ICML 2008

Proceedings, Twenty-Fifth International Conference on Machine Learning

Edited by Andrew McCallum and Sam Roweis

ISBN 978-1-60558-205-4

Printed in Madison, Wisconsin, United States of America.

Table of Contents

ICML 2008 Papers

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Preface

This volume contains the papers accepted to the 25^{th} International Conference on Machine Learning (ICML 2008). ICML is the annual conference of the International Machine Learning Society (IMLS), and provides a venue for the presentation and discussion of current research in the field of machine learning. These proceedings can also be found online at `http://www.machinelearning.org`.

This year, ICML was held July $5-9$ at the University of Helsinki, in Helsinki, Finland, and was co-located with COLT-2008, the 21^{st} Annual Conference on Computational Learning Theory, and UAI-2008, the 24^{th} Conference on Uncertainty in Artificial Intelligence.

No less than 583 papers were submitted to ICML 2008. There was a very thorough review process, in which each paper was reviewed double-blind by three program committee (PC) members. Authors were able to respond to the initial reviews, and the PC members could then modify their reviews based on online discussions and the content of this author response. There were two discussion periods led by the senior program committee (SPC), one just before and one after the submission of author responses. At the end of the second discussion period, the SPC members gave their recommendations and provided a summary review for each of their papers. Some papers were checked by the SPCs to ensure that reviewer comments had been addressed. Apart from the length restrictions on papers and the compressed time frame, the review process for ICML resembles that of many journal publications. In total, 158 papers were accepted to ICML this year, including a small number of papers which were initially conditionally accepted, yielding an overall acceptance rate of 27%.

ICML authors presented their papers both orally and in a poster session, allowing time for detailed discussions with any interested attendees of the conference. Each day of the main conference included one or two invited talks by a prominent researcher. We were very fortunate to be able to host Michael Collins, of the Massachusetts Institute of Technology; Andrew Ng, of Stanford University; and Luc De Raedt, of the Katholieke Universiteit Leuven, and John Winn of Microsoft Research Cambridge. In addition to the technical talks, ICML-2008 also included nine tutorials held before the main conference, presented by Alex Smola, Arthur Gretton, and Kenji Fukumizu; Bert Kappen and Marc Toussaint; Neil Lawrence; Martin Wainwright; Ralf Herbrich and Thore Graepel; Andreas Krause and Carlos Guestrin; Shai Shalev-Shwartz and Yoram Singer; Rob Fergus; and Matthias Seeger. This year our workshops were organized jointly with COLT and UAI as part of a special "overlap day," consisting of eleven workshops selected and arranged collaboratively by the respective workshop chairs of the three conferences. This day provided a rich opportunity for interaction among the attendees of the conferences.

This year, ICML enlarged its award offerings to match several other well-established conferences. We hope these will help build our community, celebrate our advances, and encourage applications and long-term thinking. In addition to our previously traditional "Best Paper" and "Best Student Paper" awards, we also gave awards for "Best Application Paper" and "10-year Best Paper" (for the best paper of ICML 1998, optionally given in conjunction with

a co-located conference). We thank the Machine Learning Journal for sponsoring some of our paper awards.

The organization of ICML-2008 involved efforts from many people, to whom we are extremely grateful. As program chairs, we worked closely with the general chair, William Cohen, and the local arrangements chair, Hannu Toivonen. The tutorials chair, Chris Williams, and the workshop co-chairs, Sanjoy Dasgupta and Michael Littman, also made valuable contributions to the program, and the publication chair, Ricardo Silva, performed a substantial and invaluable service in arranging for publication of the proceedings and ensuring the quality and uniformity of the papers it contained. We wish to thank Lise Getoor and Rich Caruana, the funding co-chairs for IMLS, who secured numerous sponsors for ICML; Noah Smith, the student funding chair, who dispersed student travel awards; and Matti Kääriäinen, the volunteer chair, who arranged for student volunteers, and made sure the conference ran smoothly, and Carlos Guestrin, who chaired our awards sub-committee. We also wish to thank Steven Scott, the treasurer of IMLS, for his support and advice on financial issues, and Greger Lindén for his work as webmaster for ICML-2008. We also wish to thank Rich Gerber and Paolo Gai, of SoftConf.com, who administered the START V2 software used for the conference.

For more general support, we are grateful to the members of IMLS for their advice. We are also very grateful to the many financial sponsors of ICML (who are listed elsewhere in these proceedings) for their support of this conference.

No technical conference is possible without the efforts of reviewers, so we wish to thank the Senior Program Committee, the Program Committee, and the additional reviewers for ICML-2008 for their careful and conscientious reviewing, which ensured the technical quality of the papers in these proceedings. Finally, and perhaps most importantly, we wish to thank the authors who elected to submit their work to ICML-2008, and the people who attended the conference. We hope that this volume will be a useful resource to them, and the other members of the machine learning community.

Sincerely,

Andrew McCallum and Sam Roweis
ICML 2008 Program Co-chairs

ICML 2008 Organization

General Chair

William Cohen, *Carnegie Mellon University, U.S.A.*

Program Co-chairs

Andrew McCallum, *University of Massachusetts Amherst, U.S.A.*
Sam Roweis, *University of Toronto and Google, U.S.A.*

Workshops Co-chairs

Sanjoy Dasgupta, *University of California, San Diego, U.S.A.*
Michael Littman, *Rutgers University, U.S.A.*

Tutorials Chair

Chris Williams, *University of Edinburgh, United Kingdom*

Publications Chair

Ricardo Silva, *University of Cambridge, United Kingdom*

Volunteers Chair

Matti Kääriäinen, *University of Helsinki, Finland*

Student Funding Chair

Noah Smith, *Carnegie Mellon University, U.S.A.*

Fund Raising Co-chairs

Lise Getoor, *University of Maryland, U.S.A.*
Rich Caruana, *Cornell University, U.S.A.*

Local Arrangements Chair

Hannu Toivonen, *University of Helsinki, Finland*

Treasurer and Webmaster

Greger Lindén, *University of Helsinki, Finland*

Senior Program Committee

David Blei	Léon Bottou	Michael Bowling	Carla Brodley
Wray Buntine	Michael Collins	Nello Cristianini	Luc De Raedt
Pedro Domingos	Peter Flach	David Forsyth	Johannes Fürnkranz
Lise Getoor	Mark Girolami	Amy Greenwald	Russ Greiner
Carlos Guestrin	Sven Koenig	Neil Lawrence	Ryan McDonald
Steven Minton	David Page	Diona Precup	Joaquin Quiñonero-Candela
Carl Rasmussen	Nicholas Roy	Sunita Sarawagi	Lawrence Saul
Dale Schuurmans	Michele Sebag	John Shawe-Taylor	Nathan Srebro
Richard Sutton	Yee Whye Teh	Luís Torgo	Manfred Warmuth
Max Welling	ChengXiang Zhai	Alice Zheng	

Program Committee

Douglas Aberdeen	Deepak Agarwal	Charu Aggarwal	Phaedra Agius
Edoardo Airoldi	Yasemin Altun	Stuart Andrews	Cédric Archambeau
Marta Arias	Arik Azran	Francis Bach	Suhrid Balakrishnan
Maria-Florina Balcan	Arindam Banerjee	Ziv Bar-Joseph	Kobus Barnard
Andrew Barto	Regina Barzilay	Sumit Basu	Patrick Beeson
Mikhail Belkin	Shai Ben-David	Bettina Berendt	Tamara Berg
Alina Beygelzimer	Mikhail Bilenko	Aude Billard	Gilles Blanchard
David Blei	John Blitzer	Hendrik Blockeel	Joe Bockhorst
Karsten Borgwardt	Henrik Boström	Leon Bottou	Stéphane Boucheron
Michael Bowling	Janez Brank	Carla Brodley	Emma Brunskill
Olivier Buffet	Joachim Buhmann	Vadim Bulitko	Razvan Bunescu
John Burge	Chris Burges	Colin Campbell	Nicola Cancedda
Barbara Caputo	François Caron	Miguel Carreira-Perpiñán	Xavier Carreras
Nicolò Cesa-Bianchi	Yu-Han Chang	Gal Chechik	Tanzeem Choudhury
Wei Chu	Massimiliano Ciaramita	Alex Clark	Michael Coen
David Cohn	Michael Collins	Aaron Courville	Koby Crammer
Nello Cristianini	Lehel Csato	Aron Culotta	James Cussens
Andrea Danyluk	Sanjoy Dasgupta	Denver Dash	Alexandre d'Aspremont
Hal Daumé III	Jesse Davis	Tijl De Bie	Nando de Freitas
Luc De Raedt	Ofer Dekel	Janez Demsar	Thomas Dietterich
Frank DiMaio	Chris Ding	Pedro Domingos	Kurt Driessens
Christopher Drummond	Miroslav Dudik	Jennifer Dy	Sašo Džeroski
James Early	Tina Eliassi-Rad	Charles Elkan	Yaakov Engel
Barbara Engelhardt	Hui Fang	Alan Fern	Xiaoli Fern
Cesar Ferri	Peter Flach	François Fleuret	David Forsyth
Blaz Fortuna	Eibe Frank	Paolo Frasconi	Marcus Frean
Dayne Freitag	Yoav Freund	Johannes Fürnkranz	Giorgio Fumera
Thomas Gärtner	Claudio Gentile	Lise Getoor	Pierre Geurts
Zoubin Ghahramani	Rayid Ghani	Mohammad Ghavamzadeh	Ali Ghodsi
Remi Gilleron	Aristides Gionis	Mark Girolami	Amir Globerson
Jacob Goldberger	Dilan Gorur	Yves Grandvalet	Amy Greenwald
Russ Greiner	Peter Grünwald	Marco Grzegorczyk	Yann Guermeur
Carlos Guestrin	Yuhong Guo	Zaid Harchaoui	Matthias Hein
Katherine Heller	Carl Henrik Ek	Mark Herbster	Natalia Hernandez-Gardiol
José Hernández-Orallo	Tom Heskes	Thomas Hofmann	Derek Hoiem

Vasant Honavar
Chun-Nan Hsu
Charles Isbell
Robert Jenssen
Rie Johnson
Adam Kalai
Roni Khardon
Philip Koehn
Zhenzhen Kou
Sanjiv Kumar
Michail Lagoudakis
Neil Lawrence
Wei Li
Nikolas List
Dan Lizotte
Gábor Lugosi
Alessia Mammone
Dragos Margineantu
Luke McDowell
Roland Memisevic
Steven Minton
Pawan Mudigonda
Roderick Murray-Smith
Daniel Neill
Alexandru Niculescu-Mizil
Richard Nock
David Page
Emilio Parrado-Hernández
Francisco Pereira
Slav Petrov
John Platt
Jefferson Provost
Alexander Rakhlin
Carl Rasmussen
Balaraman Ravindran
Marko Robnik-Šikonja
Dan Roth
Ruslan Salakhutdinov
Tobias Scheffer
Nicol Schraudolph
Michele Sebag
Gregory Shakhnarovich
Luo Si
David Smith
Ed Snelson
David Sontag
Amos Storkey
Charles Sutton
Prasad Tadepalli
Yee Whye Teh
Jo-Anne Ting

Pengyu Hong
Eyke Huellermeier
Tommi Jaakkola
Rong Jin
Rosie Jones
Bert Kappen
Sergey Kirshner
Sven Koenig
Stefan Kramer
Dima Kuzmin
Terran Lane
Guy Lebanon
Percy Liang
Jennifer Listgarten
Huma Lodhi
Omid Madani
Gideon Mann
Bhaskara Marthi
Amy McGovern
Donald Metzler
Dunja Mladenic
Remi Munos
Ion Muslea
Jennifer Neville
Zaiqing Nie
Guillaume Obozinski
Chris Pal
Vladimir Pavlovic
Fernando Pérez-Cruz
Bernhard Pfahringer
Barnabás Póczos
Joaquin Quiñonero-Candela
Alain Rakotomamonjy
Nathan Ratliff
Soumya Ray
Simon Rogers
Volker Roth
Sunita Sarawagi
Katya Scheinberg
Oliver Schulte
Marc Sebban
Guy Shani
David Silver
Noah Smith
Stephen Soderland
Alessandro Sperduti
Jan Struyf
Richard Sutton
Brian Tanner
Ambuj Tewari
Ivan Titov

Peter Hooper
Dirk Husmeier
Tony Jebara
Thorsten Joachims
Matti Kääriäinen
Eamonn Keogh
Jyrki Kivinen
Joost Kok
Andreas Krause
Nicolas Lachiche
John Langford
Jure Leskovec
Chih-Jen Lin
Huan Liu
Phil Long
Donato Malerba
Shie Mannor
Jon McAuliffe
Brendan McMahan
Matt Michelson
Claire Monteleoni
Kevin Murphy
Ramesh Nallapati
David Newman
Yael Niv
Miles Osborne
Ulrich Paquet
Kristiaan Pelckmans
Theodore Perkins
Petra Philips
Pascal Poupart
Gunnar Rätsch
Ganesh Ramakrishnan
Magnus Rattray
Matthew Richardson
Khashayar Rohanimanesh
Daniel Roy
Anoop Sarkar
Bruno Scherrer
Dale Schuurmans
Matthias Seeger
John Shawe-Taylor
Özgür Şimşek
Trey Smith
Peter Sollich
Nathan Srebro
Erik Sudderth
Umar Syed
Graham Taylor
Romain Thibaux
David Tolliver

Tamas Horvath
Alex Ihler
David Jensen
Mark Johnson
Ata Kaban
Kristian Kersting
Levente Kocsis
Igor Kononenko
Jeremy Kubica
Simon LaCoste-Julien
Pedro Larranaga
Christina Leslie
Charles Ling
Tie-Yan Liu
Elliot Ludvig
Mark Maloof
Daniel Marcu
Ryan McDonald
Chris Meek
Brian Milch
Joris Mooij
Iain Murray
Sriraam Natarajan
Xuanlong Nguyen
William Noble
Sarah Osentoski
Ronald Parr
Daniel Pelleg
Claudia Perlich
Joelle Pineau
Doina Precup
Ali Rahimi
Deva Ramanan
Pradeep Ravikumar
Martin Riedmiller
Romer Rosales
Nicholas Roy
Lawrence Saul
Jeff Schneider
Stephen Scott
Fei Sha
Christian Shelton
Noam Slonim
Alex Smola
Soeren Sonnenburg
Peter Stone
Ilya Sutskever
Csaba Szepesvári
Matthew Taylor
Bo Thiesson
Luís Torgo

Antonio Torralba
Bill Triggs
Antal van den Bosch
Vishy Vishwanathan
Manfred Warmuth
Eric Wiewiora
Ole Winther
Qiang Yang
Elad Yom-Tov
Hugo Zaragoza
ChengXiang Zhai
Sandra Zilles
Kristina Toutanova
Kagan Tumer
Maarten van Someren
Michalis Vlachos
Kilian Weinberger
Dana Wilkinson
Frank Wood
Yiming Yang
Kai Yu
Gerson Zaverucha
Yi Zhang
Martin Zinkevich
Volker Tresp
Lyle Ungar
Nuno Vasconcelos
Kiri Wagstaff
Gary Weiss
Oliver Williams
Jenn Wortman
Jieping Ye
Bianca Zadrozny
Luke Zettlemoyer
Alice Zheng
Konstantin Tretjakov
Raquel Urtasun
Jean-Philippe Vert
Jack Wang
Max Welling
David Wingate
Stefan Wrobel
Wen-Tau Yih
Mohammed Zaki
Hongyuan Zha
Xiaojin Zhu

External Reviewers

Amrudin Agovic
Laura Antanas
Andrew Bagnell
Asa Ben-Hur
Steffen Bickel
Mario Boley
Janez Brank
Giovanni Cavallanti
Jianhui Chen
Chad Cumby
Sanjoy Dasgupta
Kun Deng
Christos Dimitrakakis
Alain Dutech
Vojtěch Franc
Johannes Fürnkranz
Valentin Gjorgjioski
Arthur M. Greene
Roland Hafner
Ricardo Henao
Jin Huang
Dmitry Kamenetsky
Jörg Kindermann
Matjaz Kukar
Martin Lauer
Steve Lianoglou
Gaelle Loosli
Neville Mehta
Emmanuel Monfrini
Francesco Orabona
Scott Proper
Franck Rapaport
Ulrich Rückert
Rob Schapire
Oliver Serang
Blake Shaw
Edoardo Airoldi
Annalisa Appice
Dorit Baras
Eva Besada-Portas
Ronald Bjarnason
Zoran Bosnić
Agnès Braud
Michelangelo Ceci
Boris Chidlovskii
William Dabney
Kurt De Grave
Liam Mac Dermend
Shilin Ding
Marc Dymetman
Gustavo Frigo
Stanislav Funiak
Shantanu Godbole
Rahul Gupta
Peter Haider
Andrew Howard
Arya Irani
Motoaki Kawanabe
Aaron Klammer
Anshul Kundaje
Greg Lee
Christoph Lippert
Daniel Lowd
Pauli Miettinen
Joshua C. Neil
Aline Marins Paes
Yanjun Qi
Achim Rettinger
Stefan Rueping
Bruno Scherrer
Manoj Seshadrinathan
Javen (Qinfeng) Shi
Babak Alipanahi
Andreas Argyriou
Kedar Bellare
Sooraj Bhat
John Blitzer
Guillaume Bouchard
Markus Bundschus
Olivier Chapelle
Koby Crammer
Eric Dallal
Fernando De La Torre
Uwe Dick
Janardhan Rao Doppa
Jason Ernst
Elisa Fromont
Thomas Gabel
Robby Goetschalckx
Amaury Habrard
Steve Hanneke
Derek Hao Hu
Shuiwang Ji
Pooyan Khajehpour
Christine Körner
Abhimanyu Lad
Yoong Keok Lee
Jun Liu
Ryan McDonald
Olana Missura
Blaž Novak
Sinno Jialin Pan
Jian Qiu
Andrew W. Robertson
Jan Rupnik
Leander Schietgat
Fei Sha
Yanxin Shi
Galen Andrew
Aaron Arvey
Marc G. Bellemare
Indrajit Bhattacharya
Avrim Blum
Michael Brückner
Bin Cao
Jiang Chen
Aron Culotta
Amir Danak
Aloísio Carlos de Pina
Emmanuel Didiot
Hao Du
Thomas Finley
Qiang Fu
Subramaniam Ganapathy
Thore Graepel
Gholamreza Haffari
Abhay Harpale
Xu Huan
Brad Joyce
Hassan Khosravi
Petra Kralj
Sascha Lange
Xuejing Li
Nan Liu
Wannes Meert
Dunja Mladenic
Cheng Soon Ong
Sergey Plis
Maria José Ramírez-Quintana
Sushmita Roy
Konstantin Salomatin
Gabriele Schweikert
Hanhuai Shan
Pannaga Shivaswamy

Shai Shalev Shwartz	Yoram Singer	Ivica Slavkov	Le Song
Nati Srebro	Nisheeth Srivastava	Erik trumbelj	Liang Sun
Peter Sunehag	Mihai Surdeanu	Charles Sutton	Grzegorz Swirszcz
Lei Tang	Ben Taskar	Katerina Taskova	Choon Hui Teo
Vincent Thomas	Stephan Timmer	Raquel Urtasun	Stijn Vanderlooy
Joaquin Vanschoren	Shankar Vembu	Gang Wang	Xuanhui Wang
Markus Weimer	Adam White	Aaron Wilson	Chunyu Yang
Chun-Nam Yu	Jia Yuan Yu	Jin Yu	Kai Yu
Yisong Yue	Karina Zapien	Bernard Zenko	Harry H. Zhang
Peng Zhang	Xinhua Zhang	Zhenyue Zhang	Zheng Zhao
Vincent W. Zheng	Peng Zhou	Alexander Zien	

Local Arrangements Volunteers

Tapio Elomaa	Samuel Kaski	Arto Klami	Kati Kervinen
Sanna Kettunen	Jyrki Kivinen	Veli Mäkinen	Petri Myllymäki

Publications Volunteers

Finale Doshi Ryan Turner Jurgen Van Gael

Logo, Web and Cover Design

Jyri Tuulos

Registration System

Kimmo Kulovesi

Board of the International Machine Learning Society

President, Raymond J. Mooney **Secretary**, Satinder Singh **Treasurer**, Stephen Scott
Funding co-chairs, Rich Caruana and Lise Getoor

William Cohen	Luc De Raedt	Tom Dietterich
Zoubin Ghahramani	Rob Holte	Michael Jordan
Pat Langley	Nada Lavrač	Andrew McCallum
Andrew Moore	Foster Provost	Stuart Russell
Rob Schapire	Bernhard Schölkopf	Ian Witten
Stefan Wrobel		

Governing Bodies

The International Machine Learning Society (IMLS), `http://www.machinelearning.org`.

ICML 2008 Funding

The organizers of ICML 2008 would like to thank the following sponsors:

Microsoft Research	University of Helsinki	Federation of Finnish Learned Societies
Yahoo!	Intel Corporation	Google
Xerox	IBM	Machine Learning Journal/Springer
NSF	Helsinki Institute for Information Technology	Pascal

Other Funding

We would also like to thank an anonymous donor.

ICML 2008 Invited Talks

Structured Prediction Problems in Natural Language Processing

Michael Collins
Massachussets Institute of Technology, U.S.A.

Abstract:
Modeling language at the syntactic or semantic level is a key problem in natural language processing, and involves a challenging set of structured prediction problems. In this talk I'll describe work on machine learning approaches for syntax and semantics, with a particular focus on lexicalized grammar formalisms such as dependency grammars, tree adjoining grammars, and categorial grammars. I'll address key issues in the following areas: 1) the design of learning algorithms for structured linguistic data; 2) the design of representations that are used within these learning algorithms; 3) the design of efficient approximate inference algorithms for lexicalized grammars, in cases where exact inference can be very expensive. In addition, I'll describe applications to machine translation, and natural language interfaces.

STAIR: The STanford Artificial Intelligence Robot Project

Andrew Ng
Stanford University, U.S.A.

Abstract:
This talk will describe the STAIR home assistant robot project, and several satellite projects that led to key STAIR components such as (i) robotic grasping of previously unknown objects, (ii) depth perception from a single still image, and (iii) apprenticeship learning for control.

Since its birth in 1956, the AI dream has been to build systems that exhibit broad-spectrum competence and intelligence. STAIR revisits this dream, and seeks to integrate onto a single robot platform tools drawn from all areas of AI including learning, vision, navigation, manipulation, planning, and speech/NLP. This is in distinct contrast to, and also represents an attempt to reverse, the 30 year old trend of working on fragmented AI sub-fields. STAIR's goal is a useful home assistant robot, and over the long term, we envision a single robot that can perform tasks such as tidying up a room, using a dishwasher, fetching and delivering items, and preparing meals.

STAIR is still a young project, and in this talk I'll report on our progress so far on having STAIR fetch items from around the office. Specifically, I'll describe: (i) learning to grasp previously unseen objects (including its application to unloading items from a dishwasher); (ii) probabilistic multi-resolution maps, which enable the robot to open/use doors; (iii) a robotic foveal+peripheral vision system for object recognition and tracking. I'll also outline some of the main technical ideas − such as learning 3-d reconstructions from a single still image, and reinforcement learning algorithms for robotic control − that played key roles in enabling these STAIR components.

Logical and Relational Learning Revisited

Luc De Raedt
Katholieke Universiteit Leuven, Belgium

Abstract:
I use the term *logical and relational learning* (LRL) to refer to the subfield of machine learning and data mining that is concerned with learning in expressive logical or relational representations. It is the union of inductive logic programming, (statistical) relational learning and multi-relational data mining and constitutes a general class of techniques and methodology for learning from structured data (such as graphs, networks, relational databases) and background knowledge.

During the course of its existence, logical and relational learning has changed dramatically. Whereas early work was mainly concerned with logical issues (and even program synthesis from examples), in the 90s its focus was on the discovery of new and interpretable knowledge from structured data, often in the form of rules or patterns. Since then the range of tasks to which logical and relational learning has been applied has significantly broadened and now covers almost all machine learning problems and settings. Today, there exist logical and relational learning methods for reinforcement learning, statistical learning, distance- and kernel-based learning in addition to traditional symbolic machine learning approaches.

At the same time, logical and relational learning problems are appearing everywhere. Advances in intelligent systems are enabling the generation of high-level symbolic and structured data in a wide variety of domains, including the semantic web, robotics, vision, social networks, and the life sciences, which in turn raises new challenges and opportunities for logical and relational learning,

These developments have led to a new view on logical and relational learning and its role in machine learning and artificial intelligence. In this talk, I shall reflect on this view by identifying some of the lessons learned in logical and relational learning and formulating some challenges for future developments.

Probabilistic Models for Understanding Images

John Winn
Microsoft Research Cambridge, United Kingdom

Abstract:
Getting a computer to understand an image is challenging due to the numerous sources of variability that influence the imaging process. The pixels of a typical photograph will depend on the scene type and geometry, the number, shape and appearance of objects present in the scene, their 3D positions and orientations, as well as effects such as occlusion, shading and shadows. The good news is that research into physics and computer graphics has given us a detailed understanding of how these variables affect the resulting image. This understanding can help us to build the right prior knowledge into our probabilistic models of images. In theory, building a model containing all of this knowledge would solve the image understanding problem. In practice, such a model would be intractable for current inference methods. The open challenge for machine learning and machine vision researchers is to create a model which captures the imaging process as accurately as possible, whilst remaining tractable for accurate inference. To illustrate this challenge, I will show how different aspects of the imaging process can be incorporated into models for object detection and segmentation, and discuss techniques for making inference tractable in such models.

ICML 2008
Workshop and Tutorial Summaries

Overview of Workshops and Tutorials

Once again, ICML solicited and hosted world-class workshops on topics related to machine learning. This year, we were delighted to collaborate with the workshop chairs of the UAI (Nando de Freitas) and COLT (John Langford) conferences to put together an exciting joint program. We constructed a slate of 13 workshops that represent a wide range of perspectives and fields, as seen in the summaries below. All workshops were held on July 9th, immediately after the main conference days. We would like to thank all of the workshop organizers for their service to the community in putting together these high-quality meetings. We also thank the outstanding local arrangement chairs and the general and program chairs for ICML and the other conferences for creating another exciting and successful conference.

Sanjoy Dasgupta and Michael L. Littman
ICML 2008 Workshop Chairs

As in previous years we were pleased to have a strong programme of tutorials for ICML 2008. These were held on 5 July, immediately preceding the main conference. The programme featured nine tutorials covering a wide range of methods in and applications of machine learning. There were tutorials on: embedding distributions in reproducing kernel Hilbert spaces (Smola, Gretton, Fukumizu); stochastic optimal control theory (Kappen, Toussaint); probabilistic dimensionality reduction (Lawrence); message-passing and relaxations in graphical models (Wainwright); machine learning applications in computer games (Herbrich, Graepel); submodularity in machine learning (Krause, Guestrin); theory and applications of online learning (Shalev-Shwartz, Singer); visual object recognition and retrieval (Fergus); and sparse linear models (Seeger). We would like to thank the community for the high-quality tutorial proposals that were received, the presenters for their extensive efforts in preparing and delivering the selected tutorials, and the local arrangements, programme and general chairs of ICML for their hard work in organizing such a stimulating conference.

Chris Williams
ICML 2008 Tutorial Chair

Workshops

W1: Bayesian Modelling Applications

Suzanne M. Mahoney, Innovative Decisions, Inc, U.S.A.
Silja Renooij, Utrecht University, the Netherlands
Hermi J.M. Tabachneck-Schijf, Utrecht University, the Netherlands

The Bayesian Modelling Applications Workshop provides a focused but informal forum for

fruitful exchanges among theorists, practitioners and tool developers, covering research questions and insights, methodologies, techniques, and experiences with applications of Bayesian models to any particular problem domain. Apart from an overall focus on Bayesian modelling, this years 6th edition has as special theme "HOW BIASED ARE OUR NUMBERS?". This theme focusses on issues relating to (probability) biases in applications of Bayesian networks. We seek insight in and examples of and solutions to encountered biases in sources of probabilistic information, or introduced by the methods, new or existing, for obtaining and communicating the numbers. In addition, we are interested in methods for identifying biases in the numbers, and for establishing their effect on model behaviour.

W2: The 3rd Workshop on Evaluation Methods for Machine Learning

Chris Drummond, NRC Institute for Information Technology, Canada
Nathalie Japkowicz, University of Ottawa, Canada
William Klement, University of Ottawa, Canada
Sofus A. Macskassy, Fetch Technologies, U.S.A.

This workshop is the third in a series, the previous ones having taken place at AAAI over the past two years. Our continuing goal is to encourage debate within the machine learning community into how we experimentally evaluate new algorithms. The earlier workshops were successful in that they began the process of presentation, and discussion, of new ideas for evaluation. However, they did not raise all the high-level questions we believe must be addressed by the community. For this reason, we have changed the format of the workshop. First, we will hold it at ICML. Here, with access to a much larger group of ML researchers we expect to hear from many more voices that have an interesting take on the issue. Second, we solicited position papers rather than research papers. This way instead of getting lost into the nitty-gritty details of particular new evaluation methods, we can address the important, high-level, issues surrounding machine learning evaluation.

W3: International Workshop on Machine Learning and Music (MML 2008)

Rafael Ramirez, Universitat Pompeu Fabra, Spain
Christina Anagnostopoulou, University of Athens, Greece
Darrell Conklin, City University, United Kingdom
José Manuel Iñesta, Alicante University, Spain
Xavier Serra, Universitat Pompeu Fabra, Spain

With the current explosion and quick expansion of music in digital formats, research on machine learning and music is gaining increasing popularity. As complexity of the problems investigated by researchers on this area increases, there is a need to develop new algorithms and methods to solve these problems. Machine learning has proved to provide efficient solutions to many music-related problems. The application of related techniques to the development of music processing systems is an active, exciting and significant area of research which has become an established field of research. The goal of the workshop is to bring together researchers who are using machine learning in musical applications, providing the opportunity to promote, present and discuss ongoing work in the area.

W4: Machine Learning for Health Care Applications

Milos Hauskrecht, University of Pittsburgh, U.S.A.
Dale Schuurmans, University of Alberta, Canada
Csaba Szepesvári, University of Alberta, Canada

Health-care applications have been and continue to be the source of inspiration for many areas of artificial intelligence research. Many advances in various sub-specialties of AI have been inspired by challenges posed by medical problems. A new challenge for AI in general, but machine learning in particular, arises from the wealth and variety of data generated in modern medical and health-care settings. Extensive electronic health and medical records − with thousands of fields recording patient conditions, diagnostic tests, treatments, outcomes, and so on − provide an unprecedented source of information that can provide clues leading to potential improvements in disease detection, chronic disease management, design of clinical trials, and other aspects of health-care. The purpose of this workshop is to bring together machine learning researchers interested in problems and applications in health-care, with the goal of exchanging ideas and perspectives, identifying important and challenging applications, and raising awareness of potential health-care applications in the machine learning community. The workshop program will consists of presentations by invited speakers and authors of the papers submitted and accepted to the workshop. A panel session focusing on the main challenges and open problems in the field will be held at the end of the workshop.

W5: Nonparametric Bayes

Yee Whye Teh, University College London, United Kingdom
Romain Thibaux, University of California, Berkeley, U.S.A.
Athanasios Kottas, University of California, Santa Cruz, U.S.A.
Zoubin Ghahramani, University of Cambridge, United Kingdom
Michael Jordan, University of California, Berkeley, U.S.A.

One of the major problems driving current research in statistical machine learning is the search for ways to exploit highly-structured models that are both expressive and tractable. Nonparametric Bayesian methodology provides significant leverage on this problem. In the nonparametric Bayesian framework, the prior distribution is not a fixed parametric form, but is rather a general stochastic process − a distribution over a possibly uncountably infinite number of random variables. This generality makes it possible to work with prior and posterior distributions on objects such as trees of unbounded depth and breadth, graphs, partitions, sets of monotone functions, sets of smooth functions and sets of general measures. Applications of nonparametric Bayesian methods have begun to appear in disciplines such as information retrieval, natural language processing, machine vision, computational biology, cognitive science and signal processing. This workshop is intended to bring together the growing community of nonparametric Bayesian researchers. The issues we wish to address include: development of general-purpose software packages for nonparametric Bayesian models, efficient inference, and new models, methodologies, theoretical frameworks and applications.

W6: PASCAL Large Scale Learning Challenge

Soeren Sonnenburg, Fraunhofer Institute FIRST, Germany
Vojtech Franc, Fraunhofer Institute FIRST, Germany
Elad Yom-Tov, IBM Haifa Research Lab, Israel
Michele Sebag, LRI, France

With the exceptional increase in computing power, storage capacity and network bandwidth of the past decades, ever growing datasets are collected. While the data size growth leaves computational methods as the only viable way of dealing with data, it poses new challenges to ML methods. The PASCAL Large Scale Learning challenge is concerned with the scalability and efficiency of existing ML approaches with respect to computational, memory or communication resources.

Indeed many comparisons are presented in the literature; however, these usually focus on assessing few algorithms and aspects. As a result it is difficult to determine how a method compares to others in terms of test error, training time and memory requirements, which are the practically relevant criteria.

The workshop will serve to disseminate the challenge results and announce the winners of the competition. Authors of the best and most original contributions will present their work. Furthermore a panel discussion will be devoted to establishing a principled framework for the validation of large scale learning methods.

W7: Second Planning to Learn Workshop (PlanLearn)

Pavel Brazdil, University of Porto, Portugal
Avi Bernstein, University of Zurich, Switzerland
Larry Hunter, University of Colorado at Denver and Health Sciences Center, USA

The task of constructing composite systems, that is systems composed of more than one part, can be seen as interdisciplinary area which builds on expertise in different domains. The aim of this workshop is to explore the possibilities of constructing such systems with the aid of Machine Learning and exploiting the know-how of Data Mining. One way of producing composite systems is by inducing the constituents and then by putting the individual parts together. This problem can be seen as a problem of planning to resolve multiple (possibly interacting) tasks. So, one important issue that needs to be addressed is how these multiple learning processes can be coordinated. Each task is resolved using certain ordering of operations. Meta-learning and knowledge transfer can be useful in this process. It can help us to retrieve previous solutions conceived in the past and re-use them in new settings. The aim of the workshop is to explore the possibilities of this new area, offer a forum for exchanging ideas and experience concerning the state-of-the art, permit to bring in knowledge gathered in different but related and relevant areas and outline new directions for research.

W8: Prior Knowledge for Text and Language Processing

Marc Dymetman, Xerox Research Centre Europe, France
Guillaume Bouchard, Xerox Research Centre Europe, France
Hal Daumé III, University of Utah, U.S.A.
Yee Whye Teh, University College London, United Kingdom

The aim of the workshop is to present and discuss recent advances in machine learning approaches to text and natural language processing that capitalize on rich prior knowledge models in these domains.

Traditionally, in Machine Learning, a strong focus has been put on data-driven methods that assume little a priori knowledge on the part of the learning mechanism. Such techniques have proven quite effective not only for simple pattern recognition tasks, but also, more surprisingly, for such tasks as language modeling in speech recognition using basic n-gram models. However, when the structures to be learned become more complex, even large training sets become sparse relative to the task, and this sparsity can only be mitigated if some prior knowledge comes into play to constrain the space of fitted models. We currently see a strong emerging trend in the field of machine learning for text and language processing to incorporate such prior knowledge for instance in language modeling (e.g. through non-parametric Bayesian priors) or in document modeling (e.g. through hierarchical graphical models). There are complementary attempts in the field of statistical computational linguistics (e.g in statistical machine translation) to build hybrid systems that do not rely uniquely on corpus data, but also exploit some form of a priori grammatical knowledge, bridging the gap between purely data-oriented approaches and the traditional purely rule-based approaches, that do not rely on automatic corpus training, but only indirectly on human observations about linguistic data. The domain of text and language processing thus appears as a very promising field for studying the interactions between prior knowledge and raw training data, and this workshop aims at providing a forum for discussing recent theoretical and practical advances in this area.

W9: Recent Breakthroughs in Minimum Description Length Learning

Tim van Erven, CWI, the Netherlands
Peter Grünwald, CWI, the Netherlands
Petri Myllymäki, University of Helsinki, Finland
Teemu Roos, Helsinki Institute for Information Technology, Finland
Ioan Tabus, Tampere University of Technology, Finland

During the last few years (2004-2007), there have been several breakthroughs in the area of Minimum Description Length (MDL) modeling, learning and prediction. These breakthroughs concern the efficient computation and proper formulation of MDL in parametric problems based on the ”normalized maximum likelihood”, as well as altogether new, and better, coding schemes for nonparametric problems. This essentially solves the so-called AIC-BIC dilemma, which has been a central problem in statistical model selection for more than 20 years now. The goal of this workshop is to introduce these exciting new developments to the ML and UAI communities, and to foster new collaborations between interested researchers.

W10: Second Annual Reinforcement Learning Competition (RL 2008)

Shimon Whiteson, Universiteit van Amsterdam, the Netherlands
Adam White, University of Alberta, Canada
Rich Sutton, University of Alberta, Canada
Doina Precup, McGill University, Canada
Peter Stone, University of Texas at Austin, U.S.A.
Michael Littman, Rutgers University, U.S.A.
Nikos Vlassis, Technical University of Crete, Greece
Martin Riedmiller, Universität Osnabrück, Germany

The Second Annual Reinforcement Learning Competition is an opportunity for reinforcement learning researchers to rigorously compare the performance of their methods on a suite of challenging domains, including: the game of Tetris; robot soccer keepaway, based on the RoboCup simulator; a real-time strategy (RTS) game; and a helicopter control problem, based on the work of Andrew Ng and collaborators. This year's competition will utilize new evaluation paradigms designed to encourage algorithms that generalize well to previously unseen tasks. In particular, each domain will be parameterized and test parameters will differ from those used for training. As a result, only learning algorithms that are robust across a range of parameters can expect to perform well. The competition concludes with a workshop at which the winners will be announced. Top competitors will give short presentations about their methods and several moderated discussions will be held on topics including the challenges of empirical RL and the future of the competition.

W11: Sparse Optimization and Variable Selection

Irina Rish, IBM T. J. Watson Research Center, U.S.A.
Guillermo Cecchi, IBM T. J. Watson Research Center, U.S.A.
Rajarshi Das, IBM T. J. Watson Research Center, U.S.A.
Tony Jebara, University of Columbia, U.S.A.
Gerry Tesauro, IBM T. J. Watson Research Center, U.S.A.
Martin Wainwright, University of California, Berkeley, U.S.A.

Variable selection is an important issue in many applications of machine learning and statistics where the main objective is discovering predictive patterns in data that would enhance our understanding of underlying physical, biological and other natural processes, beyond just building accurate 'black-box' predictors. Examples include biomarker selection in biological applications, identifying brain areas related to various 'mental states' based on brain imaging data, identifying a small number of bottlenecks in a large-scale computer network that best explain the network performance, and so on. Recent years have witnessed a flurry of research on algorithms and theory for variable selection and estimation involving sparsity constraints. Various types of convex relaxation, particularly L1-regularization, have proven very effective: examples include the LASSO, Elastic Net, L1-regularized GLMs, sparse classifiers such as sparse (1-norm) SVM, as well as sparse dimensionality reduction methods (e.g. sparse component analysis such as sparse PCA and sparse NMF). Applications of these methods are wide-ranging, including computational biology, neuroscience, graphical model selection, and the rapidly growing area of compressed sensing. Theoretical work has provided some conditions when various relaxation methods are capable of recovering an underlying sparse

signal, provided bounds on sample complexity, and investigated trade-offs between different choices of design matrix properties that guarantee good performance. The goal of this workshop is to bring together researchers working on the methodology, theory and applications of sparse models and selection methods to share their experiences and insights into both the basic properties of the methods, and the properties of the application domains that make particular methods more (or less) suitable. We hope to further explore connections between variable selection and related areas such as dimensionality reduction, optimization and compressed sensing.

Tutorials

T1: Painless Embeddings of Distributions: the Function Space View

Alex Smola, NICTA, Australia
Arthur Gretton, Max Planck Institute for Biological Cybernetics, Germany
Kenji Fukumizu, Institute of Statistical Mathematics, Japan

In the early days of kernel machines research, the "kernel trick" was considered a useful way of constructing nonlinear algorithms from linear ones. More recently, however, it has become clear that a potentially more far reaching use of kernels is as a linear way of dealing with higher order statistics. For instance, in kernel independent component analysis, general nonlinear dependencies show up as linear correlations once they are computed in a suitable reproducing kernel Hilbert space. This tutorial provides an introduction to embeddings of probability distributions into reproducing kernel Hilbert spaces, as a way of painlessly dealing with high order statistics. We will cover both theoretical issues, such as conditions under which different probability distributions have unique mappings; as well as practical applications ranging from tests of distribution properties (homogeneity, independence, conditional independence) to density estimation to causal inference.

T2: Stochastic Optimal Control Theory

Bert Kappen, Radboud University, the Netherlands
Marc Toussaint, Technical University, Germany

Stochastic optimal control theory concerns the problem of how to act optimally when reward is only obtained at a later time. The stochastic optimal control problem is central to modeling of intelligent behavior in animals or machines. Examples are the control of multi-joint robot arms, navigation of vehicles, coordination of multi-agent systems, and decision making in financial applications. Classical optimal control theory is based on principles like the Hamilton-Jacobi-Bellman equation, the Pontryagin maximum principle, and special cases like the LQ-case and the Ricatti equations. More familiar to the Machine Learner are Reinforcement Learning or (Partially Observable) Markov Decision Processes which can be viewed as special cases of stochastic control theory. This tutorial aims to introduce to the classical principles as well as the more modern frameworks and thereby to provide an integrative view on the different notions. Special emphasis is given on newer approaches of

using inference techniques to solving stochastic optimal control problems. The tutorial is introductory and aimed at the 'average' machine learning researcher. No background in control theory and/or reinforcement learning is assumed. A basic understanding of Bayesian networks and statistical inference is assumed.

T3: Dimensionality Reduction, the Probabilistic Way

Neil Lawrence, University of Manchester, United Kingdom

The main focus of this tutorial will be probabilistic interpretations of dimensional reduction. It is aimed to complement the tutorial given by Lawrence Saul at NIPS 2005 on "Spectral Methods for Dimensional Reduction". Its particular focus will be probabilistic approaches to dimensional reduction based on generative models. These approaches have become increasingly popular in graphics and vision through the Gaussian Process Latent Variable Model. However, there also is a history to these methods which is perhaps less widely known amoungst the newer generation of researchers. In particular the Generative Topographic Mapping and Latent Density Networks. This tutorial will give grounding to these methods through unifying them in the context of probabilistic latent variable models. This will involve a introduction to these approaches through the mechanism of probabilistic PCA, then a discussion of density networks leading into the generative topographic mapping. Finally the dual interpretation of probabilistic PCA and its extension to the GP-LVM will be given. Throughout the tutorial we will develop intuition about the methods with an ongoing set of example data sets. A particular focus of these example data sets will be motion capture data. Motion capture data is a nice example to use because it is easy for the human eye to tell when samples from the model are realistic. One aspect of the tutorial will be the difference between the probabilistic approaches and the more commonly applied spectral approaches. In particular we will emphasise the distance preservation character of the probabilistic approaches: namely that local distances in the data are not necessarily preserved in the latent space. This contrasts with spectral algorithms which typically aim to preserve such local distances. These different characteristics mean that probabilistic approaches complement the spectral approaches, but the bring their own range of associated problems, in particular local minima in the optimisation space. Heuristics for avoiding these local minima will also be discussed.

T4: Graphical Models and Variational Methods: Message-passing and Relaxations

Martin Wainwright, University of California, Berkeley, U.S.A.

Graphical models provide a flexible framework for capturing dependencies among large collections of random variables, and are by now an essential component of the statistical machine learning toolbox. Any application of graphical models involves a core set of computational challenges, centered around the problems of marginalization, mode-finding, parameter estimation, and structure estimation. Although efficiently solvable for graphs without cycles (trees) and graphs of low treewidth more generally, exact solutions to these core problems are computationally challenging for general graphical models with large numbers of nodes and/or state space sizes. Consequently, many applications of graphical models require efficient methods for computing approximate solutions to these core problems. The past decade and a half has witnessed an explosion of activity on approximate algorithms for graphical

models. This tutorial will show how a wide class of methods − including mean field theory, sum-product or belief propagation algorithms, expectation-propagation, and max-product algorithms − are all variational methods, meaning that they can be understood as algorithms for solving particular optimization problems on graphs. The perspective also forges connections to convex optimization, including linear programming and other type of conic relaxations.

T5: Playing Machines: Machine Learning Applications in Computer Games

Ralf Herbrich, Microsoft Research Cambridge, United Kingdom
Thore Graepel, Microsoft Research Cambridge, United Kingdom

The tutorial will give an introduction to the emerging area of applying machine learning to computer games and of using computer games as test beds for machine learning. One of the key problems in computer games is the creation of AI driven agents that interact with the player so as to create a great interactive gaming experience. As a consequence a substantial part of the tutorial will consider adaptive and learning game AI based on supervised and reinforcement learning. However, computer games also offer a great variety of other challenges including problems in graphics, sound, networking, player rating and matchmaking, interface design, narrative generation etc. Selected problems from some of these areas will be discussed together with machine learning approaches to solve them. Since this is an application area, the tutorial will focus on past and recent applications, open problems and promising avenues for future research. It will also provide resources available to people who would like to work in this fascinating and fun research space.

T6: Beyond Convexity: Submodularity in Machine Learning

Andreas Krause, Carnegie Mellon University, U.S.A.
Carlos Guestrin, Carnegie Mellon University, U.S.A.

Convex optimization has become a main workhorse for many machine learning algorithms during the past ten years. When minimizing a convex loss function for, e.g., training a Support Vector Machine, we can rest assured to efficiently find an optimal solution, even for large problems. In recent years, another fundamental problem structure, which has similar beneficial properties, has emerged as very useful in a variety of machine learning applications: Submodularity is an intuitive diminishing returns property, stating that adding an element to a smaller set helps more than adding it to a larger set. Similarly to convexity, submodularity allows one to efficiently find provably (near-)optimal solutions. In this tutorial, we will give an introduction to the concept of submodularity, discuss algorithms for optimizing submodular functions and − as the main focus − illustrate their usefulness in solving difficult machine learning problems, such as active learning and sparse experimental design, informative path planning, structure learning, clustering, influence maximization and ranking.

T7: Tutorial on Theory and Applications of Online Learning

Shai Shalev-Shwartz, Toyota Technological Institute, U.S.A.
Yoram Singer, Google, U.S.A.

Online learning is a well established learning paradigm which has both theoretical and practical appeals. The goal of online learning is to make a sequence of accurate predictions

given knowledge of the correct answer to previous prediction tasks and possibly additional available information. The roots of online learning goes back to Hannan's work in the 50s. Online learning became of great interest to practitioners due the recent emergence of large scale web applications. Notable examples of web-based applications are online advertisement placement and online web ranking. The tutorial is targeted at people from all areas of machine learning and covers the formal foundations along with algorithmic and practical aspects of online learning. The goal is to provide a high-level, broad, and rigorous overview of the formal framework. By the end of tutorial the attendees should have acquired enough knowledge to be able to pin-point an online algorithm that best matches an application.

The tutorial starts with a simple example of predicting the next element of a binary sequence. We then formally introduce the basic definitions of online learning and the notion of regret analysis. Next we describe the problem of predicting with experts advice by analyzing a few algorithms and contrasting them with an impossibility result. This basic setting is then re-examined in the context of online learning of general linear predictors. We give a recent analysis which reveals an underlying primal-dual apparatus for the analysis of online algorithms. We conclude the formal part of the tutorial with a description of extensions and generalizations of online learning tasks while underscoring connections to game theory, information theory, and reinforcement learning. We recap the tutorial with two complete examples that demonstrate the usage of online learning for portfolio selection and for text filtering.

T8: Visual Object Recognition and Retrieval

Rob Fergus, New York University, U.S.A.

The tutorial will address the problem of recognizing visual object classes in images, currently the focus of much interest in Computer Vision. As recent innovations in the area draw heavily on machine learning concepts, the tutorial will attempt to highlight the growing intersection between the two areas. The material will be divided five sections, covering (i) bag of words models; (ii) parts and structure models; (iii) discriminative methods; (iv) combined recognition and segmentation and (v) retrieval schemes for large datasets. The emphasis will be on the important general concepts rather than in depth coverage of contemporary papers. The tutorial is a revised version of the prize-winning short course given at ICCV 2005 and CVPR 2007 in conjunction with Fei-Fei Li (Princeton) and Antonio Torralba (MIT).

T9: Sparse Linear Models: Bayesian Inference and Experimental Design

Matthias Seeger, Max Planck Institute for Biological Cybernetics, Germany

Sparse linear models are cornerstones of applied statistics, embodying fundamental ideas such as feature selection, shrinkage, and automatic relevance determination. While much progress has been made recently in understanding point estimation of sparse signals, Bayesian inference is needed to drive higher-level tasks such as experimental design, where valid uncertainties and covariances are more important than point estimates. In this tutorial, the major determnistic inference approximations to date (expectation propagation, sparse Bayesian learning, variational mean field Bayes) will be introduced for the sparse linear model, and their mathematics (scale mixtures, convex duality, moment matching) will be clarified. Sequential Bayesian design, with the application to optimizing an image measurement architecture, serves as motivation for this effort.

ICML 2008 Papers

Gaussian Process Product Models for Nonparametric Nonstationarity

Ryan Prescott Adams RPA23@CAM.AC.UK
Oliver Stegle OS252@CAM.AC.UK
Cavendish Laboratory, University of Cambridge, Cambridge CB3 0HE, UK

Abstract

Stationarity is often an unrealistic prior assumption for Gaussian process regression. One solution is to predefine an explicit nonstationary covariance function, but such covariance functions can be difficult to specify and require detailed prior knowledge of the nonstationarity. We propose the Gaussian process product model (GPPM) which models data as the pointwise product of two latent Gaussian processes to nonparametrically infer nonstationary variations of amplitude. This approach differs from other nonparametric approaches to covariance function inference in that it operates on the outputs rather than the inputs, resulting in a significant reduction in computational cost and required data for inference. We present an approximate inference scheme using Expectation Propagation. This variational approximation yields convenient GP hyperparameter selection and compact approximate predictive distributions.

1. Introduction

The Gaussian process (Rasmussen & Williams, 2006) is a useful and popular prior for nonlinear regression. It can be used to construct a distribution over scalar functions via a prior on smoothness. This prior is specified through a positive-definite kernel, which determines the covariance between two outputs as a function of their corresponding inputs. Often, this covariance function is taken to be stationary, i.e., a function only of the distance between the input points. Stationary covariance functions are appealing due to their intuitive interpretation and their relative ease of construction via Bochner's Theorem (Gibbs, 1997).

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Unfortunately, stationarity is often an unrealistic assumption. We expect many problems of interest to have nontrivial nonstationarity in the form of input-dependent noise, length scale or amplitude. While input-dependent noise and length-scale have been well-studied in the literature, nonstationarity in the form of varying amplitude has received relatively little attention.

One approach to modeling such data is to directly specify a covariance function with nonstationary properties (Gibbs, 1997; Higdon et al., 1999). In machine learning, however, we find it undesirable to need to specify the covariance nonstationarity *a priori*; rather we wish to infer it. Moreover, as the objective with Gaussian process regression is to perform nonparametric inference, we would prefer a representation of the nonstationarity which is also nonparametric.

Several approaches have been proposed to solve the problem of learning a length scale that varies across the input space. One of the first techniques was that of Sampson and Guttorp (1992), who model a spline-based mapping to a latent input space in which the data are stationary. This approach was given a nonparametric Bayesian treatment by Schmidt and O'Hagan (2003). Recently, Paciorek and Schervish (2004) extended the work of Higdon et al. (1999) to learn nonparametric variation of the covariance kernel. Other approaches involve Gaussian process mixtures (Rasmussen, 2000), augmentation of the input space (Pfingsten et al., 2006), and weighted sums of locally-stationary processes (Nott & Dunsmuir, 2002).

A related problem is input-dependent observation noise in the Gaussian process, addressed by Goldberg et al. (1998), who model a log-noise term in the covariance function with another Gaussian process, and by Le et al. (2005) who model nonstationary noise by performing regression in the natural parameter space of the exponential family. Snelson and Ghahramani (2006) achieve nonstationary noise as a side effect of the combination of input dimensionality reduction and a sparse approximation using pseudo-data.

In this paper, we propose the Gaussian process product model (GPPM) to address smooth input-dependent changes in amplitude. The GPPM models the data as the pointwise product of two latent stationary Gaussian processes. This approach has the notable computational advantage over remappings of the input space in that high dimensional problems pose no intrinsic scalability problems. Remapping the input nonparametrically while maintaining the input dimension requires at least as many latent processes as input dimensions. In contrast, the GPPM uses only a single additional GP regardless of input dimension. We develop a quadrature-based Expectation Propagation (EP) algorithm for efficient approximate inference in the GPPM model. The EP approach allows us to use the estimated marginal likelihood of the model to learn empirical settings of the Gaussian process hyperparameters. The approximate inference procedure we describe yields uncertainty in the nonstationarity, while avoiding expensive MCMC methods that are typically required. We additionally develop useful approximations for the predictive distribution arising from the EP approximation, and discuss rapid learning of a MAP estimate of the nonstationarity when observations can be considered noise free. This model is similar to that presented by Turner and Sahani (2008), who modulate sounds with Gaussian processes, however the GPPM is intended for the general regression problem and our inference approach differs significantly.

2. Gaussian Process Regression

In Gaussian process regression, we find a distribution over functions of the form $f : \mathcal{X} \to \mathbb{R}$, $\mathcal{X} = \mathbb{R}^m$. For a comprehensive introduction see Rasmussen and Williams (2006). The data consist of N input/output pairs $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}^N$, $\boldsymbol{x}_n \in \mathcal{X}$, $y_n \in \mathbb{R}$. A vector of output points has a Gaussian prior distribution with a mean function $\mu(\boldsymbol{x})$, which we take to be zero, and a positive-definite covariance function $C(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$. This construction gives an analytic Gaussian predictive distribution for an unseen output $y_\star \sim \mathcal{N}(\mu_\star, v_\star)$:

$$\mu_\star = \boldsymbol{k}_N^\mathsf{T} \boldsymbol{C}_N^{-1} \boldsymbol{y}_N, \quad v_\star = C(\boldsymbol{x}_\star, \boldsymbol{x}_\star) - \boldsymbol{k}_N^\mathsf{T} \boldsymbol{C}_N^{-1} \boldsymbol{k}_N,$$

where $\boldsymbol{k}_N = [C(\boldsymbol{x}_\star, \boldsymbol{x}_1; \boldsymbol{\theta}), \ldots, C(\boldsymbol{x}_\star, \boldsymbol{x}_N; \boldsymbol{\theta})]^\mathsf{T}$, and $\boldsymbol{C}_N$ is the covariance matrix formed from the observed data. The log evidence, or log marginal likelihood after integrating out all possible functions is

$$\mathcal{L} = -\frac{1}{2} \ln |\boldsymbol{C}_N| - \frac{1}{2} \boldsymbol{y}_N^\mathsf{T} \boldsymbol{C}_N^{-1} \boldsymbol{y}_N - \frac{N}{2} \ln 2\pi. \qquad (1)$$

Stationary covariance functions only depend on a distance measure d between $\boldsymbol{x}$ and $\boldsymbol{x}'$, for example the Mahalanobis distance $d(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{x}')^\mathsf{T} \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{x}')$ with positive definite $\boldsymbol{W}$. Covariance functions that depend only on distance are appealing due to the intuition that the outputs of the function should covary in inverse proportion to how far the inputs are from each other. The model proposed in this paper attempts to retain this intuition while providing a mechanism for the relationship between distance and covariance to vary across the input space.

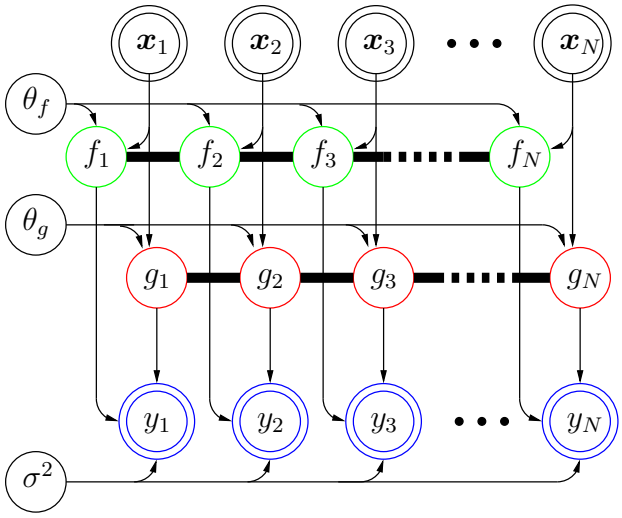


Figure 1. A graphical model describing the GPPM. The thick lines connecting the values of $\boldsymbol{f}$ and $\boldsymbol{g}$ represent undirected connections associated with the Gaussian process. The double-lined circles around the $\boldsymbol{y}$ values represent observables. Both $f(x)$ and $g(x)$ have the same input space.

3. The Gaussian Process Product Model

In the Gaussian process product model (GPPM), the observed outputs $\{y_n\}^N$ are modeled by a pointwise product of two latent functions, plus independent zero-mean Gaussian noise with variance σ^2. One latent function $f : \mathcal{X} \to \mathbb{R}$, is modulated by the other function $g : \mathcal{X} \to \mathbb{R}$ that has been exponentiated, so that

$$y_n \sim \mathcal{N}(f(\boldsymbol{x}_n) e^{g(\boldsymbol{x}_n)}, \, \sigma^2). \qquad (2)$$

We place independent zero-mean Gaussian process priors on $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$, with covariance functions $C_f(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_f)$ and $C_g(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_g)$, respectively. Figure 1 shows a graphical interpretation of this model. Our convention is that $f(\boldsymbol{x})$ captures local near-stationary variations in the observed function and $g(\boldsymbol{x})$ captures slowly-varying amplitude nonstationarity. The length-scale hyperparameters of these covariance functions (and their hyperpriors) should be chosen to reflect prior beliefs about such variations. To give the flavor of this model, Figure 2 shows several samples from the GPPM.

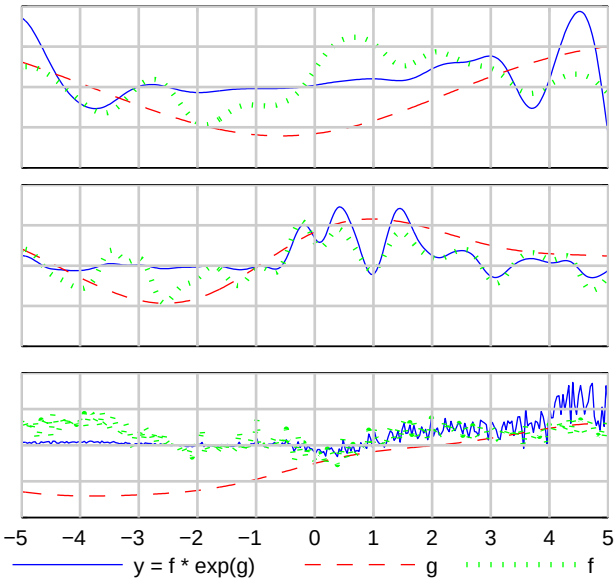


Figure 2. Three samples from the GPPM with different parameters. In the top plot, the length scales are $l_f = 0.5$ and $l_g = 4.0$. In the middle plot, both are shorter: $l_f = 0.25$ and $l_g = 2.0$. In the bottom plot, $l_f = 0.5$ and $l_g = 2.0$, but $f(x)$ also has additive noise.

Note that the pointwise product of a Gaussian process prior with any known function $a(\boldsymbol{x})$ results in a covariance function given by $C'(\boldsymbol{x}, \boldsymbol{x}') = a(\boldsymbol{x})C(\boldsymbol{x}, \boldsymbol{x}')a(\boldsymbol{x}')$ and that this function is guaranteed to be positive definite. In the GPPM we use an exponentiated form $a(\boldsymbol{x}) = \exp\{g(\boldsymbol{x})\}$ in order to reduce the multimodality of the posterior on the latent functions, but this is not critical for the validity of the covariance function. Without restricting the sign of one of the functions, there would be at least 2^N posterior modes, as each observation could be explained by the same latent function values with flipped signs.

4. Factor Inference in the GPPM

The basic GPPM inference task is to determine the posterior distribution over the values of the latent functions $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ at the input locations $\{\boldsymbol{x}_n\}^N$. These latent function values will be denoted $f_n = f(\boldsymbol{x}_n)$ and $g_n = g(\boldsymbol{x}_n)$ for brevity. Additionally we will write the vectors of these latent values in bold type: $\boldsymbol{f} = [f_1, \ldots, f_N]^\mathsf{T}$ and $\boldsymbol{g} = [g_1, \ldots, g_N]^\mathsf{T}$. With this notation and with $\boldsymbol{C}_f$ and $\boldsymbol{C}_g$ representing the GP-derived covariance matrices on $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ respectively, the posterior distribution of the latent functions is

$$p(\boldsymbol{f}, \boldsymbol{g} \,|\, \mathcal{D}, \boldsymbol{\theta}) \propto \mathcal{N}(\boldsymbol{f};\, \boldsymbol{0}, \boldsymbol{C}_f)\mathcal{N}(\boldsymbol{g};\, \boldsymbol{0}, \boldsymbol{C}_g) \times \prod_{n=1}^{N} \mathcal{N}(y_n;\, f_n e^{g_n},\, \sigma^2). \quad (3)$$

4.1. Approximate Inference

Approximate inference via variational methods is appealing due to its determinism and potential computational savings. In the GPPM, several properties affect our choice of approximation. First, we expect that the posterior will be approximately Gaussian, as we have strong Gaussian process priors and a near-Gaussian likelihood. Second, the likelihood factorizes to N independent terms, each involving one point from the two latent functions. Third, these likelihood factors introduce nontrivial dependencies between $\boldsymbol{f}$ and $\boldsymbol{g}$ so that a factorized approximation is inappropriate. We address these properties using Expectation Propagation.

4.1.1. Expectation Propagation

Expectation Propagation (Minka, 2001) makes successive *local* approximations of factors in a joint density, typically using exponential-family distributions, to yield a *global* approximation that is optimal under a divergence measure. EP is particularly well-suited for approximation of Bayesian posterior distributions with i.i.d. data as in Equation 3, as each factor only involves a few of the unknown parameters.

Our construction of the EP approximation is similar to that used by Rasmussen and Williams (2006) for binary Gaussian process classification. The prior on $\boldsymbol{f}$ and $\boldsymbol{g}$ is Gaussian with zero mean and a block covariance matrix arising from the independent Gaussian process priors. For notational convenience, we will write $\boldsymbol{\phi}$ to be the concatenation of $\boldsymbol{f}$ and $\boldsymbol{g}$ so that $\boldsymbol{\phi} = [f_1, \ldots, f_N, g_1, \ldots, g_N]^\mathsf{T}$, and $\boldsymbol{\phi}_n$ to be the nth pair $[f_n\ g_n]^\mathsf{T}$. The prior can now be written

$$p(\boldsymbol{\phi}) = \mathcal{N}(0, \boldsymbol{\Sigma}_{\mathsf{GP}}), \qquad \boldsymbol{\Sigma}_{\mathsf{GP}} = \begin{bmatrix} \boldsymbol{C}_f & 0 \\ 0 & \boldsymbol{C}_g \end{bmatrix}.$$

The aim of EP is to approximate the exact posterior distribution of Equation 3 with a tractable alternative

$$q(\boldsymbol{f}, \boldsymbol{g} \,|\, \mathcal{D}, \boldsymbol{\theta}) \propto \mathcal{N}(0, \boldsymbol{\Sigma}_{\mathsf{GP}}) \prod_{n=1}^{N} \tilde{t}_n(f_n, g_n). \quad (4)$$

Each of the exact likelihood terms

$$\mathcal{L}_n(f_n, g_n) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2\sigma^2}\left(f_n e^{g_n} - y_n\right)^2\right\}$$

is approximated with an unnormalized bivariate Gaussian on f_n and g_n:

$$\tilde{t}_n(f_n, g_n) = \tilde{Z}_n \exp\left\{-\frac{1}{2}(\boldsymbol{\phi}_n - \tilde{\boldsymbol{\mu}}_n)^\mathsf{T}\tilde{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\phi}_n - \tilde{\boldsymbol{\mu}}_n)\right\}.$$

The product of these likelihood approximations is an unnormalized Gaussian with a block-diagonal covariance matrix.

$$\prod_{n=1}^{N} \tilde{t}_n(\boldsymbol{\phi}_n) = \exp\left\{-\frac{1}{2}(\boldsymbol{\phi} - \tilde{\boldsymbol{\mu}})^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\phi} - \tilde{\boldsymbol{\mu}})\right\} \prod_{n=1}^{N} \tilde{Z}_n$$

The overall approximation is Gaussian as well, as it is the product of these Gaussian likelihood approximations and the Gaussian process prior.

$$q(\boldsymbol{f}, \boldsymbol{g} \mid \mathcal{D}, \boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{\phi} = \begin{bmatrix}\boldsymbol{f}\\ \boldsymbol{g}\end{bmatrix}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \tag{5}$$
$$\boldsymbol{\Sigma} = \left(\boldsymbol{\Sigma}_{\mathsf{GP}}^{-1} + \tilde{\boldsymbol{\Sigma}}^{-1}\right)^{-1} \qquad \boldsymbol{\mu} = \boldsymbol{\Sigma}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}}$$

The Expectation Propagation algorithm proceeds by iteratively updating the parameters of the local approximations t_n, leaving all other approximate factors fixed. In this iterative procedure the update of the nth site can be understood as the minimization of the KL divergence between two approximating distributions: the product of the cavity distribution times the *exact* local likelihood, and the product of the cavity distribution times the *approximate* local likelihood. The insight of EP is that the cavity distribution "focuses" the approximation on the most relevant area.

$$\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n = \underset{\boldsymbol{\mu}', \boldsymbol{\Sigma}'}{\operatorname{argmin}} \operatorname{KL}\Big[\mathcal{N}(\boldsymbol{\mu}_{/n}, \boldsymbol{\Sigma}_{/n}) \times \overbrace{\mathcal{L}_n(f_n, g_n)}^{\text{exact factor}} \,\Big|\Big|\, \mathcal{N}(\boldsymbol{\mu}_{/n}, \boldsymbol{\Sigma}_{/n}) \times \underbrace{\tilde{t}_n(f_n, g_n | \boldsymbol{\mu}', \boldsymbol{\Sigma}')}_{\text{approximation}}\Big]$$

The cavity distribution for site n is the product of the prior and all approximate sites excluding the nth. This is Gaussian with parameters

$$\boldsymbol{\Sigma}_{/n} = \left(\boldsymbol{\Sigma}_n^{-1} - \tilde{\boldsymbol{\Sigma}}_n^{-1}\right)^{-1} \tag{6}$$
$$\boldsymbol{\mu}_{/n} = \boldsymbol{\Sigma}_{/n}\left(\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\mu}_n - \tilde{\boldsymbol{\Sigma}}_n^{-1}\tilde{\boldsymbol{\mu}}_n\right). \tag{7}$$

As shown by Minka (2001), the minimum of an inclusive KL divergence is achieved when the moments are equal. Thus to find the best-fitting Gaussian, it is sufficient to find the first and second moments of the product of the cavity distribution and the exact likelihood. We also find the "zeroth moment," which is the normalization constant $\hat{Z}_n$. Calculation of these moments is done numerically via Gaussian quadrature, addressed in Section 4.1.2.

Once the moments of the product have been found, we use them to recover the optimal parameters of the local approximation:

$$\begin{aligned}
\tilde{\boldsymbol{\Sigma}}_n &= \left(\hat{\boldsymbol{\Sigma}}_n^{-1} - \boldsymbol{\Sigma}_{/n}^{-1}\right)^{-1} \\
\tilde{\boldsymbol{\mu}}_n &= \tilde{\boldsymbol{\Sigma}}_n\left(\hat{\boldsymbol{\Sigma}}_n^{-1}\hat{\boldsymbol{\mu}}_n - \boldsymbol{\Sigma}_{/n}^{-1}\boldsymbol{\mu}_{/n}\right) \\
\ln \tilde{Z}_n &= \ln \hat{Z}_n - \frac{1}{2}\ln|\tilde{\boldsymbol{\Sigma}}_n| + \frac{1}{2}\ln|\boldsymbol{\Sigma}_{/n}| + \frac{1}{2}\tilde{\boldsymbol{\mu}}_n^{\mathsf{T}}\tilde{\boldsymbol{\Sigma}}_n^{-1}\tilde{\boldsymbol{\mu}}_n \\
&\quad + \frac{1}{2}\boldsymbol{\mu}_{/n}^{\mathsf{T}}\boldsymbol{\Sigma}_{/n}^{-1}\boldsymbol{\mu}_{/n} - \frac{1}{2}\hat{\boldsymbol{\mu}}_n^{\mathsf{T}}\hat{\boldsymbol{\Sigma}}_n^{-1}\hat{\boldsymbol{\mu}}_n.
\end{aligned}$$

Taken together these equations define a fixed-point iteration scheme for approximating the posterior in Equation 3. We initialize the approximations so that the initial estimate of the mean of $\boldsymbol{f}$ is $\boldsymbol{y}$ and the mean of $\boldsymbol{g}$ is zero. We then iterate over each of the N local approximations, and update the overall posterior approximation using Equation 5. To facilitate convergence of EP it is helpful to use damping to update local sites, which we implement in natural parameter space. Convergence of EP is not guaranteed, but given sufficient damping it is found to convergence for the problems we considered so far. Local approximations may not necessarily be positive definite, but as long as the overall approximation remains a valid Gaussian, this does not present a problem. Following from the treatment by Minka (2001) of negative variances, we skip the update of local approximations that would result in invalid global covariance matrices. This has not appeared to affect the accuracy of the global approximation in practice. Figure 3(b) shows the result of applying the EP procedure to a synthetic data set. Marginal error bars are shown for each function and site location.

4.1.2. Gaussian Quadrature for EP

Unfortunately, the moments that minimize the KL divergence of Section 4.1.1 are not available analytically. To resolve this, we use the approach proposed by Zoeter and Heskes (2005) of approximating the moment integrals using Gaussian quadrature. When a definite integral is the product of a nonnegative "weighting function" $w(v)$ and another function $z(v)$, it can be approximated by a sum of weighted evaluations of $z(v)$

$$\int_b^a dv\; w(v)z(v) \approx \sum_{k=1}^{K} w_k z(v_k)$$

where the weights $\{w_k\}$ and abscissae $\{v_k\}$ are determined by the integration interval, the weighting function $w(v)$, and the number of evaluation points K. This sum is exact where $z(v)$ is a polynomial of degree $2K-1$. In the case of interest here, the weighting function is the Gaussian cavity distribution, which implies Gauss-Hermite quadrature.

One difficulty is that Gaussian quadrature is generally oriented towards univariate definite integrals and we must solve a two-dimensional integral. When the weighting function is factorizable, this is done straightforwardly by defining a lattice of abscissae and using the Cartesian product of the weights. In the GPPM, however, the cavity distribution has nonzero mean and is not generally factorizable, so we must transform

the integrand prior to to performing Gauss-Hermite quadrature. The factorizable form can be recovered by transforming the abscissae with the inverse Cholesky decomposition of the cavity covariance matrix and the cavity mean. The Gaussian parameters resulting from these moment calculations are denoted $\hat{\boldsymbol{Z}}_n$, $\hat{\boldsymbol{\mu}}_n$, and $\hat{\boldsymbol{\Sigma}}_n$ in Section 4.1.1.

4.2. Noise-free MAP Learning

In some applications of the GPPM, it may be that the observations can be considered noise-free. For example, one may model the noise as coming exclusively from the locally-varying function $f(x)$. The appeal of this restricted model is that proposals of the non-stationarity can now be evaluated as $O(N^2)$ rather than $O(N^3)$. This is particularly valuable for finding rapid maximum *a posteriori* (MAP) estimates of the latent modulating function $g(x)$. The computational advantage in the noise-free case comes from the deterministic coupling of the latent functions, given $\boldsymbol{y}$; we can now consider the posterior of $\boldsymbol{g}$ alone:

$$p(\boldsymbol{g} \,|\, \boldsymbol{\theta}_f, \boldsymbol{\theta}_g) \propto p(\mathcal{D} \,|\, \boldsymbol{g}, \boldsymbol{\theta}_f) p(\boldsymbol{g} \,|\, \boldsymbol{\theta}_g). \tag{8}$$

In this form, conditioning on $\boldsymbol{g}$ corresponds to a simple linear transformation of the GP prior on $\boldsymbol{f}$. Using the notational shortcut $\boldsymbol{\mathcal{G}} = \mathrm{diag}([e^{g_1}, e^{g_2}, \ldots, e^{g_N}])$, the log likelihood is

$$\ln p(\mathcal{D} \,|\, \boldsymbol{g}, \boldsymbol{\theta}_f) = -\frac{1}{2} \ln |\boldsymbol{\mathcal{G}} \boldsymbol{C}_f \boldsymbol{\mathcal{G}}| - \frac{1}{2} \boldsymbol{y}^\mathsf{T} [\boldsymbol{\mathcal{G}} \boldsymbol{C}_f \boldsymbol{\mathcal{G}}]^{-1} \boldsymbol{y} - \frac{N}{2} \ln 2\pi.$$

The log posterior over $\boldsymbol{g}$, eliminating irrelevant terms and using $\mathbf{1}$ to indicate a column vector of ones, is

$$\ln p(\boldsymbol{g} \,|\, \mathcal{D}, \boldsymbol{\theta}_f, \boldsymbol{\theta}_g) = -\boldsymbol{g}^\mathsf{T} \mathbf{1} - \frac{1}{2} \boldsymbol{y}^\mathsf{T} [\boldsymbol{\mathcal{G}} \boldsymbol{C}_f \boldsymbol{\mathcal{G}}]^{-1} \boldsymbol{y} - \frac{1}{2} \boldsymbol{g}^\mathsf{T} \boldsymbol{C}_g^{-1} \boldsymbol{g} + \text{const}$$

and the gradient in terms of $\boldsymbol{g}$ is

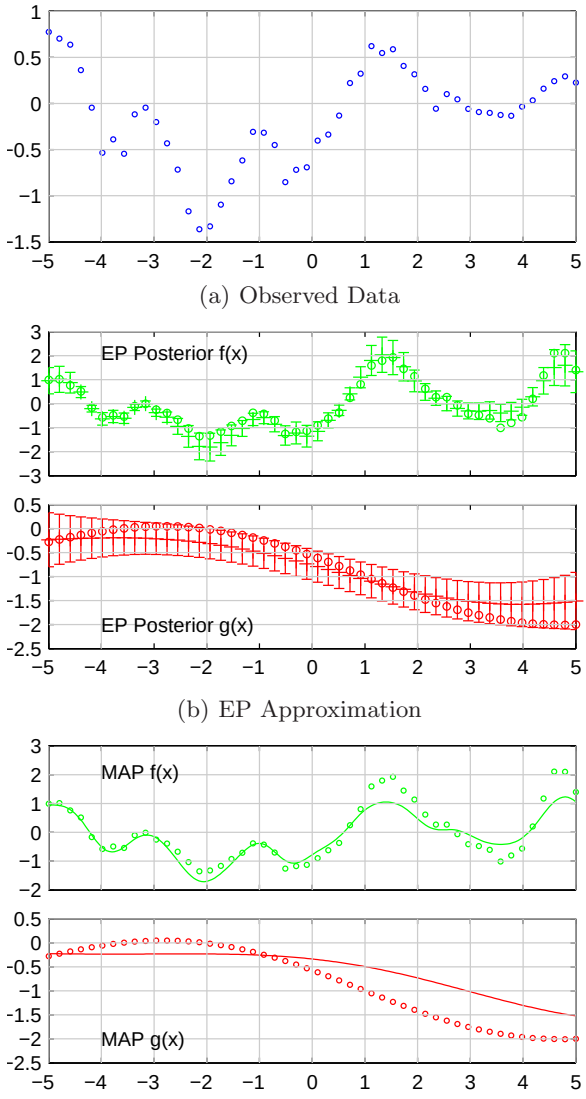
$$\frac{\partial}{\partial \boldsymbol{g}} \ln p(\boldsymbol{g} \,|\, \mathcal{D}, \boldsymbol{\theta}_f, \boldsymbol{\theta}_g) = -\mathbf{1} + \boldsymbol{Y} [\boldsymbol{\mathcal{G}} \boldsymbol{C}_f \boldsymbol{\mathcal{G}}]^{-1} \boldsymbol{y} - \boldsymbol{C}_g^{-1} \boldsymbol{g}$$

where $\boldsymbol{Y} = \mathrm{diag}(\boldsymbol{y})$. As the difficult $O(N^3)$ operations of decomposition or inversion of $\boldsymbol{C}_f$ and $\boldsymbol{C}_g$ can be done in advance, the computational complexity of taking a step in $\boldsymbol{g}$ space is $O(N^2)$. In practice, we have found the MAP estimate to be best when $f(x)$ has additive noise and $g(x)$ is smooth.

5. Making Predictions

As with the standard regression model, the primary task of interest is prediction at locations where data have not been observed. For the GPPM we must make predictions for both latent functions, and find the resulting distribution, integrating out the posterior distribution over the latent functions, as in

$$p(y^\star \,|\, \boldsymbol{x}^\star, \mathcal{D}, \boldsymbol{\theta}_f, \boldsymbol{\theta}_g) = \int_{\boldsymbol{f}, \boldsymbol{g}} p(y^\star \,|\, \boldsymbol{x}^\star, \boldsymbol{f}, \boldsymbol{g}) p(\boldsymbol{f}, \boldsymbol{g} \,|\, \mathcal{D}, \boldsymbol{\theta}_f, \boldsymbol{\theta}_g).$$



(a) Observed Data

(b) EP Approximation

(c) MAP Approximation

Figure 3. Figure 3(a) shows synthetic data generated from the GPPM with known settings and $\sigma = 0.05$. We applied the Expectation Propagation algorithm to the data and the Gaussian marginal posterior distributions over $\boldsymbol{f}$ and $\boldsymbol{g}$ are shown in Figure 3(b), along with the true $f(x)$ and $g(x)$ indicated as circles. Figure 3(c) shows the result of applying the MAP approximation to the data, despite the known observation noise. The true values are shown for comparison.

The EP scheme of Section 4.1 finds an approximate Gaussian distribution over $\boldsymbol{f}$ and $\boldsymbol{g}$, and this results in a convenient joint Gaussian distribution on $f^\star$ and $g^\star$, the values of the latent functions at $\boldsymbol{x}^\star$, with parameters

$$\boldsymbol{\mu}^\star = \boldsymbol{K}^\mathsf{T}\left(\boldsymbol{\Sigma}_{\mathsf{GP}} + \tilde{\boldsymbol{\Sigma}}\right)^{-1}\tilde{\boldsymbol{\mu}}, \quad \boldsymbol{\Sigma}^\star = \boldsymbol{\kappa} - \boldsymbol{K}^\mathsf{T}\left(\boldsymbol{\Sigma}_{\mathsf{GP}} + \tilde{\boldsymbol{\Sigma}}\right)^{-1}\boldsymbol{K},$$

where

$$\boldsymbol{K} = \begin{bmatrix} C(\boldsymbol{x}^\star, \boldsymbol{x}_1; \boldsymbol{\theta}_f) & 0 \\ C(\boldsymbol{x}^\star, \boldsymbol{x}_2; \boldsymbol{\theta}_f) & 0 \\ \vdots & \vdots \\ C(\boldsymbol{x}^\star, \boldsymbol{x}_N; \boldsymbol{\theta}_f) & 0 \\ 0 & C(\boldsymbol{x}^\star, \boldsymbol{x}_1; \boldsymbol{\theta}_g) \\ \vdots & \vdots \\ 0 & C(\boldsymbol{x}^\star, \boldsymbol{x}_N; \boldsymbol{\theta}_g) \end{bmatrix}$$

$$\boldsymbol{\kappa} = \begin{bmatrix} C(\boldsymbol{x}^\star, \boldsymbol{x}^\star; \boldsymbol{\theta}_f) & 0 \\ 0 & C(\boldsymbol{x}^\star, \boldsymbol{x}^\star; \boldsymbol{\theta}_g) \end{bmatrix}.$$

We expect that the resulting predictive distribution on $y^\star$ will be heavy-tailed and have similar properties to the noncentral Student's t distribution. To approximate the true distribution's heavy tails analytically, one approach is to generate several samples from $g^\star$ and use the conditional distribution on $f^\star$ to create a mixture of Gaussians:

$$p(y^\star \mid \boldsymbol{x}^\star, \mathcal{D}, \boldsymbol{\theta}_f, \boldsymbol{\theta}_g) \approx \sum_i \mathcal{N}(y^\star; \mu^\star_{f|g_i} e^{g^\star_i}, v^\star_{f|g_i} e^{2g^\star_i}).$$

We have used $\mu^\star_{f|g_i}$ and $v^\star_{f|g_i}$ to indicate the conditional Gaussian parameters on $f^\star$ given the ith marginal sample from $g^\star$.

If the heavy-tailed properties are not significant for the application, and a single Gaussian distribution is preferred, then a more tractable alternative is to linearize the model around the mean $\boldsymbol{\mu}^\star$. This is a similar approach to the Extended Kalman Filter (EKF) (Haykin, 2001), which uses the first terms of the Taylor series of a nonlinear function to maintain Gaussian uncertainty in latent state estimation. The resulting approximation is

$$f^\star e^{g^\star} \underset{\boldsymbol{\mu}^\star}{\approx} \mu^\star_f e^{\mu^\star_g} + \begin{bmatrix} e^{\mu^\star_g} \\ \mu^\star_f e^{\mu^\star_g} \end{bmatrix}^\mathsf{T} \begin{bmatrix} f^\star - \mu^\star_f \\ g^\star - \mu^\star_g \end{bmatrix}$$

which transforms the Gaussian on $f^\star$ and $g^\star$ into one on $y^\star$ with parameters

$$\mu^\star_y = \mu^\star_f e^{\mu^\star_g} \qquad v^\star_y = \begin{bmatrix} e^{\mu^\star_g} \\ \mu^\star_f e^{\mu^\star_g} \end{bmatrix}^\mathsf{T} \boldsymbol{\Sigma}^\star \begin{bmatrix} e^{\mu^\star_g} \\ \mu^\star_f e^{\mu^\star_g} \end{bmatrix} + \sigma^2.$$

6. Hyperparameter Learning

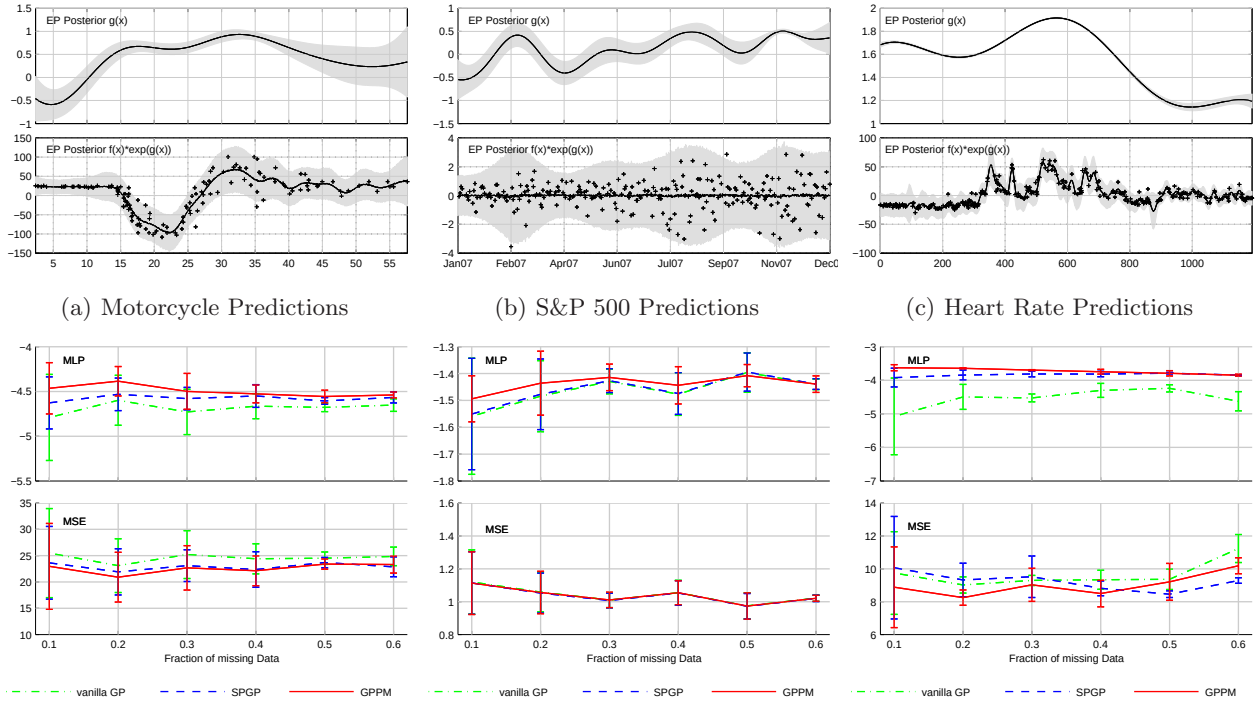
When performing Gaussian process regression, we are commonly interested in appropriate settings of the hyperparameters controlling the covariance function. These hyperparameters generally determine the length scale of correlations, the output variation (or amplitude) of the function, and the noise level. In the GPPM, we wish to find appropriate hyperparameter settings for *both* latent functions, given the data. While the vanilla Gaussian process offers the marginal likelihood analytically, it is not available directly in the GPPM. Fortunately, the EP algorithm of Section 4.1 provides a convenient estimate of the marginal likelihood, using the zeroth moments mentioned previously.

$$\ln Z_{\mathsf{EP}} = \frac{1}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}\ln|\boldsymbol{\Sigma}_{\mathsf{GP}}| - \frac{1}{2}\tilde{\boldsymbol{\mu}}^\mathsf{T}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}} + \frac{1}{2}\boldsymbol{\mu}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \sum_{n=1}^{N}\ln\tilde{Z}_n$$

In principle it is also possible to evaluate the gradients of $\ln Z_{\mathsf{EP}}$ with respect to hyperparameters following for instance (Seeger, 2005). In practice however, the quadrature-based moment calculation is numerically not stable enough to provide precise gradients. We hence reverted to gradient-free optimization methods to determine hyper parameter settings. We suggest setting hyperpriors to reflect the intuition described in Section 3 of $f(x)$ capturing local near-stationary variations and $g(x)$ capturing slowly varying nonstationarity on a larger lengthscale.

7. Results

We evaluated the GPPM model on three data sets. First, we examined the motorcycle data set (Parker & Rice, 1985), a well-studied example of a nonstationary regression task. The data are acceleration force in g's on a helmet during impact, as a function of time in milliseconds. Figure 4(a) in the upper plot shows the EP approximation found for the latent $g(x)$ function, and in the lower plot shows the Gaussian approximation to the predictive distribution, overlaid with the true data. The GPPM finds a good fit in most regions except where the $g(x)$ function becomes quite small. In these regions the uncertainty in the modulating function creates unrealistically large prediction error bars. We evaluated the accuracy of predictions using a fill-in test, where a fraction of the data are removed from the training set and compared to the model's predictions. Figure 4(d) depicts the mean log probability and the mean squared error for missing data as a function of the fraction of missing data. The GPPM outperforms both a vanilla GP and the sparse pseudo-input process (SPGP) (Snelson & Ghahramani, 2006) using either of the performance measures. We chose the SPGP for comparison to the GPPM, as it is one of the few



(a) Motorcycle Predictions (b) S&P 500 Predictions (c) Heart Rate Predictions

(d) Motorcycle Fill–in Performance (e) S&P Fill–in Performance (f) Heart Rate Fill–in Performance

Figure 4. Top panel: Predictive distribution of GPPM for three different data sets. The upper plot shows the EP approximation to the posterior of the log-modulating function $g(x)$ with 2σ error bars. The lower plot shows the raw data, along with the 2σ approximate predictive distribution. Lower panel: Fill-in test for corresponding data sets comparing three models. The upper plot shows the mean log probability of the missing data as a function of the fill-in rate. The lower plot shows the root mean squared error for these data. Both plots show mean values and 2σ error bars, calculated from four training/test splits.

methods capable of representing nonstationarity without requiring MCMC. Hyperparameters for the SPGP and the vanilla GP were set via ML-II optimization (Rasmussen & Williams, 2006). To set hyperparameters in the GPPM, a grid search was used, centered on the settings for the vanilla GP.

We also examined the performance of the GPPM for daily log returns of the S&P 500 stock index during 2007. We expect that these data will be well-modeled by a latent $f(x)$ comprised primarily of noise. The log modulating function $g(x)$ can be interpreted roughly as the log "volatility" of the stochastic process and is shown in the upper plot of Figure 4(b). The corresponding expected envelope is shown against the true data in the lower plot. Performance measures against the standard Gaussian process and the SPGP are shown in Figure 4(e). In this example mean predictions are equally good for three all models, but GPPM yields nonstationary uncertainty which results in an improved mean log probability.

As a last application we applied the GPPM to 23 hours of heart rate data, sampled at 5 minute intervals. Based on the physiological properties of heart rates, we expect correlations on a short time scale to be captured by $f(x)$. These local correlations will be modulated by an activity profile over a daily time scale. Figure 4(c) illustrates that these amplitude modulations are picked up by the latent $g(x)$ leading to improved predictive performance compared to the vanilla GP and SPGP, as shown in Figure 4(f).

8. Discussion

We have introduced the Gaussian process product model for modeling nonstationary amplitude in regression. We have presented an approximate inference algorithm using Expectation Propagation to infer the latent functions in this model and have exploited this approximation to make tractable predictions and enable hyperparameter learning. When examined on real-world data, the GPPM has yielded promising results, outperforming the vanilla Gaussian process. It has also outperformed an alternative approach to nonstationary regression in the SPGP, although it should be noted that the SPGP's focus is purely on efficient regression and not on modeling nonstationarity *per se*.

Computationally, the model we have presented, combined with the EP implementation has two appealing properties. First, as we expect the number of EP iterations to be independent of the number of data (Minka, 2001), and each local calculation is a $O(N^2)$ rank-one update of the inverse, the overall algorithm is $O(N^3)$. The GPPM is therefore only a constant multiple more expensive than performing standard Gaussian process regression. Second, in contrast to methods of modeling nonstationarity on the input side, the GPPM does not introduce additional latent spaces if the input dimensionality increases. The additional computational complexity of using the GPPM is essentially independent of input dimension.

In future work, a more comprehensive examination of inference of hyperparameters is warranted. We also expect that the basic idea of this model can be used to perform vector regression with correlation that varies across the input space.

Acknowledgements

The authors wish to thank David MacKay for helpful comments. This work was funded by the Gates Cambridge Trust.

References

Gibbs, M. (1997). *Bayesian Gaussian processes for regression and classification*. Doctoral dissertation, University of Cambridge, Cambridge.

Goldberg, P., Williams, C., & Bishop, C. (1998). Regression with input-dependent noise: a Gaussian process treatment. *Advances in Neural Processing Systems 10* (pp. 493–499). Cambridge, MA: MIT Press.

Haykin, S. (Ed.). (2001). *Kalman filtering and neural networks*. New York: John Wiley and Sons, Inc.

Higdon, D., Swall, J., & Kern, J. (1999). Nonstationary spatial modeling. In J. Bernardo, J. Berger, A. Dawid and A. Smith (Eds.), *Bayesian statistics 6*, 761–768. Oxford: Oxford University Press.

Le, Q., Smola, A., & Canu, S. (2005). Heteroscedastic Gaussian process regression. *Proceedings of the 22nd International Conference on Machine Learning (ICML)*.

Minka, T. P. (2001). *A family of algorithms for approximate Bayesian inference*. Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Nott, D., & Dunsmuir, W. (2002). Estimation of nonstationary spatial covariance structure. *Biometrika*, *89*, 819–829.

Paciorek, C., & Schervish, M. (2004). Nonstationary covariance functions for Gaussian process regression. *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.

Parker, R., & Rice, J. (1985). Discussion of "Some aspects of the spline smoothing approach to nonparametric curve fitting" by B.W. Silverman. *Journal of the Royal Statistical Society, Series B*, *47*, 40–42.

Pfingsten, T., Kuss, M., & Rasmussen, C. (2006). Nonstationary Gaussian process regression using a latent extension of the input space. www.kyb.mpg.de/publication.html?publ=3985.

Rasmussen, C. (2000). The infinite Gaussian mixture model. *Advances in Neural Information Processing Systems 12* (pp. 554–560). Cambridge, MA: MIT Press.

Rasmussen, C., & Williams, C. (2006). *Gaussian processes for machine learning*. Cambridge, MA: MIT Press.

Sampson, P., & Guttorp, P. (1992). Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, *87*, 108–119.

Schmidt, A. M., & O'Hagan, A. (2003). Bayesian inference for nonstationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society, Series B*, *65*, 745–758.

Seeger, M. (2005). *Expectation propagation for exponential families* (Technical Report). Technical report, University of California at Berkeley, 2005. See www. kyb. tuebingen. mpg. de/bs/people/seeger.

Snelson, E., & Ghahramani, Z. (2006). Variable noise and dimensionality reduction for sparse Gaussian processes. *Proceedings of the 22nd International Conference on Uncertainty in Artificial Intelligence*.

Turner, R., & Sahani, M. (2008). Modeling natural sounds with modulation cascade processes. *Advances in Neural Information Processing Systems 21*. Cambridge, MA: MIT Press.

Zoeter, O., & Heskes, T. (2005). Gaussian quadrature based expectation propagation. *Proceedings of Artificial Intelligence and Statistics 2005*.

Sequence Kernels for Predicting Protein Essentiality

Cyril Allauzen ALLAUZEN@GOOGLE.COM
Google Research, New York, NY

Mehryar Mohri MOHRI@CS.NYU.EDU
Courant Institute of Mathematical Sciences and Google Research, New York, NY

Ameet Talwalkar AMEET@CS.NYU.EDU
Courant Institute of Mathematical Sciences, New York, NY

Abstract

The problem of identifying the minimal gene set required to sustain life is of crucial importance in understanding cellular mechanisms and designing therapeutic drugs. This work describes several kernel-based solutions for predicting essential genes that outperform existing models while using less training data. Our first solution is based on a semi-manually designed kernel derived from the Pfam database, which includes several Pfam domains. We then present novel and general *domain-based* sequence kernels that capture sequence similarity with respect to several domains made of large sets of protein sequences. We show how to deal with the large size of the problem – several thousands of domains with individual domains sometimes containing thousands of sequences – by representing and efficiently computing these kernels using automata. We report results of extensive experiments demonstrating that they compare favorably with the Pfam kernel in predicting protein essentiality, while requiring no manual tuning.

1. Motivation

Identifying the minimal gene set required to sustain life is of crucial importance for understanding the fundamental requirements for cellular life and for selecting therapeutic drug targets. Gene knockout studies and RNA interference are experimental techniques for identifying an organism's "essential" genes, or the set of genes whose removal is lethal to the organism. However, these techniques are expensive and time-consuming. Recent work has attempted to extract from experimental knockout studies relevant features of essentiality, which aid in identifying essential genes in organisms lacking experimental results.

Several features have been proposed as indicators for essentiality, including evolutionary conservation, protein size, and number of paralogs (Chen & Xu, 2005). Using these basic features, Chen and Xu (2005) constructed a model of essentiality for *S. cerevisiae* (baker's yeast). Using Naive Bayes Classifiers (NBC), Gustafson et al. (2006) subsequently created a model of essentiality for *S. cerevisiae* and *E. coli* using an extended set of features generated from sequence data.

This work presents kernel methods to improve upon existing models. We first use several sequence kernels recently introduced by the computational biology community and show that the Pfam kernel (Ben-Hur & Noble, 2005) is most effective in selecting essential genes for *S. cerevisiae*. The Pfam kernel has recently been applied successfully in several biologically motivated learning tasks, and is generated from the Pfam database, the leading resource for storing protein family classification and protein domain data. However, the Pfam database is an ad-hoc solution relying on semi-manually tuned information.

In the second part of this work, we design general sequence kernels that produce effective similarity measures while bypassing the manual tuning of the Pfam database. We present two sequence kernels that are instances of rational kernels, a class of sequence kernels defined by weighted automata that are effective for analyzing variable-length sequences (Cortes et al., 2004). Using automata to represent and compute these ker-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

nels is crucial in order to handle the large number of Pfam domains and the size of each of domain – we work with 6190 domains with the largest domain containing over 3000 protein sequences. These novel kernels are designed from the same domain-specific data used by the Pfam library, and we show how they compare favorably to the Pfam kernel at predicting protein essentiality. They are general *domain-based* kernels that can be used in many problems in bioinformatics or other applications where similarity needs to be defined in terms of proximity to several large sets of sequences.

The remainder of the paper is organized as follows. Section 2 describes the various sequence kernels and outlines the model used to improve prediction accuracy of protein essentiality in *S. cerevisiae*. Section 3 describes and analyzes the novel rational kernels we present as alternatives to the Pfam kernel. Section 4 presents the results of extensive experiments comparing these domain-based kernels to the Pfam kernel.

2. Pfam-Based Solution

Our first model uses Support Vector Machines (SVMs) (Cortes & Vapnik, 1995) to predict protein essentiality with choices of kernels including the RBF kernel as well as three sequence kernels. In the following subsections, we define the sequence kernels, outline the experimental design, and present our first results.

2.1. Sequence Kernels

Pfam Kernel

The Pfam database is a collection of multiple sequence alignments and Hidden Markov Models (HMMs) representing many common protein domains and families. Pfam version 10.0 contains 6190 domains, and for each domain an HMM is constructed from a set of proteins experimentally determined to be part of the domain ('seed' proteins). Each HMM is trained using a manually-tuned multiple alignment of the seed proteins with gaps inserted to normalize sequence length. Once constructed, the HMM is evaluated in an ad-hoc fashion and the entire process is repeated if the alignment is deemed 'unsatisfactory.' See (Sonnhammer et al., 1997) for further details.

When applied to a test sequence, a Pfam domain HMM generates an E-value statistic that measures the likelihood of the test sequence containing the domain. Given a dataset of protein sequences, the Pfam sequence kernel matrix is constructed by representing each protein in the dataset as a vector of 6190 log E-values and computing explicit dot products from these feature vectors (Ben-Hur & Noble, 2005). The

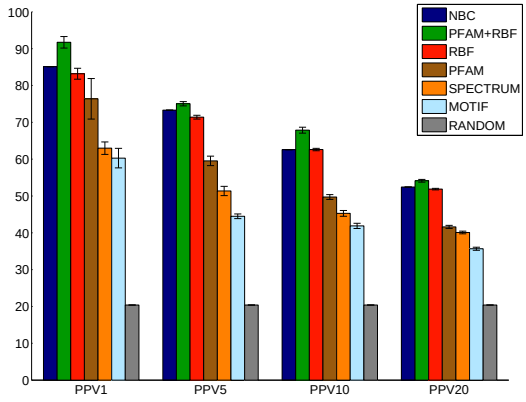


Figure 1. SVM's performance for RBF and Sequence Kernels using a reduced training set. Note that accuracy for NBC corresponds to a model trained on 50% of training data.

Pfam kernel has recently been applied successfully in learning tasks ranging from protein function (Lanckriet et al., 2004) to protein-protein interaction (Ben-Hur & Noble, 2005).

Spectrum and Motif Kernels

The Spectrum and Motif kernels are two recently proposed sequence kernels used in learning tasks to estimate protein similarity (Leslie & Kuang, 2004; Ben-Hur & Brutlag, 2003). Both kernels model a protein in a feature space of subsequences, with each feature measuring the extent to which the protein contains a specific subsequence. The Spectrum kernel models proteins in a feature space of all possible n-grams, representing each protein as a vector of n-gram counts (in our studies we set $n = 3$). Alternatively, the Motif kernel uses a feature space consisting of a set of discrete sequence motifs (we use a set of motifs extracted from the eMotif database (Ben-Hur & Noble, 2005)). For both kernels, the resulting kernel matrices are computed as an explicit dot product using these features.

2.2. Data

We used the dataset of *S. cerevisiae* proteins from Gustafson et al. (2006), consisting of 4728 yeast proteins of which 20.4% were essential. We constructed the RBF kernel matrix from a set of 16 features generated directly from protein sequences, corresponding to the 'easily attainable' features from Gustafson et al. (2006). We used data from Ben-Hur and Noble (2005) to construct the Pfam, Spectrum and Motif kernel matrices, each of which was constructed following the steps outlined in Section 2.1 and subsequently centered and normalized. In addition to the RBF and the three sequence kernels, we also used a combined Pfam/RBF

kernel, which we computed by additively combining the RBF kernel matrix with the normalized Pfam kernel matrix (RBF kernels are by definition normalized).

2.3. Experimental Design

We ran experiments with 100 trials. For each trial, we randomly chose 8.3% of the data as a training set and used the remaining points as a test set, subject to the constraint that an equal number of essential proteins were in each set.[1] We used the training set to train an SVM, and used the resulting model to make predictions on the test set in the form of probabilities of essentiality. We used libsvm's functionality (Chang & Lin, 2001) to estimate the outputs of an SVM as probabilities by fitting its results to a sigmoid function (Platt, 2000). To calculate the predicted probability of essentiality for each protein, we took the average over the predictions from each trial in which the protein appeared in the test set.

We measured the accuracy of the model for the proteins with the highest predicted probability of essentiality, using positive predictive value (PPV) as a performance indicator. Grid search was used to determine the optimal values for parameters C and gamma. Standard deviations were calculated from 10 'super-trials,' each corresponded to a 100-trial experiment described above. The Naive Bayes classifier (NBC) results were taken from Gustafson et al. (2006) and standard deviations were not reported.

2.4. First Results

Figure 1 shows SVM's performance using the set of kernels outlined above. The results show that the Pfam kernel is the most effective of the three sequence kernels at predicting essentiality. They also clearly show that the combined Pfam/RBF kernel outperforms all other models. The importance of the phyletic retention feature is a possible reason for the superior performance of the combined kernel compared with Pfam alone. As shown by Gustafson et al. (2006) and verified in our work, phyletic retention (a measure of gene conservation across species) is a powerful predictor of essentiality. This feature is used by RBF but not by Pfam (or by the domain-based kernels defined in Section 3) because it requires comparing sequences across organisms.

These results improve upon the leading model for prediction of protein essentiality while reducing the size of the training set more than five fold. Further, this is the first result showing the effectiveness of the Pfam kernel for this task, a fact that motivates the following sections of this paper, in which we seek a more general alternative to the Pfam kernel.

[1] Gustafson et al. (2006) used 50% of the data for training, but otherwise, our experimental designs are identical.

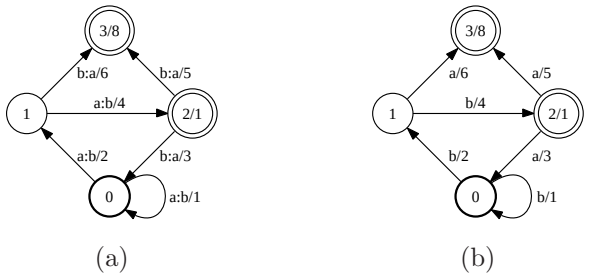


Figure 2. (a) Example of weighted transducer T. (b) Example of weighted automaton A. A can be obtained from T by projection on the output and $T(aab, bba) = A(bba) = 1 \times 2 \times 6 \times 8 + 2 \times 4 \times 5 \times 8$.

3. Domain-Based Sequence Kernels

In the previous section, we tested various sequence kernels, all introduced precisely to compute the similarity between protein sequences. Our results showed that the Pfam kernel was the most effective of these kernels, and we now aim to find a more general solution free of the manual tuning associated with the Pfam database.

Specifically, we wish to determine a method to extract similarities between protein sequences based on their similarities to several domains, each represented by a set of sequences, i.e., Pfam domains. Although several sequence kernels have been recently introduced in the machine learning community, e.g., mismatch, gappy, substitution and homology kernels (Leslie & Kuang, 2004; Eskin & Snir, 2005), none of these kernels provides a solution to our domain-based learning problem. Indeed, these kernels are not designed to efficiently compute the similarity between a string and a large set of strings, which in our case consists of 6190 Pfam domains each containing tens to thousands of sequences.

Alternatively, large sets of strings, such as the Pfam domains, can be efficiently represented by minimal deterministic automata. Hence, an efficient way to define a similarity measure between such sets is to use automata-based kernels. This leads us to consider the framework for automata-based kernels proposed by Cortes et al. (2004). An additional benefit of this framework is that most commonly used string kernels are special instances of this scheme.

3.1. Representation and Algorithms

We will follow the definitions and terminology given by Cortes et al. (2004). The representation and computation of the Domain-based kernels are based on *finite-state transducers*, which are finite automata in which each transition is augmented with an output label in addition to the familiar input label (Salomaa & Soittola, 1978). Input (output) labels are concatenated along a path to form an input (output) sequence. *Weighted transducers* are finite-state transducers in which each transition carries some weight in addition to the input and output labels. The weights of the transducers considered in this paper are real values.

Figure 2(a) shows an example of a weighted finite-state transducer. In this figure, the input and output labels of a transition are separated by a colon delimiter, and the weight is indicated after the slash separator. A weighted transducer has a set of initial states represented in the figure by a bold circle, and a set of final states, represented by double circles. A path from an initial state to a final state is an accepting path.

The weight of an accepting path is obtained by first multiplying the weights of its constituent transitions and multiplying this product by the weight of the initial state of the path (which equals one in our work) and the weight of the final state of the path (displayed after the slash in the figure). The weight associated by a weighted transducer T to a pair of strings $(x, y) \in \Sigma^* \times \Sigma^*$ is denoted by $T(x, y)$ and is obtained by summing the weights of all accepting paths with input label x and output label y. For example, the transducer of Figure 2(a) associates the weight 416 to the pair (aab, bba) since there are two accepting paths labeled with input aab and output bba: one with weight 96 and another one with weight 320.

The standard operations of sum $+$, product or concatenation $\cdot$, and Kleene-closure * can be defined for weighted transducers (Salomaa & Soittola, 1978). For any pair of strings (x, y),

$$\begin{aligned}(T_1 + T_2)(x, y) &= T_1(x, y) + T_2(x, y)\\ (T_1 \cdot T_2)(x, y) &= \sum_{\substack{x_1x_2=x\\ y_1y_2=y}} T_1(x_1, y_1) \cdot T_2(x_2, y_2). \end{aligned} \quad (1)$$

For any transducer T, T^{-1} denotes its *inverse*, that is the transducer obtained from T by swapping the input and output labels of each transition. For all $x, y \in \Sigma^*$, we have $T^{-1}(x, y) = T(y, x)$.

The *composition* of two weighted transducers T_1 and T_2 with matching input and output alphabets Σ, is a weighted transducer denoted by $T_1 \circ T_2$ when the sum:

$$(T_1 \circ T_2)(x, y) = \sum_{z \in \Sigma^*} T_1(x, z) \cdot T_2(z, y) \quad (2)$$

is well-defined and in $\mathbb{R}$ for all x, y (Salomaa & Soittola, 1978).

Weighted automata can be defined as weighted transducers A with identical input and output labels, for any transition. Since only pairs of the form (x, x) can have a non-zero weight associated to them by A, we denote the weight associated by A to (x, x) by $A(x)$ and call it the *weight associated by A to x*. Similarly, in the graph representation of weighted automata, the output (or input) label is omitted. Figure 2(b) shows an example of a weighted automaton. When A and B are weighted automata, $A \circ B$ is called the *intersection* of A and B. Omitting the input labels of a weighted transducer T results in a weighted automaton which is said to be the *output projection of T*.

3.2. Automata-Based Kernels

A string kernel $K : \Sigma^* \times \Sigma^* \to \mathbb{R}$ is *rational* if it coincides with the function defined by a weighted transducer U, that is for all $x, y \in \Sigma^*$, $K(x, y) = U(x, y)$. Not all rational kernels are *positive definite and symmetric* (PDS), or equivalently verify the Mercer condition, which is crucial for the convergence of training for discriminant classification algorithms such as SVMs. But, for any weighted transducer T, $U = T \circ T^{-1}$ is guaranteed to define a PDS kernel (Cortes et al., 2004).

Furthermore, most rational kernels used in computational biology and natural language processing are of this form (Haussler, 1999; Leslie & Kuang, 2004; Lodhi et al., 2002; Zien et al., 2000; Collins & Duffy, 2001; Cortes & Mohri, 2005). For instance, the n-gram kernel is a rational kernel. The n-gram kernel K_n is defined as

$$K_n(x, y) = \sum_{|z|=n} c_x(z)c_y(z), \quad (3)$$

where $c_x(z)$ is the number of occurrences of z in x. K_n is a PDS rational kernel since it corresponds to the weighted transducer $T_n \circ T_n^{-1}$ where the transducer T_n is defined such that $T_n(x, z) = c_x(z)$ for all $x, z \in \Sigma^*$ with $|z| = n$. The transducer T_2 for $\Sigma = \{a, b\}$ is shown in Figure 3.

We will now extend this formalism to measure the similarity between domains, or sets of strings represented by an automaton. Let us define the count of a string z in a weighted automaton A as:

$$c_A(z) = \sum_{u \in \Sigma^*} c_u(z)A(u). \quad (4)$$

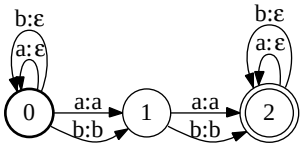


Figure 3. Counting transducer T_2 for $\Sigma = \{a, b\}$.

The similarity between two sets of strings represented by the weighted automata A and B according to n-gram kernel K_n can then be defined by:

$$\begin{aligned} K_n(A, B) &= \sum_{x,y \in \Sigma^*} (A \circ T_n \circ T_n^{-1} \circ B)(x, y) \\ &= \sum_{|z|=n} c_A(z) c_B(z). \end{aligned} \tag{5}$$

Other rational kernels can be extended into a similarity measure between sets of strings in the same way. We will now define two families of kernels that can be used in a variety of applications where similarity with respect to domains is needed.

3.3. Independent Domain Kernel

The Independent Domain kernel (IDK) measures the similarity between two sequences in our dataset $\mathcal{D}$ by comparing their similarities to each domain, e.g., Pfam domains.[2] For the i-th Pfam domain (with $1 \leq i \leq P = 6190$), let $\mathcal{P}_i$ be the set of all seed protein sequences for that domain. Each sequence in $\mathcal{P}_i$ is represented as a string in an alphabet, Σ, consisting of $|\Sigma| = 21$ characters, 20 for different amino acids plus an optional gap character used to represent gaps in the seed alignment. We denote by D_i the minimal deterministic automaton representing the set of strings $\mathcal{P}_i$. For a given sequence x in our dataset, we can use the n-gram kernel K_n to compute the similarity between x and the i-th Pfam domain $\mathcal{P}_i$: $K_n(x, D_i)$. This leads to an overall similarity measure vector $s(x) \in \mathbb{R}^P$ between x and the set of domains: $s(x) = (K_n(x, D_1), \ldots, K_n(x, D_P))$. We now define the IDK K_I as, for all x, y in Σ^*:

$$\begin{aligned} K_I(x, y) &= \sum_{i=1}^{P} K_n(x, D_i) K_n(y, D_i) \\ &= \sum_{i=1}^{P} \Big(\sum_{|z|=n} c_x(z) c_{D_i}(z) \Big) \Big(\sum_{|z|=n} c_y(z) c_{D_i}(z) \Big). \end{aligned} \tag{6}$$

K_I is PDS since it is constructed via an explicit dot-product. Any PDS kernel K with positive eigenvalues can be normalized to take values between 0 and 1 by defining K' as

$$K'(x, y) = \frac{K(x, y)}{\sqrt{K(x, x) K(y, y)}}. \tag{7}$$

We apply this normalization to K_I to account for the varying lengths of proteins in our dataset, since longer proteins will contain more n-grams and will thus tend to have more n-gram similarity with every domain.

The kernel K_I can be efficiently computed by computing $K_n(x, D_i)$ for all $1 \leq i \leq P$ as follows:

1. Compute D_i for each $\mathcal{P}_i$ by representing each sequence in $\mathcal{P}_i$ by an automaton and determinizing and minimizing the union of these automata.
2. For all $1 \leq i \leq P$ compute $T_n \circ D_i$, and for each sequence x in the dataset compute $T_n \circ X$, where X is the automaton representing x. Optimize the results by projecting onto outputs, applying epsilon-removal, determinizing and minimizing to obtain weighted automata A_i and Y_x.
3. Compute $K_n(x, D_i)$ by intersecting A_i and Y_x and using a generic single-source shortest-distance algorithm (Cortes et al., 2004) to compute the sum of all the paths in the resulting automaton.

The complexity of computing $K_n(x, D_i)$ for a fixed set of domains grows linearly in the length of x, hence the complexity of computing $K_I(x, y)$ grows linearly in the sum of the length of x and y, *i.e.* in $O(|x| + |y|)$. Thus, this kernel is efficient to compute. However, it does not capture the similarity of two sequences in as fine a way as the next kernel we present.

3.4. Joint Domain Kernel

Let us consider two sequences x and y and a given domain $\mathcal{P}_i$. Let $\mathcal{X}$ be the set of n-grams in common between x and $\mathcal{P}_i$, and $\mathcal{Y}$ the set of n-grams in common between y and $\mathcal{P}_i$. When computing the similarity between x and y according to $\mathcal{P}_i$, the IDK K_I takes into account all n-grams in common between x and $\mathcal{P}_i$ and between y and $\mathcal{P}_i$, i.e., all the n-grams in $\mathcal{X} \cup \mathcal{Y}$, regardless of whether these n-grams appear in both x and y. Thus, K_I may indicate that x and y are similar according to $\mathcal{P}_i$ even if $\mathcal{X}$ and $\mathcal{Y}$ differ significantly, or in other words, even if x and y are similar to $\mathcal{P}_i$ for different reasons. In contrast, the Joint Domain kernel (JDK) only takes into consideration the n-grams in common to x, y and $\mathcal{P}_i$, that is the n-grams in $\mathcal{X} \cap \mathcal{Y}$, when determining the similarity between x and y. It will thus declare x and y similar according to $\mathcal{P}_i$ iff x and y are similar to $\mathcal{P}_i$ for the same reason.

[2]Both the IDK and spectrum kernels represent sequences as vectors of n-gram counts but only the IDK accounts for the n-gram spectrums of the domains of interest.

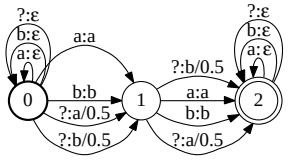


Figure 4. Counting transducer $\overline{T}_2$ with $\Sigma = \{a, b\}$, '?' representing the gap symbol and an expansion penalty weight of 0.5.

For each domain $\mathcal{P}_i$, the JDK defines a kernel K_i that measures the similarity between two sequences x and y according to $\mathcal{P}_i$, using as a similarity measure the count of the n-grams in common among x, y and $\mathcal{P}_i$. More precisely, we define $K_i : \Sigma^* \times \Sigma^* \to \mathbb{R}$ as follows:

$$K_i(x, y) = \sum_{|z|=n} c_x(z) c^2_{D_i}(z) c_y(z). \tag{8}$$

Each K_i is normalized as shown in Equation 7. We then combine these P kernels to obtain the kernel $K_J : \Sigma^* \times \Sigma^* \to \mathbb{R}$ defined as follows:

$$\begin{aligned} K_J(x, y) &= \sum_{i=1}^{P} K_i(x, y) \\ &= \sum_{i=1} \sum_{|z|=n} c_x(z) c^2_{D_i}(z) c_y(z). \end{aligned} \tag{9}$$

We will now show that each K_i and thus K_J is a PDS rational kernel. Let A_i be the weighted automata obtained by composing D_i with T_n and projecting the result onto its output: $A_i = \pi_2(D_i \circ T_n)$. From the definition of T_n, it follows that $A_i(z) = c_{D_i}(z)$ and $c_x(z) = T_n(x, z)$ for all $|z| = n$. Thus, for all (x, y), $K_i(x, y)$ can be rewritten as:

$$\begin{aligned} K_i(x, y) &= \sum_{|z|=n} T_n(x, z) A_i(z) A_i(z) T_n(y, z) \\ &= (T_n \circ A_i \circ A_i \circ (T_n)^{-1})(x, y). \end{aligned} \tag{10}$$

Observe that $(T_n \circ A_i)^{-1} = A_i^{-1} \circ T_n^{-1} = A_i \circ T_n^{-1}$ since for an automaton the inverse A_i^{-1} coincides with A_i. Thus, $K_i(x, y) = \big((T_n \circ A_i) \circ (T_n \circ A_i)^{-1}\big)(x, y)$, which is of the form $T \circ T^{-1}$ and thus K_i is PDS. Since PDS kernels are closed under sum, K_J is also PDS.

The computation of the kernel K_J is more costly than that of K_I since a full $D \times D$ kernel matrix needs to be computed for each Pfam domain. This leads to $D^2 \times P$ rational kernel computations to compute K_J, compared to only $D \times P$ rational kernel computations for K_I. This is significant when $P = 6190$. Thus, it is important to determine an efficient way to compute the kernels K_i. The following is an efficient method for computing K_J that we adopt in our experiments, in which the complexity of computing $K_J(x, y)$ for a fixed set of domains linearly depends on the product of the length of x and y, *i.e.* in $O(|x||y|)$:

1. Compute each A_i and optimize using epsilon-removal, determinization and minimization.
2. For each sequence x in the dataset, compute $Y_x = \pi_2(T_n \circ X)$ where X is the automaton representing x and optimize Y_x using epsilon-removal, determinization and minimization.
3. $K_i(x, y)$ is obtained by computing $A_i \circ Y_x$ and $A_i \circ Y_y$, computing the intersection of the resulting automata and using a generic single-source shortest-distance algorithm (Cortes et al., 2004) to compute the sum of all paths in the resulting automaton.

Gap Symbol Handling

The sequence alignments in the Pfam domain ($\mathcal{P}_i$) contain a gap symbol used to pad the alignments. In the previous two sections, we either ignored the gap symbol (when dealing with raw sequence data) or treated it as a regular symbol (when dealing with aligned sequences). In the latter case, since this symbol does not appear in the sequences in the dataset, the result is that all n-grams containing the gap symbol are ignored during the computation of K_I and K_J.

Alternatively, we can treat the gap symbol as a wildcard, allowing it to match any regular symbol. This can be achieved by modifying the transducer T_n to match any gap symbol on its input to any regular symbol on its output (these transitions can also be assigned a weight to penalize gap expansion). We denote by $\overline{T}_n$ the resulting transducer and replace T_n by $\overline{T}_n$ when composing with D_i. Figure 4 shows $\overline{T}_2$ for $|\Sigma| = \{a, b\}$ with the symbol '?' representing the gap symbol and an expansion penalty weight of 0.5.

3.5. Domain Kernels Based on Moments of Counts

Although counting common n-grams leads to informative kernels, this technique affords a further generalization that is particularly suitable when defining kernels for domains. We can view the sum of the counts of an n-gram in a domain as its average count after normalization. One could extend this idea to consider higher moments of the counts of an n-gram, as this could capture useful information about how similarity varies across the sequences within a single domain.

Remarkably, it is possible to design efficiently computable domain kernels based on these quantities by

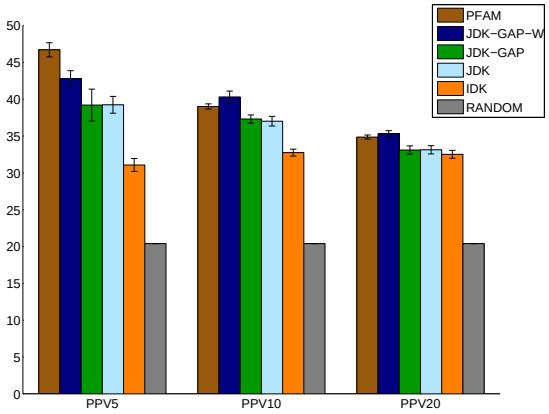


Figure 5. SVM's performance with various kernels averaged over all datasets.

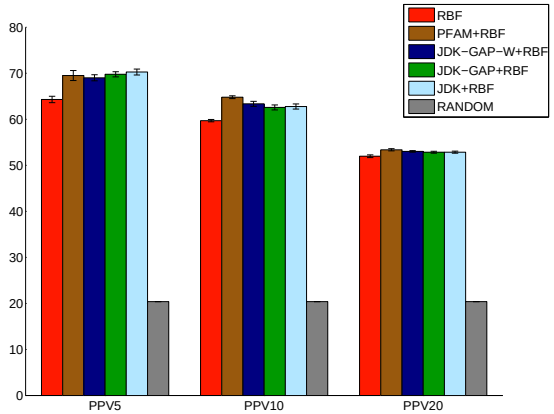


Figure 6. SVM's performance with various kernels combined with RBF kernel averaged over all datasets.

generalizing the domain kernels from Sections 3.3 and 3.4 in a way similar to what is described by Cortes and Mohri (2005). Let m be a positive integer. We can define the *m-th moment of the count* of the sequence z in a weighted automata A, denoted by $c_{A,m}(z)$, as:

$$c_{A,m}(z) = \sum_{u \in \Sigma^*} c_u^m(z) A(u). \tag{11}$$

Both of our kernel families can then be generalized to a similarity measure based on the m-th moment of the n-gram counts as follows:

$$K_m^I(x,y) = \sum_{i=1}^{P} \Big(\sum_{|z|=n} c_{x,m}(z) c_{D_i,m}(z) \Big) \Big(\sum_{|z|=n} c_{y,m}(z) c_{D_i,m}(z) \Big)$$

$$K_m^J(x,y) = \sum_{i=1} \sum_{|z|=n} c_{x,m}(z) c_{D_i,m}^2(z) c_{y,m}(z).$$

These kernels can be efficiently computed by using, in place of T_n, a transducer T_m^n that can be defined such that $T_m^n(x,z) = (c_x(z))^m = c_{x,m}(z)$ for all $x, z \in \Sigma^*$ with $|z| = n$.

4. Experimental Results

We evaluated the domain-based kernels described in Section 3 (with $n = 3$) using an experimental design similar to Section 2.3. In order to test these kernels under various conditions, we chose to work with datasets sampled from the yeast dataset used in Section 2. We constructed 10 datasets, each containing 500 sampled data points randomly chosen from the 4728 initial points, subject to the constraint that we maintained the same ratio of positively and negatively labeled points. We worked on a large cluster machines and used the OpenFst and OpenKernel libraries to construct similarity matrices for each sample dataset for varying kernels (Allauzen et al., 2007; Allauzen & Mohri, 2007). Generating similarity matrices took less than 30 minutes for IDK, 1 hour for JDK, and 2.5 hours for JDK with gaps treated as wildcards. We do not show results for the top 1% since it is an unreliable statistic when working with 500 points. In all reported results we exclude results from one sampled dataset that generated pathological results for all sequence kernels.

Figure 5 shows the average prediction performance over the sampled datasets for various kernels. The figure shows that the average performance of the JDK with gaps treated as wildcards (JDK-GAPS-W) is slightly better than the Pfam kernel, as it outperforms the Pfam kernel for the top 10% and 20% predictions. The figure also presents results for variants of the JDK that either ignore gaps in the seed alignment (JDK) or treat them as a distinct symbol (JDK-GAPS). The results show that, regardless of the treatment of gaps, the JDK drastically outperforms the IDK.

Based on these results, we next tested the effectiveness of the JDK combined with the RBF kernel. Similar to the results in Figure 1, average prediction performance over the sampled datasets was better using combination kernels in contrast to any kernel alone.[3] Figure 6 shows that the combined JDK is comparable to the combined Pfam kernel. Further, in contrast to the results in Figure 5, the treatment of gaps by the JDK does not significantly alter prediction efficiency. In other words, the JDK is able to match the best results of the Pfam kernel using only raw Pfam sequence data (JDK), while completely ignoring the hand-curated multiple sequence alignments that are vital to parameterizing the HMMs of the Pfam Library. We did not perform experiments using higher moments of the count, as described in Section 3.5, though we suspect

[3] As in Figure 1, RBF alone outperforms all sequence kernels alone, possibly due to the phyletic retention feature.

that these more refined kernels would lead to further improvements over the Pfam kernel.

5. Conclusion

We presented novel domain-based sequence kernels that require no hand-crafted information, in contrast to the Pfam kernel. The joint domain kernels we defined were shown to match or outperform the best previous results for predicting protein essentiality. These kernels and their generalization based on moments of counts can be used for any application requiring similarity between sequences that may be extracted from proximity to several large sequence domains. In bioinformatics, such applications may include remote homology prediction, subcellular localization, and prediction of protein-protein interaction.

6. Acknowledgments

This work was partially supported by the National Science Foundation award number MCB-0209754 and the New York State Office of Science Technology and Academic Research (NYSTAR), and was also sponsored in part by the Department of the Army Award Number W81XWH-04-1-0307. The U.S. Army Medical Research Acquisition Activity, 820 Chandler Street, Fort Detrick MD 21702-5014 is the awarding and administering acquisition office. The content of this material does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

References

Allauzen, C., & Mohri, M. (2007). OpenKernel library. `http://www.openkernel.org`.

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., & Mohri, M. (2007). OpenFst: a general and efficient weighted finite-state transducer library. *CIAA 2007* (pp. 11–23). Springer. `http://www.openfst.org`.

Ben-Hur, A., & Brutlag, D. L. (2003). Remote homology detection: a motif based approach. *ISMB (Supplement of Bioinformatics)* (pp. 26–33).

Ben-Hur, A., & Noble, W. (2005). Kernel methods for predicting protein-protein interactions. *Bioinformatics*, *21*, 38–46.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chen, Y., & Xu, D. (2005). Understanding protein dispensability through machine learning analysis of high-throughput data. *Bioinformatics*, *21*, 575–581.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. *NIPS 2001* (pp. 625–632).

Cortes, C., Haffner, P., & Mohri, M. (2004). Rational Kernels: Theory and Algorithms. *Journal of Machine Learning Research*, *5*, 1035–1062.

Cortes, C., & Mohri, M. (2005). Moment kernels for regular distributions. *Machine Learning*, *60*, 117–134.

Cortes, C., & Vapnik, V. N. (1995). Support-Vector Networks. *Machine Learning*, *20*, 273–297.

Eskin, E., & Snir, S. (2005). The Homology Kernel: A Biologically Motivated Sequence Embedding into Euclidean Space. *CIBCB* (pp. 179–186).

Gustafson, A., Snitkin, E., Parker, S., DeLisi, C., & Kasif, S. (2006). Towards the identification of essential genes using targeted genome sequencing and comparative analysis. *BMC:Genomics*, *7*, 265.

Haussler, D. (1999). *Convolution Kernels on Discrete Structures* (Technical Report UCSC-CRL-99-10). University of California at Santa Cruz.

Lanckriet, G., Deng, M., Cristianini, N., Jordan, M., & Noble, W. (2004). Kernel-based data fusion and its application to protein function prediction in yeast. *Pacific Symposium on Biocomputing* (pp. 300–311).

Leslie, C. S., & Kuang, R. (2004). Fast String Kernels using Inexact Matching for Protein Sequences. *Journal of Machine Learning Research*, *5*, 1435–1455.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watskins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, *2*, 419–44.

Platt, J. (2000). Probabilities for support vector machines. In *Advances in large margin classifiers*. Cambridge, MA: MIT Press.

Salomaa, A., & Soittola, M. (1978). *Automata-Theoretic Aspects of Formal Power Series*. Springer.

Sonnhammer, E., Eddy, S., & Durbin, R. (1997). Pfam: A comprehensive database of protein domain families based on seed alignments. *Proteins: Structure, Function and Genetics*, *28*, 405–420.

Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T., & Müller, K.-R. (2000). Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites. *Bioinformatics*, *16*, 799–807.

Hierarchical Kernel Stick-Breaking Process for Multi-Task Image Analysis

Qi An QA@EE.DUKE.EDU
Chunping Wang CW36@EE.DUKE.EDU
Ivo Shterev IS33@EE.DUKE.EDU
Eric Wang EW28@EE.DUKE.EDU
Lawrence Carin LCARIN@EE.DUKE.EDU
Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708

David B. Dunson DUNSON@STAT.DUKE.EDU
Department of Statistical Science, Duke University, Durham, NC 27708

Abstract

The kernel stick-breaking process (KSBP) is employed to segment general imagery, imposing the condition that patches (small blocks of pixels) that are spatially proximate are more likely to be associated with the same cluster (segment). The number of clusters is not set *a priori* and is inferred from the hierarchical Bayesian model. Further, KSBP is integrated with a shared Dirichlet process prior to simultaneously model multiple images, inferring their inter-relationships. This latter application may be useful for sorting and learning relationships between multiple images. The Bayesian inference algorithm is based on a hybrid of variational Bayesian analysis and local sampling. In addition to providing details on the model and associated inference framework, example results are presented for several image-analysis problems.

1. Introduction

The segmentation of general imagery is a problem of long-standing interest. There have been numerous techniques developed for this purpose, including K-means and associated vector quantization methods (Ding & He, 2004), statistical mixture models (McLachlan & Basford, 1988), as well as spectral clustering (Ng et al., 2001). This list of existing methods is not exhaustive, although these methods share attributes associated with most existing algorithms. First, the clustering is based on the features of the image, and when clustering these features one typically does not account for their physical location within the image (although the location may be appended as a feature component). Secondly, the segmentation or clustering of images is typically performed one image at a time, and therefore there is no attempt to relate the segments of one image to segments in other images (*i.e.*, to learn inter-relationships between multiple images). Finally, in many of the techniques cited above one must *a priori* set the number of anticipated segments or clusters. The techniques developed in this paper seek to perform clustering or segmentation in a manner that explicitly accounts for the physical locations of the features within the image, and multiple images are segmented simultaneously (termed "multi-task learning") to infer their inter-relationships. Moreover, the analysis is performed in a semi-parametric manner, in the sense that the number of segments or clusters is not set *a priori*, and is inferred from the data. There has been recent research wherein spatial information has been exploited when clustering (Figueiredo et al., 2007), but that segmentation has been performed one image at a time, and therefore not in a multi-task setting.

To address the goals elucidated above within a statistical setting, we employ a class of hierarchical models related to the Dirichlet process (DP) (Ferguson, 1973). The Dirichlet process is a statistical prior that may be summarized succinctly as follows. Assume that the n-th patch is represented by feature vector $\boldsymbol{x}_n$, and the total image is composed of N such feature vectors $\{\boldsymbol{x}_n\}_{n=1,N}$. The feature vector associated with each patch is assumed drawn from a parametric distribution $f(\phi_n)$, where ϕ_n represents the parameters associated with the n-th feature vector. A DP prior can be placed on ϕ_n, which is characterized by the non-negative parameter α and the "base" distribution G_o. We adopt the stick-breaking construction developed by Sethuraman (Sethuraman, 1994), and the hierarchical model may be expressed as

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

$$
\begin{aligned}
\boldsymbol{x}_n|\phi_n &\overset{ind}{\sim} f(\phi_n) \\
\phi_n|G &\overset{iid}{\sim} G \\
G &= \sum_{h=1}^{\infty} \pi_h \delta_{\theta_h} \\
\pi_h &= V_h \prod_{l=1}^{h-1} (1 - V_l) \\
V_h &\overset{iid}{\sim} Beta(1, \alpha) \\
\theta_h &\overset{iid}{\sim} G_o.
\end{aligned} \tag{1}
$$

This is termed a "stick-breaking" representation of DP because one sequentially breaks off "sticks" of length π_h from an original stick of unit length ($\sum_{h=1}^{\infty} \pi_h = 1$). As a consequence of the properties of the distribution $Beta(1, \alpha)$, for relatively small α it is likely that only a relatively small set of sticks π_h will have appreciable weight/size, and therefore when drawing parameters ϕ_n from the associated G it is probable multiple ϕ_n will share the same "atoms" θ_h (those associated with the large-amplitude sticks). The parameter α therefore plays an important role in defining the number of clusters that are constituted, and therefore in practice one typically places a non-informative Gamma prior on α (Xue et al., 2007).

The form of the model in (1) imposes the prior belief that the feature vectors $\{\boldsymbol{x}_n\}_{n=1,N}$ associated with an image should cluster, and the data are used to infer the most probable clustering distribution, via the posterior distribution on the parameters $\{\phi_n\}_{n=1,N}$. Such semi-parametric clustering has been studied successfully in many settings (Xue et al., 2007; Rasmussen, 2000). However, there are two limitations of such a model, with these defining the focus of this paper. First, while the model in (1) captures our belief that the feature vectors should cluster, it does not impose our additional belief that the probability that two feature vectors are in the same cluster should increase as their physical locations within the image become more proximate; this is an important factor when one is interested in segmenting an image into contiguous regions. Secondly, typical semi-parametric clustering has been performed one image or dataset at a time, and here we wish to cluster multiple images simultaneously, to infer the inter-relationships between clusters in different images, thereby inferring the inter-relationships between the associated multiple images themselves.

As an extension of the DP-based mixture model, we here consider the recently developed kernel stick-breaking process (KSBP) (Dunson & Park, 2008), introduced by Dunson and Park. As detailed below, this model is similar to that in (1), but now the stick-breaking process is augmented to employ a kernel function to quantify the prior belief associated with spatially proximate patches. In (Dunson & Park, 2008) a Markov chain Monte Carlo (MCMC) sampler was used to estimate the posterior on the model parameters. In the work considered here we are interested in relatively large data sets, and therefore we develop an inference engine that exploits ideas from variational Bayesian analysis (Beal, 2003).

There are problems for which one may wish to perform segmentation on multiple images simultaneously, with the goal of inferring the inter-relationships between the different images. This is referred to as multi-task learning (MTL) (Thrun & O'Sullivan, 1996; Xue et al., 2007), where here each "task" corresponds to clustering feature vectors from a particular image. As presented below, it is convenient to simultaneously cluster/segment multiple images by linking the multiple associated KSBP models with an overarching DP. There are at least three applications of MTL in the context of image analysis: (i) one may have a set of images, some of which are labeled, and others of which are unlabeled, and by performing an MTL analysis on all of the images one may infer labels for the unlabeled image segmentation, by drawing upon the relationships to the labeled imagery; (ii) by inferring the inter-relationships between the different images, one may sort the images as well as sort components within the images; (iii) one may identify abnormal images and locations within an image in an unsupervised manner, by flagging those locations that are allocated to a segmentation component that is locally rare. A similar scenario has been studied in (Sudderth et al., 2006), where the spatial translations are handled with transformed Dirichlet processes.

2. Kernel Stick-Breaking Process

2.1. KSBP prior for image processing

The stick-breaking representation of the Dirichlet process (DP) was summarized in (1), and this has served as the basis of a number of generalizations of the DP. The dependent DP (DDP) proposed by MacEachern (MacEachern, 1999) assumes a fixed set of weights, $\boldsymbol{\pi}$, while allowing the atoms $\theta = \{\theta_1, \cdots, \theta_N\}$ to vary with the predictor $\boldsymbol{x}$ according to a stochastic process. Dunson and Park (Dunson & Park, 2008) have proposed the kernel stick-breaking process (KSBP), which is particularly attractive for image-processing applications. Rather than simply considering the feature vector $\{\boldsymbol{x}_n\}_{n=1,N}$, we now consider $\{\boldsymbol{x}_n, \boldsymbol{r}_n\}_{n=1,N}$, where $\boldsymbol{r}_n$ is tied to the location of the pixel or block of pixels used to constitute feature vector $\boldsymbol{x}_n$. We let $K(\boldsymbol{r}, \boldsymbol{r}', \psi) \rightarrow [0, 1]$ define a bounded kernel function with parameter ψ, where $\boldsymbol{r}$ and $\boldsymbol{r}'$ represent general locations in the image of interest. One may choose to place a prior on the kernel parameter ψ; this issue is revisited be-

low. A draw $G_{\boldsymbol{r}}$ from a KSBP prior is a function of position $\boldsymbol{r}$, and is represented as

$$\begin{aligned} G_{\boldsymbol{r}} &= \sum_{h=1}^{\infty} \pi_h(\boldsymbol{r}; V_h, \Gamma_h, \psi)\delta_{\theta_h} \\ \pi_h(\boldsymbol{r}; V_h, \Gamma_h, \psi) &= V_h K(\boldsymbol{r}, \Gamma_h, \psi) \textstyle\prod_{l=1}^{h-1}[1 - V_l K(\boldsymbol{r}, \Gamma_l, \psi)] \\ V_h &\overset{iid}{\sim} Beta(a, b) \\ \Gamma_l &\overset{iid}{\sim} H \\ \theta_h &\overset{iid}{\sim} G_o. \end{aligned} \tag{2}$$

Dunson and Park (Dunson & Park, 2008) prove the validity of $G_{\boldsymbol{r}}$ as a probability measure. Comparing (1) and (2), both priors take the general form of a stick-breaking representation, while the KSBP prior possesses several interesting properties. For example, the stick weights $\pi_h(\boldsymbol{r}; V_h, \Gamma_h, \psi)$ are a function of $\boldsymbol{r}$. Therefore, although the atoms $\{\theta_h\}_{h=1,\infty}$ are the same for all $\boldsymbol{r}$, the weights effectively shift the probabilities of different θ_h based on $\boldsymbol{r}$. The basis functions Γ_h serve to localize in the space of $\boldsymbol{r}$ regions (clusters) in which the weights $\pi_h(\boldsymbol{r}; V_h, \Gamma_h, \psi)$ are relatively constant, with the size of these regions tied to the kernel parameter ψ.

If $f(\phi_n)$ is the parametric model (with parameter ϕ_n) responsible for the generation of $\boldsymbol{x}_n$, we now assume that the augmented data $\{\boldsymbol{x}_n, \boldsymbol{r}_n\}_{n=1,N}$ are generated as

$$\begin{aligned} \boldsymbol{x}_n &\overset{ind}{\sim} f(\phi_n) \\ \phi_n &\overset{ind}{\sim} G_{\boldsymbol{r}_n} \\ G_{\boldsymbol{r}} &\sim KSBP(a, b, \psi, G_o, H). \end{aligned} \tag{3}$$

The notation $G_{\boldsymbol{r}} \sim KSBP(a, b, \psi, G_o, H)$ is meant to convey that $G_{\boldsymbol{r}}$ is drawn *one* time from the KSBP, and is a parametric function of location $\boldsymbol{r}$, and it is evaluated at specific locations $\{\boldsymbol{r}_n\}_{n=1,N}$.

The generative model in (3) states that two feature vectors that come from the same region in the image (defined via $\boldsymbol{r}$) will have similar $\pi_h(\boldsymbol{r}; V_h, \Gamma_h, \psi)$, and therefore they are likely to share the same atoms θ_h. The settings of a and b control how much similarity there will be in drawn atoms for a given spatial cluster centered about a particular Γ_h. If we set $a = 1$ and $b = \alpha$, analogous to the DP, small concentration parameter α and/or small kernel parameter ψ will impose that π_h is likely to be near one, and therefore only a relatively small number of atoms θ_h are likely to be dominant for a given cluster spatial center Γ_h. On the other hand, if two features are generated from distant parts of a given image, the associated atoms θ_h that may be prominent for each feature vector are likely to be different, and therefore it is of relatively low probability that these feature vectors would have been generated via the same parameters ϕ. It is possible that the model may infer two distinct and widely separated clusters/segments with similar parameters (atoms); if the G_o within the KSBP is itself drawn from a DP, as it will be below when analyzing multiple images, widely separated clusters may share the exact same atoms.

For the case $a = 1$ and $b = \alpha$, which we consider below, we employ the notation $G_{\boldsymbol{r}} \sim KSBP(\alpha, \psi, G_o, H)$. Below we will also assume that $f(\phi)$ corresponds to a multivariate Gaussian distribution.

2.2. Spatial correlation properties

As indicated above, the functional form of the kernel function is important and needs to be chosen carefully. A commonly used kernel is given as $K(\boldsymbol{r}, \Gamma, \psi) = \exp(-\psi\|\boldsymbol{r} - \Gamma\|^2)$ for $\psi > 0$, which allows the associated stick weight to change continuously from $V_h \prod_{l=1}^{h-1}(1-V_l)$ to 0 conditional on the distance between $\boldsymbol{r}$ and Γ. By choosing a kernel we are also implicitly imposing the dependency between the priors of two samples, $G_{\boldsymbol{r}}$ and $G_{\boldsymbol{r}'}$. Specifically, both priors are encouraged to share the same atoms θ_h if $\boldsymbol{r}$ and $\boldsymbol{r}'$ are close, with this discouraged otherwise. Dunson and Park (Dunson & Park, 2008) derive the correlation coefficient between two probability measures $G_{\boldsymbol{r}}$ and $G_{\boldsymbol{r}'}$ to be

$$\begin{aligned} &corr\{G_{\boldsymbol{r}}, G_{\boldsymbol{r}'}\} \\ &= \frac{\sum_{h=1}^{\infty} \pi_h(\boldsymbol{r}; V_h, \Gamma_h, \psi)\pi_h(\boldsymbol{r}'; V_h, \Gamma_h, \psi)}{\sqrt{\sum_{h=1}^{\infty} \pi_h(\boldsymbol{r}; V_h, \Gamma_h, \psi)^2}\sqrt{\sum_{h=1}^{\infty} \pi_h(\boldsymbol{r}'; V_h, \Gamma_h, \psi)^2}}. \end{aligned}$$

The coefficient approaches unity in the limit as $\boldsymbol{r} \to \boldsymbol{r}'$. Since the correlation is a strong function of the kernel parameter ψ, below we will consider a distinct ψ_h for each stick. This implies that the spatial extent within the image over which a given stick is important will vary as a function of the stick (to accommodate textural regions of different sizes).

3. Multi-Task Image Segmentation with a Hierarchical KSBP

We now consider the problem for which we wish to jointly segment M images, where each image has an associated set of feature vectors with location information, in the sense discussed above. Aggregating the data across the M images, we have the set of feature vectors $\{\boldsymbol{x}_{nm}, \boldsymbol{r}_{nm}\}_{n=1,N_m;\ m=1,M}$. The image sizes may be different, and therefore the number of feature vectors N_m may vary between images. The premise of the model discussed below is that the cluster or segment characteristics may be

similar between multiple images, and the inference of these inter-relationships may be of value. Note that the assumption is that sharing of clusters may be of relevance for the feature vectors, but not for the associated locations.

3.1. Model

A relatively simple means of sharing feature-vector clusters between the different images is to let each image be processed with a separate $KSBP(\alpha_m, \psi_m, G_m, H_m)$. To achieve the desired sharing of feature-vector clusters between the different images, we impose that $G_m \equiv G$ and G is drawn $G \sim DP(\gamma, G_o)$. Recalling the stick-breaking form of a draw from $DP(\gamma, G_o)$, we have $G = \sum_{h=1}^{\infty} \pi_h \delta_{\theta_h}$, in the sense summarized in (1). The discrete form of G is very important, for it implies that the different $G_{\boldsymbol{r}}$ will share the same set of discrete atoms $\{\theta_h\}_{h=1,\infty}$. It is interesting to note that for the case in which the kernel parameter ψ is set such that $K(\boldsymbol{r}, \Gamma_h, \psi) \rightarrow 1$, the hierarchical KSBP (H-KSBP) model reduces to the hierarchical Dirichlet process (HDP) (Teh et al., 2005).

Therefore, the H-KSBP model is represented as

$$\begin{aligned} \boldsymbol{x}_{nm} &\overset{ind}{\sim} \mathcal{N}(\phi_{nm}) \\ \phi_{nm} &\overset{ind}{\sim} G_{\boldsymbol{r}_{nm}} \\ G_{\boldsymbol{r}} &\sim KSBP(\alpha_m, \psi_m, G, H_m) \\ G &\sim DP(\gamma, G_o), \end{aligned} \tag{4}$$

where $\mathcal{N}(\cdot)$ is a Gaussian distribution. Assume that G is composed of the atoms $\{\theta_h\}_{h=1,\infty}$, from the perspective of the stick-breaking representation in (1). These same atoms are shared across all $\{G_{\boldsymbol{r}_{nm}}\}_{n=1,N_m;m=1,M}$ drawn from the associated KSBPs, but with respective stick weights unique to the different images, and a function of position within a given image. The posterior inference allows one to infer which clusters of features are unique to a particular image, and which clusters are shared between multiple images. The density functions H_m are tied to the support of the m-th image, and in practice this is set as uniform across the image extent. The distinct α_m, for each of which a Gamma hyper-prior may be imposed, encourages that the number of clusters (segments) may vary between the different images, although one may simply wish to set $\alpha_m = \alpha$ for all M tasks.

For notational convenience, in (4) it was assumed that the kernel parameter ψ_m varied between tasks, but was fixed for all sticks within a given task; this is overly restrictive. In the implementation that follows the parameter ψ_{hm} may vary across tasks *and* across the task-specific KSBP sticks.

3.2. Posterior inference

For inference purposes, we truncate the number of sticks in the KSBP to T, and the number of sticks in the truncated DP to K (the truncation properties of the stick-breaking representation of DP are discussed in (Ishwaran & James, 2001), although we emphasize that when truncating KSBP one must take into account the draws from the Beta distribution *and* the properties of the kernel, to assure that the truncated set of sticks sum to one). Due to the discreteness of $G = \sum_{k=1}^{K} \beta_k \delta_{\theta_k}$, each draw of the KSBP, $G_{\boldsymbol{r}_{nm}} = \sum_{h=1}^{T} \pi_{hm} \delta_{\phi_{hm}}$, can only take atoms $\{\phi_{hm}\}_{h=1,T;\, m=1,M}$ from K unique possible values $\{\theta_k\}_{k=1,K}$; when drawing atoms ϕ_{hm} from G, the respective probabilities for $\{\theta_k\}_{k=1,K}$ are given by $\{\beta_k\}_{k=1,K}$, and for a given $\boldsymbol{r}_{nm}$ the respective probabilities for different $\{\phi_{hm}\}_{h=1,T;\, m=1,M}$ are defined by $\{\pi_{hm}\}_{h=1,T;\, m=1,M}$. In order to reflect the correspondences between the data and atoms explicitly, we further introduce two auxiliary indicator variables. One is z_{nm}, this indicating which stick of the KSBP the feature vector $\boldsymbol{x}_{nm}$ is associated, and the other is t_{hm}, this indicating which mixing component θ_k the atom ϕ_{hm} is associated with.

With this specification we can represent our H-KSBP mixture model via a stick-breaking characterization. A graphical representation of the proposed H-KSBP model is provided in Figure 1.

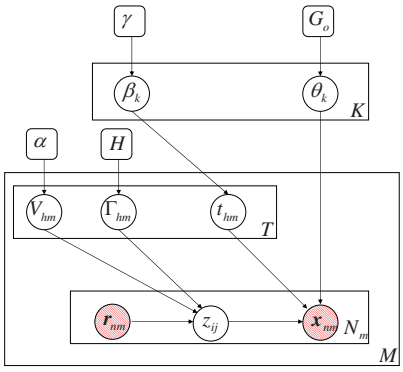


Figure 1. A graphical representation of the H-KSBP mixture model.

For the large-scale problems of interest here we employ variational Bayesian (VB) inference, which has proven to be a relatively fast (compared to MCMC) and accurate inference tool for many models and applications (Beal, 2003; Blei & Jordan, 2004). To employ VB, a conjugate prior is required for all variables in the model. In the proposed model, we however cannot obtain a closed form for the variational posterior distribution of the node V_{hm}, because of the the kernel function. Alternatively, motivated by

the Monte Carlo Expectation Maximization (MCEM) algorithm (Wei & Tanner, 1990), we develop a Monte Carlo Variational Bayesian (MCVB) inference algorithm, where the intractable nodes are approximated with Monte Carlo samples from their conditional posterior distributions. The resulting algorithm combines the benefits of both MCMC and VB, and has proven to be effective for the examples we have considered (some of which are presented here).

Given the H-KSBP mixture model detailed in Section 3.1, we can follow standard variational Bayesian inference (Beal, 2003) to infer the variables of interests. All the updates are analytical except for V_{hm}, which is estimated with the samples from its conditional posterior distributions. Due to the limited space, we only consider the update for V_{hm}. To obtain the conditional posterior distribution of V_{hm}, we rewrite $z_{nm} = min\{h : A_{nm,h} = B_{nm,h} = 1\}$, with two auxiliary variables defined as: $A_{nm,h} \sim Bernoulli(V_{hm})$ and $B_{nm,h} \sim Bernoulli(K(\boldsymbol{r}_{nm}, \Gamma_{hm}, \psi_m))$.

The conditional posterior distributions of V_{hm} are

$$Beta(1 + \sum_{n:z_{nm} \geq h} A_{nm,h}, \alpha + \sum_{n:z_{nm} \geq h} (1 - A_{nm,h})),$$

where

$$p(A_{nm,h} = B_{nm,h} = 0) = \frac{(1-V_{hm})(1-K(\boldsymbol{r}_{nm},\Gamma_{hm},\psi_m))}{1-V_{hm}K(\boldsymbol{r}_{nm},\Gamma_{hm},\psi_m)}$$
$$p(A_{nm,h} = 0, B_{nm,h} = 1) = \frac{(1-V_{hm})K(\boldsymbol{r}_{nm},\Gamma_{hm},\psi_m)}{1-V_{hm}K(\boldsymbol{r}_{nm},\Gamma_{hm},\psi_m)}$$
$$p(A_{nm,h} = 1, B_{nm,h} = 0) = \frac{V_{hm}(1-K(\boldsymbol{r}_{nm},\Gamma_{hm},\psi_m))}{1-V_{hm}K(\boldsymbol{r}_{nm},\Gamma_{hm},\psi_m)},$$

for $h = 1, 2, \cdots, z_{nm} - 1$, and $A_{nm,h} = B_{nm,h} = 1$ for $h = z_{nm}$.

The hyper-parameters α, γ, and $\boldsymbol{\psi}$ are assumed to be constant for inference of the other parameters. However, since the model performance may be sensitive to the settings of those hyper-parameters, we can relax this assumption by placing non-informative priors. The updates are straightforward (Beal, 2003) and therefore omitted here.

3.3. Convergence

To monitor the convergence of our MCVB algorithm, we compute the lower bound of the log model evidence at each iteration. Because of the sampling of some variables, the lower bound does not in general increase monotonically, but we observed in all experiments that the lower bound increases sequentially for the first several iterations, with generally small fluctuations after it has converged to the local optimal solution.

4. Experimental Results

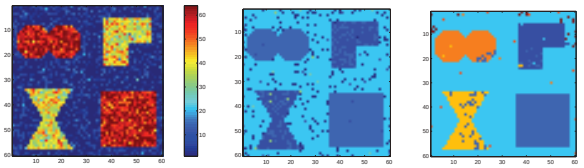
We have applied the H-KSBP multi-task image-segmentation algorithm to both synthetic and real images. We first present results on synthesized imagery, wherein we compare KSBP-based clustering of a single image with associated DP-based clustering. We then consider H-KSBP as applied to actual imagery, taken from a widely utilized database. The hyper-priors in the model for the examples are set as follows: Gamma priors, $G(\tau_{10}, \tau_{20})$ and $G(\tau_{30}, \tau_{40})$, for α and γ with parameter $\tau_{10} = 1e^{-2}$, $\tau_{20} = 1e^{-2}$, $\tau_{30} = 3e^{-2}$, $\tau_{40} = 3e^{-2}$, respectively; a normal-Wishart prior, $N(\boldsymbol{\mu}_k|\boldsymbol{\mu}_0, \eta_0\boldsymbol{\Sigma}_k)W(\boldsymbol{\Sigma}_k|w_*, \boldsymbol{\Sigma}_*)$, conjugate to the Gaussian distribution with $\boldsymbol{\mu}_0 = \mathbf{0}$, $\eta_0 = 1$, $w_* = d + 2$, $\boldsymbol{\Sigma}_* = 5 \times \boldsymbol{I}$; the discrete priors for $\boldsymbol{\Gamma}$ and $\boldsymbol{\psi}$ with uniform weights over all candidates. The stick-breaking truncations are $K = 40, T = 40$.

4.1. Single image segmentation

In this simple illustrative example, each feature vector is associated with a particular pixel, and the feature is simply a real number, corresponding to its intensity; the pixel location is the auxiliary information within the KSBP, while this information is not employed by the DP-based segmentation algorithm. Figure 2 shows the original image and the segmentation results of both algorithms. In Figure 2(a) we note that there are five contiguous regions for which the intensities are similar. There is a background region with a relatively fixed intensity, and within this are four distinct contiguous sub-regions, and of these there are pairs for which the intensities are comparable. The data in Figure 2(a) were generated as follows. Each pixel in each region is generated independently as a draw from a Gaussian distribution; the standard deviation of each of the Gaussians is 10, and the background has mean intensity 5, and the two pairs are generated with mean intensities of 40 and 60. The color bar in Figure 2(a) denotes the pixel amplitudes. The DP and KSBP segmentation results are shown in Figures 2(b) and 2(c), respectively. A distinct color is associated with distinct cluster parameters. In the DP results we note that the four subregions are generally properly segmented, but there is significant speckle in the background region. The KSBP segmentation algorithm is beset by far less speckle. Further, in the KSBP results there are five distinct clusters (dominant KSBP sticks), where in the DP results there are principally three distinct sticks (in the DP, the spatially separated segments with the same features are treated as one cluster, while in the KSBP each contiguous region is represented by its own stick).

In the next set of results, on real imagery, we employ the H-KSBP algorithm, and therefore at the task level segmentation is performed as in Figure 2(c). Alternatively, using the HDP model (Teh et al., 2005), at the task level one em-

ploys clustering of the form in Figure 2(b). The relative performance of H-KSBP and HDP is analyzed.



(a) (b) (c)

Figure 2. A synthetic image example. (a) Original synthetic image, (b) image-segmentation results of DP-based model, and (c) image-segmentation results of KSBP-based model.

4.2. H-KSBP applied to a set of real images

Within the subsequent image analysis we employ features constituted by the independent feature subspace analysis (ISA) technique, developed by Hyvärinen and Hoyer (Hyvärinen & Hoyer, 2000). These features have proven to be relatively shift or translation invariant, which enables them to be widely applicable to many type of images.

We test the H-KSBP model on a subset of images from Microsoft Research Cambridge, available at http://research.microsoft.com/vision/cambridge/recognition/. There are seven types of images used in this database: buildings, clouds, countryside, faces, fireworks, offices and urban. Twenty images are randomly selected from the database for each type, yielding a total of 140 images. To capture textural information within the features, we first divided each image into a contiguous 24×24-pixel non-overlapping patches (more than 70,000 patches in total) and then extract ISA features from each patch; color images are considered, and the RGB colors are handled within ISA feature extraction as in (Hoyer & Hyvärinen, 2000). Concerning learning the ISA independent feature subspaces, we randomly select 150 patches out of each of the 140 images from the seven classes, and these 150 image patches are used for basis training. The posterior on the H-KSBP (and HDP) model parameters is inferred based on the proposed MCVB algorithm, processing all 140 images simultaneously; as discussed in Section 2, the HDP analysis is performed by a special setting of the H-KSBP parameters. To mitigate the influence of random samples and VB initialization, we perform the experiment ten times and report the average results.

Borrowing the successful "bag of words" assumption in text analysis (Blei & Lafferty, 2005), we assume each image is a bag of atoms, which results in a measurable quantity of inter-relationship between images, specifically similar images should share similar distribution over those mixture components. An important aspect of the H-KSBP algorithm is that while in text analysis the "bag of words" may be set *a priori*, here the "bag of atoms" is inferred from the data itself, within the clustering process. Related concepts have been employed previously in image analysis (Quelhas et al., 2007), but in that work one had to set the canonical set of image atoms (shapes) *a priori*, which is somewhat *ad hoc*.

As an example, for the data considered, we show one realization of H-KSBP in Figure 3. In the figure, we display canonical atom usage across all 140 images. Figure 3 is a count matrix, where each square represents the relative number of counts in a given image for a particular atom (atoms indexed along the vertical axis in Figure 3).

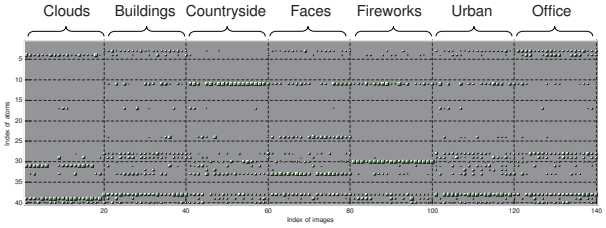


Figure 3. Matrix on the usage of atoms across the different images. The size of each box represents the relative frequency with which a particular atom is manifested in a given image. These results are computed via H-KSBP.

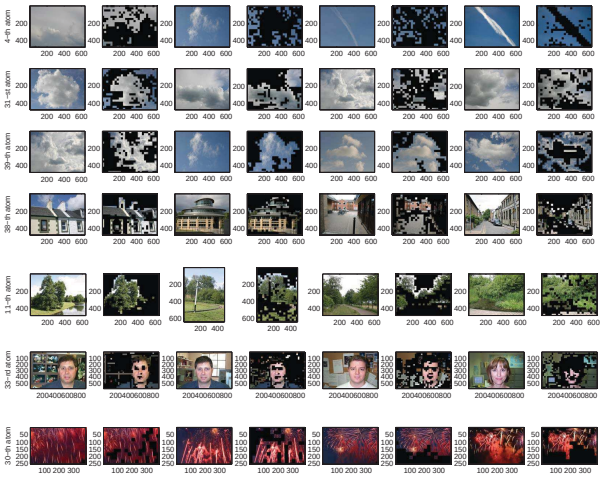


Figure 4. Demonstration of different atoms as inferred by an example run of the H-KSBP algorithm. Each row of the figure corresponds to one atom. Every two images form a set, with the original images at left and areas assigns to a particular atom shown at right.

Figure 4 gives a representation of most of the atoms. For example the 4-th, 31-st and 39-th atoms are associated with clouds and sky; the 38-th atom is principally modeling

buildings; and the 11-th atom is associated with trees and grasses. While performing the experiment, we also noticed it was relatively easy to segment clouds, fireworks, countryside, and urban images while harder to obtain contiguous segments within office images (these typically have far more details, and less large regions of smooth texture; this latter issue may be less an issue of the H-KSBP, but rather of the features employed). An example of this difficulty is observable in Figure 5, as office images are composed of many different atoms. Fortunately, the office images still tend to share similar usage of atoms so that they can be grouped together (sorted) when quantifying similarities between images based on the histogram over atoms (discussed next).

The results in Figure 5, in which both H-KSBP and HDP segmentation results are presented, demonstrate general properties observed when analyzing the images considered here: (i) the segmentation characteristics of HDP were generally good, but on some occasions they were markedly worse (less detailed) than those of H-KSBP; and (ii) the H-KSBP was generally more sensitive to detailed textural differences in the images, thereby generally inferring a larger number of principal atoms (increased number of large sticks).

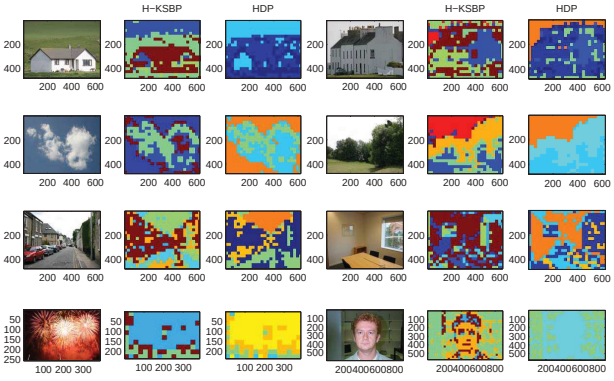


Figure 5. Representative set of segmentation results, comparing H-KSBP and HDP. While these two algorithms tend to generally yield comparable segmentations for the images considered, the H-KSBP is generally more sensitive to details, with this sometimes yielding better segmentations ($e.g.$, the top-level and bottom-right results).

To demonstrate the image-sorting potential of the H-KSBP, we compute the Kullback-Leibler (KL) divergence on the histogram over atoms between any two images, by averaging histograms of the form in Figure 3 over ten random MCVB initializations. For each image, we rank its similarity to all other images based on the associated KL divergence. Performance is addressed quantitatively as follows. For each of the 140 images, we quantify via KL divergence its similarity to all other 139 images, wherein we achieve in ordered list. In Figure 6 we present a confusion matrix, which represents the fraction of the top-ten members of this ordered list that are within the same class (among seven classes) as the image under test.

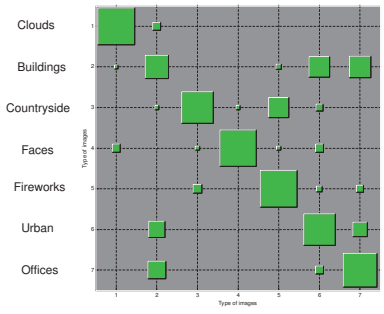


Figure 6. The confusion matrix over image types, generated using H-KSBP.

As demonstrated in Figure 6, the H-KSBP performs well in distinguishing clouds, faces and fireworks images. The buildings and urban images often share some similar atoms, mainly representing buildings, and therefore these are somewhat confused (reasonably, it is felt). The offices images are often related to other relatively complex scenes. Some typical image ranking results are given in Figure 7. It was found that the HDP produced similar sorting results as produced by H-KSBP ($e.g.$, the associated confusion matrix for HDP is similar to that in Figure 6), and therefore the HDP sorting results are omitted here for brevity. This indicates that while in some cases the HDP segmentation results are inferior to those of H-KSBP, in general the ability of HDP and H-KSBP to sort images is comparable (at least for the set of images considered).

The H-KSBP results on the 140-image database were performed in non-optimized $Matlab^{TM}$ software, on a PC with 3 GHz CPU and 2 GB memory. It required about 3 hours to compute one run of the MCVB code for 80 iterations, with typically 40-50 iterations required to achieve convergence. The H-KSBP and HDP algorithms were run with comparable computation times.

5. Conclusions

The kernel stick-breaking process has been extended for use in image segmentation. The algorithm explicitly imposes the belief that feature vectors that are generated from proximate locations in an image are more likely to be associated with the same image segment. We have also extended the KSBP algorithm to the MTL setting, exploring the inter-relationship of images by sharing the same mixing components. Generally superior segmentation performance of H-KSBP was observed relative to HDP, when

Figure 7. Sample image sorting results, as generated by H-KSBP. The top left image is the original image followed by the five most similar images and then the five most dissimilar images.

segmenting multiple images simultaneously. In addition to segmenting multiple images, the H-KSBP and HDP algorithms also yield information about the inter-relationships between the images, based on the underlying sharing mechanisms inferred among the associated clusters. For the images considered, it was found that the H-KSBP and HDP yielded very similar sorting results.

References

Beal, M. (2003). *Variational algorithms for approximate Bayesian inference*. Doctoral dissertation, Gatsby Computational Neuroscience Unit, University College London.

Blei, D., & Jordan, M. (2004). Variational methods for the Dirichlet process. *Proc. the 21st International Conference on Machine Learning*.

Blei, D., & Lafferty, J. (2005). Correlated topic models. *Advances in Neural Information Processing System*.

Ding, C., & He, X. (2004). K-means clustering via principal component analysis. *Proc. the International Conference on Machine Learning* (pp. 225–232).

Dunson, D., & Park, J.-H. (2008). Kernel stick-breaking process. *Biometrika*.

Ferguson, T. (1973). A bayesian analysis of some nonparametric problems. *Annals of Statistics*, *1*.

Figueiredo, M., Cheng, D., & Murino, V. (2007). Clustering under prior knowledge with application to image segmentation. *Advances in Neural Information Processing System*.

Hoyer, P., & Hyvärinen, A. (2000). Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, *11*, 191–210.

Hyvärinen, A., & Hoyer, P. (2000). Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, *12*, 1705–1720.

Ishwaran, H., & James, L. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, *96*, 161–173.

MacEachern, S. (1999). Dependent nonparametric process. *ASA Proceeding of the Section on Bayesian Statistical Science*. Alexandria, VA.

McLachlan, G., & Basford, K. (1988). *Mixture models: Inference and applications to clustering*. Marcel Dekker.

Ng, A., Jordan, M., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 13*.

Quelhas, P., F. Monay, J.-M. O., Gatica-Perez, D., & Tuytelaars, T. (2007). A thosand words in a scenes. *IEEE Trans. Pattern Analysis Machine Intell.*, *9*, 1575–1589.

Rasmussen, C. (2000). The infinite gaussian mixture model. *Advances in Neural Information Processing System* (pp. 554–560).

Sethuraman, J. (1994). A constructive definition of dirichlet priors. *Statistica Sinica*, *4*, 639–650.

Sudderth, E. B., Torralba, A., Freeman, W. T., & Willsky, A. S. (2006). Describing visual scenes using transformed Dirichlet processes. *NIPS 18* (pp. 1297–1304).

Teh, Y., Jordan, M., Beal, M., & Blei, D. (2005). Hierarchical dirichlet processes. *Journal of the American Statistical Association*, *101*, 1566–1582.

Thrun, S., & O'Sullivan, J. (1996). Discovering structure in multiple learning tasks: The TC algorithm. *Proc. the 13th International Conference on Machine Learning*.

Wei, G., & Tanner, M. (1990). A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, *85*, 699–704.

Xue, Y., Liao, X., Carin, L., & Krishnapuram, B. (2007). Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, *8*, 35–63.

Graph Kernels between Point Clouds

Francis R. Bach FRANCIS.BACH@MINES.ORG
INRIA - WILLOW Project-Team, Laboratoire d'Informatique de l'Ecole Normale Supérieure, Paris, France

Abstract

Point clouds are sets of points in two or three dimensions. Most kernel methods for learning on sets of points have not yet dealt with the specific geometrical invariances and practical constraints associated with point clouds in computer vision and graphics. In this paper, we present extensions of graph kernels for point clouds, which allow one to use kernel methods for such objects as shapes, line drawings, or any three-dimensional point clouds. In order to design rich and numerically efficient kernels with as few free parameters as possible, we use kernels between covariance matrices and their factorizations on probabilistic graphical models. We derive polynomial time dynamic programming recursions and present applications to recognition of handwritten digits and Chinese characters from few training examples.

1. Introduction

In recent years, kernels for structured data have been designed in many domains, such as bioinformatics (Vert et al., 2004), text processing (Lodhi et al., 2002) and computer vision (Harchaoui & Bach, 2007; Parsana et al., 2008). They provide an elegant way of including known *a priori* information, by using directly the natural topological structure of objects. Using *a priori* knowledge through kernels on structured data have proved beneficial because it allows (a) to reduce the number of training examples, (b) to reuse existing data representations that are already well developed by experts of those domains and (c) to bring to bear the rapidly developing kernel machinery, and in particular semi-supervised learning—see, e.g., Chapelle et al. (2006)—and hyperparameter learning for supervised kernel methods—see, e.g., Bach et al. (2004).

In this paper, we propose a positive definite kernel between point clouds, with applications to classification of line drawings—such as handwritten digits (LeCun et al., 1998) or Chinese characters (Srihari et al., 2007)—or shapes (Belongie et al., 2002). The natural geometrical structure of point clouds is hard to represent in a few real-valued features (see, e.g., Forsyth and Ponce (2003)), in particular because of (a) the required local or global invariances by rotation, scaling, and/or translation, (b) the lack of preestablished registrations of the point clouds (i.e., points from one cloud are not given matched to points from another cloud), and (c) the noise and occlusion that impose that only portions of two point clouds ought to be compared.

One of the leading principles for designing kernels between structured objects is to decompose each object into parts and to compare all parts of one object to all parts of another object (Shawe-Taylor & Cristianini, 2004). Even if there is an exponential number of such decompositions, which is a common case, this is numerically possible under two conditions: (a) the object must lead itself to an efficient enumeration of subparts, and (b) the similarity function between subparts (i.e., the *local kernel*), beyond being a positive definite kernel, must be simple enough so that the sum over a potentially exponential number of terms can be recursively performed in polynomial time through factorization.

One of the most striking instantiations of this design principle are the *string kernels* (see, e.g., Shawe-Taylor and Cristianini (2004)), which consider all substrings of a given string but still allow efficient computation in polynomial time. The same principle can also be applied to graphs: intuitively, the *graph kernels* (Ramon & Gärtner, 2003; Kashima et al., 2004; Borgwardt et al., 2005) consider all possible subgraphs and compare and count matching subgraphs. However, the set of subgraphs (or even the set of paths) has exponential size and cannot be efficiently described recursively. By choosing appropriate substructures, such as *walks* or *tree-walks*, and fully factorized local kernels, matrix inversion formulations (Kashima et al., 2004) and efficient dynamic programming recursions (Harchaoui & Bach, 2007) allow one to sum over an exponential number of substructures in polynomial time (for more details

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

on graph kernels, see Section 2.1).

In this paper, we consider the application of graph kernels to point clouds. Indeed, we assume that each point cloud has a graph structure (most often a neighborhood graph); then, our graph kernels consider all partial matches between two neighborhood graphs and sum over those. However, the straightforward application of graph kernels poses a major problem: in the context of computer vision, substructures correspond to matched sets of points, and dealing with local invariances by rotation and/or translation imposes to use a local kernel that cannot be readily expressed as a product of separate terms for each pair of points, and the usual dynamic programming and matrix inversion approaches cannot then be directly applied. One of the main contributions of this paper is to design a local kernel that is not fully factorized but can be instead factorized according to the graph underlying the substructure. This is naturally done through probabilistic graphical models and the design of positive definite kernels for covariance matrices that factorize on graphical models (see Section 3). With this novel local kernel, we derive new polynomial time dynamic programming recursions in Section 4. In Section 5, we present simulations on handwritten character recognition.

2. Graph Kernels

In this section, we consider two labelled undirected graphs $G = (V, E, a, x)$ and $H = (W, F, b, y)$, where V, W are vertex sets, E, F are edge sets and a, b, x, y are vertex labelling functions (Diestel, 2005). Two types of labels are considered: *attributes*, which are denoted $a(v) \in \mathcal{A}$ for vertex $v \in V$ and $b(w) \in \mathcal{A}$ for vertex $w \in W$ and *positions*, which are denoted $x(v) \in \mathcal{X}$ and $y(w) \in \mathcal{X}$. We assume that the graphs have no self-loops. Our motivating examples are line drawings, where $\mathcal{X} = \mathcal{A} = \mathbb{R}^2$ (i.e., the position is itself also an attribute). In this case, the graph is naturally obtained from the drawings by considering 4-connectivity or 8-connectivity (Forsyth & Ponce, 2003). In other cases, graphs can be easily obtained from nearest-neighbor graphs.

2.1. Related work

Graph data occur in many application domains, and kernels for attributed graphs have received increased interest in the applied machine learning literature, in particular in bioinformatics (Kashima et al., 2004; Borgwardt et al., 2005) and computer vision (Harchaoui & Bach, 2007). Note that in this paper, we only consider kernels between graphs (each data point is a graph), as opposed to kernels for a single dataset with associated graph information between data points (see, e.g., Shawe-Taylor and Cristianini (2004)).

Current graph kernels can roughly be divided in two classes: the first class is composed of non positive definite similarity measures based on existing techniques from the graph matching literature, that can be made positive definite by *ad hoc* matrix transformations; this includes the edit-distance kernel (Neuhaus & Bunke, 2006) and the optimal assignment kernel (Fröhlich et al., 2005; Vert, 2008).

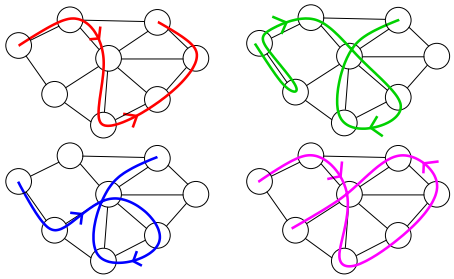


Figure 1. (top left) path, (top right) 1-walk which is not a 2-walk, (bottom left) 2-walk which is not a 3-walk, (bottom right) 4-walk.

Another class of graph kernels relies on a set of substructures of the graphs. The most natural ones are paths, subtrees and more generally subgraphs; however, they do not lead to positive definite kernels with polynomial time computation algorithms—see, in particular, NP-hardness results by Ramon and Gärtner (2003)—and recent work has focused on larger sets of substructures. In particular, *random walk* kernels consider all possible walks and sum a local kernel over all possible walks of the graphs (with all possible lengths). With a proper length-dependent factor, the computation can be achieved by solving a large sparse linear system (Kashima et al., 2004; Borgwardt et al., 2005), whose running time complexity has been recently reduced (Vishwanathan et al., 2007). When considering fixed-length walks, efficient dynamic programming recursions can de derived (Harchaoui & Bach, 2007) that drive down the computation time, at the cost of considering a smaller feature space. These however have the advantage of allowing extensions to other types of substructures, namely "tree-walks" (Ramon & Gärtner, 2003), that we now present.

2.2. Paths, Walks, Subtrees and Tree-walks

Given an undirected graph G with vertex set V, a *path* is a sequence of distinct connected vertices, while a *walk* is a sequence of possibly non distinct connected vertices. In order to prevent the walks from going back and forth too quickly (a phenomenon referred to as *tottering* by Mahé and Vert (2006)), we further restrain the set of walks; that is, for any positive integer β, we define β-walks as walks such that any $\beta+1$ successive vertices are distinct (1-walks are regular walks); see examples in Figure 1. Note that when the graph G is a tree (no cycles), then the set of 2-walks is equal to the set of paths. More generally, for any graph, β-walks of length $\beta + 1$ are exactly paths of length $\beta+1$. Note that the integer β corresponds to the "memory"

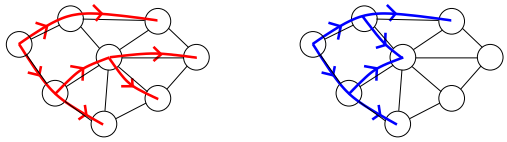


Figure 2. (left) binary 2-tree-walk, which in fact a subtree, (right) binary 1-tree-walk which is not a 2-tree-walk.

of the walk, i.e., the number of past vertices it needs to remember before going on.

A subtree of G is a subgraph of G with no cycles. A subtree of G can thus be seen as a connected subset of distinct nodes of G with an underlying tree structure. The notion of walk is extending the notion of path by allowing nodes to be equal; similarly, we can extend the notion of subtrees to *tree-walks*, which can have nodes that are equal. More precisely, we define an α-ary tree-walk of depth γ of G as a rooted labelled α-ary tree of depth γ with nodes labelled by vertices in G, and such that the labels of neighbors in the tree-walk must be neighbors in G (we refer to all allowed such set of labels as *consistent* labels). We assume that the tree-walks are not necessarily complete trees, i.e., each node may have less than α children. Tree-walks can be plotted on top of the original graph, as shown in Figure 2, and may be represented by a tree structure T over the vertex set $\{1, \dots, |T|\}$ and a tuple of consistent but possibly non distinct labels $I \in V^{|T|}$ (i.e., the labels of neighboring vertices in T must be neighboring vertices in G). Finally, in this paper, we consider only rooted subtrees, i.e., subtrees where a specific node is identified as the root; moreover, all the trees that we consider are unordered trees (i.e., no order is considered among siblings).

We can also define β-tree-walks, as tree-walks such that for each node in T, its label (which is an element of the original vertex set V) and the ones of all its descendants up to the β-th generation are all distinct. With that definition, 1-tree-walks are regular tree-walks (see Figure 2), and if $\alpha = 1$, we get back β-walks. From now on, we refer to the descendants up to the β-th generation as the β-descendants.

We let denote $\mathcal{T}_{\alpha,\gamma}$ the set of rooted tree structures of depth less than γ and with at most α children per node; for example, $\mathcal{T}_{1,\gamma}$ is exactly the set of chain graphs of length less than γ. For $T \in \mathcal{T}_{\alpha,\gamma}$, we denote $\mathcal{J}_\beta(T, G)$ the set of consistent labellings of T by vertices in V leading to β-tree-walks. With these definitions, a β-tree-walk of G is characterized by (a) a tree structure $T \in \mathcal{T}_{\alpha,\gamma}$ and (b) a labelling $I \in \mathcal{J}_\beta(T, G)$.

2.3. Graph Kernels

We assume that we are given a positive definite kernel between tree-walks that share the same tree structure, which we refer to as the *local kernel*. This kernel depends on the tree structure T and the set of attributes and positions associated with the nodes in the tree-walks (remember that each node of G and H has two labels, a position and an attribute). Given a tree structure T and consistent labellings $I \in \mathcal{J}_\beta(T, G)$ and $J \in \mathcal{J}_\beta(T, H)$, we let denote $q_{T,I,J}(G, H)$ the value of the local kernel between two tree-walks defined by the same structure T and labellings I and J.

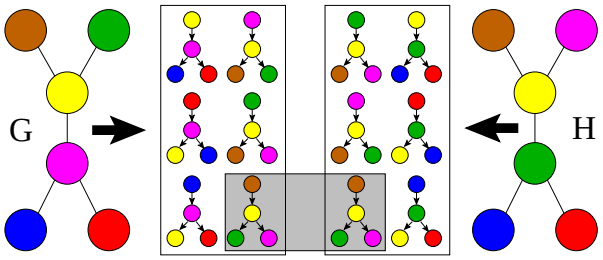


Figure 3. Graph kernels between two graphs (each color represents a different label). We display all binary 1-tree walks with a specific tree structure, extracted from two simple graphs; the graph kernels is computing and summing the local kernels between all those extracted tree-walks. In the case of the Dirac kernel (hard matching), only one pair of tree-walks is matched (for both labels and structures).

Following Ramon and Gärtner (2003), we can define the *tree-kernel* as the sum over all matching tree-walks of G and H of the local kernel, i.e.:

$$k^{\mathcal{T}}_{\alpha,\beta,\gamma}(G, H) = \sum_{T \in \mathcal{T}_{\alpha,\gamma}} f_{\lambda,\nu}(T) \times \sum_{I \in \mathcal{J}_\beta(T,G)} \sum_{J \in \mathcal{J}_\beta(T,H)} q_{T,I,J}(G, H). \quad (1)$$

When considering 1-walks (i.e., $\alpha = \beta = 1$), and letting the maximal walk length γ tend to $+\infty$, we get back the random walk kernel (Ramon & Gärtner, 2003; Kashima et al., 2004). If the kernel $q_{T,I,J}(G, H)$ has nonnegative values and is equal to 1 if the two tree-walks are equal, it can be seen as a soft matching indicator, and then the kernel in Eq. (1) simply counts the softly matched tree-walks in the two graphs (see Figure 3 for an illustration with hard matching).

We add a nonnegative penalization $f_{\lambda,\nu}(T)$ depending only on the tree-structure. Besides the usual penalization of the number of nodes $|T|$, we also add a penalization of the number of leaf nodes $\ell(T)$ (i.e., nodes with no children). More precisely, we use the penalization $f_{\lambda,\nu} = \lambda^{|T|}\nu^{\ell(T)}$. This penalization, suggested by Mahé and Vert (2006), is essential in our situation to avoid that trees with nodes of higher degrees dominate the sum.

If $q_{T,I,J}(G, H)$ is obtained from a positive definite kernel between (labelled) tree-walks, then $k^{\mathcal{T}}_{\alpha,\beta,\gamma}(G, H)$ also defines a positive definite kernel. The kernel $k^{\mathcal{T}}_{\alpha,\beta,\gamma}(G, H)$ sums the *local kernel* $q_{T,I,J}(G, H)$ over all tree-walks of G and H that share the same tree structure; the number

of such matching tree-walks is exponential in the depth γ, thus, in order to deal with potentially deep trees, a recursive definition is needed. As we now detail, it requires a specific type of local kernels, which can be decomposed according to tree structures.

2.4. Local Kernels

The local kernel is used between tree-walks which can have large depths (note that everything we propose will turn out to have linear time complexity in the depth γ). We use the product of a kernel for attributes and a kernel for positions. For attributes, we use the following usual factorized form $q_{\mathcal{A}}(a(I), b(J)) = \prod_{p=1}^{|I|} k_{\mathcal{A}}(a(I_p), b(J_p))$, where $k_{\mathcal{A}}$ is a positive definite kernel on $\mathcal{A} \times \mathcal{A}$. This allows the separate comparison of each matched pair of points and efficient dynamic programming recursions (Harchaoui & Bach, 2007). However, for our local kernel on positions, we need a kernel that *jointly* depends on the whole vectors $x(I) \in \mathcal{X}^{|I|}$ and $y(J) \in \mathcal{X}^{|J|}$, and not only on the p pairs $(x(I_p), y(J_p)) \in \mathcal{X} \times \mathcal{X}$. Indeed, we do not assume that the pairs are *registered*, i.e., we do not know the matching between points indexed by I in the first graph and the ones indexed by J in the second graph.

In this paper, we focus on $\mathcal{X} = \mathbb{R}^d$ and *translation invariant* local kernels, which implies that the local kernel for positions may only depend on differences $x(i) - x(i')$ and $y(j) - y(j')$ for $(i, i') \in I \times I$ and $(j, j') \in J \times J$. We further reduce these to kernel matrices corresponding to a translation invariant positive definite kernel $k_{\mathcal{X}}(x_1 - x_2)$. Depending on the application, $k_{\mathcal{X}}$ may or may not be rotation invariant. In simulations, we use the rotation invariant Gaussian kernel of the form $k_{\mathcal{X}}(x_1, x_2) = e^{-\upsilon \|x_1 - x_2\|^2}$.

Thus, we reduce the set of all positions in $\mathcal{X}^{|V|}$ and $\mathcal{X}^{|W|}$ to full kernel matrices $K \in \mathbb{R}^{|V| \times |V|}$ and $L \in \mathbb{R}^{|W| \times |W|}$ for each graph, defined as $K(v, v') = k_{\mathcal{X}}(x(v) - x(v'))$ (and similarly for L). These matrices are by construction symmetric positive semi-definite and, for simplicity, we assume that these matrices are positive definite (i.e., invertible), which can be enforced by adding a multiple of the identity matrix. The local kernel will thus only depend on the submatrices $K_I = K_{I,I}$ and $L_J = L_{J,J}$, which are positive definite matrices. Note that we use kernel matrices K and L to represent the geometry of each graph, and that we use a positive definite kernel on such kernel matrices.

We consider the following positive definite kernel on positive matrices K and L, the (squared) Bhattacharyya kernel $k_{\mathcal{B}}$, defined as (Kondor & Jebara, 2003):

$$k_{\mathcal{B}}(K, L) = |K|^{1/2} |L|^{1/2} \left| \tfrac{K+L}{2} \right|^{-1}, \qquad (2)$$

where $|K|$ denotes the determinant of K.

By taking the product of the attribute-based local kernel and the position-based local kernel, we get the following local kernel $q^0_{T,I,J}(G, H) = k_{\mathcal{B}}(K_I, L_J) q_{\mathcal{A}}(a(I), b(J))$. However, this local kernel $q^0_{T,I,J}(G, H)$ does not yet depend on the tree structure T and the recursion may be efficient only if $q^0_{T,I,J}(G, H)$ can be computed recursively. The factorized term $q_{\mathcal{A}}(a(I), b(J))$ does not cause any problems; however, for the term $k_{\mathcal{B}}(K_I, L_J)$, we need an approximation based on T. As we show in Section 3, this can be obtained by a factorization according to the appropriate graphical model, i.e., we will replace each kernel matrix of the form K_I by a projection onto a subset of kernel matrices which allow efficient recursions.

3. Positive Matrices and Graphical Models

The main idea underlying the factorization of the kernel is to consider symmetric positive definite matrices as covariance matrices and to look at probabilistic graphical models defined for Gaussian random vectors with those covariance matrices. The goal of this section is to show that by appropriate graphical model techniques, we can design properly factorized approximations of Eq. (2), namely through Eq. (6) and Eq. (7).

More precisely, we assume that we have n random variables $Z_1, \dots, Z_n$ with probability distribution $p(z) = p(z_1, \dots, z_n)$. Given a kernel matrix K (in our case defined as $K_{ij} = e^{-\upsilon \|x_i - x_j\|^2}$, for positions $x_1, \dots, x_n$), we consider jointly Gaussian distributed random variables $Z_1, \dots, Z_n$ such that $\operatorname{cov}(Z_i, Z_j) = K_{ij}$. In this section, with this identification, we consider covariance matrices as kernel matrices, and vice-versa.

3.1. Graphical Models and Junction Trees

Graphical models provide a flexible and intuitive way of defining factorized probability distributions. Given any undirected graph Q with vertices in $\{1, \dots, n\}$, the distribution $p(z)$ is said to factorize in Q if it can be written as a product of potentials over all cliques (completely connected subgraphs) of the graph Q. When the distribution is Gaussian with covariance matrix $K \in \mathbb{R}^{n \times n}$, the distribution factorizes if and only if $(K^{-1})_{ij} = 0$ for each (i, j) which is not an edge in Q (Lauritzen, 1996).

In this paper, we only consider *decomposable* graphical models, for which the graph Q is *triangulated* (i.e., there exists no chordless cycle of length strictly larger than 3). In this case, the joint distribution is uniquely defined from its marginals $p_C(z_C)$ on the cliques C of the graph Q. Namely, if $\mathcal{C}(Q)$ is the set of maximal cliques of Q, we can build a tree of cliques, a *junction tree*, such that $p(z) = \prod_{C \in \mathcal{C}(Q)} p_C(z_C) / \prod_{C, C' \in \mathcal{C}(Q), C \sim C'} p_{C \cap C'}(z_{C \cap C'})$ (see Figure 4 for an example of a graphical model and a junction tree). The sets $C \cap C'$ are usually referred to as *separators* and we let denote $\mathcal{S}(Q)$ the set of such separators. Note that for a zero mean normally distributed vector, the marginals

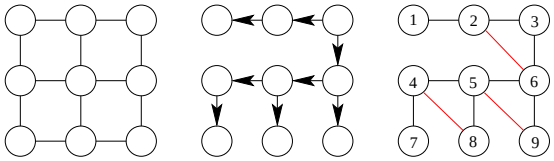


Figure 4. (left) original graph, (middle) a single extracted tre-walk, (right) decomposable graphical model $Q_1(T)$ with added edges in red, defined in Section 3.4. The junction tree is a chain composed of the cliques $\{1,2\},\{2,3,6\},\{5,6,9\},\{4,5,8\},\{4,7\}$.

$p_C(z_C)$ are characterized by the marginal covariance matrix $K_C = K_{C,C}$. Projecting onto a graphical model will preserve the marginal over all maximal cliques, and thus preserve the local kernel matrices, while imposing zeros in the inverse of K.

3.2. Graphical Models and Projections

We let denote $\Pi_Q(K)$ the covariance matrix that factorizes in Q which is closest to K for the Kullback-Leibler divergence between normal distributions. In this paper, we essentially replace K by $\Pi_Q(K)$; i.e., we project all our covariance matrices onto a graphical model, which is a classical tool in probabilistic modelling (Lauritzen, 1996). We leave the study of the approximation properties of such a projection (i.e., for a given K, how dense the graph should be to approximate the full local kernel correctly?) to future work—see, e.g., Caetano et al. (2006) for related results.

Practically, since our kernel on kernel matrices involves determinants, we simply need to compute $|\Pi_Q(K)|$ efficiently. For decomposable graphical models, $\Pi_Q(K)$ can be obtained in closed form (Lauritzen, 1996) and its determinant has the following simple expression:

$$\log|\Pi_Q(K)| = \sum_{C\in\mathcal{C}(Q)} \log|K_C| - \sum_{S\in\mathcal{S}(Q)} \log|K_S|. \quad (3)$$

The determinant $|\Pi_Q(K)|$ is thus a ratio of terms (determinants over cliques and separators), which will restrict the applicability of the projected kernels (see Proposition 1). In order to keep only products, we consider the following equivalent form: if the junction tree is rooted (by choosing any clique as the root), then for each clique but the root, a unique parent clique is defined, and we have:

$$\begin{aligned}\log|\Pi_Q(K)| &= \textstyle\sum_{C\in\mathcal{C}(Q)} \log\frac{|K_C|}{|K_{p_Q(C)}|} \\ &= \textstyle\sum_{C\in\mathcal{C}(Q)} \log|K_{C|p_Q(C)}|, \end{aligned} \quad (4)$$

where $p_Q(C)$ is the parent clique of Q (and $\varnothing$ for the root clique) and the conditional covariance matrix is defined, as usual, as $K_{C|p_Q(C)} = K_{C,C} - K_{C,p_Q(C)}K^{-1}_{p_Q(C),p_Q(C)}K_{p_Q(C),C}$ (Lauritzen, 1996).

3.3. Graphical Models and Kernels

We now propose several ways of defining a kernel adapted to graphical models. All of them are based on replacing determinants $|M|$ by $|\Pi_Q(M)|$, and their different decompositions in Eq. (3) and Eq. (4). Simply using Eq. (3), we obtain the similarity measure:

$$k^Q_{\mathcal{B},0}(K,L) = \prod_{C\in\mathcal{C}(Q)} k_{\mathcal{B}}(K_C,L_C) \prod_{S\in\mathcal{S}(Q)} k_{\mathcal{B}}(K_S,L_S)^{-1}. \quad (5)$$

which turns out not to be a positive definite kernel for general covariance matrices:

Proposition 1 *For any decomposable model Q, the kernel $k^Q_{\mathcal{B},0}$ defined in Eq. (5) is a positive definite kernel on the set of covariance matrices K such that for all separators $S\in\mathcal{S}(Q)$, $K_{S,S} = I$. In particular, when all separators have cardinal one, this is a kernel on correlation matrices.*

In order to remove the condition on separators (i.e., we want more sharing between cliques than through a single variable), we consider the rooted junction tree representation in Eq. (4). A straightforward kernel is to compute the product of the Bhattacharyya kernels $k_{\mathcal{B}}(K_{C|p_Q(C)}, L_{C|p_Q(C)})$ for each conditional covariance matrix. However, this does not lead to a true distance on covariance matrices that factorize on Q because the set of conditional covariance matrices do not characterize entirely those distributions. Rather, we consider the following kernel:

$$k^Q_{\mathcal{B}}(K,L) = \textstyle\prod_{C\in\mathcal{C}(Q)} k^{C|p_Q(C)}_{\mathcal{B}}(K,L); \quad (6)$$

for the root clique, we define $k^{R|\varnothing}_{\mathcal{B}}(K,L) = k_{\mathcal{B}}(K_R,L_R)$ and the kernels $k^{C|p_Q(C)}_{\mathcal{B}}(K,L)$ are defined as kernels between conditional Gaussian distributions of Z_C given $Z_{p_Q(C)}$. We use

$$k^{C|p_Q(C)}_{\mathcal{B}}(K,L) = \frac{|K_{C|p_Q(C)}|^{1/2}|L_{C|p_Q(C)}|^{1/2}}{\left|\frac{1}{2}K_{C|p_Q(C)}+\frac{1}{2}L_{C|p_Q(C)}+MM^\top\right|}, \quad (7)$$

where the additional term M is equal to $\frac{1}{2}(K_{C,p_Q(C)}K^{-1}_{p_Q(C)} - L_{C,p_Q(C)}L^{-1}_{p_Q(C)})$. This exactly corresponds to putting a prior with identity covariance matrix on variables $Z_{p_Q(C)}$ and considering the kernel between the resulting joint covariance matrices on variables indexed by $(C,p_Q(C))$. We now have a positive definite kernel on all covariance matrices:

Proposition 2 *For any decomposable model Q, the kernel $k^Q_{\mathcal{B}}(K,L)$ defined in Eq. (6) and Eq. (7) is a positive definite kernel on the set of covariance matrices.*

Note that the kernel is not invariant by the choice of the particular root of the junction tree. However, in our setting, this is not an issue because we have a natural way of rooting the junction trees (i.e, following the rooted tree-walk, see Section 3.4). Note that these kernels could be useful in other domains than point clouds and computer vision.

In Section 4, we will use the notation $k_{\mathcal{B}}^{I_1|I_2,J_1|J_2}(K,L)$ for $|I_1| = |I_2|$ and $|J_1| = |J_2|$ to denote the kernel between covariance matrices $K_{I_1\cup I_2}$ and $L_{I_1\cup I_2}$ adapted to the conditional distributions $I_1|I_2$ and $J_1|J_2$, defined through Eq. (7).

3.4. Choice of Graphical Models

Given the rooted tree structure T of a β-tree-walk, we now need to define the graphical model $Q_\beta(T)$ that we use to project our kernel matrices. A natural candidate is T itself; however, as shown in Section 4, in order to compute efficiently the kernel we simply need that the local kernel is a product of terms that only involve a node and its β-descendants. The densest graph (remember that denser graphs lead to better approximations when projecting onto the graphical model) we may use is exactly the following: we define $Q_\beta(T)$ such that for all nodes in T, the node together with all its β-descendants form a clique, i.e., a node is connected to its β-descendants and all β-descendants are also mutually connected (see Figure 4 for example for $\beta = 1$): the set of cliques are thus the set of *families* of depth $\beta+1$ (i.e., with $\beta+1$ generations). Thus, our final kernel is:

$$k_{\alpha,\beta,\gamma}^{\mathcal{T}}(G,H) = \sum_{T\in\mathcal{T}_{\alpha,\gamma}} f_{\lambda,\nu}(T)\times$$
$$\sum_{I\in\mathcal{J}_\beta(T,G)}\ \sum_{J\in\mathcal{J}_\beta(T,H)} k_{\mathcal{B}}^{Q_\beta(T)}(K_I,L_J)q_{\mathcal{A}}(a(I),b(J)). \quad (8)$$

The main intuition behind this definition is to sum local similarities over all matching subgraphs. In order to obtain a tractable formulation, we simply needed (a) to extend the set of subgraphs (to tree-walks of depth γ) and (b) to factorize the local similarities along the graphs. We now show how these elements can be combined to derive efficient recursions.

4. Dynamic Programming Recursions

In order to derive dynamic programming recursions, we follow Mahé and Vert (2006) and rely on the fact that α-ary β-tree-walks of G can essentially be defined through 1-tree-walks on the augmented graph of all rooted subtrees of G of depth at most β and arity less than α. We thus consider the set $V_{\alpha,\beta}$ of non complete rooted (unordered) subtrees of $G = (V,E)$, of depths less than β and arity less than α. Given two different rooted unordered labelled trees, they are said *equivalent* (or isomorphic) if they share the same tree structure, and this is denoted $\sim_t$.

On this set $V_{\alpha,\beta}$, we define a *directed* graph with edge set $E_{\alpha,\beta}$ as follows: $R_0 \in V_{\alpha,\beta}$ is connected to $R_1 \in V_{\alpha,\beta}$ if "the tree R_1 extends the tree R_0 one generation further", i.e., if and only if (a) the first $\beta-1$ generations of R_1 are exactly equal to one of the complete subtree of R_0 rooted at a child of the root of R_0, and (b) the nodes of depth β of R_1 are distinct from the nodes in R_0. This defines a graph $G_{\alpha,\beta} = (V_{\alpha,\beta}, E_{\alpha,\beta})$ and a neighborhood $\mathcal{N}_{G_{\alpha,\beta}}(R)$ for $R \in V_{\alpha,\beta}$ (see Figure 5 for an example). Similarly we define a graph $H_{\alpha,\beta} = (W_{\alpha,\beta}, F_{\alpha,\beta})$ for the graph H. Note that when $\alpha = 1$, $V_{1,\beta}$ is the set of paths of length less than or equal to β.

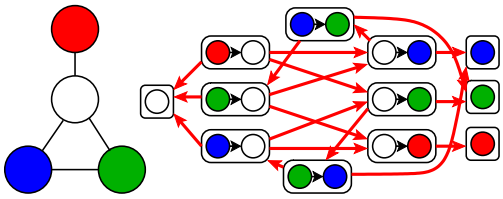


Figure 5. (left) undirected graph G, (right) graph $G_{1,2}$.

For a β-tree-walk, the root with its β-descendants must have distinct vertices and thus corresponds exactly to an element of $V_{\alpha,\beta}$. We denote $k_{\alpha,\beta,\gamma}^{\mathcal{T}}(G,H,R_0,S_0)$ the same kernel as defined in Eq. (8), but restricted to tree-walks that start respectively with R_0 and S_0. Note that if R_0 and S_0 are not equivalent, then $k_{\alpha,\beta,\gamma}^{\mathcal{T}}(G,H,R_0,S_0) = 0$.

We obtain the following recursion between depths γ and depth $\gamma-1$, for all $R_0 \in V_{\alpha,\beta}$ and and $S_0 \in W_{\alpha,\beta}$ such that $R_0 \sim_t S_0$:

$$k_{\alpha,\beta,\gamma}^{\mathcal{T}}(G,H,R_0,S_0) = k_{\alpha,\beta,\gamma-1}^{\mathcal{T}}(G,H,R_0,S_0)$$
$$+\sum_{p=1}^{\alpha} \sum_{\substack{R_1,\dots,R_p \in \mathcal{N}_{G_{\alpha,\beta}}(R_0) \\ R_1,\dots,R_p \text{ disjoint}}} \sum_{\substack{S_1,\dots,S_p \in \mathcal{N}_{H_{\alpha,\beta}}(S_0) \\ S_1,\dots,S_p \text{ disjoint}}}$$
$$\left[\lambda \prod_{i=1}^{p} k_{\mathcal{A}}(a(\mathrm{root}(R_i)),b(\mathrm{root}(S_i)))\times \right.$$
$$\left. \frac{k_{\mathcal{B}}^{\cup_{i=1}^p R_i|R_0,\cup_{i=1}^p S_i|S_0}(K,L)}{\prod_{i=1}^p k_{\mathcal{B}}^{R_i,S_i}(K,L)} \left(\prod_{i=1}^{p} k_{\alpha,\beta,\gamma-1}^{\mathcal{T}}(G,H,R_i,S_i)\right)\right].$$

Note that if any of the trees R_i is not equivalent to S_i, it does not contribute to the sum. The recursion is initialized with $k_{\alpha,\beta,\gamma}^{\mathcal{T}}(G,H,R_0,S_0) = \lambda^{|R_0|}\nu^{\ell(R_0)}q_{\mathcal{A}}(a(R_0),b(S_0))k_{\mathcal{B}}(K_{R_0},L_{S_0})$ while the final kernel is obtained by summing over all R_0 and S_0, i.e, $k_{\alpha,\beta,\gamma}^{\mathcal{T}}(G,H) = \sum_{R_0\sim_t S_0} k_{\alpha,\beta,\gamma}^{\mathcal{T}}(G,H,R_0,S_0)$.

Computational Complexity The complexity of computing one kernel between two graphs is linear in γ (the depth of the tree-walks), and quadratic in the size of $V_{\alpha,\beta}$ and $W_{\alpha,\beta}$. However, those sets may have exponential size in β and α in general (in particular if graphs are densely connected). And thus, we are limited to small values (typically $\alpha \leqslant 3$ and $\beta \leqslant 6$) which are sufficient for good classification performance (in particular, higher β or α do not necessarily mean better performance, see Section 5). Overall, one can deal with any graph size, as long as the "sufficient statistics" (i.e., the unique local neighorhoods in $V_{\alpha,\beta}$) are not too numerous.

Figure 6. For digits and Chinese characters: (left) original characters, (right) thinned and subsampled characters.

For example, for the handwritten digits we use in simulations, the average number of nodes in the graphs is 18 ± 4, while the average cardinal of $V_{\alpha,\beta}$ and running times[1] for one kernel evaluation are, for walk kernels of depth 24: $|V_{\alpha,\beta}| = 36$, $T = 2$ ms ($\alpha = 1$, $\beta = 2$), $|V_{\alpha,\beta}| = 37$, $T = 3$ ms ($\alpha = 1$, $\beta = 4$); and for tree-kernels: $|V_{\alpha,\beta}| = 56$, $T = 25$ ms ($\alpha = 2$, $\beta = 2$), $|V_{\alpha,\beta}| = 70$, $T = 32$ ms ($\alpha = 2$, $\beta = 4$).

Finally, we may reduce the computational load by considering a set of trees of smaller arity in the previous recursions; i.e., we can consider $V_{1,\beta}$ instead of $V_{\alpha,\beta}$ with tree-kernels of arity $\alpha > 1$.

5. Application to Character Recognition

We have tested our new kernels on the task of isolated handwritten character recognition, handwritten arabic numerals (MNIST dataset) and Chinese characters (ETL9B dataset). We selected the first 100 examples for the ten classes in the MNIST dataset, while for the ETL9B dataset, we selected the five hardest classes to discriminate among 3,000 classes (by computing distances between class means) and then selected the first 50 examples per class. Our learning task it to classify those characters; we use a one-vs-rest multiclass scheme with 1-norm support vector machines (see, e.g., Shawe-Taylor and Cristianini (2004)).

We consider characters as drawings in $\mathbb{R}^2$, which are sets of possibly intersecting contours. Those are naturally represented as undirected planar graphs. We have thinned and subsampled uniformly each character to reduce the sizes of the graphs (see two examples in Figure 6).

The kernel on positions is $k_{\mathcal{X}}(x, y) = \exp(-\tau\|x-y\|^2) + \kappa\delta(x, y)$, but could take into account different weights on horizontal and vertical directions. We add the positions from the center of the bounding box as features, to take into account the global positions, i.e., we use $k_{\mathcal{A}}(x, y) = \exp(-\upsilon\|x-y\|^2)$. This is necessary because the problem of handwritten character recognition is not globally translation invariant.

[1]Those do not take into account preprocessing and were evaluated on an Intel Xeon 2.33 GHz processor from MATLAB/C code, and are to be compared to the simplest recursions which correspond to the usual random walk kernel ($\alpha = 1$, $\beta = 1$), where $T = 1$ ms.

In this paper, we have defined a family of kernels, corresponding to different values of the following free parameters (shown with their possible values): arity of tree-walks ($\alpha = 1, 2$), order of tree-walks ($\beta = 1, 2, 4, 6$), depth of tree-walks ($\gamma = 1, 2, 4, 8, 16, 24$), penalization on number of nodes ($\lambda = 1$), penalization on number of leaf nodes ($\nu = .1, .01$), bandwidth for kernel on positions ($\tau = .05, .01, .1$), ridge parameter ($\kappa = .001$), bandwidth for kernel on attributes ($\upsilon = .05, .01, .1$).

The first two sets of parameters $(\alpha, \beta, \gamma, \lambda, \nu)$ are parameters of the graph kernel, independent of the application, while the last set (τ, κ, ν) are parameters of the kernels for attributes and positions. Note that with only a few important scale parameters (τ and ν), we are able to characterize complex interactions between the vertices and edges of the graphs. In practice, this is important to avoid considering many more distinct parameters for all sizes and topologies of subtrees.

In simulations, we performed two loops of 5-fold cross-validation: in the outer loop, we consider 5 different training folds with their corresponding testing folds. On each training fold, we consider all possible values of α and β. For all of those values, we select all other parameters (including the regularization parameters of the SVM) by 5-fold cross-validation (the inner folds). Once the best parameters are found only by looking only at the training fold, we train on the whole training fold, and test on the testing fold. We output the means and standard deviations of the testing errors for each testing fold. We show in Figure 7 the performance for various values of α and β. We compare those favorably to three baseline kernels with hyperparameters learned by cross-validation in the same way: (a) the *Gaussian-RBF kernel* on the vectorized original images, which leads to testing errors of $11.6 \pm 5.4\%$ (MNIST) and $50.4 \pm 6.2\%$ (ETL9B); (b) the regular *random walk kernel* which sums over all walk lengths, which leads to testing errors of $8.6 \pm 1.3\%$ (MNIST) and $34.8 \pm 8.4\%$ (ETL9B); and (c) the *pyramid match kernel* (Grauman & Darrell, 2007), which is commonly used for image classification and leads here to testing errors of $10.8 \pm 3.6\%$ (MNIST) and $45.2 \pm 3.4\%$ (ETL9B).

These results show that our new family of kernels that use the natural structure of line drawings are outperforming other kernels on structured data (regular random walk kernel and pyramid match kernel) as well as the "blind" Gaussian-RBF kernel which does not take into account explicitly the structure of images but still leads to very good performance with more training data (LeCun et al., 1998). Note that for arabic numerals, higher arity does not help, which is not surprising since most digits have a linear structure (i.e, graphs are chains). On the contrary, for Chinese characters, which exhibit higher connectivity, best performance is achieved for binary tree-walks.

	MNIST $\alpha = 1$	MNIST $\alpha = 2$	ETL9B $\alpha = 1$	ETL9B $\alpha = 2$
$\beta = 1$	11.6 ± 4.6	9.2 ± 3.9	36.8 ± 4.6	32 ± 8.4
$\beta = 2$	5.6 ± 3.1	5.6 ± 3.0	29.2 ± 8.8	$\mathbf{25.2 \pm 2.7}$
$\beta = 4$	$\mathbf{5.4 \pm 3.6}$	$\mathbf{5.4 \pm 3.1}$	32.4 ± 3.9	29.6 ± 4.3
$\beta = 6$	5.6 ± 3.3	6 ± 3.5	29.6 ± 4.6	28.4 ± 4.3

Figure 7. Error rates (multiplied by 100) on handwritten character classification tasks.

6. Conclusion

We have presented a new kernel for point clouds which is based on comparisons of local subsets of the point clouds. Those comparisons are made tractable by (a) considering subsets based on tree-walks and walks, and (b) using a specific factorized form for the local kernels between tree-walks, namely a factorization on a properly defined probabilistic graphical model.

Moreover, we have reported applications to handwritten character recognition where we showed that the kernels were able to capture the relevant information to allow good predictions from few training examples. We are currently investigating other domains of applications of points clouds, such as shape mining in computer vision (Belongie et al., 2002), and prediction of protein functions from their three-dimensional structures (Qiu et al., 2007).

Acknowledgements

We would like to thank Zaïd Harchaoui and Jean-Philippe Vert for fruitful discussions related to this work.

References

Bach, F. R., Lanckriet, G. R. G., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. *Proc. ICML*.

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Trans. PAMI, 24*, 509–522.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., & Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics, 21*.

Caetano, T., Caelli, T., Schuurmans, D., & Barone, D. (2006). Graphical models and point pattern matching. *IEEE Trans. PAMI, 28*, 1646–1663.

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.). (2006). *Semi-supervised learning (adaptive computation and machine learning)*. MIT Press.

Diestel, R. (2005). *Graph theory*. Springer-Verlag.

Forsyth, D. A., & Ponce, J. (2003). *Computer vision: A modern approach*. Prentice Hall.

Fröhlich, H., Wegner, J. K., Sieker, F., & Zell, A. (2005). Optimal assignment kernels for attributed molecular graphs. *Proc. ICML*.

Grauman, K., & Darrell, T. (2007). The pyramid match kernel: Efficient learning with sets of features. *J. Mach. Learn. Res., 8*, 725–760.

Harchaoui, Z., & Bach, F. (2007). Image classification with segmentation graph kernels. *Proc. CVPR*.

Kashima, H., Tsuda, K., & Inokuchi, A. (2004). Kernels for graphs. *Kernel Methods in Comp. Biology*. MIT Press.

Kondor, R. I., & Jebara, T. (2003). A kernel between sets of vectors. *Proc. ICML*.

Lauritzen, S. (1996). *Graphical models*. Oxford U. Press.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE, 86*, 2278–2324.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *J. Mach. Learn. Res., 2*, 419–444.

Mahé, P., & Vert, J.-P. (2006). *Graph kernels based on tree patterns for molecules* (Tech. report HAL-00095488).

Neuhaus, M., & Bunke, H. (2006). Edit distance based kernel functions for structural pattern classification. *Pattern Recognition, 39*, 1852–1863.

Parsana, M., Bhattacharyya, C., Bhattacharya, S., & Ramakrishnan, K. R. (2008). Kernels on attributed pointsets with applications. *Adv. NIPS*.

Qiu, J., Hue, M., Ben-Hur, A., Vert, J.-P., & Noble, W. S. (2007). A structural alignment kernel for protein structures. *Bioinformatics, 23*, 1090–1098.

Ramon, J., & Gärtner, T. (2003). Expressivity versus efficiency of graph kernels. *First International Workshop on Mining Graphs, Trees and Sequences*.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge Univ. Press.

Srihari, S. N., Yang, X., & Ball, G. R. (2007). Offline Chinese handwriting recognition: A survey. *Frontiers of Computer Science in China*.

Vert, J.-P. (2008). *The optimal assignment kernel is not positive definite* (Tech. report HAL-00218278).

Vert, J.-P., Saigo, H., & Akutsu, T. (2004). Local alignment kernels for biological sequences. *Kernel Methods in Comp. Biology*. MIT Press.

Vishwanathan, S. V. N., Borgwardt, K. M., & Schraudolph, N. (2007). Fast computation of graph kernels. *Adv. NIPS*.

Bolasso: Model Consistent Lasso Estimation through the Bootstrap

Francis R. Bach FRANCIS.BACH@MINES.ORG
INRIA - WILLOW Project-Team, Laboratoire d'Informatique de l'Ecole Normale Supérieure, Paris, France

Abstract

We consider the least-square linear regression problem with regularization by the ℓ_1-norm, a problem usually referred to as the Lasso. In this paper, we present a detailed asymptotic analysis of model consistency of the Lasso. For various decays of the regularization parameter, we compute asymptotic equivalents of the probability of correct model selection (i.e., variable selection). For a specific rate decay, we show that the Lasso selects all the variables that should enter the model with probability tending to one exponentially fast, while it selects all other variables with strictly positive probability. We show that this property implies that if we run the Lasso for several bootstrapped replications of a given sample, then intersecting the supports of the Lasso bootstrap estimates leads to consistent model selection. This novel variable selection algorithm, referred to as the Bolasso, is compared favorably to other linear regression methods on synthetic data and datasets from the UCI machine learning repository.

1. Introduction

Regularization by the ℓ_1-norm has attracted a lot of interest in recent years in machine learning, statistics and signal processing. In the context of least-square linear regression, the problem is usually referred to as the *Lasso* (Tibshirani, 1994). Much of the early effort has been dedicated to algorithms to solve the optimization problem efficiently. In particular, the *Lars* algorithm of Efron et al. (2004) allows to find the entire regularization path (i.e., the set of solutions for all values of the regularization parameters) at the cost of a single matrix inversion.

Moreover, a well-known justification of the regularization by the ℓ_1-norm is that it leads to *sparse* solutions, i.e., loading vectors with many zeros, and thus performs model selection. Recent works (Zhao & Yu, 2006; Yuan & Lin, 2007; Zou, 2006; Wainwright, 2006) have looked precisely at the model consistency of the Lasso, i.e., if we know that the data were generated from a sparse loading vector, does the Lasso actually recover the sparsity pattern when the number of observed data points grows? In the case of a fixed number of covariates, the Lasso does recover the sparsity pattern if and only if a certain simple condition on the generating covariance matrices is verified (Yuan & Lin, 2007). In particular, in low correlation settings, the Lasso is indeed consistent. However, in presence of strong correlations between relevant variables and irrelevant variables, the Lasso cannot be consistent, shedding light on potential problems of such procedures for variable selection. Adaptive versions where data-dependent weights are added to the ℓ_1-norm then allow to keep the consistency in all situations (Zou, 2006).

In this paper, we first derive a detailed asymptotic analysis of sparsity pattern selection of the Lasso estimation procedure, that extends previous analysis (Zhao & Yu, 2006; Yuan & Lin, 2007; Zou, 2006), by focusing on a specific decay of the regularization parameter. Namely, we show that when the decay is proportional to $n^{-1/2}$, where n is the number of observations, then the Lasso will select all the variables that should enter the model (the *relevant* variables) with probability tending to one exponentially fast with n, while it selects all other variables (the *irrelevant* variables) with strictly positive probability. If several datasets generated from the same distribution were available, then the latter property would suggest to consider the intersection of the supports of the Lasso estimates for each dataset: all relevant variables would always be selected for all datasets, while irrelevant variables would enter the models randomly, and intersecting the supports from sufficiently many different datasets would simply eliminate them. However, in practice, only one dataset is given; but resampling methods such as the *bootstrap* are exactly dedicated to mimic the availability of several datasets by resampling from the same unique dataset (Efron & Tibshirani, 1998). In this paper, we show that when using the bootstrap and intersecting the supports, we actually get a consistent

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

model estimate, *without* the consistency condition required by the regular Lasso. We refer to this new procedure as the *Bolasso* (**bo**otstrap-enhanced **l**east **a**bs**o**lute **s**hrinkage **o**perator). Finally, our Bolasso framework could be seen as a voting scheme applied to the supports of the bootstrap Lasso estimates; however, our procedure may rather be considered as a consensus combination scheme, as we keep the (largest) subset of variables on which *all* regressors agree in terms of variable selection, which is in our case provably consistent and also allows to get rid of a potential additional hyperparameter.

The paper is organized as follows: in Section 2, we present the asymptotic analysis of model selection for the Lasso; in Section 3, we describe the Bolasso framework, while in Section 4, we illustrate our results on synthetic data, where the true sparse generating model is known, and data from the UCI machine learning repository. Sketches of proofs can be found in Appendix A.

Notations For a vector $v \in \mathbb{R}^p$, we denote $\|v\|_2 = (v^\top v)^{1/2}$ its ℓ_2-norm, $\|v\|_\infty = \max_{i\in\{1,\dots,p\}} |v_i|$ its ℓ_∞-norm and $\|v\|_1 = \sum_{i=1}^p |v_i|$ its ℓ_1-norm. For $a \in \mathbb{R}$, $\operatorname{sign}(a)$ denotes the sign of a, defined as $\operatorname{sign}(a) = 1$ if $a > 0$, -1 if $a < 0$, and 0 if $a = 0$. For a vector $v \in \mathbb{R}^p$, $\operatorname{sign}(v) \in \mathbb{R}^p$ denotes the the vector of signs of elements of v.

Moreover, given a vector $v \in \mathbb{R}^p$ and a subset I of $\{1,\dots,p\}$, v_I denotes the vector in $\mathbb{R}^{\operatorname{Card}(I)}$ of elements of v indexed by I. Similarly, for a matrix $A \in \mathbb{R}^{p\times p}$, $A_{I,J}$ denotes the submatrix of A composed of elements of A whose rows are in I and columns are in J.

2. Asymptotic Analysis of Model Selection for the Lasso

In this section, we describe existing and new asymptotic results regarding the model selection capabilities of the Lasso.

2.1. Assumptions

We consider the problem of predicting a response $Y \in \mathbb{R}$ from covariates $X = (X_1,\dots,X_p)^\top \in \mathbb{R}^p$. The only assumptions that we make on the joint distribution P_{XY} of (X,Y) are the following:

(**A1**) The cumulant generating functions $\mathbb{E}\exp(s\|X\|_2^2)$ and $\mathbb{E}\exp(sY^2)$ are finite for some $s > 0$.

(**A2**) The joint matrix of second order moments $\mathbf{Q} = \mathbb{E}XX^\top \in \mathbb{R}^{p\times p}$ is invertible.

(**A3**) $\mathbb{E}(Y|X) = X^\top\mathbf{w}$ and $\operatorname{var}(Y|X) = \sigma^2$ a.s. for some $\mathbf{w} \in \mathbb{R}^p$ and $\sigma \in \mathbb{R}_+^*$.

We let denote $\mathbf{J} = \{j, \mathbf{w}_j \neq 0\}$ the sparsity pattern of $\mathbf{w}$, $\mathbf{s} = \operatorname{sign}(\mathbf{w})$ the sign pattern of $\mathbf{w}$, and $\varepsilon = Y - X^\top\mathbf{w}$ the additive noise.[1] Note that our assumption regarding cumulant generating functions is satisfied when X and ε have compact supports, and also when the densities of X and ε have light tails.

We consider *independent and identically distributed* (i.i.d.) data $(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}$, $i = 1,\dots,n$, sampled from P_{XY}; the data are given in the form of matrices $\overline{Y} \in \mathbb{R}^n$ and $\overline{X} \in \mathbb{R}^{n\times p}$.

Note that the i.i.d. assumption, together with (**A1-3**), are the simplest assumptions for studying the asymptotic behavior of the Lasso; and it is of course of interest to allow more general assumptions, in particular growing number of variables p, more general random variables, etc., which are outside the scope of this paper—see, e.g., Meinshausen and Yu (2008); Zhao and Yu (2006); Lounici (2008).

2.2. Lasso Estimation

We consider the square loss function $\frac{1}{2n}\sum_{i=1}^n (y_i - w^\top x_i)^2 = \frac{1}{2n}\|\overline{Y} - \overline{X}w\|_2^2$ and the regularization by the ℓ_1-norm defined as $\|w\|_1 = \sum_{i=1}^p |w_i|$. That is, we look at the following Lasso optimization problem (Tibshirani, 1994):

$$\min_{w\in\mathbb{R}^p} \tfrac{1}{2n}\|\overline{Y} - \overline{X}w\|_2^2 + \mu_n\|w\|_1, \qquad (1)$$

where $\mu_n \geqslant 0$ is the regularization parameter. We denote $\hat{w}$ any global minimum of Eq. (1)—it may not be unique in general, but will with probability tending to one exponentially fast under assumption (**A2**).

2.3. Model Consistency - General Results

In this section, we detail the asymptotic behavior of the Lasso estimate $\hat{w}$, both in terms of the difference in norm with the population value $\mathbf{w}$ (i.e., regular consistency) and of the *sign pattern* $\operatorname{sign}(\hat{w})$, for all asymptotic behaviors of the regularization parameter μ_n. Note that information about the sign pattern includes information about the *support*, i.e., the indices i for which $\hat{w}_i$ is different from zero; moreover, when $\hat{w}$ is consistent, consistency of the sign pattern is in fact equivalent to the consistency of the support.

We now consider five mutually exclusive possible situations which explain various portions of the regularization path (we assume (**A1-3**)); many of these results appear elsewhere (Yuan & Lin, 2007; Zhao & Yu, 2006; Fu & Knight, 2000; Zou, 2006; Bach, 2008; Lounici, 2008) but some of the finer results presented below are new (see Section 2.4).

[1] Throughout this paper, we use boldface fonts for population quantities.

1. If μ_n tends to infinity, then $\hat{w} = 0$ with probability tending to one.

2. If μ_n tends to a finite strictly positive constant μ_0, then $\hat{w}$ converges in probability to the unique global minimum of $\frac{1}{2}(w - \mathbf{w})^\top \mathbf{Q}(w - \mathbf{w}) + \mu_0 \|w\|_1$. Thus, the estimate $\hat{w}$ never converges in probability to $\mathbf{w}$, while the sign pattern tends to the one of the previous global minimum, which may or may not be the same as the one of $\mathbf{w}$.[2]

3. If μ_n tends to zero slower than $n^{-1/2}$, then $\hat{w}$ converges in probability to $\mathbf{w}$ (regular consistency) and the sign pattern converges to the sign pattern of the global minimum of $\frac{1}{2} v^\top \mathbf{Q} v + v_{\mathbf{J}}^\top \operatorname{sign}(\mathbf{w}_{\mathbf{J}}) + \|v_{\mathbf{J}^c}\|_1$. This sign pattern is equal to the population sign vector $\mathbf{s} = \operatorname{sign}(\mathbf{w})$ if and only if the following consistency condition is satisfied:

$$\|\mathbf{Q}_{\mathbf{J}^c\mathbf{J}} \mathbf{Q}_{\mathbf{J}\mathbf{J}}^{-1} \operatorname{sign}(\mathbf{w}_{\mathbf{J}})\|_\infty \leqslant 1. \tag{2}$$

   Thus, if Eq. (2) is satisfied, the probability of correct sign estimation is tending to one, and to zero otherwise (Yuan & Lin, 2007).

4. If $\mu_n = \mu_0 n^{-1/2}$ for $\mu_0 \in (0, \infty)$, then the sign pattern of $\hat{w}$ agrees on $\mathbf{J}$ with the one of $\mathbf{w}$ with probability tending to one, while for all sign patterns consistent on $\mathbf{J}$ with the one of $\mathbf{w}$, the probability of obtaining this pattern is tending to a limit in $(0, 1)$ (in particular strictly positive); that is, all patterns consistent on $\mathbf{J}$ are possible with positive probability. See Section 2.4 for more details.

5. If μ_n tends to zero faster than $n^{-1/2}$, then $\hat{w}$ is consistent (i.e., converges in probability to $\mathbf{w}$) but the support of $\hat{w}$ is equal to $\{1, \dots, p\}$ with probability tending to one (the signs of variables in $\mathbf{J}^c$ may be negative or positive). That is, the ℓ_1-norm has no sparsifying effect.

Among the five previous regimes, the only ones with consistent estimates (in norm) and a sparsity-inducing effect are μ_n tending to zero and $\mu_n n^{1/2}$ tending to a limit $\mu_0 \in (0, \infty]$ (i.e., potentially infinite). When $\mu_0 = +\infty$, then we can only hope for model consistent estimates if the consistency condition in Eq. (2) is satisfied. This somewhat disappointing result for the Lasso has led to various improvements on the Lasso to ensure model consistency even when Eq. (2) is not satisfied (Yuan & Lin, 2007; Zou, 2006). Those are based on adaptive weights based on the non regularized least-square estimate. We propose in Section 3 an alternative way which is based on resampling.

[2] Here and in the third regime, we do not take into account the pathological cases where the sign pattern of the limit in unstable, i.e., the limit is exactly at a hinge point of the regularization path.

In this paper, we now consider the specific case where $\mu_n = \mu_0 n^{-1/2}$ for $\mu_0 \in (0, \infty)$, where we derive new asymptotic results. Indeed, in this situation, we get the correct signs of the relevant variables (those in $\mathbf{J}$) with probability tending to one, but we also get all possible sign patterns consistent with this, i.e., all other variables (those not in $\mathbf{J}$) may be non zero with asymptotically strictly positive probability. However, if we were to repeat the Lasso estimation for many datasets obtained from the same distribution, we would obtain for each μ_0, a set of active variables, all of which include $\mathbf{J}$ with probability tending to one, but potentially containing all other subsets. By intersecting those, we would get exactly $\mathbf{J}$.

However, this requires multiple copies of the samples, which are not usually available. Instead, we consider bootstrapped samples which exactly mimic the behavior of having multiple copies. See Section 3 for more details.

2.4. Model Consistency with Exact Root-n Regularization Decay

In this section we present detailed new results regarding the pattern consistency for μ_n tending to zero exactly at rate $n^{-1/2}$ (see proofs in Appendix A):

Proposition 1 *Assume (**A1-3**) and $\mu_n = \mu_0 n^{-1/2}$, with $\mu_0 > 0$. Then for any sign pattern $s \in \{-1, 0, 1\}^p$ such that $s_{\mathbf{J}} = \operatorname{sign}(\mathbf{w}_{\mathbf{J}})$, $\mathbb{P}(\operatorname{sign}(\hat{w}) = s)$ tends to a limit $\rho(s, \mu_0) \in (0, 1)$, and we have:*

$$\mathbb{P}(\operatorname{sign}(\hat{w}) = s) - \rho(s, \mu_0) = O(n^{-1/2} \log n).$$

Proposition 2 *Assume (**A1-3**) and $\mu_n = \mu_0 n^{-1/2}$, with $\mu_0 > 0$. Then, for any pattern $s \in \{-1, 0, 1\}^p$ such that $s_{\mathbf{J}} \neq \operatorname{sign}(\mathbf{w}_{\mathbf{J}})$, there exist a constant $A(\mu_0) > 0$ such that*

$$\log \mathbb{P}(\operatorname{sign}(\hat{w}) = s) \leqslant -nA(\mu_0) + O(n^{-1/2}).$$

The last two propositions state that we get all relevant variables with probability tending to one *exponentially fast*, while we get exactly get all other patterns with probability tending to a limit *strictly* between zero and one. Note that the results that we give in this paper are valid for *finite* n, i.e., we can derive actual bounds on probability of sign pattern selections with known constants that explictly depend on $\mathbf{w}$, $\mathbf{Q}$ and the joint distribution P_{XY}.

3. Bolasso: Bootstrapped Lasso

Given the n i.i.d. observations $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, $i = 1, \dots, n$, put together into matrices $\overline{X} \in \mathbb{R}^{n \times p}$ and $\overline{Y} \in \mathbb{R}^n$, we consider m *bootstrap* replications of the n data points (Efron & Tibshirani, 1998); that is, for $k = 1, \dots, m$, we consider a *ghost sample* $(x_i^k, y_i^k) \in \mathbb{R}^p \times \mathbb{R}$, $i = 1, \dots, n$, given by matrices $\overline{X}^k \in \mathbb{R}^{n \times p}$ and $\overline{Y}^k \in \mathbb{R}^n$.

The n pairs (x_i^k, y_i^k), $i = 1, \dots, n$, are sampled uniformly at random *with replacement* from the n original pairs in $(\overline{X}, \overline{Y})$. The sampling of the nm pairs of observations is independent. In other words, we defined the distribution of the ghost sample $(\overline{X}^*, \overline{Y}^*)$ by sampling n points with replacement from $(\overline{X}, \overline{Y})$, and, given $(\overline{X}, \overline{Y})$, the m ghost samples are independently sampled i.i.d. from the distribution of $(\overline{X}^*, \overline{Y}^*)$.

The asymptotic analysis from Section 2 suggests to estimate the supports $J_k = \{j,\ \hat{w}_j^k \neq 0\}$ of the Lasso estimates $\hat{w}^k$ for the bootstrap samples, $k = 1, \dots, m$, and to intersect them to define the Bolasso model estimate of the support: $J = \bigcap_{k=1}^m J_k$. Once J is selected, we estimate w by the unregularized least-square fit restricted to variables in J. The detailed algorithm is given in Algorithm 1. The algorithm has only one extra parameter (the number of bootstrap samples m). Following Proposition 3, $\log(m)$ should be chosen growing with n asymptotically slower than n. In simulations, we always use $m = 128$ (except in Figure 3, where we study the influence of m).

Algorithm 1 Bolasso

Input: data $(\overline{X}, \overline{Y}) \in \mathbb{R}^{n \times (p+1)}$
number of bootstrap replicates m
regularization parameter μ
for $k = 1$ **to** m **do**
  Generate bootstrap samples $(\overline{X}^k, \overline{Y}^k) \in \mathbb{R}^{n \times (p+1)}$
  Compute Lasso estimate $\hat{w}^k$ from $(\overline{X}^k, \overline{Y}^k)$
  Compute support $J_k = \{j,\ \hat{w}_j^k \neq 0\}$
end for
Compute $J = \bigcap_{k=1}^m J_k$
Compute $\hat{w}_J$ from $(\overline{X}_J, \overline{Y})$

Note that in practice, the Bolasso estimate can be computed simultaneously for a large number of regularization parameters because of the efficiency of the Lars algorithm (which we use in simulations), that allows to find the entire regularization path for the Lasso at the (empirical) cost of a single matrix inversion (Efron et al., 2004). Thus the computational complexity of the Bolasso is $O(m(p^3 + p^2 n))$.

The following proposition (proved in Appendix A) shows that the previous algorithm leads to consistent model selection.

Proposition 3 *Assume (**A1-3**) and $\mu_n = \mu_0 n^{-1/2}$, with $\mu_0 > 0$. Then, for all $m > 1$, the probability that the Bolasso does not exactly select the correct model, i.e., $\mathbb{P}(J \neq \mathbf{J})$, has the following upper bound:*

$$\mathbb{P}(J \neq \mathbf{J}) \leqslant m A_1 e^{-A_2 n} + A_3 \frac{\log(n)}{n^{1/2}} + A_4 \frac{\log(m)}{m},$$

where A_1, A_2, A_3, A_4 are strictly positive constants.

Therefore, if $\log(m)$ tends to infinity slower than n when n tends to infinity, the Bolasso asymptotically selects with overwhelming probability the correct active variable, and by regular consistency of the restricted least-square estimate, the correct sign pattern as well. Note that the previous bound is true whether the condition in Eq. (2) is satisfied or not, but could be improved on if we suppose that Eq. (2) is satisfied. See Section 4.1 for a detailed comparison with the Lasso on synthetic examples.

4. Simulations

In this section, we illustrate the consistency results obtained in this paper with a few simple simulations on synthetic examples and some medium scale datasets from the UCI machine learning repository (Asuncion & Newman, 2007).

4.1. Synthetic examples

For a given dimension p, we sampled $X \in \mathbb{R}^p$ from a normal distribution with zero mean and covariance matrix generated as follows: (a) sample a $p \times p$ matrix G with independent standard normal distributions, (b) form $\mathbf{Q} = GG^\top$, (c) scale $\mathbf{Q}$ to unit diagonal. We then selected the first $\mathrm{Card}(\mathbf{J}) = r$ variables and sampled non zero loading vectors as follows: (a) sample each loading signs in $\{-1, 1\}$ uniformly at random and (b) rescale those by a scaling which is uniform at random between $\frac{1}{3}$ and 1 (to ensure $\min_{j \in \mathbf{J}} |\mathbf{w}_j| \geqslant 1/3$). Finally, we chose a constant noise level σ equal to 0.1 times $(\mathbb{E}(\mathbf{w}^\top X)^2)^{1/2}$, and the additive noise ε is normally distributed with zero mean and variance σ^2. Note that the joint distribution on (X, Y) thus defined satisfies with probability one (with respect to the sampling of the covariance matrix) assumptions (**A1-3**).

In Figure 1, we sampled two distributions P_{XY} with $p = 16$ and $r = 8$ relevant variables, one for which the consistency condition in Eq. (2) is satisfied (left), one for which it was not satisfied (right). For a fixed number of sample $n = 1000$, we generated 256 replications and computed the empirical frequencies of selecting any given variable for the Lasso as the regularization parameter μ varies. Those plots show the various asymptotic regimes of the Lasso detailed in Section 2. In particular, on the right plot, although no μ leads to perfect selection (i.e., exactly variables with indices less than $r = 8$ are selected), there is a range where all relevant variables are always selected, while all others are selected with probability within $(0, 1)$.

In Figure 2, we plot the results under the same conditions for the Bolasso (with a fixed number of bootstrap replications $m = 128$). We can see that in the Lasso-consistent case (left), the Bolasso widens the consistency region, while in the Lasso-inconsistent case (right), the Bolasso "creates" a consistency region.

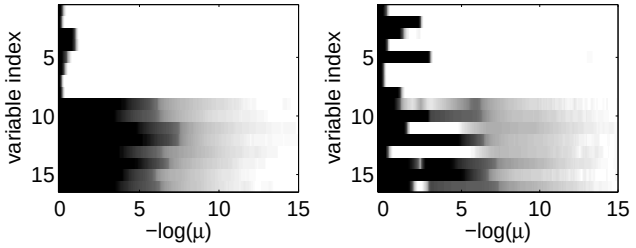


Figure 1. **Lasso**: log-odd ratios of the probabilities of selection of each variable (white = large probabilities, black = small probabilities) vs. regularization parameter. Consistency condition in Eq. (2) satisfied (left) and not satisfied (right).

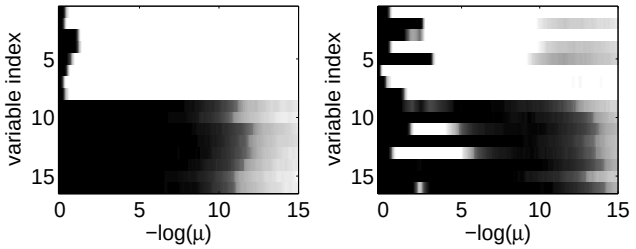


Figure 2. **Bolasso**: log-odd ratios of the probabilities of selection of each variable (white = large probabilities, black = small probabilities) vs. regularization parameter. Consistency condition in Eq. (2) satisfied (left) and not satisfied (right).

In Figure 3, we selected the same two distributions and compared the probability of exactly selecting the correct support pattern, for the Lasso, and for the Bolasso with varying numbers of bootstrap replications (those probabilities are computed by averaging over 256 experiments with the same distribution). In Figure 3, we can see that in the Lasso-inconsistent case (right), the Bolasso indeed allows to fix the unability of the Lasso to find the correct pattern. Moreover, increasing m looks always beneficial; note that although it seems to contradict the asymptotic analysis in Section 3 (which imposes an upper bound for consistency), this is due to the fact that not selecting (at least) the relevant variables has very low probability and is not observed with only 256 replications.

Finally, in Figure 4, we compare various variable selection procedures for linear regression, to the Bolasso, with two distributions where $p = 64$, $r = 8$ and varying n. For all the methods we consider, there is a natural way to select exactly r variables with no free parameters (for the Bolasso, we select the most stable pattern with r elements, i.e., the pattern which corresponds to most values of μ). We can see that the Bolasso outperforms all other variable selection methods, even in settings where the number of samples becomes of the order of the number of variables, which requires additional theoretical analysis, subject of ongoing

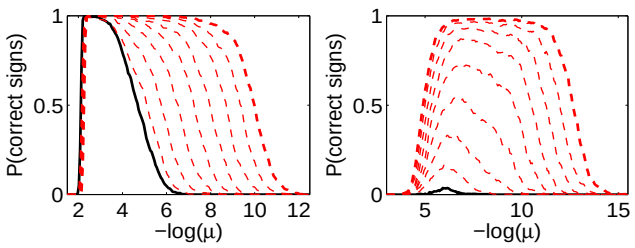


Figure 3. Bolasso (red, dashed) and Lasso (black, plain): probability of correct sign estimation vs. regularization parameter. Consistency condition in Eq. (2) satisfied (left) and not satisfied (right). The number of bootstrap replications m is in $\{2, 4, 8, 16, 32, 64, 128, 256\}$.

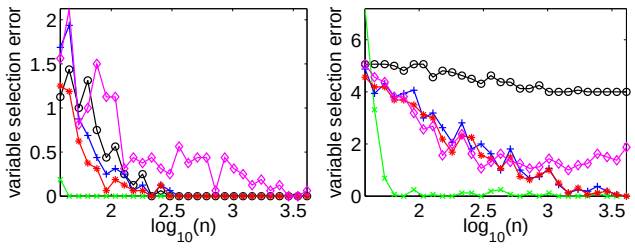


Figure 4. Comparison of several variable selection methods: Lasso (black circles), Bolasso (green crosses), forward greedy (magenta diamonds), thresholded LS estimate (red stars), adaptive Lasso (blue pluses). Consistency condition in Eq. (2) satisfied (left) and not satisfied (right). The averaged (over 32 replications) variable selection error is computed as the square distance between sparsity pattern indicator vectors.

research. Note in particular that we compare with bagging of least-square regressions (Breiman, 1996a) followed by a thresholding of the loading vector, which is another simple way of using bootstrap samples: the Bolasso provides a more efficient way to use the extra information, not for usual stabilization purposes (Breiman, 1996b), but directly for model selection. Note finally, that the bagging of Lasso estimates requires an additional parameter and is thus not tested.

4.2. UCI datasets

The previous simulations have shown that the Bolasso is succesful at performing model selection in synthetic examples. We now apply it to several linear regression problems and compare it to alternative methods for linear regression, namely, ridge regression, Lasso, bagging of Lasso estimates (Breiman, 1996a), and a soft version of the Bolasso (referred to as Bolasso-S), where instead of intersecting the supports for each bootstrap replications, we select those which are present in at least 90% of the bootstrap replications. In Table 1, we consider data randomly generated as in Section 4.1 (with $p = 32$, $r = 8$, $n = 64$), where

the true model is known to be composed of a sparse loading vector, while in Table 2, we consider regression datasets from the UCI machine learning repository, for which we have no indication regarding the sparsity of the best linear predictor. For all of those, we perform 10 replications of 10-fold cross validation and for all methods (which all have one free regularization parameter), we select the best regularization parameter on the 100 folds and plot the mean square *prediction* error and its standard deviation.

Note that when the generating model is actually sparse (Table 1), the Bolasso outperforms all other models, while in other cases (Table 2) the Bolasso is sometimes too strict in intersecting models, i.e., the softened version works better and is more competitive with other methods. Studying the effects of this softened scheme (which is more similar to usual voting schemes), in particular in terms of the potential trade-off between good model selection and low prediction error, and under conditions where p is large, is the subject of ongoing work.

Table 1. Comparison of least-square estimation methods, data generated as described in Section 4.1, with $\kappa = \|\mathbf{Q}_{J^cJ}\mathbf{Q}_{JJ}^{-1}\mathbf{s}_J\|_\infty$ (cf. Eq. (2)). Performance is measured through mean squared prediction error (multiplied by 100).

κ	0.93	1.20	1.42	1.28
Ridge	8.8 ± 4.5	4.9 ± 2.5	7.3 ± 3.9	8.1 ± 8.6
Lasso	7.6 ± 3.8	4.4 ± 2.3	4.7 ± 2.5	5.1 ± 6.5
Bolasso	5.4 ± 3.0	3.4 ± 2.4	3.4 ± 1.7	3.7 ± 10.2
Bagging	7.8 ± 4.7	4.6 ± 3.0	5.4 ± 4.1	5.8 ± 8.4
Bolasso-S	5.7 ± 3.8	3.0 ± 2.3	3.1 ± 2.8	3.2 ± 8.2

Table 2. Comparison of least-square estimation methods, UCI regression datasets. Performance is measured through mean squared prediction error (multiplied by 100).

	Autompg	Imports	Machine	Housing
Ridge	18.6 ± 4.9	7.7 ± 4.8	5.8 ± 18.6	28.0 ± 5.9
Lasso	18.6 ± 4.9	7.8 ± 5.2	5.8 ± 19.8	28.0 ± 5.7
Bolasso	18.1 ± 4.7	20.7 ± 9.8	4.6 ± 21.4	26.9 ± 2.5
Bagging	18.6 ± 5.0	8.0 ± 5.2	6.0 ± 18.9	28.1 ± 6.6
Bolasso-S	17.9 ± 5.0	8.2 ± 4.9	4.6 ± 19.9	26.8 ± 6.4

5. Conclusion

We have presented a detailed analysis of the variable selection properties of a boostrapped version of the Lasso. The model estimation procedure, referred to as the Bolasso, is provably consistent under general assumptions. This work brings to light that poor variable selection results of the Lasso may be easily enhanced thanks to a simple parameter-free resampling procedure. Our contribution also suggests that the use of bootstrap samples by L. Breiman in Bagging/Arcing/Random Forests (Breiman, 1998) may have been so far slightly overlooked and considered a minor feature, while using boostrap samples may actually be a key computational feature in such algorithms for good model selection performances, and eventually good prediction performances on real datasets.

The current work could be extended in various ways: first, we have focused on a fixed total number of variables, and allowing the numbers of variables to grow is important in theory and in practice (Meinshausen & Yu, 2008). Second, the same technique can be applied to similar settings than least-square regression with the ℓ_1-norm, namely regularization by block ℓ_1-norms (Bach, 2008) and other losses such as general convex classification losses. Finally, theoretical and practical connections could be made with other work on resampling methods and boosting (Bühlmann, 2006).

A. Proof of Model Consistency Results

In this appendix, we give sketches of proofs for the asymptotic results presented in Section 2 and Section 3. The proofs rely on the well-known property of the Lasso optimization problems, namely that if the sign pattern of the solution is known, then we can get the solution in closed form.

A.1. Optimality Conditions

We let denote $\overline{\varepsilon} = \overline{Y} - \overline{X}\mathbf{w} \in \mathbb{R}^n$, $Q = \overline{X}^\top \overline{X}/n \in \mathbb{R}^{p\times p}$ and $q = \overline{X}^\top \overline{\varepsilon}/n \in \mathbb{R}^p$. First, we can equivalently rewrite Eq. (1) as:

$$\min_{w\in\mathbb{R}^p} \tfrac{1}{2}(w-\mathbf{w})^\top Q(w-\mathbf{w}) - q^\top(w-\mathbf{w}) + \mu_n\|w\|_1. \quad (3)$$

The optimality conditions for Eq. (3) can be written in terms of the sign pattern $s = s(w) = \operatorname{sign}(w)$ and the sparsity pattern $J = J(w) = \{j,\ w_j \neq 0\}$ (Yuan & Lin, 2007):

$$\|(Q_{J^cJ}Q_{JJ}^{-1}Q_{J\mathbf{J}} - Q_{J^c\mathbf{J}})\mathbf{w}_\mathbf{J} + (Q_{J^cJ}Q_{JJ}^{-1}q_J - q_{J^c}) + \mu_n Q_{J^cJ}Q_{JJ}^{-1}s_J\|_\infty \leqslant \mu_n, \quad (4)$$

$$\operatorname{sign}(Q_{JJ}^{-1}Q_{J\mathbf{J}}\mathbf{w}_\mathbf{J} + Q_{JJ}^{-1}q_J - \mu_n Q_{JJ}^{-1}s_J) = s_J. \quad (5)$$

In this paper, we focus on regularization parameters μ_n of the form $\mu_n = \mu_0 n^{-1/2}$. The main idea behind the results is to consider that (Q, q) are distributed according to their limiting distributions, obtained from the law of large numbers and the central limit theorem, i.e., Q converges to $\mathbf{Q}$ a.s. and $n^{1/2}q$ is asymptotically normally distributed with mean zero and covariance matrix $\sigma^2\mathbf{Q}$. When assuming this, Propositions 1 and 2 are straightforward. The main effort is to make sure that we can safely replace (Q, q) by

their limiting distributions. The following lemmas give sufficient conditions for correct estimation of the signs of variables in $\mathbf{J}$ and for selecting a given pattern s (note that all constants could be expressed in terms of $\mathbf{Q}$ and $\mathbf{w}$, details are omitted here):

Lemma 1 *Assume (**A2**) and* $\|Q-\mathbf{Q}\|_2 \leqslant \lambda_{\min}(\mathbf{Q})/2$. *Then* $\operatorname{sign}(\hat{w}_{\mathbf{J}}) \neq \operatorname{sign}(\mathbf{w}_{\mathbf{J}})$ *implies* $\|\mathbf{Q}^{-1/2}q\|_2 \geqslant C_1 - \mu_n C_2$, *where* $C_1, C_2 > 0$.

Lemma 2 *Assume (**A2**) and let* $s \in \{-1,0,1\}^p$ *such that* $s_{\mathbf{J}} = \operatorname{sign}(\mathbf{w}_{\mathbf{J}})$. *Let* $J = \{j, s_j \neq 0\} \supset \mathbf{J}$. *Assume*

$$\|\mathbf{Q}-Q\|_2 \leqslant \min\{\eta_1, \lambda_{\min}(\mathbf{Q})/2\}, \tag{6}$$

$$\|\mathbf{Q}^{-1/2}q\|_2 \leqslant \min\{\eta_2, C_1 - \mu_n C_4\}, \tag{7}$$

$$\|\mathbf{Q}_{J^cJ}\mathbf{Q}_{JJ}^{-1}q_J - q_{J^c} - \mu_n \mathbf{Q}_{J^cJ}\mathbf{Q}_{JJ}^{-1}s_J\|_\infty \leqslant \mu_n - C_5\eta_1\mu_n - C_6\eta_1\eta_2, \tag{8}$$

$$\forall i \in J\backslash\mathbf{J}, s_i\left[Q_{JJ}^{-1}(q_J - \mu_n s_J)\right]_i \geqslant \mu_n C_7\eta_1 + C_8\eta_1\eta_2, \tag{9}$$

with C_4, C_5, C_6, C_7, C_8 *are positive constants. Then* $\operatorname{sign}(\hat{w}) = \operatorname{sign}(\mathbf{w})$.

Those two lemmas are useful because they relate optimality of certain sign patterns to quantities from which we can derive concentration inequalities.

A.2. Concentration Inequalities

Throughout the proofs, we need to provide upper bounds on the following quantities $\mathbb{P}(\|\mathbf{Q}^{-1/2}q\|_2 > \alpha)$ and $\mathbb{P}(\|Q-\mathbf{Q}\|_2 > \eta)$. We obtain, following standard arguments (Boucheron et al., 2004): if $\alpha < C_9$ and $\eta < C_{10}$ (where $C_9, C_{10} > 0$ are constants),

$$\mathbb{P}(\|\mathbf{Q}^{-1/2}q\|_2 > \alpha) \leqslant 4p\exp\left(-\tfrac{n\alpha^2}{2pC_9}\right).$$

$$\mathbb{P}(\|Q-\mathbf{Q}\|_2 > \eta) \leqslant 4p^2\exp\left(-\tfrac{n\eta^2}{2p^2C_{10}}\right).$$

We also consider multivariate *Berry-Esseen inequalities* (Bentkus, 2003); the probability $\mathbb{P}(n^{1/2}q \in \mathcal{C})$ can be estimated as $\mathbb{P}(t \in \mathcal{C})$ where t is normal with mean zero and covariance matrix $\sigma^2\mathbf{Q}$. The error $|\mathbb{P}(n^{1/2}q \in \mathcal{C}) - \mathbb{P}(t \in \mathcal{C})|$ is then *uniformly* (for all convex sets $\mathcal{C}$) upperbounded by:

$$400p^{1/4}n^{-1/2}\lambda_{\min}(\mathbf{Q})^{-3/2}\mathbb{E}|\varepsilon|^3\|X\|_2^3 = C_{11}n^{-1/2}.$$

A.3. Proof of Proposition 1

By Lemma 2, for any A and n large enough, the probability that the sign is different from s is upperbounded by

$$\mathbb{P}\left(\|\mathbf{Q}^{-1/2}q\|_2 > \tfrac{A(\log n)^{1/2}}{n^{1/2}}\right) + \mathbb{P}\left(\|\mathbf{Q}-Q\|_2 > \tfrac{A(\log n)^{1/2}}{n^{1/2}}\right) + \mathbb{P}\{t \notin \mathcal{C}(s, \mu_0(1-\alpha))\} + 2C_{11}n^{-1/2},$$

where $\mathcal{C}(s,\beta)$ is the set of t such that (a) $\|\mathbf{Q}_{J^cJ}\mathbf{Q}_{JJ}^{-1}t_J - t_{J^c} - \beta\mathbf{Q}_{J^cJ}\mathbf{Q}_{JJ}^{-1}s_J\|_\infty \leqslant \beta$ and (b) for all $i \in J\backslash\mathbf{J}, s_i\left[Q_{JJ}^{-1}(t_J - \beta s_J)\right]_i \geqslant 0$. Note that with $\alpha = O((\log n)n^{-1/2})$, which tends to zero, we have: $\mathbb{P}\{t \notin \mathcal{C}(s,\mu_0(1-\alpha))\} \leqslant \mathbb{P}\{t \notin \mathcal{C}(s,\mu_0)\} + O(\alpha)$. All terms (if A is large enough) are thus $O((\log n)n^{-1/2})$.

This shows that $\mathbb{P}(\operatorname{sign}(\hat{w}) = \operatorname{sign}(\mathbf{w})) \geqslant \rho(s,\mu_0) + O((\log n)n^{-1/2})$ where $\rho(s,\mu_0) = \mathbb{P}\{t \in \mathcal{C}(s,\mu_0)\} \in (0,1)$–the probability is strictly between 0 and 1 because the set and its complement have non empty interiors and the normal distribution has a positive definite covariance matrix $\sigma^2\mathbf{Q}$. The other inequality can be proved similarly. Note that the constant in $O((\log n)n^{-1/2})$ depends on μ_0 but by carefully considering this dependence on μ_0, we can make the inequality uniform in μ_0 as long as μ_0 tends to zero or infinity at most at a logarithmic speed (i.e., μ_n deviates from $n^{-1/2}$ by at most a logarithmic factor). Also, it would be interesting to consider uniform bounds on portions of the regularization path.

A.4. Proof of Proposition 2

From Lemma 1, the probability of not selecting any of the variables in $\mathbf{J}$ is upperbounded by

$$\mathbb{P}(\|\mathbf{Q}^{-1/2}q\|_2 > C_1 - \mu_n C_2) + \mathbb{P}(\|\mathbf{Q}-Q\|_2 > \lambda_{\min}(\mathbf{Q})/2),$$

which is straightforwardly upper bounded (using Section A.2) by a term of the required form.

A.5. Proof of Proposition 3

In order to simplify the proof, we made the simplifying assumption that the random variables X and ε have compact supports. Extending the proofs to take into account the looser condition that $\|X\|^2$ and ε^2 have non uniformly infinite cumulant generating functions (i.e., assumption (**A1**)) can be done with minor changes. The probability that $\bigcap_{k=1}^m J_k$ is different from $\mathbf{J}$ is upper bounded by the sum of the following probabilities:

(a) Probability of missing at least one variable in J in any of the m replications: by Lemma 1, the probability that for the k-th replication, one index in $\mathbf{J}$ is not selected, is upper bounded by

$$\mathbb{P}(\|\mathbf{Q}^{-1/2}q^*\|_2 > C_1/2) + \mathbb{P}(\|\mathbf{Q}-Q^*\|_2 > \lambda_{\min}(\mathbf{Q})/2),$$

where q^* corresponds to the ghost sample; as common in theoretical analysis of the bootstrap, we relate q^* to q as follows: $\mathbb{P}(\|\mathbf{Q}^{-1/2}q^*\|_2 > C_1/2) \leqslant \mathbb{P}(\|\mathbf{Q}^{-1/2}(q^*-q)\|_2 > C_1/4) + \mathbb{P}(\|\mathbf{Q}^{-1/2}q\|_2 > C_1/4)$ (and similarly for $\mathbb{P}(\|\mathbf{Q}-Q^*\|_2 > \lambda_{\min}(\mathbf{Q})/2)$). Because we have assumed that X and ε have compact supports, the bootstrapped variables have also compact support and we can use concentration inequalities (given the original variables $\overline{X}$, and also

after expectation with respect to $\overline{X}$). Thus the probability for one bootstrap replication is upperbounded by Be^{-Cn} where B and C are strictly positive constants. Thus the overall contribution of this part is less than mBe^{-Cn}.

(b) Probability of not selecting exactly J in all replications: note that this is not tight at all since on top of the relevant variables which are selected with overwhelming probability, different additional variables may be selected for different replications and cancel out when intersecting.

Our goal is thus to bound $\mathbb{E}\left\{\mathbb{P}(\mathbf{J}^* \neq \mathbf{J}|\overline{X})^m\right\}$. By Lemma 2, we have that $\mathbb{P}(\mathbf{J}^* \neq \mathbf{J}|\overline{X})$ is upper bounded by

$$\mathbb{P}\left(\|\mathbf{Q}^{-1/2}q^*\|_2 > \tfrac{A(\log n)^{1/2}}{n^{1/2}}|\overline{X}\right)$$

$$+\mathbb{P}\left(\|\mathbf{Q} - Q^*\|_2 > \frac{A(\log n)^{1/2}}{n^{1/2}}|\overline{X}\right)$$

$$+\mathbb{P}(t^* \notin \mathcal{C}(\mu_0)|\overline{X}) + 2C_{11}n^{-1/2} + O(\tfrac{\log n}{n^{1/2}}),$$

where now, given $\overline{X}, \overline{Y}$, t^* is normally distributed with mean $n^{1/2}q$ and covariance matrix $\frac{1}{n}\sum_{i=1}^n \varepsilon_i^2 x_i x_i^\top$.

As in (a), the first two terms and the last two ones are uniformly $O(\frac{\log n}{n^{1/2}})$ (if A is large enough). We then have to consider the remaining term. We have $\mathcal{C}(\mu_0) = \{t^* \in \mathbb{R}^p, \|\mathbf{Q}_{\mathbf{J}^c\mathbf{J}}\mathbf{Q}_{\mathbf{J}\mathbf{J}}^{-1}t^*_{\mathbf{J}} - t^*_{\mathbf{J}^c} - \mu_0\mathbf{Q}_{\mathbf{J}^c\mathbf{J}}\mathbf{Q}_{\mathbf{J}\mathbf{J}}^{-1}\mathbf{s}_{\mathbf{J}}\|_\infty \leqslant \mu_0\}$. By Hoeffding's inequality, we can replace the covariance matrix that depends on $\overline{X}$ and $\overline{Y}$ by $\sigma^2\mathbf{Q}$, at cost $O(n^{-1/2})$. We thus have to bound $\mathbb{P}(n^{1/2}q + y \notin \mathcal{C}(\mu_0)|q)$ for y normally distributed and $\mathcal{C}(\mu_0)$ a fixed compact set. Because the set is compact, there exist constants $A, B > 0$ such that, if $\|n^{1/2}q\|_2 \leqslant \alpha$ for α large enough, then $\mathbb{P}(n^{1/2}q + y \notin \mathcal{C}(\mu_0)|q) \leqslant 1 - Ae^{-B\alpha^2}$. Thus, by truncation, we obtain a bound of the form:

$$\mathbb{E}\left\{\mathbb{P}(\mathbf{J}^* \neq \mathbf{J}|\overline{X})^m\right\} \leqslant (1 - Ae^{-B\alpha^2} + F\frac{\log n}{n^{1/2}})^m + Ce^{-B\alpha^2}$$

$$\leqslant \exp(-mAe^{-B\alpha^2} + mF\frac{\log n}{n^{1/2}}) + Ce^{-B\alpha^2},$$

where we have used Hoeffding's inequality to upper bound $\mathbb{P}(\|n^{1/2}q\|_2 > \alpha)$. By minimizing in closed form with respect to $e^{-B\alpha^2}$, i.e., with $e^{-B\alpha^2} = \frac{F\log n}{An^{1/2}} + \frac{\log(mA/C)}{mA}$, we obtain the desired inequality.

Acknowledgements

I would like to thank Zaïd Harchaoui and Jean-Yves Audibert for fruitful discussions related to this work. This work was supported by a French grant from the Agence Nationale de la Recherche (MGA Project).

References

Asuncion, A., & Newman, D. (2007). UCI machine learning repository.

Bach, F. R. (2008). Consistency of the group Lasso and multiple kernel learning. *J. Mac. Learn. Res.*, to appear.

Bentkus, V. (2003). On the dependence of the Berry–Esseen bound on dimension. *Journal of Statistical Planning and Inference*, *113*, 385–402.

Boucheron, S., Lugosi, G., & Bousquet, O. (2004). Concentration inequalities. *Advanced Lectures on Machine Learning*. Springer.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (1996b). Heuristics of instability and stabilization in model selection. *Ann. Stat.*, *24*, 2350–2383.

Breiman, L. (1998). Arcing classifier. *Ann. Stat.*, *26*, 801–849.

Bühlmann, P. (2006). Boosting for high-dimensional linear models. *Ann. Stat.*, *34*, 559–583.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Ann. Stat.*, *32*, 407.

Efron, B., & Tibshirani, R. J. (1998). *An introduction to the bootstrap*. Chapman & Hall.

Fu, W., & Knight, K. (2000). Asymptotics for Lasso-type estimators. *Ann. Stat.*, *28*, 1356–1378.

Lounici, K. (2008). Sup-norm convergence rate and sign concentration property of Lasso and Dantzig estimators. *Electronic Journal of Statistics*, *2*.

Meinshausen, N., & Yu, B. (2008). Lasso-type recovery of sparse representations for high-dimensional data. *Ann. Stat.*, to appear.

Tibshirani, R. (1994). Regression shrinkage and selection via the Lasso. *J. Roy. Stat. Soc. B*, *58*, 267–288.

Wainwright, M. J. (2006). *Sharp thresholds for noisy and high-dimensional recovery of sparsity using ℓ_1-constrained quadratic programming* (Tech. report 709). Dpt. of Statistics, UC Berkeley.

Yuan, M., & Lin, Y. (2007). On the non-negative garrotte estimator. *J. Roy. Stat. Soc. B*, *69*, 143–161.

Zhao, P., & Yu, B. (2006). On model selection consistency of Lasso. *J. Mac. Learn. Res.*, *7*, 2541–2563.

Zou, H. (2006). The adaptive Lasso and its oracle properties. *J. Am. Stat. Ass.*, *101*, 1418–1429.

Learning All Optimal Policies with Multiple Criteria

Leon Barrett BARRETT@ICSI.BERKELEY.EDU
Srini Narayanan SNARAYAN@ICSI.BERKELEY.EDU
1947 Center St. Ste. 600, Berkeley, CA 94704

Abstract

We describe an algorithm for learning in the presence of multiple criteria. Our technique generalizes previous approaches in that it can learn optimal policies for all linear preference assignments over the multiple reward criteria at once. The algorithm can be viewed as an extension to standard reinforcement learning for MDPs where instead of repeatedly backing up maximal expected rewards, we back up the set of expected rewards that are maximal for some set of linear preferences (given by a weight vector, $\overrightarrow{w}$). We present the algorithm along with a proof of correctness showing that our solution gives the optimal policy for any linear preference function. The solution reduces to the standard value iteration algorithm for a specific weight vector, $\overrightarrow{w}$.

1. Introduction

In Reinforcement Learning (RL), an agent interacts with the environment to learn optimal behavior. (Sutton & Barto, 1998) Most RL techniques are based on a scalar reward, i.e., they aim to optimize an objective that is expressed as a function of a scalar reinforcement. A natural extension to traditional RL techniques is thus the case where there are multiple rewards. In many realistic domains, actions depend on satisfying multiple objectives simultaneously (such as achieving performance while keeping costs low, a robot moving efficiently toward a goal while being close to a recharging station, or a government funding both military and social programs). Learning optimal policies in many real-world domains thus depends on the ability to learn in the presence of multiple rewards. However, the resulting policies depend heavily on the preferences over these rewards, and they may change swiftly as preferences vary. We present both an algorithm for the general case of learning all optimal policies under all assignments of linear priorities for the reward components, and a proof showing the correctness of our algorithm.

We start with a motivating example of a simple task with multiple rewards in Section 2. The paper then proceeds to the main algorithm in Section 3. We address related work in Section 4, and then Section 5 discusses the complexity of our algorithm including realistic and tractable specializations of our algorithm. Section 6 describes the application of this algorithm to an example domain, and Section 7 discusses extensions to this technique, such as implementations using other RL methods (such as temporal difference methods) and applications of our algorithm to infer another agent's preferences based on observing their behavior. Section 8 outlines the proof of the algorithm's correctness.

2. Explanation and Motivating Example

We assume that instead of getting a single reward signal, the agent gets a reward divided up into several components, a reward vector. That is, we decompose the reward signal $r(s,a)$ (where s is a state and a is an action) into a vector $\overrightarrow{r}(s,a) = [r_1(s,a), r_2(s,a), \ldots, r_n(s,a)]$. An agent could potentially optimize many different functions of these rewards, but the simplest function is a weighted sum: for every fixed weight vector $\overrightarrow{w}$ we obtain a total reward scalar $r_{\overrightarrow{w}}(s,a) = \overrightarrow{w} \cdot \overrightarrow{r}(s,a)$. There is thus an optimal policy $\pi^*_{\overrightarrow{w}}$ for *each* weight vector $\overrightarrow{w}$.

Consider, for example, a lab guinea pig running a familiar maze, shown in Figure 1. The guinea pig runs through the maze to one of four stashes of food. Once it has reached a stash and eaten the food, the experimenter takes it out of the maze and returns it to its cage, so it can only hope to eat one of the stashes per run of the maze. Assume that there are

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

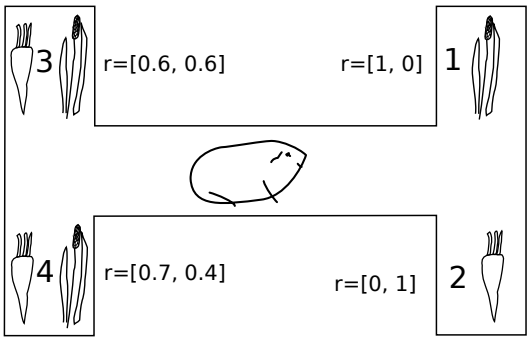


Figure 1. An example maze with rewards, split into 2 components, at 3 different locations

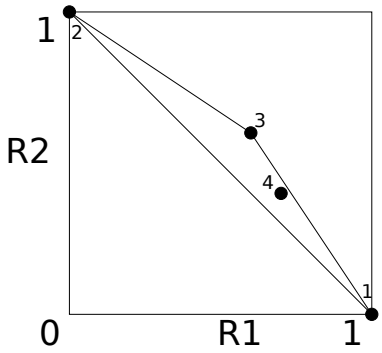


Figure 2. The potential reward vectors in the guinea pig example

only 2 types of food provided (hay and carrot), so reward vectors take the form [hay,carrot]. Location 1 contains hay ($\overrightarrow{r} = [1, 0]$), location 2 contains carrot ($\overrightarrow{r} = [0, 1]$), and locations 3 and 4 contain a little of both ($\overrightarrow{r} = [0.6, 0.6]$ and $[0.7, 0.4]$, respectively). Because the maze is familiar, the animal knows where the food is placed and what sort of food is in each location.

The experimenter has several different guinea pigs and has discovered that each has different tastes. For instance, Chester likes only hay ($\overrightarrow{w} = [1, 0]$), and Milo likes only carrot ($\overrightarrow{w} = [0, 1]$), but greedy Louis likes both equally ($\overrightarrow{w} = [0.5, 0.5]$). (Without loss of generality, assume that all animals' weight vectors satisfy $\sum_i w_i = 1$: they describe relative preferences, not absolute utilities.) So, if Chester goes to location 4 ($\overrightarrow{r} = [0.7, 0.4]$), then he gets reward $r = \overrightarrow{w} \cdot \overrightarrow{r} = 0.7$. Milo would get 0.4, and Louis would get a reward of 0.55.

Looking at the maze, we see that although there are 4 possible strategies (with rewards shown in Figure 2, only 3 of them are optimal for any values of $\overrightarrow{w}$. One strategy occurs when the weight vector has $w_0 > 0.6$ (and hence $w_1 = 1 - w_0 < 0.4$): then the guinea pig should head straight for location 1, because the reward elsewhere will be no more than 0.6. By the exact same logic, when the weight vector has $w_1 > 0.6$ (and $w_0 < 0.4$), then the animal should go to location 2. In all other cases ($0.4 \leq w_0 \leq 0.6$), it will optimize its reward by going to location 3. Under no circumstances would an optimal agent go to location 4! No matter what its weight vector, some other location dominates location 4. We would like to determine exactly this: which policies are viable and which are not (even without knowing $\overrightarrow{w}$).

Our method learns the set of optimal policies for *all* $\overrightarrow{w}$ at the same time. Once the agent has learned all these policies, it can change reward weights at runtime to get a new optimal behavior, without having to do any relearning. For a fixed priority scheme (fixed weight vector $\overrightarrow{w}$) over the multiple reward components, our algorithm results in the standard recurrence for Q-values that is analogous to the equation for the average weighted reward case as in (Natarajan & Tadepalli, 2005):

$$Q^*_{\overrightarrow{w}}(s, a) = \mathbb{E}\left[\overrightarrow{w} \cdot \overrightarrow{r}(s, a) + \gamma \max_{a'} Q^*_{\overrightarrow{w}}(s', a') | s, a\right]$$

In the general case, where we do not know the relative priorities over the reward components, our algorithm exploits the fact that the extrema of the set of Q-values vectors (Q vectors that are maximal for some weight setting) is the same as the *convex hull* of the Q-value vectors. (The convex hull is defined as the smallest convex set that contains all of a set of points. In this case, we mean the points that lie on the boundary of this convex set, which are of course the extreme points–the ones that are maximal in some direction. This is somewhat similar to the Pareto curve, since both are maxima over trade-offs in linear domains.) Now we can rewrite the general RL recurrence in terms of operations on the convex hull of Q-values, and we show this recurrence to be correct and convergent to the value iteration algorithm in the fixed weight vector case. Many standard RL algorithms in the literature can be seen as limiting cases of our more general algorithm. While the worst-case complexity of our general algorithm is exponentially higher than that of fixed-$\overrightarrow{w}$ cases, it not only solves all the fixed-$\overrightarrow{w}$ cases but also determines which cases are worth solving. We also give some constraints and techniques that can help reduce the complexity.

3. Convex Hull Value Iteration

In this section, we introduce the problem definition in the context of a traditional MDP setting and our approach and algorithm.

3.1. Preliminaries and Notation

Our approach is based on an MDP which is a tuple $(S, A, T, \gamma, \overrightarrow{r})$, where S is a finite set of N states, $A = \{a_1, \ldots, a_k\}$ is a set of k actions, $T = \{\mathrm{P}_{sa}(s')\}$ is the set of state transition probabilities ($\mathrm{P}_{sa}(s')$ is the transition probability of going to state $s' \in S$ by taking action $a \in A$ from state $s \in S$), $\gamma \in [0, 1)$ is the discount factor, and $\overrightarrow{r} : S \times A \mapsto \mathbb{R}^d$ is the reward function giving d-component reward vector $\overrightarrow{r}(s, a)$. This differs from the standard formulation only in that reward now comes as a vector.

A policy, π, is the map $S \mapsto A$, and the value function for any policy π, evaluated at some state s_i is the vector

$$\overrightarrow{V}^{\pi}(s_i) = \mathbb{E}[\overrightarrow{r}(s_i, a_i) + \gamma \overrightarrow{r}(s_{i+1}, a_{i+1}) + \ldots | \pi] \quad (1)$$

where the expectation is over the distribution of the state and reward sequence $(s_i, \overrightarrow{r}_i, s_{i+1}, \overrightarrow{r}_{i+1}, \ldots)$, that is obtained on executing the policy π starting from the state s_i. The Q-function is the vector

$$\overrightarrow{Q}^{\pi}(s, a) = \mathbb{E}_{\overrightarrow{r}(s,a), s' \sim P_{sa}} \left[\overrightarrow{r}(s, a) + \gamma \overrightarrow{V}^{\pi}(s') \right] \quad (2)$$

where $\overrightarrow{r}(s, a), s' \sim P_{sa}$ means that the expectation with respect to s' and $\overrightarrow{r}(s, a)$ distributed according to P_{sa}. The optimal Q function for a weight $\overrightarrow{w}$ is $Q^*_{\overrightarrow{w}}(s, a) = sup_{\pi} \overrightarrow{w} \cdot \overrightarrow{Q}^{\pi}(s, a)$.

3.2. Approach: Convex Hulls

Given some $\overrightarrow{w}$, the resulting reward for taking an action is $r(s, a) = \overrightarrow{w} \cdot \overrightarrow{r}(s, a)$. This gives us the following recurrence for optimal Q-values, which is exactly equivalent to the equation for a single reward component:

$$Q_{\overrightarrow{w}}(s, a) = \mathbb{E}\left[\overrightarrow{w} \cdot \overrightarrow{r}(s, a) + \gamma \max_{a'} Q_{\overrightarrow{w}}(s', a') | s, a \right]$$

We can solve this recurrence directly, or we can use it to get converging approximations to the optimal value function—this gives rise to the value iteration method, Q-learning, and so on.

An alternative view is that each possible policy gives a different expected reward $\overrightarrow{Q}(s, a)$, and we simply want to select a policy by maximizing the dot product of this with $\overrightarrow{w}$. For a fixed $\overrightarrow{w}$, only one such $\overrightarrow{Q}(s, a)$ can be optimal, but in general we might care about any $\overrightarrow{Q}$s that are maximal for some $\overrightarrow{w}$. But this set of Q-values that are extrema is exactly the convex hull of the Q-values! This allows us to use standard convex hull operations to pare down the set of points we consider and gives rise to the following proposition.

Proposition 1. *The convex hull over Q-values contains the optimal policy over the average expected reward $r(s, a) = \overrightarrow{w} \cdot \overrightarrow{r}(s, a)$ for any $\overrightarrow{w}$.*

To make this operational and derive an algorithm that maintains all optimal policies for any weight vector $\overrightarrow{w}$, we need a few definitions for relevant operations on the convex hull.

We write $\overset{\circ}{Q}(s, a)$ to represent the vertices of the convex hull of possible Q-value vectors for taking action a at state s. We then define the following operations on convex hulls which will be used to construct our learning algorithm.

Definition 1. Translation and scaling operations

$$\overrightarrow{u} + b\overset{\circ}{Q} \equiv \{\overrightarrow{u} + b\overrightarrow{q} : \overrightarrow{q} \in \overset{\circ}{Q}\} \quad (3)$$

Definition 2. Summing two convex hulls

$$\overset{\circ}{Q} + \overset{\circ}{U} \equiv \text{hull}\{\overrightarrow{q} + \overrightarrow{u} : \overrightarrow{q} \in \overset{\circ}{Q}, \overrightarrow{u} \in \overset{\circ}{U}\} \quad (4)$$

Definition 3. Extracting the Q-value *To extract the best Q-value for a given $\overrightarrow{w}$, we perform a simple maximum:*

$$Q_{\overrightarrow{w}}(s, a) \equiv \max_{\overrightarrow{q} \in \overset{\circ}{Q}(s,a)} \overrightarrow{w} \cdot \overrightarrow{q} \quad (5)$$

Given these definitions, we are now ready to illustrate the basic algorithm.

3.3. Convex Hull Value Iteration Algorithm

Our algorithm extends the single-$\overrightarrow{w}$ case (which is the standard expected discounted reward framework (Bellman, 1957)) into the following recurrence:

$$\overset{\circ}{Q}(s, a) = \mathbb{E}\left[\overrightarrow{r}(s, a) + \gamma \,\text{hull} \bigcup_{a'} \overset{\circ}{Q}(s', a') | s, a \right] \quad (6)$$

That is, instead of repeatedly backing up maximal expected rewards, we back up the set of expected rewards that are maximal for some $\overrightarrow{w}$. While the expectation over hulls looks awkward, it is the natural equivalent of an expectation of maxima, and it arises for the same reason. We must take an expectation over s', but once in s', we can choose the best action, no matter what our $\overrightarrow{w}$. The expectation's computation can be broken down, in the usual way, into the scalings and sums we have already defined.

This leads us to define Algorithm 1, which extends the value iteration algorithm (Bellman, 1957) to learn optimal Q-values for all possible $\overrightarrow{w}$. A proof of its correctness is given in Section 8.

Algorithm 1 Value iteration algorithm modified from that of Bellman (1957)

Initialize $\overset{\circ}{Q}(s,a)$ arbitrarily $\forall s, a$
while not converged **do**
  for all $s \in S$, $a \in A$ **do**
    $\overset{\circ}{Q}(s,a) \leftarrow \mathbb{E}[\overrightarrow{r}(s,a)$
      $+\gamma \,\mathrm{hull} \bigcup_{a'} \overset{\circ}{Q}(s',a')|s,a]$
  end for
end while

return $\overset{\circ}{Q}$

4. Related Work

There is now a body of work addressing multi-reward reinforcement learning. There have been algorithms that assume a fixed ordering between different rewards, such as staying alive and not losing food (Gabor et al., 1998), techniques based on formulating the multiple reward problem as optimizing a weighted sum of the discounted total rewards for multiple reward types (Feinberg & Schwartz, 1995), and techniques that decompose the reward function into multiple components which are learned independently (with a single policy) (Russell & Zimdars, 2003). In all these cases, the preference over rewards is assumed to be fixed and time-invariant. In a slightly more flexible formulation, Mannor and Shimkin (2004) take multiple reward components and perform learning that results in expected rewards lying in a particular region of reward space.

More recently, (Natarajan & Tadepalli, 2005) formulate the multiple reward RL problem as we do, using a weighted expected discounted reward framework, and they store both the currently best policy and its Q-values as vectors. When priorities change dynamically (as reflected in changes in the weight vector), the agent can calculate new reward scalars from the vectors and thus start from the Q-values of the best policy learned so far rather than resetting entirely. As far as we are aware, none of the techniques proposed tackle the general case of learning optimal policies for all linear preference assignments over the multiple reward components.

4.1. Relation to POMDPs

Our problem, and hence its solution, is closely related to the standard partially observable Markov decision process (POMDP) formulation. In a POMDP, we have a model of both observed and unobserved variables and use Bayesian reasoning to infer a joint distribution over the hidden variables. Then, we must choose an optimal action based on both the observed state and the continuous beliefs. (Kaelbling et al., 1998)

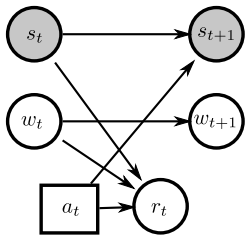


Figure 3. A POMDP formulation of multiple reward components

Consider the POMDP shown in Figure 3; here, the reward depends on an unobserved multinomial random variable, so $\mathbb{E}[r] = \sum_i \mathrm{P}(w = i) r_i$. If we define $\mathrm{P}(w_t|w_{t-1})$ to be the identity, the distribution of w will not change with t. Then, the expected reward depends linearly on our prior distribution over w, and the dual of the usual POMDP maximum-hyperplane algorithm corresponds to a convex hull operation over reward components. It is thus possible to write our multiple-reward problem as a POMDP problem. This suggests a natural route to extend our algorithm to operate on POMDPs. It remains future work, however, to see if the approximation algorithms used for solving POMDPs can yield useful results in our domain.

5. Complexity

This algorithm relies on four convex hull operations, whose complexity we will analyze in terms of the number of points on the hull, n; in the limit, this number converges to the number of optimal policies in the environment. We must both scale (by probabilities and discounts) and translate (by rewards) our convex hulls; these operations only require touching every point once, resulting in a complexity of $O(n)$. We must also merge two or more convex hulls. This takes time at most $O((kn)^{\lfloor d/2 \rfloor})$ if $d > 3$, where k is the number of hulls involved, n is the number of points in each hull, and d is the dimension (number of reward components) (Clarkson & Shor, 1989). Finally, we must add two convex hulls. If done naively by adding all pairs of points and taking a hull, this takes time at most $O(n^{2\lfloor d/2 \rfloor})$. All these operations must be performed whenever we back up Q-values, so we multiply the complexity of ordinary reinforcement learning by $O(n^{2\lfloor d/2 \rfloor})$. (However, in the $d = 2$ and $d = 3$ cases, there are efficient ways to perform these operations.)

In the long run, the number of points on each convex hull, n, must converge to a limit as the Q-values con-

		R1	E2	
		E1		R2
		H		

Figure 4. A resource-collection domain

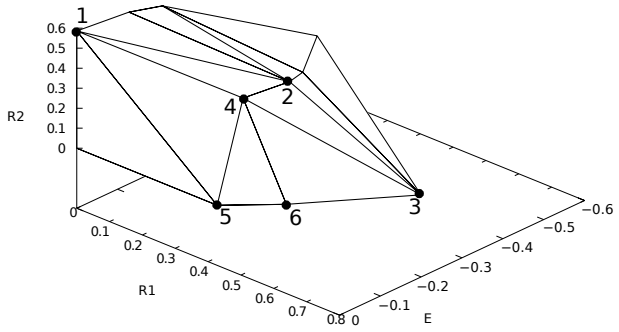


Figure 5. Optimal rewards in the resource-collection domain

verge to their optimal values. Eventually, there will be exactly one point on each convex hull for each optimal policy. However, in the short term, the number of short-range policies we might have to track might be much lower or even higher. Also, the number of optimal policies n depends on the environment in a complicated way, with the worst case being that all policies ($|A|^{|S|}$ of them) may be optimal for some weight vector.

5.1. Reducing the Complexity

The complexity result of our algorithmic modifications is an exponential blowup with the number of reward components. There are a few main ways of tackling this. The first is to simply restrict the number of reward components; with only, say, 5 or fewer, this additional computation is likely not to be an undue burden. In practice, there are currently very few problems studied with more reward components than this.

When we must handle a high-dimensional problem, we can reduce the complexity by applying constraints on the weight vectors that we might optimize for. Given the geometric nature of our approach, if we have knowledge about the directions of allowable vectors, such as $\overrightarrow{a} \cdot \overrightarrow{w} > 0$, then we can simply take a partial convex hull. This will, on average, reduce the complexity of the convex hull computation by half. So, if we know that all d elements of $\overrightarrow{w}$ must be positive, then we can write that as d such constraints to divide the convex hull complexity by 2^d.

In addition, the convergence of Q-values means that we are essentially performing the same convex hull operations again and again; this means that we might be able to reuse the information from the last iteration. The idea is to annotate each point with a "witness", or proof of its status: if a point is not on the convex hull, then we note down a set of faces that enclose it, and if it is on the hull, we note down a direction in which it is the extremum. Then, on the next iteration, when these points have moved slightly and we must compute a convex hull again, we can simply check these proofs

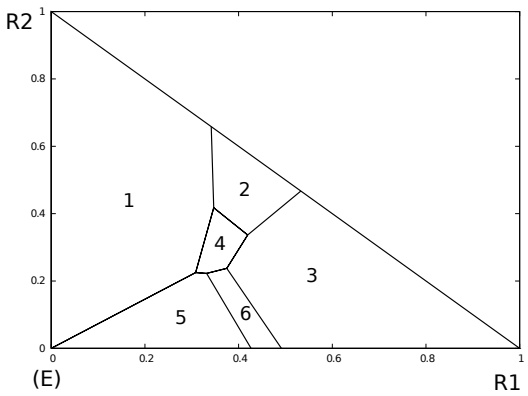


Figure 6. Regions of preference space in which policies are optimal. Axes are reward components $R1$ and $R2$; the enemy weight is $E = 1 - R1 - R2$.

(in at most $O(n^2)$ time). If all the proofs are correct, then our convex hull remains correct and the locations of the points have moved only slightly. On the other hand, if any proof is violated, we can simply rebuild the convex hull in the ordinary, expensive way. In the limit as the Q-values and policy converge, the policy must stop changing, so this trick may greatly reduce the complexity of refining Q-values.

#	**policy**
1	Go directly to R2, dodging Es
2	Go to both Rs, through both Es
3	Go to R1, through E1 both ways
4	Go to both Rs, dodging E1 but through E2
5	Go to R1, dodging all Es
6	Go to R1, going through E1 only once

Table 1. The optimal policies for the example domain

6. Example Application: Resource Gathering

In order to demonstrate the application of this method, we have tested it on a resource-collecting problem similar to that of many strategy games. We model this as a resource-collecting agent moving (in the 4 cardinal directions) around in a grid environment shown in Figure 4, starting from the home base, labelled H. Its goal is to gather resources and take them back to the home base. If it reaches location R1, it then picks up resource 1, and at R2 it gets resource 2; it can carry both at the same time. When the agent returns to H, it receives a reward for each resource it brings back. Also, if it steps on one of the two enemy spaces, labeled E1 and E2, with a 10% probability it will be attacked, receiving a penalty and resetting to the home space, losing all it carries. Its reward space is then [enemy, resource1, resource2], so it can get a penalty of [-1,0,0] for being attacked, or a reward of [0,1,0], [0,0,1], or [0,1,1] for bringing back one or both resources. We use a discounting rate of $\gamma = 0.9$.

Depending on the relative values of the resources and attack, the agent may find different policies to be valuable. The convex hull of values starting at H, $\overset{\circ}{V}(H)$, is shown in Figure 5. The points on the hull correspond to optimal policies, described in Table 1; each policy is valid for some range of preferences $\overrightarrow{w}$, which are shown in Figure 6.[1]

7. Extensions and Current Work

This same convex-hull technique can be used with other RL algorithms, such as the temporal difference learning algorithm. The critical thing to recall is that because we are learning more than one policy at once, we can use only off-policy learning algorithms.

Our solution can also be used for inferring the preference function from observation data. This is closely related to the inverse reinforcement learning problem (Ng & Russell, 2000; Abeel & Ng, 2004). The basic idea behind inverse reinforcement learning is to use observed behavior to infer weights from a user that can then be used to find optimal policies. In our case, the method for learning all policies at once can also be used in reverse to learn the range of reward weights that an agent must have. If we assume that an agent we observe is rational and uses a policy that is optimal for its reward weights, then we can use our observations of the agent to infer its reward weights. We simply repeatedly observe its choice of action a and use our knowledge of $\overset{\circ}{Q}(s,a)$ to identify which values of $\overrightarrow{w}$ are consistent with that action. Then, we take the intersection of the constraints.

The multi-criterion RL approach also allows us to examine reward at different time scales. Instead of having a single discounting factor γ, we could have a discounting factor γ_i for each component. This allows us to use a sum of exponentials with different time constants to approximate non-exponential discounting rates, which are helpful in explaining the preferences of humans (Ainslie, 2001). With our convex hull method, we can find what policies are optimal for a whole range of discounting rates.

[1] We do not show the ranges of policies optimal where the values of the rewards are less than 0 ($w_i < 0$); these policies, while sometimes interesting, are not valuable for the task.

8. Appendix: Proof of Correctness

We prove that $\forall \overrightarrow{w}$ Algorithm 1 gives the optimal policy by reducing the recurrence to the standard value iteration recurrence for any $\overrightarrow{w}$. First, recall the basic recurrence of our algorithm, Equation 6.

$$\overset{\circ}{Q}(s,a) \leftarrow \mathbb{E}\left[\overrightarrow{r}(s,a) + \gamma \,\mathrm{hull} \bigcup_{a'} \overset{\circ}{Q}(s',a') | s,a\right]$$

Now apply Equation 5 to the both sides (to extract the optimal value for $\overrightarrow{w}$):

$$\begin{aligned} Q_{\overrightarrow{w}}(s,a) \quad \leftarrow \quad & \max\{\overrightarrow{w}\cdot\overrightarrow{q} : \overrightarrow{q} \in \mathbb{E}\Big[\overrightarrow{r}(s,a) \\ & +\gamma \,\mathrm{hull} \bigcup_{a'} \overset{\circ}{Q}(s',a') | s,a\Big]\}. \end{aligned}$$

Next, apply the definition of an expectation

$$\begin{aligned} \leftarrow \max \quad & \{\overrightarrow{w}\cdot\overrightarrow{q} : \overrightarrow{q} \in \textstyle\sum_{s',\overrightarrow{r}(s,a)} \mathrm{P}(s',\overrightarrow{r}(s,a)|s,a) \\ & \cdot \left(\overrightarrow{r}(s,a) + \gamma \,\mathrm{hull} \textstyle\bigcup_{a'} \overset{\circ}{Q}(s',a')\right)\}, \end{aligned}$$

then use Equations 3 and 4 and rewrite

$$\begin{aligned} \leftarrow \quad & \max\{\overrightarrow{w}\cdot\overrightarrow{q} : \overrightarrow{q} \in \mathrm{hull}\Big\{ \\ & \sum_{i,\overrightarrow{r}(s,a)} \mathrm{P}(s'_i,\overrightarrow{r}(s,a)|s,a)\left(\overrightarrow{r}(s,a) + \gamma \overrightarrow{q}'_{s'_i}\right) \\ & : \overrightarrow{q}'_{s'_1} \in \mathrm{hull} \bigcup_{a'} \overset{\circ}{Q}(s'_1,a'), \ldots\Big\}\}. \end{aligned}$$

$$\leftarrow \max\Bigg\{\overrightarrow{w}\cdot\sum_{i,\overrightarrow{r}(s,a)}\mathrm{P}(s'_i,\overrightarrow{r}(s,a)|s,a)\cdot\Big(\overrightarrow{r}(s,a)+\gamma\,\overrightarrow{q}'_{s'_i}\Big):\ \overrightarrow{q}'_{s'_1}\in\mathrm{hull}\bigcup_{a'}\overset{\circ}{Q}(s'_1,a'),\ldots\Bigg\}$$

$$\leftarrow \max\Bigg\{\mathbb{E}\Big[\overrightarrow{w}\cdot\overrightarrow{r}(s,a)\Big|s,a\Big]+\gamma\sum_i\mathrm{P}(s'_i|s,a)\overrightarrow{w}\cdot\overrightarrow{q}'_{s'_i}:\ \overrightarrow{q}'_{s'_1}\in\mathrm{hull}\bigcup_{a'}\overset{\circ}{Q}(s'_1,a'),\ldots\Bigg\}.$$

Pull $\overrightarrow{r}(s,a)$ (added independently to the entire set) and γ (non-negative) out of the maximum.

$$\leftarrow \mathbb{E}[\overrightarrow{w}\cdot\overrightarrow{r}(s,a)|s,a]+\gamma\max\Bigg\{\sum_i\mathrm{P}(s'_i|s,a)\overrightarrow{w}\cdot\overrightarrow{q}'_{s'_i}:\ \overrightarrow{q}'_{s'_1}\in\mathrm{hull}\bigcup_{a'}\overset{\circ}{Q}(s'_1,a'),\ldots\Bigg\}$$

But the max of a sum over different sets is the sum of the sets' maxima, which we simplify.

$$\leftarrow \mathbb{E}[\overrightarrow{w}\cdot\overrightarrow{r}(s,a)|s,a]+\gamma\sum_i\mathrm{P}(s'_i|s,a)\max\Bigg\{\overrightarrow{w}\cdot\overrightarrow{q}'_{s'_i}:\ \overrightarrow{q}'_{s'_i}\in\mathrm{hull}\bigcup_{a'}\overset{\circ}{Q}(s'_i,a')\Bigg\}$$

$$\leftarrow \mathbb{E}[\overrightarrow{w}\cdot\overrightarrow{r}(s,a)|s,a]+\gamma\sum_i\mathrm{P}(s'_i|s,a)\max\Bigg\{\overrightarrow{w}\cdot\overrightarrow{q}'_{s'_i}:\ \overrightarrow{q}'_{s'_i}\in\overset{\circ}{Q}(s'_i,a'),a'\in A(s'_i)\Bigg\}$$

$$\leftarrow \mathbb{E}[\overrightarrow{w}\cdot\overrightarrow{r}(s,a)|s,a]+\gamma\sum_i\mathrm{P}(s'_i|s,a)\max_{a'}\max_{q'_{s'_i}\in\overset{\circ}{Q}(s'_i,a')}\overrightarrow{w}\cdot\overrightarrow{q}'_{s'_i}$$

But we re-order the maxima and rewrite an expectation, and so we recover our recurrence for a single $\overrightarrow{w}$.

$$Q_{\overrightarrow{w}}(s,a)\leftarrow\mathbb{E}\Big[\overrightarrow{w}\cdot\overrightarrow{r}(s,a)+\gamma\max_{a'}\overset{\circ}{Q}_{\overrightarrow{w}}(s',a')\Big|s,a\Big].$$

Given a $\overrightarrow{w}$, at any point in the algorithm, this gives the same Q-value as ordinary value iteration. Therefore, the proof of convergence for the value iteration algorithm applies to our method, and our method converges exactly as quickly as ordinary value iteration (for every $\overrightarrow{w}$).

References

Abeel, P., & Ng, A. (2004). Apprentice learning via inverse reinforcement learning. *Proc. ICML-04*.

Ainslie, G. (2001). *Breakdown of will*. Cambridge, Massachusetts: Cambridge University Press.

Bellman, R. E. (1957). *Dynamic programming*. Princeton: Princeton University Press.

Clarkson, K. L., & Shor, P. W. (1989). Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry*, *4*, 387–421.

Feinberg, E., & Schwartz, A. (1995). Constrained markov decision models with weighted discounted rewards. *Mathematics of Operations Research*, *20*, 302–320.

Gabor, Z., Kalmar, Z., & Szepesvari, C. (1998). Multi-criteria reinforcement learning. *Proc. ICML-98*.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*.

Mannor, S., & Shimkin, N. (2004). A geometric approach to multi-criterion reinforcement learning. *Journal of Machine Learning Research*, 325–360.

Natarajan, S., & Tadepalli, P. (2005). Dynamic preferences in mult-criteria reinforcement learning. *Proc. ICML-05*. Bonn, Germany.

Ng, A., & Russell, S. (2000). Algorithms for inverse reinforcement learning. *Proc. ICML-00*.

Russell, S., & Zimdars, A. (2003). Q-decomposition for reinforcement learning agents. *Proc. ICML-03*. Washington, DC.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, Massachusetts: The MIT Press.

Multiple Instance Ranking

Charles Bergeron CHBERGERON@GMAIL.COM
Mathematical Sciences Department, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180 USA

Jed Zaretzki ZARETJ@RPI.EDU
Curt Breneman BRENEC@RPI.EDU
Chemistry Department, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180 USA

Kristin P. Bennett BENNEK@RPI.EDU
Mathematical Sciences Department, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180 USA

Abstract

This paper introduces a novel machine learning model called multiple instance ranking (MIRank) that enables ranking to be performed in a multiple instance learning setting. The motivation for MIRank stems from the hydrogen abstraction problem in computational chemistry, that of predicting the group of hydrogen atoms from which a hydrogen is abstracted (removed) during metabolism. The model predicts the preferred hydrogen group within a molecule by ranking the groups, with the ambiguity of not knowing which hydrogen atom within the preferred group is actually abstracted. This paper formulates MIRank in its general context and proposes an algorithm for solving MIRank problems using successive linear programming. The method outperforms multiple instance classification models on several real and synthetic datasets.

1. Introduction

This paper introduces a new machine learning paradigm called multiple instance ranking (MIRank), bringing the concept of ranking to the framework of multiple instance learning. Some problems that MIRank could potentially solve based on prior data are:

1. For a given country, predict the city that contains the most profitable store.
2. For a given state, predict the congressional district that contains the politician that delivers the most subsidies.
3. For a given document, predict the paragraph/passage that contains the most pertinent sentence/phrase/word.
4. For a given molecular class, predict the molecule with the conformation having the highest human immunodeficiency virus (HIV) inhibition efficacy.
5. For a given state, predict the division that contains the town with the highest median housing unit price.
6. For a given molecule, predict the site of metabolism from which a hydrogen atom is abstracted (removed).

It is this last application, that of hydrogen abstraction from the field of computational chemistry, that motivated this work. The fifth application, which involves making predictions from the census, is also explored here. Later in this paper, a general formulation for multiple instance ranking is provided, an algorithm for MIRank is proposed, and this algorithm is tested on datasets that stem from both applications as well as synthetic data.

As introduced by Dietterich et al. (1997), the setup for multiple instance learning differs somewhat from the standard learning framework. In standard classification, the task is to predict the class of each item. Each item has a corresponding binary classification label, and features defined for each item are used to build the model. In multiple instance classification (MIC), each item belongs to a bag. The task is to predict the class of each bag of items. Features are defined for

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

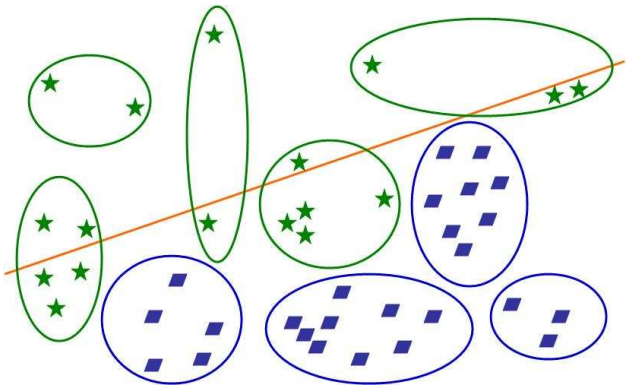


Figure 1. Schematic of multiple instance classification. Bags are ellipses, active bags contain stars and inactive bags contain parallelograms.

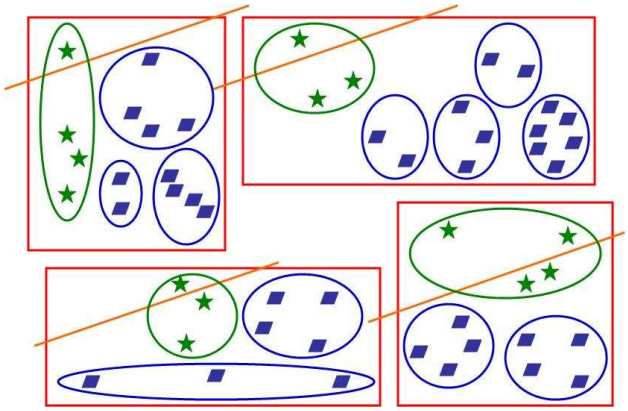


Figure 2. Schematic of multiple instance ranking. Boxes are rectangles, bags are ellipses, preferred bags contain stars, and other bags contain parallelograms.

each item, but the class label is assigned to each bag. For simplicity of presentation, assume there are two classes: active and inactive. By definition, an active bag must contain at least one active item, while an inactive bag contains exclusively inactive items. It is not known which item is active.

Figure 1 illustrates MIC, in which bags are ellipses, items in active bags are represented as stars, and items in inactive bags are marked as parallelograms. The straight line is the separating line representing the classification function. Notice that at least one item from each active bag is found above the line, while all items in inactive bags are located below the line.

The difficulty is that there exists an ambiguity as to which items in an active bag are actually active. For example, consider the drug discovery application (Dietterich et al., 1997), with molecules as bags and conformations (three-dimensional molecular shapes that differ from each other by the rotation of atom groups about one or more bonds) as items. If a molecule possesses one—or possibly several—conformations that are active, then it is known that the molecule is active. However, it is not known which conformation is active. On the other hand, if none of a molecule's conformations are active, then the molecule is deemed inactive, and in this case, it is inferred that all of that molecule's conformations are inactive.

Other applications of MIC include automatic image annotation (Andrews et al., 2003), context-based image indexing (Maron & Ratan, 1998), text categorization (Andrews et al., 2003) and hard-drive failure prediction (Murray et al., 2005). Algorithms for MIC stem from diverse density (Maron & Ratan, 1998; Zhang & Goldman, 2001), neural networks (Ramon & Raedt, 2000), and generalisations of support vector machines (Andrews et al., 2003; Mangasarian & Wild, 2008). The drug discovery application later inspired Ray & Davis (2001) to formulate multiple instance regression, where this time the response assigned to each bag is a real number quantifying the activity of the molecules.

Multiple instance ranking differs in that a classification label is not known for each bag. Rather, some preference information is available for pairs of bags. For example, it may be known that bag A ranks higher than both bags B and C, while the relative ranking of bags B and C may not be known. In many applications, even more structure exists. In these cases, it is convenient to think of every bag as belonging to a box. Within each box, exactly one bag ranks higher than the other ones in the box, and this bag is designated the preferred bag. It is not known how the other bags in the box rank with respect to each other. Further, it is not known how bags rank with respect to each other across boxes. Additionally, there remains the ambiguity of which items in the preferred bags are preferred and which ones are not preferred. Figure 2 illustrates the situation. Large rectangles represent boxes. As was the case in Figure 1, bags are ellipses, items in preferred bags are represented as stars and items in the other bags are marked as parallelograms. Instead of being fixed, the separating line (representing the ranking function) slides from one box to the next. For each box, the ranking function separates at least one item of the preferred bag from the remaining items of the box.

The hydrogen abstraction application fits perfectly into this framework. For each molecule (box), the task

is to find the group (bag) from which a hydrogen atom (item) is abstracted. It is not known which hydrogen atom is abstracted, only to which group it belongs.

The organization of this paper is as follows. Section 2 defines some mathematical notation. Section 3 motivates multiple instance ranking through the computational chemistry problem of hydrogen abstraction. Multiple instance ranking is formulated, and an algorithm for MIRank is proposed, in Section 4. The model and algorithm are generalized to nonlinear MIRank problems in Section 5. Section 6 describes the datasets used in this paper, and Section 7 specifies the modeling results. Finally, Sections 8 and 9 constitute a discussion and outlook, respectively.

2. Notation

Let $\boldsymbol{x}$ denote a vector in R^n and let $\boldsymbol{x}^T$ mark the transpose of $\boldsymbol{x}$. Let $\mathbf{0}$ denote the vector of all zeros and $\boldsymbol{e}$ denote the vector of all ones. Let $|\boldsymbol{x}|$ denote the cardinality of $\boldsymbol{x}$, that is, the number of entries in the vector. Let $\|\boldsymbol{x}\|_1$ denote the 1-norm of $\boldsymbol{x}$, equal to the sum of the absolute values of the entries of the vector. If $\boldsymbol{x}$ has nonnegative entries, then this equals $\boldsymbol{e}^T\boldsymbol{x}$. Let $X \in R^{k\times n}$ and $H \in R^{m\times n}$ denote matrices. I and J indicate index sets. The cardinality of the set I is indicated by $|I|$. The matrix X_I indicates the matrix in $R^{|I|\times n}$ with rows restricted to the index set I. A kernel matrix $K(X, H')$ maps $R^{k\times n}$ and $R^{n\times m}$ into $R^{k\times m}$. Each entry of the mapping results from a function (such as the radial basis function) applied to one row of X applied to one row of H.

3. Motivating application

Bioavailability of a drug, or its ability to be administered orally, is a major concern to the pharmaceutical industry. This characteristic depends on a drug's capability to withstand degradation by intestinal and hepatic enzymes during first-pass metabolism in order to cross the intestinal lining and make it into the bloodstream so that its medicinal effect may be felt (Thummel et al., 1997). Hence, this process of drug metabolism needs to be better understood. More specifically, it is important to discover the attributes of molecules that identify sites which are vulnerable to enzymatic degradation.

Cytochrome CYP3A4 is but one of many metabolising enzymes found in the human liver and small intestine, yet this enzyme metabolises nearly 50% of marketed drugs (Guengerich, 1999; Rendic, 1997). For CYP3A4 substrates, approximately half of the known metabolism reactions occur via hydroxylation, the rate limiting step of which is hydrogen atom abstraction (Sheridan et al., 2007). Knowing where a molecule is preferentially oxidized by this cytochrome would aid the modification of compounds to improve their kinetic or pharmacological profiles (Afzelius et al., 2007).

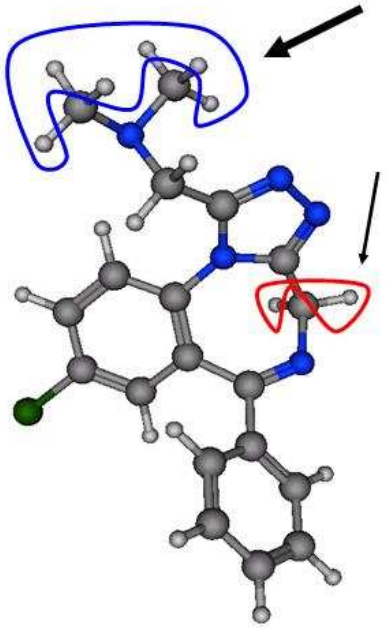


Figure 3. Stick model of an Adinazolam molecule. Large spheres represent nonhydrogen atoms while small spheres represent hydrogen atoms. Two groups of hydrogens are evidenced. The top group, indicated by a thick arrow, has a hydrogen abstracted during metabolism. The lower group, indicated by a thin arrow, does not.

Normally, experimental techniques are used to identify the molecular sites susceptible to metabolism. This is a time- and labor-intensive process. While *in vitro* studies are increasingly high throughput, the *in silico* identification of metabolic liability early on in the drug discovery process will help prevent taking forward poor drug candidates. In addition, the constraints of the biological problem fit perfectly into the framework of a MIRank application, leading to a potential *in silico* solution.

The goal is to build a model that predicts, for each molecule, the site of abstraction of a hydrogen atom during metabolism. In order to accomplish this, individual hydrogen atoms are first grouped together according to molecular equivalence: hydrogens are placed within the same group if and only if the abstraction of any hydrogen from within the group would result in the same metabolised molecule. In this way, groups are equivalent representations of potential sites

of metabolism. Note that experimental data do not show which individual hydrogen is abstracted during metabolism, but rather to which group this hydrogen atom belongs. This setup perfectly fits that of multiple instance ranking. Molecules can be viewed as boxes, groups as bags, and individual hydrogens as items. Figure 3 illustrates these using a stick representation of a molecule.

Two prior modeling attempts are described. Firstly, Singh et al. (2003) chose the hydrogen atom that has the minimum estimated abstraction energy, with a sufficiently large surface area (of 8 squared Angstroms), as the abstracted hydrogen. Allowing 1 guess per molecule, their rule-based model performed correctly in 44% of molecules. Sheridan et al. (2007) later reported that this model has a prediction rate of 51%, allowing for 2 guesses per molecule. Secondly, Sheridan et al. (2007) assembled a database of 316 molecules (including the 50 molecules used by Singh et al. (2003)). They used a random forest applied to molecular descriptors, and found a model that correctly predicted the site of abstraction for 77% of molecules, allowing for 2 guesses per molecule.

4. Formulation

Let (I, J) denote an ordered pair of bags where I and J are lists of indices referring to their items. Let $\boldsymbol{x}_i$ denote a vector of n features for an item i, and let matrix X_I's rows contain the features for each index in I. Further let f denote the ranking function. Then the statement *bag I is preferred over bag J* is expressed mathematically as

$$\max_{i \in I} f(\boldsymbol{x}_i) > \max_{j \in J} f(\boldsymbol{x}_j).$$

The maximum operator on the right hand side can be replaced with all of the items it operates over, hence the inequality is rewritten as

$$\max_{i \in I} f(\boldsymbol{x}_i) > f(\boldsymbol{x}_j) \ \ \forall \ \ j \in J.$$

The maximum operator on the left hand side is also replaced. A convex combination of the items in bag I is taken, following the lead of Mangasarian & Wild (2008) in their formulation of MIC. This convex combination is achieved through vector $\boldsymbol{v_{I,J}}$ whose cardinality is that of I. In a slight abuse of notation, $\boldsymbol{v_{I,J}}$ means the vector corresponding to the pair of bags (I, J). This vector is nonnegative $\boldsymbol{v_{I,J}} \geq 0$, and its entries sum to one: $\boldsymbol{e}^T \boldsymbol{v_{I,J}} = 1$. This vector multiplies matrix X_I:

$$f(X_I^T \boldsymbol{v_{I,J}}) > f(\boldsymbol{x}_j) \ \ \forall \ \ j \in J.$$

Let the model be linear defined by vector $\boldsymbol{w}$, i.e.

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w}. \tag{1}$$

In this case, we have

$$\boldsymbol{v_{I,J}}^T X_I \boldsymbol{w} > \boldsymbol{x}_j^T \boldsymbol{w}.$$

This paper focuses on linear models, because chemists are interested model interpretation. However, this formulation is readily kernelized, as discussed in Section 5.

Now introduce an empirical risk scalar $\xi_{I,j}$ based on the hinge-loss, allowing for errors in training the model:

$$\boldsymbol{v_{I,J}}^T X_I \boldsymbol{w} - \boldsymbol{x}_j^T \boldsymbol{w} \geq 1 - \xi_{I,j}.$$

This inequality resembles the main constraint in Joachims' ranking support vector machine (2002). It is also the key constraint in an optimization problem whose objective function is to minimize

$$\nu \mathcal{L}_{emp}(\boldsymbol{\xi}) + \mathcal{L}_{reg}(\boldsymbol{w})$$

where $\nu > 0$ is the tradeoff parameter and $\mathcal{L}_{emp}$ and $\mathcal{L}_{reg}$ are arbitrary loss functions.

Choosing the 1-norm for both loss functions makes the objective linear in the variables, a choice that was also made by Mangasarian & Wild (2008). Furthermore, using the 1-norm on $\boldsymbol{w}$ makes for sparse models, facilitating the interpretability of linear models. Thereupon, the 1-norm MIRank optimization problem is

$$\min_{\boldsymbol{\xi}, \boldsymbol{w}, \boldsymbol{v_{I,J}}} \quad \nu \boldsymbol{e}^T \boldsymbol{\xi} + \|\boldsymbol{w}\|_1 \tag{2}$$

subject to

$$\boldsymbol{v_{I,J}}^T X_I \boldsymbol{w} - \boldsymbol{x}_j^T \boldsymbol{w} \geq 1 - \xi_{I,j} \ \ \forall \ \ (I, J, j) \tag{3}$$

$$\boldsymbol{e}^T \boldsymbol{v_{I,J}} = 1 \ \ \forall \ \ (I, J) \tag{4}$$

$$\boldsymbol{v_{I,J}} \geq \boldsymbol{0} \ \ \forall \ \ (I, J) \tag{5}$$

$$\boldsymbol{\xi} \geq \boldsymbol{0}. \tag{6}$$

The entries of empirical risk vector $\boldsymbol{\xi}$ are $\xi_{I,j}$ as they appear in the first constraint. This notation signifies that, for each pair (I, J), there is an empirical risk contribution from each item $j \in J$. These are nonnegative quantities, as per 6. Note that there are as many vectors $\boldsymbol{v_{I,J}}$ as there are pairs (I, J). These vectors are forced to be nonnegative and to sum to one in constraints 4 and 5.

Since the first constraint is linear and the remaining terms are linear, this is a bilinear optimization problem. We use the successive linear programming algorithm given in Algorithm 1 to find a locally optimal

Algorithm 1 Multiple instance ranking algorithm

Select tolerance τ and tradeoff parameter ν.
Initialise $\boldsymbol{u_{I,J}} = \frac{\boldsymbol{e}}{|I|} \ \forall \ (I, J)$.
repeat
  Set $\boldsymbol{v_{I,J}} = \boldsymbol{u_{I,J}} \ \forall \ (I, J)$.
  Fix the $\boldsymbol{v_{I,J}}$'s and solve the linear program 2-6 for $\boldsymbol{\xi}$ and $\boldsymbol{w}$.
  Fix $\boldsymbol{w}$ and solve the linear program 2-6 for $\boldsymbol{\xi}$ and the $\boldsymbol{u_{I,J}}$'s.
until $\|\boldsymbol{v_I} - \boldsymbol{u_I}\|_1 \leq \tau \ \forall \ (I, J)$

solution of the bilinear problem. This proposed MI-Rank algorithm belongs to a family of algorithms that has proven to find good local solutions on a variety of bilinear machine learning problems. The subproblem solutions are not necessarily unique, but this has no impact on algorithm convergence.

The convergence proof for the MIC algorithm in Mangasarian & Wild (2008) can be readily adapted to Algorithm 1. Specifically, the algorithm converges because the sequence of objective function values

$$\{\nu \boldsymbol{e}^T \boldsymbol{\xi} + \|\boldsymbol{w}\|_1\}$$

at each iteration is nonincreasing and bounded below by zero, and every accumulation point satisfies a local minima property. The formal proof is omitted for brevity; see Mangasarian & Wild (2008).

Algorithm 1, as well the Mangasarian & Wild (2008) algorithm for MIC, were implemented in Matlab using the linear programming solver MOSEK (`www.mosek.com`).

5. Nonlinear Formulation

A nonlinear MIRank function can be generated by kernel transformations (Shawe-Taylor & Cristianini, 2004). We adopt the notation and direct kernel approach used for MIC in Mangasarian & Wild (2008). The linear ranking function 1 is replaced by the nonlinear function:

$$f(\boldsymbol{x}) = K(\boldsymbol{x}^T, H^T)\boldsymbol{\alpha} \tag{7}$$

where $\boldsymbol{x} \in R^n$ is an item, $\boldsymbol{\alpha} \in R^m$ are the dual variables and the matrix $H \in R^{n \times m}$ has as its rows all of the m items found collectively in all of the bags and boxes, and $K(\boldsymbol{x}^T, H^T)$ is an arbitrary kernel map. The bilinear program generating the nonlinear MI-Rank function becomes:

$$\min_{\boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{v_{I,J}}} \quad \nu \boldsymbol{e}^T \boldsymbol{\xi} + \|\boldsymbol{w}\|_1 \tag{8}$$

subject to

$$\boldsymbol{v_{I,J}}^T K(X_I, H^T)\boldsymbol{w} - K(\boldsymbol{x}_j^T, H^T)\boldsymbol{\alpha} \geq 1 - \xi_{I,j} \ \forall \ (I, J, j) \tag{9}$$

$$\boldsymbol{e}^T \boldsymbol{v_{I,J}} = 1 \ \forall \ (I, J) \tag{10}$$

$$\boldsymbol{v_{I,J}} \geq \boldsymbol{0} \ \forall \ (I, J) \tag{11}$$

$$\boldsymbol{\xi} \geq \boldsymbol{0}. \tag{12}$$

The kernel formulation remains a bilinear program and thus can be solved using Algorithm 1 by substituting $\boldsymbol{\alpha}$ for $\boldsymbol{w}$ and bilinear program 8-12 for bilinear program 2-6.

6. Datasets

In addition to the hydrogen abstraction dataset, several additional datasets are used in modeling experiments. All three are described here.

6.1. CYP3A4 substrate dataset

The CYP3A4 substrate dataset is made up of 227 small drug-like compounds. A series of 36 descriptors for each hydrogen atom for all molecules are calculated:

- the charge of the hydrogen;
- the surface area of the hydrogen;
- the non hydrogen surface area of the base atom the hydrogen is attached to;
- the hydrophobic moment: the hydrogen's location with regards to the hydrophobic or hydrophilic end of the molecule;
- the span: a measure of whether the candidate hydrogen is located at the end or within the middle of the molecule.
- the topological neighborhood: the distributions of atom types within a various topological distances from the hydrogen.

Recall that, for each molecule, the goal is to predict from which group a hydrogen atom is abstracted, and it is not known which hydrogen from the abstracted site is removed.

These 227 molecules form are a subset of the 305 non-proprietary molecules used by Sheridan et al. (2007), and represent all those for which descriptor generation could be completed.

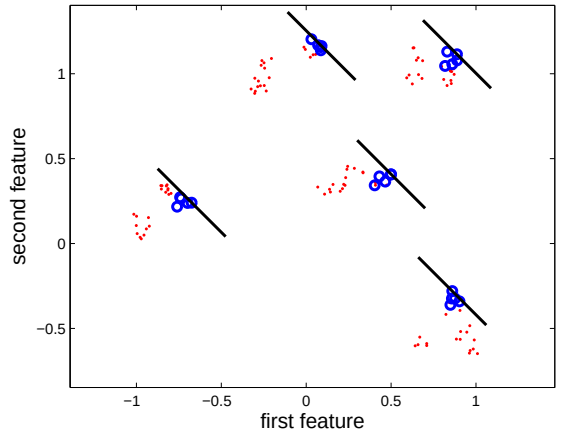


Figure 4. Synthetic dataset visualisation. Preferred bags contain circles and other bags contain dots. Sliding line represents the ranking function found by MIRank that separates at least one circle from remaining items in each box.

6.2. Synthetic datasets

This dataset consists of 227 boxes, five bags per box and five items per bag. There are two features. Each feature is calculated as follows:

$$\mu_i^{box} + \mu_j^{bag} + \mu_k^{item}$$

with μ_i^{box} drawn from the uniform distribution $\mathcal{U}(-1, 1)$, μ_j^{bag} drawn from the distribution $\mathcal{U}(-A, A)$ and μ_k^{item} drawn from the distribution $\mathcal{U}(-B, B)$. Put in words, the center of each box is chosen from a uniform distribution, and the center of each bag with respect to its box is chosen from a different uniform distribution, and each item with respect to its bag is chosen from yet another uniform distribution. Parameters A and B characterize these synthetic datasets. For each item, the response is the sum of the features. The goal is, for each box, to find the bag containing the item of greatest response. Five boxes of this dataset are portrayed as Figure 4. It illustrates the difficulty in constructing a linear function separating at least one circle from each box from the remaining circles and dots, as MIC attempts to do. On the other hand, it it possible to find a ranking function (the sliding line) that does this for each box, as MIRank does.

Different values for dataset parameters A and B were attempted:

- Synthetic-1 set $A = B = 0.01$.
- Synthetic-2 set $A = 0.1$ and $B = 0.01$.

Table 1. Prediction accuracies

Dataset	MIC	MIRank
CYP3A4 substrate	$67.1\% \pm 7.1$	$70.9\% \pm 6.9$
Synthetic-1	$90.8\% \pm 8.6$	$99.8\% \pm 0.53$
Synthetic-2	$96.8\% \pm 4.6$	$99.1\% \pm 1.8$
Synthetic-3	$95.5\% \pm 8.3$	$99.9\% \pm 0.38$
Synthetic-4	$95.7\% \pm 5.2$	$99.7\% \pm 0.91$
Census-16h	$52.8\% \pm 17.4$	$60.3\% \pm 15.1$
Census-16l	$46.2\% \pm 17.7$	$57.5\% \pm 16.0$

- Synthetic-3 set $A = 0.01$ and $B = 0.1$.
- Synthetic-4 set $A = B = 0.1$.

6.3. Census datasets

The two census datasets (census-16h and census-16l) belong to the *Data for Evaluating Learning in Valid Experiments* (DELVE, `http://www.cs.toronto.edu/~delve/`) repository. It consists of 22784 towns spread amongst the 50 states of the United States of America. This study only considered the 3054 towns of more than 10000 inhabitants. Each town is assigned a 5-digit Federal Information Processing Standard (FIPS) place code (that is not a zip code). Typically, this dataset is used in a regression setting to model the response—which is the town's median housing unit price. The census-16h and census-16l datasets differ in their features: each consists of 16 features drawn from the 1990 census.

These datasets are fitted into the multiple instance ranking framework as follows. States are boxes, divisions of towns are bags and towns are items. For each state, towns whose place code begin with the same number are assigned to the same division. As no place code commences with the number 9, there are up to 9 divisions per state. The task is to predict, for each state, the division that contains the town with the highest median housing unit price.

7. Results

For each dataset, results were obtained using both the MIC and MIRank algorithms. For MIC, preferred bags were treated as active bags and other bags were treated as inactive bags. All results reported are for linear functions.

The experimental design is as follows. Each dataset was randomly split into training, validation and testing subsets consisting of 60%, 20% and 20% of the boxes, respectively. The training subset was used to

Table 2. Hypothesis testing

Dataset	P-value
CYP3A4 substrate	$5.59 \cdot 10^{-3}$
Synthetic-1	$1.62 \cdot 10^{-6}$
Synthetic-2	$1.31 \cdot 10^{-2}$
Synthetic-3	$5.84 \cdot 10^{-3}$
Synthetic-4	$1.46 \cdot 10^{-4}$
Census-16h	$4.51 \cdot 10^{-2}$
Census-16l	$3.92 \cdot 10^{-4}$

train both MIC and MIRank models for 19 values of tradeoff parameter ν spread logarithmically over the range $[10^{-3}, 10^{6}]$. The model corresponding to the value of ν that resulted in the best prediction accuracy over the validation set was retained, and a prediction using this model was obtained for the testing subset. This process was repeated 32 times, and the average performance across these 32 testing subsets is reported in Table 1, along with the standard deviation as a measure of spread.

All results in Table 1 are presented as a percentage of boxes for which the preferred bag was accurately predicted, allowing for 2 guesses per box, which is the metric employed by Sheridan et al. (2007). The algorithm tolerance τ defined in Algorithm 1 was set to 10^{-3}.

For all datasets, the hypothesis that MIC and MIRank results are statistically equal is dismissed using paired t-testing at a 5% significance level. The p-values are reported in Table 2.

8. Discussion

The results of Section 7 make a strong case supporting the hypothesis that these problems, when framed in a multiple instance ranking paradigm, are better solved by an algorithm that is designed to solve problems of that paradigm over one that is not. Forcing MIRank problems into a MIC paradigm was not as successful. In other words, the improvement is due to choosing a model that better fits the problem.

The MIRank result for the CYP3A4 substrate dataset reported in this paper compare favourably with existing approaches to hydrogen abstraction. It clearly outperforms the results of Singh et al. (2003). Their results are reproducible and their reported error holds on new molecules. Comparison with Sheridan et al. (2007) is more difficult. Reproduction of their results is challenging since since their descriptors are not public and the details of the learning and model selection methods they used are not entirely clear. Our descriptors attempt to reproduce those of Sheridan et al. (2007), but could not be generated for all molecules. Hence, we regard their results as optimistic.

A future controlled experiment is needed to fully compare the approaches of Sheridan et al. (2007) and those of this paper. This experiment would validate which descriptor set and modeling paradigm is most well suited for this chemistry application. To facilitate future investigations into MIRank and hydrogen abstraction, the datasets and Matlab source codes used in this paper are available from `http://www.rpi.edu/~bennek/MIRank/`.

9. Conclusion

This paper introduced a framework that tackles a novel machine learning question arising from an important chemistry problem. A first working algorithm produces excellent results on it and other problems. We believe that this first paper for MIRank will generate future research into new algorithms and applications. This section explores several possible extensions.

In the chemistry domain, we often restrict ourselves to sparse and linear models because model interpretability is a desired property in the particular application of drug discovery. However, this interpretability analysis is a paper of its own, and does not appear here.

Hydrogen abstraction is an important application of MIRank modeling of great practical value for drug discovery. We are working to expand the efficacy and applicability of the MIRank hydrogen abstraction models in several ways. First, we are increasing the number of molecules in the database of CYP3A4 substrates that can be used to develop and test new MIRank models. Second, we hope to build databases and models for new substrates, such as CYP2D6 and CYP2C9. Third, we are developing novel descriptors that are believed to be indicative of hydrogen abstraction.

We are working to improve the MIRank modeling paradigm and investigating other potential multiple instance ranking problems. Reports here are limited to the linear MIRank models, but as discussed the approach can be readily applied with nonlinear models using kernel functions. Research is needed to investigate how modeling results are affected by changing the loss functions in the empirical risk and/or regularization terms of the optimization problem.

Finally, further improvements to the MIRank algo-

rithm are possible. More scalable and efficient algorithms for finding locally optimal solutions could be developed by exploiting recent developments in large scale support vector machine algorithms. In addition, integer programming or cutting plane algorithms could be used to find global minima of the optimization problem, but at much greater computational cost.

Acknowledgments

Charles Bergeron is under fellowship from the *Fonds québécois de la recherche sur la nature et les technologies*. This work was done under, and all authors belong to, the Rensselaer Exploratory Center for Cheminformatics Research (RECCR, `reccr.chem.rpi.edu`).

References

Afzelius, L., Arnby, C. H., Broo, A., Carlsson, L., Isaksson, C., Jurva, U., Kjellander, B., Kolmodin, K., Nilsson, K., Raubacher, F., & Weidolf, L. (2007). State-of-the-art tools for computational site of metabolism predictions: Comparative analysis, mechanistical insights, and future applications. *Drug Metabolism Reviews*, *39*, 61–86.

Andrews, S., Tsochantaridis, I., & Hofmann, T. (2003). Support vector machines for multiple-instance learning. *Advances in Neural Information Processing Systems 15*.

Dietterich, T. G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Guengerich, F. P. (1999). Cytochrome p-450 3A4: Regulation and role in drug metabolism. *Annual Review of Pharmacology and Toxicology*, *39*, 1–7.

Joachims, T. (2002). Optimizing search engines using clickthrough data. *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 133–142).

Mangasarian, O. L., & Wild, E. W. (2008). Multiple instance classification via successive linear programming. *Journal of Optimization Theory and Applications*, Accepted.

Maron, O., & Ratan, A. L. (1998). Multiple-instance learning for natural scene classification. *Proceedings of the 15th International Machine Learning Conference*.

Murray, J. F., Hughes, G. F., & Kreutz-Delgado, K. (2005). Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning*, *6*, 783–816.

Ramon, J., & Raedt, L. D. (2000). Multi instance neural networks. *Proceedings of the 17th International Machine Learning Conference*.

Ray, S., & Page, D. (2001). Multiple instance regression. *Proceedings of the 18th International Machine Learning Conference*.

Rendic, S. (1997). Summary of information on human CYP enzymes: human P450 metabolism data. *Drug Metabolism Reviews*, *34*, 83–448.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.

Sheridan, R. P., Korzekwa, K. R., Torres, R. A., & Walker, M. J. (2007). Empirical regioselectivity models for human cytochromes P450 3A4, 2D6, and 2C9. *Journal of Medicinal Chemistry*, *50*, 3173–3184.

Singh, S. B., Shen, L. Q., Walker, M. J., & Sheridan, R. P. (2003). A model for predicting likely sites of CYP3A4-mediated metabolism on drug-like molecules. *Journal of Medicinal Chemistry*, *46*, 1330–1336.

Thummel, K. E., Kunzea, K. L., & Shen, D. D. (1997). Enzyme-catalyzed processes of first-pass hepatic and intestinal drug extraction. *Advanced Drug Delivery Reviews*, *27*, 99–127.

Zhang, Q., & Goldman, S. A. (2001). EM-DD: An improved multiple-instance learning technique. *Advances in Neural Information Processing Systems 14*.

Multi-Task Learning for HIV Therapy Screening

Steffen Bickel BICKEL@MPI-INF.MPG.DE
Jasmina Bogojeska JASMINA@MPI-INF.MPG.DE
Thomas Lengauer LENGAUER@MPI-INF.MPG.DE
Tobias Scheffer SCHEFFER@MPI-INF.MPG.DE
Max Planck Institute for Computer Science, Saarbrücken, Germany

Abstract

We address the problem of learning classifiers for a large number of tasks. We derive a solution that produces resampling weights which match the pool of all examples to the target distribution of any given task. Our work is motivated by the problem of predicting the outcome of a therapy attempt for a patient who carries an HIV virus with a set of observed genetic properties. Such predictions need to be made for hundreds of possible combinations of drugs, some of which use similar biochemical mechanisms. Multi-task learning enables us to make predictions even for drug combinations with few or no training examples and substantially improves the overall prediction accuracy.

1. Introduction

In *multi-task learning* one seeks to solve many classification problems in parallel. Some of the classification problems will likely relate to one another, but one cannot assume that the tasks share a joint conditional distribution of the class label given the input variables. The challenge of multi-task learning is to come to a good generalization across tasks: each task should benefit from the wealth of data available for the entirety of tasks, but the optimization criterion needs to remain tied to the individual task at hand.

Our work is motivated by the problem of predicting the therapeutic success of a given combination of drugs for a given strain of the *Human Immunodeficiency Virus-1* (HIV-1). HIV is associated with the *acquired immunodeficiency syndrome* (AIDS). Being a disease that claimed more than 25 million lives since 1981, AIDS is one of the most destructive epidemics in recorded history. Currently there are more than 33 million people infected with HIV (UNAIDS/WHO, 2007).

Antiretroviral therapy is hampered by HIV's strong ability to mutate and develop viral quasi-species that can quickly be dominated by resistant variants. In order to decide on a course of therapy, virus samples taken from each individual patient are tested for a set of resistance-relevant mutations. Given this set of identified mutations together with the patient's medication history, a medical practitioner needs to decide which combination of drugs to administer. The large number of genetic mutations and the wide array of available drug combinations render the process of predicting the success of a potential therapy difficult, at best, for a human doctor.

Historic treatment records of HIV patients cover only a small portion of all possible drug combinations. For many of these combinations, only few treatments have been recorded. This scarceness of training data precludes separate training of a powerful prediction model for each combination from only records of treatments which used the same drug combination. Distinct combinations can have similar effects when they intersect in jointly contained drugs, or when they include drugs that use similar mechanisms to affect the virus. Therefore, in order to predict the outcome of a given drug combination, it is desirable to exploit data from related combinations and thereby achieve generalization over both virus mutations and combinations of drugs.

We contribute a new multi-task learning model that can handle arbitrarily different data distributions for different tasks without making assumptions about the data generation process or the relation between tasks. We show that by appropriately weighting each instance in the pool of all examples, one can match the distribution that governs the pool of examples of all tasks to each of the single task distributions. We show

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

how appropriate weights can be obtained by discriminating the labeled sample for a given task against the pooled sample.

The rest of this paper is structured as follows. After formalizing the problem setting in Section 2, we review related transfer learning models in Section 3. We devise the model for multi-task learning by distribution matching in Section 4. In Section 5 we describe the data sets and the experimental setting and report on experimental results. Section 6 concludes.

2. Problem Setting

In *supervised multi-task learning*, each of several tasks z is characterized by an unknown joint distribution $p(\mathbf{x}, y|z)$ of features $\mathbf{x}$ and label y given the task z. The joint distributions of different tasks may differ arbitrarily but usually some tasks have similar distributions. A training sample $D = \langle(\mathbf{x}_1, y_1, z_1), \ldots, (\mathbf{x}_m, y_m, z_m)\rangle$ collects examples from all tasks. There may be tasks with no data. For each example, input attributes $\mathbf{x}_i$, class label y_i, and the originating task z_i are known. The entire sample D is governed by the mixed joint density $p(z)p(\mathbf{x}, y|z)$. The prior $p(z)$ specifies the task proportions.

The goal is to learn a hypothesis $f_z : \mathbf{x} \mapsto y$ for each task z. This hypothesis $f_z(\mathbf{x})$ should correctly predict the true label y of unseen examples drawn from $p(\mathbf{x}|z)$ for all z. That is, it should minimize the expected loss

$$\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|z)}[\ell(f_z(\mathbf{x}), y)]$$

with respect to the unknown joint distribution $p(\mathbf{x}, y|z)$ for each individual z.

This abstract problem setting models the HIV therapy screening application as follows. Input $\mathbf{x}$ describes the genotype of the virus that a patient carries, together with the patient's treatment history. Genotype information is encoded as a binary vector indicating the presence and absence of each out of a predefined set of resistance-relevance mutations, respectively. The treatment history can be represented as a binary vector indicating which drugs have been administered over the course of past treatments. A candidate combination of drugs plays the role of the task z: each task has an associated binary vector $\mathbf{z}$ that indicates a set of drugs that a medical practitioner is currently giving consideration. The binary class label y indicates whether the therapy will be successful.

In addition to training data, we may have prior knowledge on the similarity of tasks which is encoded in a kernel function $k(z, z')$. Prediction models for different drug combinations can be similar because the sets of drugs intersect (we will later refer to this as the drug feature kernel), or because similar sets of mutations in the virus render the drugs in the set ineffective (mutation table kernel).

3. Prior Work

One obvious strategy for multi-task learning is to learn independent models for each target task t by minimizing an appropriate loss function on the portion of $D_t = \{(\mathbf{x}_i, y_i, z_i) \in D : z_i = t\}$. The other extreme could be a one-size-fits-all model $f_*(\mathbf{x})$ trained on the entire sample.

In many applications, task-level descriptions or prior knowledge on task similarity encoded in a kernel are available. Bonilla et al. (2007) study an extension of the one-size-fits-all model and find that training with a kernel defined as the multiplication of an input feature kernel and a task-level kernel outperforms a gating network. Task-level features have also been utilized for task clustering and for a task-dependent prior on the model parameters (Bakker & Heskes, 2003).

Another simple extension to the one-size-fits-all model would be to train a model for a target task from all data with weighted examples from other tasks, using one fixed uniform weight for each task. Such a model is described by Wu and Dietterich (2004).

Our work is inspired by learning under covariate shift. In the covariate shift setting the marginals $p_{train}(\mathbf{x})$ and $p_{test}(\mathbf{x})$ of training and test distributions differ, but the conditionals are identical $p_{train}(y|\mathbf{x}) = p_{test}(y|\mathbf{x})$. If training and test distributions were known, then the loss on the test distribution could be minimized by weighting the loss on the training distribution with an instance-specific factor. Shimodaira (2000) illustrates that the scaling factor has to be $\frac{p_{test}(\mathbf{x})}{p_{train}(\mathbf{x})}$. Bickel et al. (2007) derive a discriminative expression for this marginal density ratio that can be estimated – without estimating the potentially high-dimensional densities of training and test distributions – by discriminating training against test data.

Hierarchical Bayesian models for multi-task learning are based on the assumption that task-specific model parameters are drawn from a common prior. The task dependencies are captured by estimating the common prior. Yu et al. (2005) impose a normal-inverse Wishart hyperprior on the mean and covariance of a Gaussian process prior that is shared by all task-specific regression functions. Mean and covariance of the Gaussian process are estimated using the EM algorithm. A Dirichlet process can serve as prior in a hierarchical Bayesian model and cluster the tasks (Xue

et al., 2007); all tasks in one cluster share the same model parameters. Evgeniou and Pontil (2004) derive a kernel that is based on a hierarchical Bayesian model with Gaussian prior (covariance matrix is scalar) on the parameters of a regularized regression.

Larder et al. (2007) tackle the problem of predicting virological response to a given HIV drug combination with neural networks. Lathrop and Pazzani (1999) apply combinatorial optimization to the same problem using features extracted from the viral genotype and the drugs in the combination. Altmann et al. (2007) approach the problem by including various phenotypic information and an estimate of future evolutionary development of the virus in the learning process.

4. Multi-Task Learning by Distribution Matching

In learning a classifier $f_t(\mathbf{x})$ for target task t, we seek to minimize the loss function with respect to $p(\mathbf{x}, y|t)$. Simply pooling the available data for all tasks would create a sample governed by $\sum_z p(z)p(\mathbf{x}, y|z)$. Our approach now is to create a task-specific resampling weight $r_t(\mathbf{x}, y)$ for each element of the pool of examples. The sampling weights match the pool to the target distribution $p(\mathbf{x}, y|t)$. The weighted sample is governed by the correct target distribution, but is still larger as it draws from the sample pool for all tasks.

Instead of sampling from the pool, one can weight the loss incurred by each instance by the resampling weight. The expected weighted loss with respect to the mixture distribution that governs the pool equals the loss with respect to the target distribution $p(\mathbf{x}, y|t)$. Equation 1 defines the resampling weights.

$$\begin{aligned} &\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|t)}[\ell(f(\mathbf{x},t),y)] \qquad (1)\\ &\quad = \mathbf{E}_{(\mathbf{x},y)\sim \sum_z p(z)p(\mathbf{x},y|z)}\left[r_t(\mathbf{x},y)\ell(f(\mathbf{x},t),y)\right] \end{aligned}$$

In the following, we will show that

$$r_t(\mathbf{x}, y) = \frac{p(\mathbf{x}, y|t)}{\sum_z p(z)p(\mathbf{x}, y|z)}$$

satisfies Equation 1. Equation 2 expands the expectation and introduces a fraction that equals one. Equation 3 expands the sum over z in the numerator to run over the entire expression because the integral over $(\mathbf{x}, y)$ is independent of z. Equation 4 is the expected loss over the distribution of all tasks weighted by $\frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$.

$$\begin{aligned} &\mathbf{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|t)}[\ell(f(\mathbf{x},t),y)] \qquad (2)\\ &\quad = \int \frac{\sum_z p(z)p(\mathbf{x},y|z)}{\sum_{z'} p(z')p(\mathbf{x},y|z')} p(\mathbf{x},y|t)\ell(f(\mathbf{x},t),y)d\mathbf{x}dy\\ &\quad = \int \sum_z \Bigg(p(z)p(\mathbf{x},y|z)\frac{p(\mathbf{x},y|t)}{\sum_{z'} p(z')p(\mathbf{x},y|z')} \qquad (3)\\ &\qquad\qquad \ell(f(\mathbf{x},t),y)\Bigg) d\mathbf{x}dy\\ &\quad = \mathbf{E}_{(\mathbf{x},y)\sim \sum_z p(z)p(\mathbf{x},y|z)} \qquad (4)\\ &\qquad \left[\frac{p(\mathbf{x},y|t)}{\sum_{z'} p(z')p(\mathbf{x},y|z')}\ell(f(\mathbf{x},t),y)\right] \end{aligned}$$

Equation 4 signifies that we can train a hypothesis for task t by minimizing the expected loss over the distribution of all tasks weighted by $r_t(\mathbf{x}, y)$. This amounts to minimizing the expected loss with respect to the target distribution $p(\mathbf{x}, y|t)$.

Equation 4 leaves us with the problem of estimating the joint density ratio $r_t(\mathbf{x}, y) = \frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$. One might be tempted to train density estimators for $p(\mathbf{x}, y|t)$ and $\sum_z p(z)p(\mathbf{x}, y|z)$. However, obtaining estimators for potentially high-dimensional densities is unnecessarily difficult because ultimately only a scalar weight is required for each example.

4.1. Discriminative Density Ratio Model

In this section, we derive a discriminative model that directly estimates the resampling weights $r_t(\mathbf{x}, y) = \frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$ without estimating the individual densities. We reformulate the density ratio $\frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$ in terms of a conditional model $p(t|\mathbf{x}, y)$. This conditional has the following intuitive meaning: Given that an instance $(\mathbf{x}, y)$ has been drawn at random from the pool $\cup_z D_z = D$ of samples for all tasks (including D_t); the probability that $(\mathbf{x}, y)$ originates from D_t is $p(t|\mathbf{x}, y)$. The following equations assume that the prior on the size of the target sample is greater than zero, $p(t) > 0$. In Equation 6 Bayes' rule is applied twice and in Equation 7 $p(\mathbf{x}, y)$ and $p(z)$ are canceled out. Equation 8 follows by $\sum_z p(z|\mathbf{x}, y) = 1$.

$$\begin{aligned} r_t(\mathbf{x}, y) &= \frac{p(\mathbf{x}, y|t)}{\sum_z p(z)p(\mathbf{x}, y|z)} \qquad (5)\\ &= \frac{p(t|\mathbf{x}, y)p(\mathbf{x}, y)}{p(t)} \frac{1}{\sum_z p(z)\frac{p(z|\mathbf{x},y)p(\mathbf{x},y)}{p(z)}} \qquad (6)\\ &= \frac{p(t|\mathbf{x}, y)}{p(t)\sum_z p(z|\mathbf{x}, y)} \qquad (7)\\ &= \frac{p(t|\mathbf{x}, y)}{p(t)} \qquad (8) \end{aligned}$$

The significance of Equation 8 is that it shows how the resampling weights $r_t(\mathbf{x}, y) = \frac{p(\mathbf{x},y|t)}{\sum_z p(z)p(\mathbf{x},y|z)}$ can be determined without knowledge of any of the task densities $p(\mathbf{x}, y|z)$. The right hand side of Equation 8 can be evaluated based on a model $p(t|\mathbf{x}, y)$ that discriminates labeled instances of the target task against labeled instances of the pool of examples for all tasks. Intuitively, $p(t|\mathbf{x}, y)$ characterizes how much more likely $(\mathbf{x}, y)$ is to occur in the target distribution than it is to occur in the mixture distribution of all tasks. Instead of potentially high-dimensional densities $p(\mathbf{x}, y|t)$ and $p(\mathbf{x}, y|z)$, a conditional distribution with a single variable needs to be modeled. One can apply any probabilistic classifier to model this conditional distribution.

4.2. Soft-Max Model for Density Ratio Estimation

We model $p(t|\mathbf{x}, y)$ of Equation 8 for all tasks jointly with a soft-max model (the multi-class generalization of the logistic model) with model parameters $\mathbf{v}$, displayed in Equation 9. The parameter vector $\mathbf{v}$ is a concatenation of task-specific subvectors $\mathbf{v}_z$, one for each task z. With this model an estimate for $p(t|\mathbf{x}, y)$ is given by $p(z = t|\mathbf{x}, y, \mathbf{v})$; this is the evaluation of the soft-max model with respect to task t.

$$p(z|\mathbf{x}, y, \mathbf{v}) = \frac{\exp(\mathbf{v}_z^\mathsf{T}\Phi(\mathbf{x}, y))}{\sum_{z'} \exp(\mathbf{v}_{z'}^\mathsf{T}\Phi(\mathbf{x}, y))} \quad (9)$$

Equation 9 requires a problem-specific feature mapping $\Phi(\mathbf{x}, y)$. Without loss of generality we define this mapping for binary labels $y \in \{+1, -1\}$ in Equation 10; δ is the Kronecker delta. In the absence of prior knowledge about the similarity of classes, input features $\mathbf{x}$ of examples with different class labels y are mapped to disjoint subsets of the feature vector.

$$\Phi(\mathbf{x}, y) = \begin{bmatrix} \delta(y, +1)\Phi(\mathbf{x}) \\ \delta(y, -1)\Phi(\mathbf{x}) \end{bmatrix} \quad (10)$$

With this feature mapping the models for positive and negative examples do not interact and can be trained independently.

For training the soft-max model we maximize the regularized log-likelihood of the data. Prior knowledge on the similarity of tasks in the form of a positive semi-definite kernel function $k(z, z')$ can be be encoded in the covariance matrix of a Gaussian prior $N(0, \Sigma)$ on parameter vector $\mathbf{v}$. We set all main diagonal entries of Σ to the scalar parameter $\sigma_\mathbf{v}^2$ and set the secondary diagonal entries corresponding to the covariances between $\mathbf{v}_z$ and $\mathbf{v}'_z$ to $k(z, z')\rho\sigma_\mathbf{v}^2$ (assuming kernel values $0 \leq k(z, z') \leq 1$). Parameter $\sigma_\mathbf{v}^2$ specifies the variance of each element in $\mathbf{v}$. $k(z, z')\rho$ is the correlation coefficient between elements of subvectors $\mathbf{v}_z$ and $\mathbf{v}'_z$; parameter ρ specifies the strength of this correlation. The covariance matrix Σ is required to be invertible and therefore $0 \leq \rho < 1$. All other entries of Σ are set to zero. When prior knowledge on the task similarities is encoded in the prior on the model parameters, then this prior knowledge dominates the optimization criterion for small samples while the data-driven portion of the criterion becomes dominant and overrides prior beliefs as more data arrives.

Optimization Problem 1 *Over parameters* $\mathbf{v}$*, maximize*

$$\sum_{(\mathbf{x}_i, y_i, z_i) \in D} \log(p(z_i|\mathbf{x}_i, y_i, \mathbf{v})) + \mathbf{v}^\mathsf{T}\Sigma^{-1}\mathbf{v}.$$

The solution of Optimization Problem 1 is a maximum *a posteriori* estimation of the soft-max model (Equation 9) over the model parameters $\mathbf{v}$ using a Gaussian prior with covariance matrix Σ. Tasks with no training examples are covered naturally in Optimization Problem 1. In this case, the Gaussian prior with the task kernel $k(z, z')$ encoded in the covariance matrix determines the model.

For our experiments we use a kernelized variant of Optimization Problem 1 by applying the representer theorem. Details on the kernelization of multi-class logistic regression can be learned from Zhu and Hastie (2002).

4.3. Weighted Empirical Loss and Target Model

The multi-task learning procedure first determines resampling weights $r_z(\mathbf{x}, y)$ for all tasks and instances by solving Optimization Problem 1. In this section we describe the second step of training an array of target models, one for each task, using weighted examples.

With the results of Optimization Problem 1 the discriminative expression for the weights of Equation 8 can be estimated. Using these weights we can evaluate the expected loss over the weighted training data as displayed in Equation 11. It is the regularized empirical counterpart of Equation 4.

$$\mathbf{E}_{(\mathbf{x},y)\sim D}\left[\frac{p(t|\mathbf{x}, y, \mathbf{v})}{p(t)}\ell(f(\mathbf{x}, t), y)\right] + \frac{\mathbf{w}_t^\mathsf{T}\mathbf{w}_t}{2\sigma_\mathbf{w}^2} \quad (11)$$

An instance of Optimization Problem 2 is solved for each task independently to produce a separate model for this task. Optimization Problem 2 minimizes Equation 11, the weighted regularized loss over the training data using a standard Gaussian log-prior with variance $\sigma_\mathbf{w}^2$ on the parameters $\mathbf{w}_t$. Each example is weighted by the discriminatively estimated density

fraction from Equation 8 using the solution of Optimization Problem 1.

Optimization Problem 2 *For task t: over parameters $\mathbf{w}_t$, minimize*

$$\sum_{(\mathbf{x}_i,y_i)\in D} \frac{p(t|\mathbf{x}_i,y_i,\mathbf{v})}{p(t)}\ell(f(\mathbf{x}_i,\mathbf{w}_t),y_i) + \frac{\mathbf{w}_t^\mathsf{T}\mathbf{w}_t}{2\sigma_\mathbf{w}^2}.$$

5. HIV Therapy Screening

We model HIV therapy screening as a multi-task learning problem. The input $\mathbf{x}$ to the prediction problem is given by attributes of the viral genotype and the patient's treatment history. The combination of drugs z plays the role of the task. Success or failure of the therapy constitutes class-label y.

In the next subsections we describe the data sets, reference methods, and the empirical results of our study.

5.1. Data Sets and Prior Knowledge on Task Similarity

We use data from the EuResist project (Rosen-Zvi et al., 2008). The data set comprises a total number of 52846 treatment records from the treatment histories of 16999 HIV patients treated in hospitals in the period of 1977 through 2007.

We use two different definitions of therapeutic success and failure to tag the data: *virus load labeling* and *multi-conditional labeling*.

According to our *virus load labeling* definition a therapy is successful if the viral load (number of virus copies per ml blood plasma, cp/ml) drops below the established level of virus detection of 400 cp/ml during the time of the treatment. Otherwise the treatment is a failure. In *multi-conditional labeling*, a therapy is successful if the viral load measured in the time range between 28 and 84 days after the start of the therapy decreases by at least 2 orders of magnitude compared to the most recent viral load measured one to three months before the start of the therapy, or the viral load drops below 400 cp/ml 56 days after the start of the therapy. A drawback of this definition is that due to the strict time intervals it imposes on the measurements, class labels that adhere to this labeling are only available for a small number of records. The *virus load labeling* does not require these strict time intervals by making use of any viral load measurement during the course of therapy to label it.

Out of all available treatment records we extract two different data sets using the two labelings. With the virus load labeling we extract 3260 and with the multi-conditional labeling 2011 treatment records with corresponding ratios of 65.7% and 64.1% successful treatments. The size of these data sets is much smaller than the size of the original data due to missing viral load measurements, or missing virus sequence information.

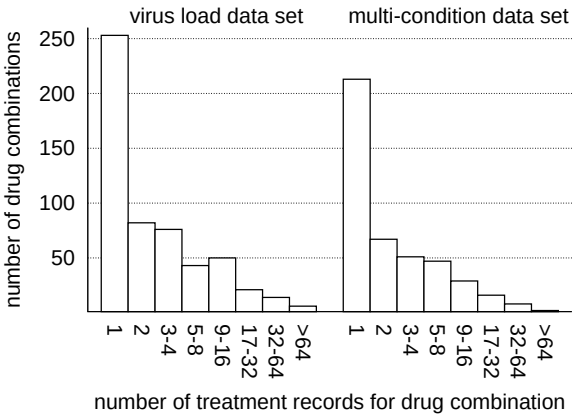


Figure 1. Histogram over number of treatment records for drug combinations (tasks) in the virus load data set (left) and multi-condition data set (right).

A number of 545 distinct drug combinations (tasks z) occur at least once in the virus load data set; 433 occur in the the multi-conditional data set. The histogram over sample sizes per task is displayed in Figure 1. For many combinations, only a few examples occur in the data. For instance, in the virus load data set we observe 253 out of 545 drug combinations with only one data point and 411 with less than 5 instances. Similarly, the multi-conditional data set has 213 out of 433 drug combinations with a single data point and 331 with less than 5 observations.

We extract two types of features for each instance: a genotypic description of the virus and information about the treatment history of the patient. We use the viral genotype taken from the patient shortly before the treatment and represent it by a binary vector indicating the presence of resistance-relevant mutations of the viral sequence (Johnson et al., 2007). Drug-resistant viral quasi-species evolve during the course of the treatment due to selective pressure imposed by the drug. As they remain in the patient's body, the treatment history plays an important role for predicting the outcome of a potential treatment. Hence, we extract all drugs given to the patient in previous treatments and use a binary vector representation with a one entry for each drug given to the patient in the treatment history. The 82-dimensional feature vector $\mathbf{x}$ for each data point results from the concatenation

Table 1. Classification accuracies with standard errors of differences to distribution matching method (ste.Δ). Symbols (•,○,∗,◇) indicate statistical significance according to a paired t-test with significance level $\alpha = 0.05$, (•) compared to separate baseline, (○) compared to pooled baseline, (∗) compared to hierarchical Bayesian kernel baseline, (◇) compared to hierarchical Bayesian Gaussian process baseline.

data set	prior knowledge	separate	ste.Δ	pooled	ste.Δ	hier. Bayes kernel	ste.Δ	hier. Bayes Gauss. proc.	ste.Δ	distribution matching
virus load	none	67.87%	1.80	75.00%	1.47	76.69%	1.39	76.53%	1.36	• ○ ∗ ◇ **79.14%**
	drug.feat.	67.87%	1.76	75.46%	1.39	75.31%	1.34			• ○ ∗ **77.91%**
	mut.table	67.87%	1.78	75.61%	1.37	76.84%	1.16			• ○ ∗ **79.29%**
multi-condition	none	64.64%	2.41	76.67%	1.13	77.17%	1.29	76.43%	1.44	• ○ ∗ ◇ **79.40%**
	drug.feat.	64.64%	2.29	**78.41%**	1.63	75.19%	1.44			• ∗ **78.16%**
	mut.table	64.64%	2.38	**78.66%**	1.11	**77.42%**	1.24			• **79.16%**

of 65 genotypic and 17 historic treatment features.

We have prior knowledge about the similarity of combinations and encode this knowledge into two different task similarity kernels $k(z, z')$. The binary drug indicator vector has an entry for each drug; entries of one indicate the presence of a drug in the combination. The *drug indicator kernel* is the inner product between the normalized drug indicator vectors of two combinations. The *mutation table kernel* is based on tables about the resistance-associated mutations of single drugs (Johnson et al., 2007). We construct binary vectors indicating resistance-relevant mutations for the set of drugs occurring in a combination. The kernel computes the normalized inner product between such binary vectors for two drug combinations.

5.2. Reference Methods

The first reference method is training of a separate logistic regression model for each task without any interaction ("separate"). Tasks without any training examples get a constant classifier that assigns each test example with 50% to each of both classes.

The next baseline is a one-size-fits-all model; all examples are pooled and only one common logistic regression is trained for all tasks ("pooled"). For the experiments with prior knowledge on task similarity we multiply the feature kernel with the task kernel values $k(\mathbf{x}, \mathbf{x}')(k(z, z') + 1)$ and train one model using this kernel (Bonilla et al., 2007). For task kernels that can have a value of zero we include a "+1" term to ensure that the feature kernel does not vanish.

The third reference method ("hier. Bayes kernel") is a logistic regression with the hierarchical Bayesian kernel $k_{hBayes}(\mathbf{x}, \mathbf{x}') = (\lambda + \delta(z, z'))k(\mathbf{x}, \mathbf{x}')$ of Evgeniou and Pontil (2004); $\delta(z, z')$ is the Kronecker delta and λ is a tuning parameter. For the experiments with task similarity kernel the hierarchical Bayes and the task kernel are multiplied. As second hierarchical Bayesian method ("hier. Bayes Gauss. proc.") we use the Gaussian process regression of Yu et al. (2005).

5.3. Experimental Setting and Results

In our experiments we study the benefit of distribution matching for HIV therapy screening compared to the reference methods described in Section 5.2. Optimization Problem 1 is solved with limited-memory BFGS and Optimization Problem 2 with Newton gradient descent using a logistic loss. For the prior term $p(t)$ required in Optimization Problem 2 we use a MAP estimate $\frac{|D_t|+\gamma}{\sum_z(|D_z|+\gamma)}$ with a symmetric Dirichlet prior. We use RBF kernels for all methods.

We apply a training-test split of the data consistent with the dates of the treatment records. We sort the treatment records by date and use the first 80% of the records as training data and the last 20% as test data. This procedure yields 653 and 403 test examples for the virus load and multi-conditional data set, respectively. The date consistent split is necessary because new drugs get approved over time, and under pressure of new drugs the viral population evolves. In such environments, the prediction models should be able to learn from data seen in the *past* and perform well on unseen data in the *future*.

We tune the prior and regularization parameters of all methods, the Dirichlet parameter γ, and the variance of the RBF kernels on tuning data resulting from a date consistent split of the training data.

The evaluation measure is the accuracy of predicting the correct label (success or failure of a treatment)

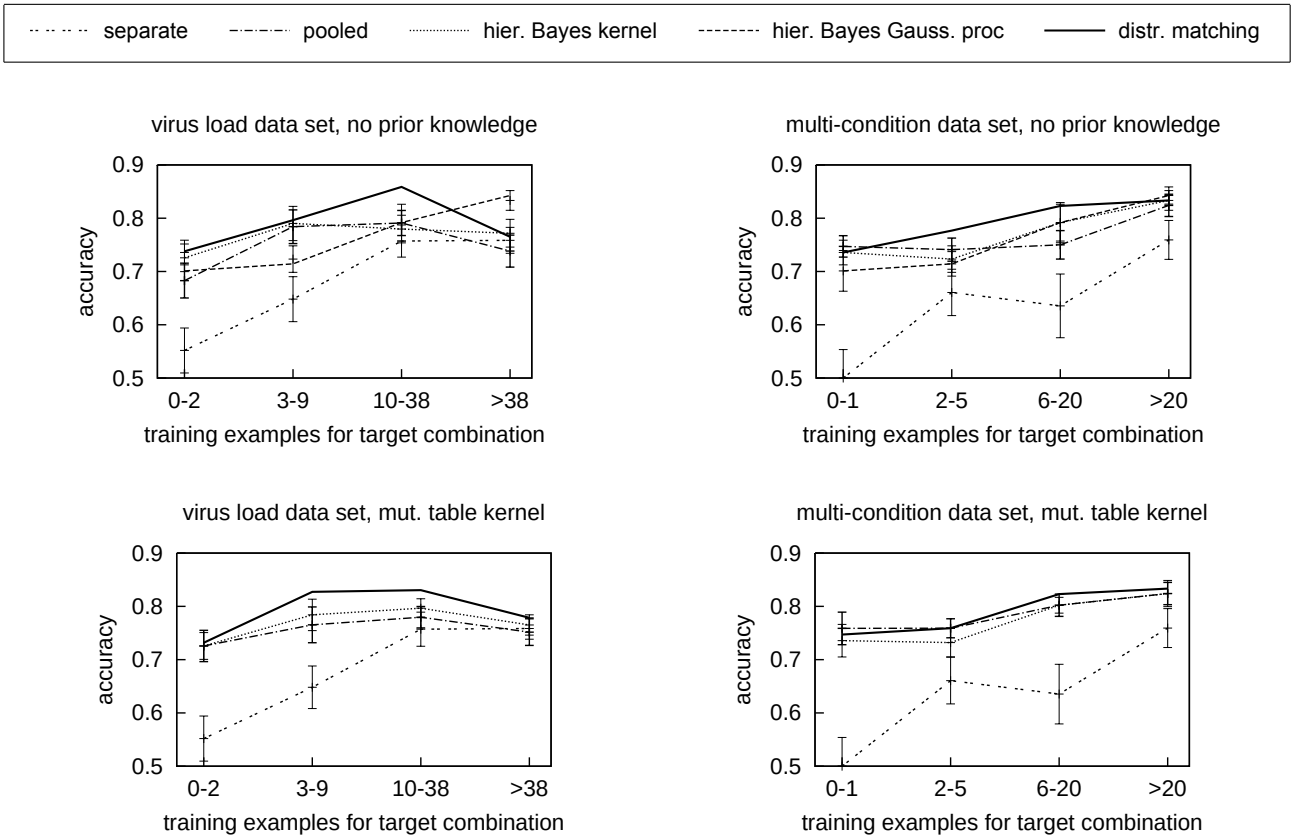


Figure 2. Accuracy over different number of training examples for target combination; virus load data set (left), multi-condition data set (right). Error bars indicate the standard error of the differences to distribution matching. The key can be found in the box right above the diagrams.

on the test set. Table 1 shows the results of the prediction accuracy for all methods over both data sets without and with two different types of prior knowledge on combination similarity. The columns "ste.Δ" placed next to the accuracy columns display the standard error of the differences to the distribution matching method.

Multi-task learning by distribution matching outperforms, or is as good as, the best alternative method in all cases. The improvement over the separate model baseline is about 10-14%. We can reject the null hypothesis that the pooled and the hierarchical Bayesian kernel baseline is at least as accurate as distribution matching in four and five cases respectively out of six according to a paired t-test at $\alpha = 0.05$.

For distribution matching, prior knowledge does not improve the accuracy. The pooled baseline benefits from prior knowledge for the multi-condition data set. For the case without prior knowledge we do not observe a statistically significant difference of the two hierarchical Bayesian methods, but they are both significantly worse than distribution matching according to the paired t-test. Note that the Gaussian process baseline is a regression model; all other methods are classification models.

Figure 2 displays the accuracy over the combinations in the test set grouped by the number of available examples for the settings without and with the mutation table kernel. For instance, an accuracy of 74% for the first group "0-2" means, that only test examples from combinations are selected that have zero, one, or two training examples each, and the accuracy on this subset of the test examples is 74%. Each of the four groups covers about the same number of test examples. The error bars indicate the standard error of the differences to the distribution matching method. Note, that the statistical tests described above are based on all test data and are not directly related to the group-specific error bars in the diagrams.

All methods benefit from larger numbers of training

examples per drug combination. The slightly decreasing accuracy for the virus load data set with ">38" training examples is surprising. Further analysis reveals that in this case there is an accumulation of test examples with history profiles very different from the training examples of the same combination.

For all methods that generalize over the tasks the benefit compared to the separate model baseline is the largest for the smallest group ("0-2" and "0-1" training examples respectively).

6. Conclusion

We devised a multi-task learning method that centers around resampling weights which match the distribution of the pool of examples of multiple tasks to the target distribution for a given task at hand. The method creates a weighted sample that reflects the desired target distribution and exploits the entire corpus of training data for all tasks. We showed how appropriate weights can be obtained by discriminating the labeled sample for a given task against the pooled sample. After weighting the pooled sample, a classifier for the given task can be trained. In our experiments on HIV therapy screening we found that the distribution matching method improves on the prediction accuracy over independently trained models by 10-14%. According to a paired t-test, distribution matching is significantly better than the reference methods for 17 out of 20 experiments.

A combination of drugs is the standard way of treating HIV patients. The accuracy to which the likely outcome of a combination therapy can be anticipated can therefore directly impact the quality of HIV treatments.

Acknowledgment

We thank Kai Yu for providing his Gaussian process implementation and Barbara Pogorzelska who adapted this code and conducted the experiments. We also thank the EuResist project with contract number EU-STREP IST-2004-027173 for providing the data. We gratefully acknowledge support from the German Science Foundation DFG.

References

Altmann, A., Beerenwinkel, N., Sing, T., Savenkov, I., Doumer, M., Kaiser, R., Rhee, S., Fessel, W., Shafer, W., & Lengauer, T. (2007). Improved prediction of response to antiretroviral combination therapy using the genetic barrier to drug resistance. *Antiviral Therapy*, *12*, 169–178.

Bakker, B., & Heskes, T. (2003). Task clustering and gating for Bayesian multitask learning. *The Journal of Machine Learning Research*, *4*, 83–99.

Bickel, S., Brückner, M., & Scheffer, T. (2007). Discriminative learning for differing training and test distributions. *Proceedings of the International Conference on Machine Learning*.

Bonilla, E., Agakov, F., & Williams, C. (2007). Kernel multi-task learning using task-specific features. *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Evgeniou, T., & Pontil, M. (2004). Regularized multi–task learning. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 109–117.

Johnson, V., Brun-Vezinet, F., Clotet, B., Günthrad, H., Kuritzkes, D., Pillay, D., Schapiro, J., Telenti, A., & Richman, D. (2007). Update of the drug resistance mutations in HIV-1: 2007. *Top HIV Med.*, *15*, 119–125.

Larder, B., Wang, D., Revell, A., Montaner, J., Harrigan, R., De Wolf, F., Lange, J., Wegner, S., Ruiz, L., Prez-Elas, M., Emery, S., Gatell, J., D'Arminio Monforte, A., Torti, C., Zazzi, M., & Lane, C. (2007). The development of artificial neural networks to predict virological response to combination HIV therapy. *Antiviral Therapy*, *12*, 15–24.

Lathrop, R., & Pazzani, M. (1999). Combinatorial optimization in rapidly mutating drug-resistant viruses. *Journal of Combinatorial Optimization*, *3*, 301–320.

Rosen-Zvi, M., Altmann, A., Prosperi, M., E., A., Neuvirth, H., Snnerborg, A., Schlter, E., Struck, D., Peres, Y., Incardona, F., Kaiser, R., Zazzi, M., & Lengauer, T. (2008). Selecting anti-HIV therapies based on a variety of genomic and clinical factors. *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, *90*, 227–244.

UNAIDS/WHO (2007). AIDS Epidemic Update.

Wu, P., & Dietterich, T. (2004). Improving SVM accuracy by training on auxiliary data sources. *Proceedings of the International Conference on Machine Learning*.

Xue, Y., Liao, X., Carin, L., & Krishnapuram, B. (2007). Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research*, *8*, 35–63.

Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning Gaussian processes from multiple tasks. *Proceedings of the International Conference on Machine Learning*.

Zhu, J., & Hastie, T. (2002). Kernel Logistic Regression and the Import Vector Machine. *Advances in Neural Information Processing Systems 14*. MIT Press.

Nonnegative Matrix Factorization via Rank-One Downdate

Michael Biggs MBIGGS@ALUMNI.UWATERLOO.CA
Ali Ghodsi AGHODSIB@UWATERLOO.CA
Stephen Vavasis VAVASIS@UWATERLOO.CA
University of Waterloo, Waterloo, ON N2L 3G1

Abstract

Nonnegative matrix factorization (NMF) was popularized as a tool for data mining by Lee and Seung in 1999. NMF attempts to approximate a matrix with nonnegative entries by a product of two low-rank matrices, also with nonnegative entries. We propose an algorithm called rank-one downdate (R1D) for computing an NMF that is partly motivated by the singular value decomposition. This algorithm computes the dominant singular values and vectors of adaptively determined submatrices of a matrix. On each iteration, R1D extracts a rank-one submatrix from the original matrix according to an objective function. We establish a theoretical result that maximizing this objective function corresponds to correctly classifying articles in a nearly separable corpus. We also provide computational experiments showing the success of this method in identifying features in realistic datasets. The method is also much faster than other NMF routines.

1. Nonnegative Matrix Factorization

Several problems in information retrieval can be posed as low-rank matrix approximation. The seminal paper by Deerwester et al. (1990) on latent semantic indexing (LSI) showed that approximating a term-document matrix describing a corpus of articles via the SVD led to powerful query and classification techniques. A drawback of LSI is that the low-rank factors in general will have both positive and negative entries, and there is no obvious statistical interpretation of the negative entries. This led Lee and Seung (1999) among others to propose *nonnegative matrix factorization* (NMF), that is, approximation of a matrix $A \in \mathbf{R}^{m\times n}$ as a product of two factors WH^T, where $W \in \mathbf{R}^{m\times k}$, $H \in \mathbf{R}^{n\times k}$, both have nonnegative entries, and $k \leq \min(m, n)$. Lee and Seung showed intriguing results with a corpus of images. In a related work, Hofmann (1999) showed the application of NMF to text retrieval. Nonnegative matrix factorization has its roots in work of Gregory and Pullman (1983), Paatero and Tapper (1994) and Cohen and Rothblum (1993).

Since the problem is NP-hard (Vavasis, 2007), it is not surprising that no algorithm is known to solve NMF to optimality. Heuristic algorithms proposed for NMF have generally been based on incrementally improving the objective $\|A - WH^T\|$ in some norm using local moves. A particularly sophisticated example of local search is due, e.g., to Kim and Park (2007). A drawback of local search is that it is sensitive to initialization and it is also sometimes difficult to establish convergence.

We propose an NMF method based on greedy rank-one downdating that we call R1D. R1D is partly motived by Jordan's algorithm for computing the SVD, which is described in Section 2. Unlike local search methods, greedy methods do not require an initial guess. In Section 3, we compare our algorithm to Jordan's SVD algorithm, which is the archetypal greedy downdating procedure. Previous work on greedy downdating algorithms for NMF is the subject of Section 4. In Section 5, we present the main theoretical result of this paper, which states that in a certain model of text due to Papadimitriou et al. (2000), optimizing our objective function means correctly identifying a topic in a text corpus; and Section 6 discusses the complexity of this problem. We then turn to computational experiments: in Section 7, we present results for R1D on image datasets, and in Section 8, we present results on text.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Algorithm and Objective Function

Rank-one downdate (R1D) is based on the simple observation that the leading singular vectors of a nonnegative matrix are nonnegative. This is a consequence of the Perron-Frobenius theorem (Golub & Van Loan, 1996). Based on this observation, it is trivial to compute a rank-one NMF. This idea can be extended to approximate a higher order NMF. Suppose we compute the rank-one NMF and then subtract it from the original matrix. The original matrix will not be nonnegative any more but all negative entries can be forced to be zero or positive and the procedure can be repeated.

An improvement on this idea takes only a submatrix of the original matrix and applies the Perron-Frobenius theorem. The point is that taking the whole matrix will in some sense average the features, whereas a submatrix can pick out particular features. A second reason to take a submatrix is that a correctly chosen submatrix may be very close to having a rank of one, so the step of forcing the residuals to be zero will not introduce significant inaccuracy (since they will already be close to zero).

The outer loop of the R1D algorithm may be described as follows.

Algorithm 1 R1D

input $A \in \mathbf{R}^{m\times n}$, $k > 0$
output $W \in \mathbf{R}^{m\times k}$, $H \in \mathbf{R}^{n\times k}$

1: **for** $\mu = 1$ **to** k **do**
2: $[M, N, \mathbf{u}, \mathbf{v}, \sigma] =$ `ApproxRankOneSubmatrix`(A)
3: $W(M, \mu) = \mathbf{u}(M)$
4: $H(N, \mu) = \sigma\mathbf{v}(N)$
5: $A(M, N) = 0$
6: **end for**

Here, M is a subset of $\{1, \ldots, m\}$, N is a subset of $\{1, \ldots, n\}$, $\mathbf{u} \in \mathbf{R}^m$, $\mathbf{v} \in \mathbf{R}^n$ and $\sigma \in \mathbf{R}$, and $\mathbf{u}, \mathbf{v}$ are both unit vectors. The function `ApproxRankOneSubmatrix` selects these five values so that the submatrix of A indexed by rows M and N is approximately rank one, and in particular, is approximately equal to $\mathbf{u}(M)\sigma\mathbf{v}^T(N)$. We follow Matlab subscripting conventions, so that $A(M, N)$ denotes this particular submatrix.

This outer loop for R1D may be called "greedy rank-one downdating" since it greedily tries to fill the columns of W and H from left to right by finding good rank-one submatrices of A and subtracting them from A. The classical greedy rank-one downdating algorithm is Jordan's algorithm for the SVD, described in Section 3. Related work on greedy rank-one downdating for NMF is the topic of Section 4.

The subroutine `ApproxRankOneSubmatrix`, presented later in this section, is a heuristic routine to maximize the following objective function:

$$f(M, N, \mathbf{u}, \sigma, \mathbf{v}) = \|A(M, N)\|_F^2 - \gamma\|A(M, N) - \mathbf{u}\sigma\mathbf{v}^T\|_F^2 \tag{1}$$

Here, γ is a penalty parameter. The Frobenius norm of an $m \times n$ matrix B, denoted $\|B\|_F$ is defined to be $\sqrt{B(1,1)^2 + B(1,2)^2 + \cdots + B(m,n)^2}$. The rationale for (1) is as follows: the first term in (1) expresses the objective that $A(M, N)$ should be large, while the second term penalizes departure of $A(M, N)$ from being a rank-one matrix.

Since the optimal $\mathbf{u}, \sigma, \mathbf{v}$ come from the SVD (once M, N are fixed), the above objective function can be rewritten just in terms of M and N as

$$\begin{aligned} f(M, N) &= \sum_{i=1}^{p} \sigma_i(A(M, N))^2 - \gamma \sum_{i=2}^{p} \sigma_i(A(M, N))^2 \\ &= \sigma_1(A(M, N))^2 \\ &\quad - (\gamma - 1) \cdot \left(\sigma_2(A(M, N))^2 \right. \\ &\quad \left. + \cdots + \sigma_p(A(M, N))^2\right), \end{aligned} \tag{2}$$

where $p = \min(|M|, |N|)$. The penalty parameter γ should be greater than 1 so that the presence of low-rank contributions is penalized rather than rewarded.

We conjecture that maximizing (1) is NP-hard (see Section 6), so we instead propose a heuristic routine for optimizing it. The procedure alternates improving M, N, $\mathbf{u}$, σ and $\mathbf{v}$ cyclically. First, observe that if M, N are already known, then the optimal choice of $\mathbf{u}$, σ, $\mathbf{v}$ can be found with the SVD. For fixed $(\mathbf{v}, N)$, the objective function (1) is separable by rows of the matrix. In particular, the contribution of row $i \in M$ is

$$\|A(i, N)\|^2 - \gamma\|A(i, N) - \beta_i\mathbf{v}^T\|^2,$$

where $\beta_i = u_i\sigma$. Note that β_i may be undefined if $i \notin M$. Nonetheless, given $\mathbf{v}$, the optimal β_i (i.e., the choice that minimizes $\|A(i, N) - u_i\mathbf{v}^T\|$) is easy to compute: it is $A(i, N)\mathbf{v}$, the solution to a simple least-squares minimization. Thus, we conclude that putting column i into index set M is favorable for the overall objective function provided that $f_i > 0$, where

$$f_i = \|A(i, N)\|^2 - \gamma\|A(i, N) - A(i, N)\mathbf{v}\mathbf{v}^T\|^2.$$

The formula for f_i can be simplified as follows:

$$\begin{aligned} f_i &= A(i, N)A(i, N)^T - \gamma(A(i, N) \\ &\quad - A(i, N)\mathbf{v}\mathbf{v}^T)(A(i, N) - A(i, N)\mathbf{v}\mathbf{v}^T)^T \\ &= -(\gamma - 1)A(i, N)A(i, N)^T + \gamma(A(i, N)\mathbf{v})^2. \end{aligned}$$

If we rescale by $\gamma - 1$ (which does not affect the acceptance criterion), and we define new penalty parameters $\bar{\gamma} := \gamma/(\gamma - 1)$, then we see that row i is accepted provided that

$$\bar{\gamma}(A(i,N)\mathbf{v})^2 - A(i,N)A(i,N)^T > 0.$$

A similar analysis applies to the columns, and leads to the conclusion that, given values for M and $\mathbf{u}$, column j should be accepted provided that

$$\bar{\gamma}(\mathbf{u}^T A(M,j))^2 - A(M,j)^T A(M,j) > 0.$$

The next issue is the choice of a starting guess for $M, N, \mathbf{u}, \sigma, \mathbf{v}$. The algorithm should be initialized with a starting guess that has a positive score, or else the rules for discarding rows and columns could conceivably discard all rows or columns. To put this more strongly, in order to improve the score of a converged solution, it seems sensible to select a starting guess with a high score. For this reason, R1D uses a single column of A as its starting guess, and in particular, the column of A with the greatest norm. (A single row may also be chosen.) It then chooses $\mathbf{u}$ to be the normalization of this column. This column is exactly rank one, so for the correct values of σ and $\mathbf{v}$ the first penalty term of (1) is zero. We have derived the following algorithm for the subroutine `ApproxRankOneSubmatrix` occuring in statement ⟨2⟩ in R1D.

Algorithm 2 ApproxRankOneSubmatrix

input $A \in \mathbf{R}^{m\times n}$, **parameter** $\bar{\gamma} > 1$
output $M \subset \{1,\ldots,m\}$, $N \subset \{1,\ldots,n\}$,
$\mathbf{u} \in \mathbf{R}^m$, $\mathbf{v} \in \mathbf{R}^n$, $\sigma \in \mathbf{R}$

1: Select $j_0 \in \{1,\ldots,n\}$ to maximize $\|A(:,j_0)\|$
2: $M = \{1,\ldots,m\}$
3: $N = \{j_0\}$
4: $\sigma = \|A(:,j_0)\|$
5: $\mathbf{u} = A(:,j_0)/\sigma$
6: **repeat**
7: Let $\bar{\mathbf{v}} = A(M,:)^T\mathbf{u}(M)$
8: $N = \{j : \bar{\gamma}\bar{v}(j)^2 - \|A(M,j)\|^2 > 0\}$
9: $\mathbf{v}(N) = \bar{\mathbf{v}}(N)/\|\bar{\mathbf{v}}(N)\|$
10: Let $\bar{\mathbf{u}} = A(:,N)\mathbf{v}(N)$
11: $M = \{i : \bar{\gamma}\bar{u}(i)^2 - \|A(i,N)\|^2 > 0\}$
12: $\sigma = \|\mathbf{u}(M)\|$
13: $\mathbf{u}(M) = \bar{\mathbf{u}}(M)/\sigma$
14: **until** stagnation in $M, N, \mathbf{u}, \sigma, \mathbf{v}$

The 'Repeat' loop is guaranteed to make progress because each iteration increases the value of the objective function. On the other hand, there does not seem to be any easy way to derive a useful prior upper bound on its number of iterations. In practice, it proceeds quite quickly, usually converging in 10–15 iterations. But to guarantee fast termination, monotonicity can be forced on M and N by requiring M to shrink and N to grow. In other words, statement ⟨8⟩ can be replaced by

$$N = N \cup \{j : \bar{\gamma}\bar{v}(j)^2 - \|A(M,j)\|^2 > 0\},$$

and statement ⟨11⟩ by

$$M = M - \{i : \bar{\gamma}\bar{u}(i)^2 - \|A(i,N)\|^2 \le 0\}.$$

Our experiments indicate that this change does not have a major impact on the performance of R1D.

Another possible modification to the algorithm is as follows: we modify the objective function by adding a second penalty term $-\rho|M|\cdot|N|$ to (1) where $\rho > 0$ is a parameter. The purpose of this term is to penalize very low-norm rows or columns from being inserted into $A(M,N)$ since they are probably noisy. For data with larger norm, the first term of (1) should dominate this penalty. Notice that this penalty term is also separable so it is easy to implement: the formula in ⟨8⟩ is changed to $\bar{\gamma}\bar{v}(j)^2 - \|A(M,j)\|^2 - \bar{\rho}|M| > 0$ while the formula in ⟨11⟩ becomes $\bar{\gamma}\bar{u}(i)^2 - \|A(i,N)\|^2 - \bar{\rho}|N| > 0$, where $\bar{\rho} = \rho/(\gamma - 1)$. A good value for $\bar{\rho}$ is to set it so that in the initial starting point, the third penalty term is a small fraction (say $\bar{\eta} = 1/20$) of the other terms. This leads to the following definition for ρ:

$$\rho = \bar{\eta}(\bar{\gamma} - 1)\sigma^2/m,$$

which may be computed immediately after ⟨4⟩.

Greedy rank-one downdating appears to be much faster than other NMF algorithms. Generating each column of W and H requires approximately 20 matrix-vector multiplications; these multiplications are always at least as sparse as the original data. There is no iterative improvement phase. It can also be much faster than the SVD, especially for sparse data.

3. Relationship to the SVD

The classical rank-one greedy downdating algorithm is Jordan's algorithm for computing the singular value decomposition (SVD) (Stewart, 1993). Recall that the SVD takes as input an $m \times n$ matrix A and returns three factors U, Σ, V such that $U \in \mathbf{R}^{m\times k}$ and U has orthonormal columns (i.e., $U^TU = I$), $\Sigma \in \mathbf{R}^{k\times k}$ and is diagonal with nonnegative diagonal entries, and $V \in \mathbf{R}^{n\times k}$ also with orthonormal columns, such that $U\Sigma V^T$ is the optimal rank-k approximation to A in either the 2-norm or Frobenius norm. (Recall that the 2-norm of an $m \times n$ matrix B, denoted $\|B\|_2$, is

Algorithm 3 JordanSVD

input $A \in \mathbf{R}^{m \times n}$ and $k \leq \min(m, n)$
output U, Σ, V as above.

for $\mu = 1$ **to** k **do**
  Select a random nonzero $\bar{\mathbf{u}} \in \mathbf{R}^m$
  $\sigma = \|\bar{\mathbf{u}}\|$
  $\mathbf{u} = \bar{\mathbf{u}}/\sigma$
  repeat {power method}
    $\bar{\mathbf{v}} = A^T\mathbf{u}$
    $\mathbf{v} = \bar{\mathbf{v}}/\|\bar{\mathbf{v}}\|$
    $\bar{\mathbf{u}} = A\mathbf{v}$
    $\sigma = \|\bar{\mathbf{u}}\|$
    $\mathbf{u} = \bar{\mathbf{u}}/\sigma$
  until stagnation in $\mathbf{u}, \sigma, \mathbf{v}$
  $A = A - \mathbf{u}\sigma\mathbf{v}^T$
  $U(:, \mu) = \mathbf{u}$
  $V(:, \mu) = \mathbf{v}$
  $\Sigma(\mu, \mu) = \sigma$
end for

defined to be $\sqrt{\lambda_{\max}(B^T B)}$, where $\lambda_{\max}$ denotes the maximum eigenvalue.)

Thus, we see that R1D is quite similar to the SVD. The principal difference is that R1D tries to find a submatrix indexed by $M \times N$ at the same time that it tries to identify the optimal $\mathbf{u}$ and $\mathbf{v}$. Hence, the formulas for $\mathbf{u}$ and $\mathbf{v}$ occurring in ⟨9⟩ and ⟨13⟩ of subroutine `ApproxRankOneSubmatrix`, which were presented earlier as solutions to a least-squares problem, may also be regarded as steps in a power method. In particular, this means that if M and N are fixed, then the inner repeat loop of this subroutine will indeed converge to the dominant singular triple of $A(M, N)$.

As mentioned earlier, a shortcoming of the SVD is that its factors contain both positive and negative numbers. It has another subtler shortcoming when used for clustering which is as follows: because the SVD always operates on the entire matrix, it can return a singular vector that averages the results from two nearly disjoint topics in a corpus (see Biggs et al. (2008) for an example). R1D avoids this pitfall by seeking a submatrix that is approximately rank-one as it applies the power method.

4. Related Work

As mentioned in the introduction, most algorithms proposed in the literature are based on forming an initial W and H and then improving them by local search on an objective function. The objective function usually includes a term of the form $\|A - WH^T\|$ in some norm, and may include other terms.

A few previous works follow an approach similar to ours, namely, greedy subtraction of rank-one matrices. This includes the work of Bergmann et al. (2003), who identify the rank-one matrix to subtract as the fixed point of an iterative process. Asgarian and Greiner (2006) find the dominant singular pair and then truncate it. Gillis (2006) finds a rank-one understimator and subtracts that. Boutsidis and Gallopoulos (2007) consider the use of a greedy algorithm for initializing other algorithm and make the following interesting observation: The nonnegative part of a rank-one matrix has rank at most 2.

The main innovation herein is the idea that the search for the rank-one submatrix should itself be an optimization subproblem. This observation allows us to compare one candidate submatrix to another. (Gillis also phrases his subproblem as optimization, although his optimization problem does not explicitly seek submatrices like ours.) A second innovation is our analysis in Section 5 showing that if the subproblem were solved optimally, then R1D would be able to accurately find the topics in the model of ϵ-separable corpora (Papadimitriou et al., 2000).

5. Behavior of this objective function on a nearly separable corpus

In this section, we establish the main theoretical result of the paper, namely, that the objective function given by (1) is able to correctly identify a topic in a nearly separable corpus. We define our *text model* as follows. There is a universe of *terms* numbered $1, \ldots, m$. There is also a set of *topics* numbered $1, \ldots, t$. Topic k, for $k = 1, \ldots, t$, is a probability distribution over the terms. Let $P(i, k)$ denote the probability of term i occurring in topic k. Thus, P is a singly stochastic matrix, i.e., it has nonnegative entries with column sums exactly 1. We assume also that there is a probability distribution over topics; say the probability of topic k is τ_k, for $k = 1, \ldots, t$. The text model is thus specified by P and $\tau_1, \ldots, \tau_t$. We use the Zipf distribution as the model of document length. In particular, there is a number L such that all documents have length less than L, and the probability that a document of length l occurs is

$$\frac{1/l}{1 + 1/2 + \cdots + 1/(L-1)}.$$

We have checked that the Zipf model is a good fit for several common datasets.

A *document* is generated from this text model as follows. First, topic k is chosen at random according to the probability distribution $\{\tau_1, \ldots, \tau_t\}$. Then, a

length l is chosen at random from $\{1, \ldots, L-1\}$ according to the Zipf distribution. Finally, the document itself is chosen at random by selecting l terms independently according to the probability distribution $P(:, k)$. A *corpus* is a set of n documents chosen independently using this text model. Its *term-document matrix* is the $m \times n$ matrix A such that $A(i, j)$ is the frequency of term i in document j.

We further assume that the text model is ϵ-*separable*, meaning that each topic k is associated with a set of terms $T_k \subset \{1, \ldots, m\}$, that $T_1, \ldots, T_t$ are mutually disjoint, and that $P(i, k) \leq \epsilon$ for $i \notin T_k$, i.e., the probability that a document on topic k will use a term outside of T_k is small. Let $P_{\min} = \min\{P(i, k) : i \in T_k, k = 1, \ldots, t\}$. Without loss of generality, $P_{\min} > 0$ since any row $i \in T_k$ such that $P(i, k) = 0$ may be removed from T_k without affecting the validity of the model. Parameter ϵ must satisfy an inequality mentioned below. This corpus model is quite similar to that of Papadimitriou et al. (2000). One difference is in the the document length model. Our model also relaxes several assumptions of Papadimitriou et al.

Our main theorem is that the objective function given by (1) correctly finds documents associated with a particular topic in a corpus.

Theorem 1. *Let $(P, (\tau_1, \ldots, \tau_t))$ specify a text model, and let $\alpha > 0$ be chosen arbitrarily. Assume $\epsilon > 0$ is chosen smaller than a function $\epsilon(P_{\min}, m, t, \alpha)$ (see Biggs et al. (2008) for this function). Suppose that the text-model is ϵ-separable with respect to $T_1, \ldots, T_t$, the subsets of terms defining the topics. Let A be the term-document matrix of a corpus of n documents drawn from this model when the document-length parameter is L.*

Choose $\gamma = 4$ in (1). Then with probability tending to 1 as $n \to \infty$ and $L \to \infty$, the optimizing pair (M, N) of (1) satisfies the following. Let $D_1, \ldots, D_t$ be the partitioning of the columns of A according to topics. There exists a topic $k \in \{1, \ldots, t\}$ such that $A(M, N)$ and $A(T_k, D_k)$ are nearly coincident in the following sense.

$$\sum_{(i,j) \in (M \times N) \triangle (T_k \times D_k)} A(i, j)^2 \leq \alpha \sum_{(i,j) \in M \times N} A(i, j)^2.$$

Here, $X \triangle Y$ denotes the set-theoretic symmetric difference $(X - Y) \cup (Y - X)$. The proof of this theorem is lengthy and appears in Biggs et al. (2008). It relies on Chernoff-Hoeffding estimates and perturbation results for singular vectors such as Theorem 8.6.5 of Golub and Van Loan (1996).

6. On the complexity of maximizing $f(M, N)$

In this section, we observe that the problem of globally maximizing (2) is NP-hard at least in the case that γ is treated as an input parameter. This observation explains why R1D settles for a heuristic maximization of (2) rather than exact maximization. First, observe that the maximum biclique (MBC) problem is NP-hard as proved by Peeters (2003). We show that the MBC problem can be transformed to an instance of (2).

Let us recall the definition of the MBC problem. The input is a bipartite graph G. The problem is to find an (m, n)-complete bipartite subgraph K (sometimes called a *biclique*) of G such that mn is maximized, i.e., the number of edges of K is maximized.

Suppose we are given G, an instance of the maximum biclique problem. Let A be the left-right adjacency matrix of G, that is, if $G = (U, V, E)$ where $U \cup V$ is the bipartition of the node set, then A has $|U|$ rows and $|V|$ columns, and $A(i, j) = 1$ if $(i, j) \in E$ for $i \in U$ and $j \in V$, else $A(i, j) = 0$.

Consider maximizing (2) for this choice of A. We require the following preliminary lemmas whose proofs are omitted.

Lemma 2. *Let A be a matrix that has either of the following as a submatrix:*

$$U_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ or } U_2 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}. \tag{3}$$

Then $\sigma_2(A) > 0.618$.

This lemma leads to the following lemma.

Lemma 3. *Suppose all entries of $A \in \mathbf{R}^{m \times n}$ are either 0 or 1, and suppose and at least one entry is 1. Suppose M, N are the optimal solution for maximizing $f(M, N)$ given by (2). Suppose also that the parameter γ is chosen to be $2.7mn + 1$ or larger. Then the optimal choice of M, N must yield a matrix $A(M, N)$ of all 1's, possibly augmented with some rows or columns that are entirely zeros.*

Now consider the main claim, namely, that optimize (M, N) of the objective function for this A corresponds to the max biclique. If $A(M, N)$ includes a row or column entirely of zeros, then this row or column may be dropped without affecting the value of the objective function (2). Hence it follows from the lemma that without loss of generality that the optimizer (M, N) of (2) indexes a matrix of all 1's. In that case, $\sigma_1(A(M, N)) = \sqrt{|M| \cdot |N|}$ while $\sigma_2(A(M, N)) =$

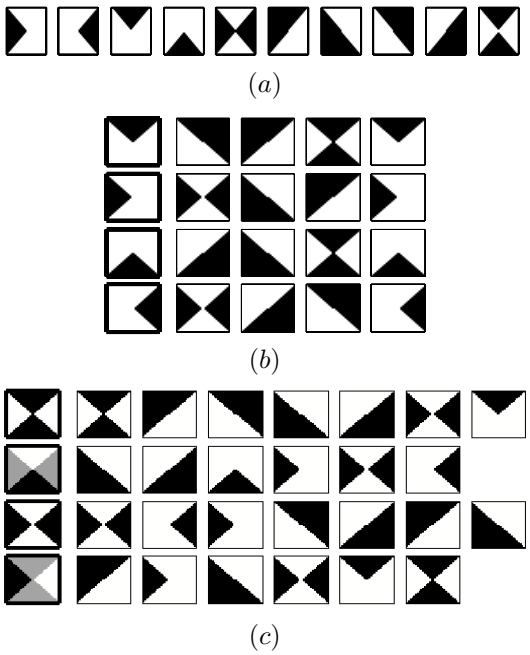


Figure 1. A binary image dataset is depicted in (a); white indicates zeros. The result of R1D on this dataset is shown in (b), and LSI in (c).

$\cdots = \sigma_p(A(M,N)) = 0$ (where $p = \min(|M|,|N|)$), and hence $f(M,N) = |M| \cdot |N|$. Thus, the value of the objective function corresponds exactly to the number of edges in the biclique. This completes the proof that biclique is reducible in polynomial time to maximizing (2).

We note that Gillis (2006) also uses the result of Peeters for a similar purpose, namely, to show that the subproblem arising in his NMF algorithm is also NP-hard.

The NP-hardness result in this section requires that γ be an input parameter. We conjecture that (2) is NP-hard even when γ is fixed (say $\gamma = 4$ as used herein).

7. Image dataset test cases

We first demonstrate the performance of R1D on a simple binary image dataset, depicted in Figure 1 (a). Each of the ten dataset images is composed of one or two "basis" triangles. The results of R1D (with parameter $\bar{\gamma} = 4$) and LSI on this dataset are shown in Figure 1 (b) and (c), respectively, and the interpretation is as follows. The leftmost column illustrates the four leading columns of W, which are the learned features. For each of these, the images on the right are the dataset images with the largest entries in the corresponding column of H; they should be closely associated with the feature on the left.

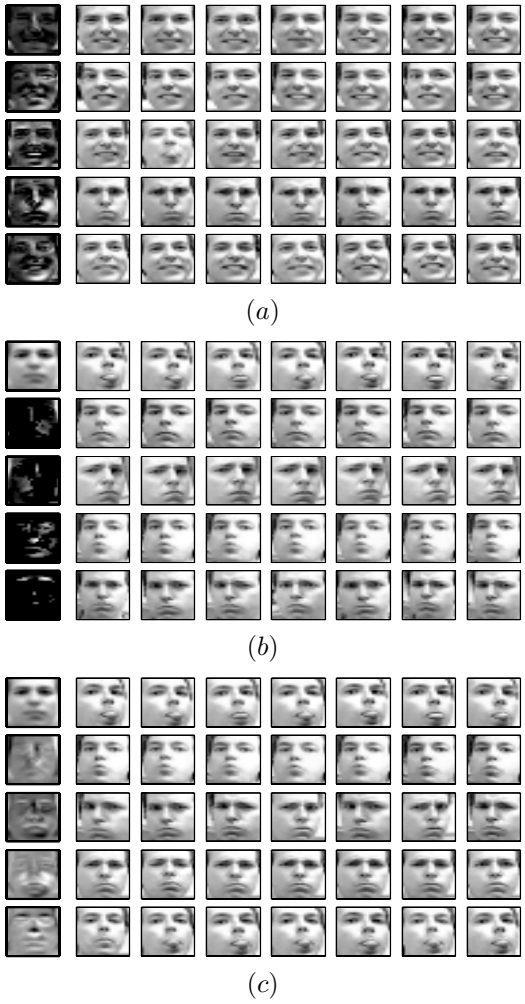


Figure 2. Three algorithms applied to the Frey face dataset (black indicates zeros): (a) NMF with divergence criterion, (b) our R1D algorithm for NMF, and (c) LSI

R1D discovered the four triangles as a basis, and to each it associated exactly the dataset images which contain the appropriate triangle. Alternatively, the LSI factorization is not as interpretable.

We have also compared results against NMFDIV from nmfpack (Hoyer, 2000; Hoyer, 2004). NMFDIV requires k, the number of basis vectors to compute, as an input parameter which globally affects the factors W and H. If k is correctly set to 4, NMFDIV is able to compute the same correct result as R1D. Otherwise, some or all of the basis vectors will appear incorrect, including the first ones. R1D and LSI will each compute the same leading columns regardless of k, and on this dataset they will not compute more than 4 columns; all subsequent columns of W and H will be

Table 1. The amount of sparsity in the NMF computed by R1D ($\bar{\gamma} = 2$) on the Frey face dataset. It is presented as the percentage of zero values in the first few columns of W and H.

Column	% zeros in W	% zeros in H
1	0.00	0.00
2	0.82	0.69
3	0.69	0.68
4	0.82	0.88
5	0.94	0.73

zero.

Figure 2 conducts a similar experiment on the Frey face dataset, which consists of 1965 registered face images of size 28×20. Again, the leading columns of W present the "eigenfaces" or "features" discovered in the dataset, and the corresponding column of H selects dataset images that are classified as carrying the feature most prominently. R1D seems to be the most successful at finding features and classifying images; in each case, the column of W shows a particular highlight that distinguishes some images in the dataset from others. NMFDIV appears to be slightly inferior to R1D, while LSI is noticeably worse.

In this experiment, the algorithms computed 30 basis vectors of the NMF. NMFDIV was allowed 500 iterations which took 727 seconds; in contrast, LSI required 20 seconds and R1D took 47 seconds.

Additionally, R1D is effective at finding a sparse factorization. Table 1 demonstrates the sparsity in the first few columns of W and H. The first column of W and H is fully dense, because the data matrix appears to be approximately rank-one; its first singular value is dominant. Apart from this, the other columns of the NMF are sparse, and the sparsity can be controlled by the $\bar{\gamma}$ parameter (here we have used $\bar{\gamma} = 2$). Alternatively, both NMFDIV and LSI perform a dense factorization with very few values near zero in any column.

8. Text dataset test cases

In Tables 2 and 3 we illustrate LSI versus R1D (with parameter $\bar{\gamma} = 4$) on the TDT Pilot Study (TDT Study, 1997). The columns of each table are the leading columns of W, with the leading terms per column displayed. The LSI results show that the topics are not properly separated and terms from different topics recur or are mixed. The columns in the R1D table are clearly identifiable topics, and the terms in each columns are all correctly associated with the given topics.

Table 2. Topics found by LSI on the TDT Pilot Study corpus (tf-idf normalization).

Topic 1	Topic 2	Topic 3	Topic 4
SIMPSON	ISRAEL	ISRAEL	BOSNIAN
PRESIDENT	ISRAELI	ISRAELI	SERBS
CLINTON	BOSNIAN	PALESTINIAN	SERB
POLICE	PEACE	GAZA	SARAJEVO
HOUSE	SERBS	ARAFAT	BOSNIA
ISRAEL	BOSNIA	PLO	NATO
BOSNIAN	SERB	JERUSALEM	SIMPSON
HAITI	SARAJEVO	PEACE	BIHAC
UNITED	PALESTINIAN	PALESTINIANS	AIR
GOVERNMENT	NATO	SIMPSON	TROOPS

Table 3. Topics found by R1D on the TDT Pilot Study corpus (tf-idf normalization). Note that all words in a column do in fact refer to the same news event.

Topic 1	Topic 2	Topic 3	Topic 4
SIMPSON	MASTERS	KOREA	DENG
JUDGE	PAIRINGS	KOREAN	XIAOPING
ITO	AUGUSTA	NORTH	RONG
JURY	AMATEUR	KIM	PARAMOUNT
DEFENSE	TOURNAMENT	PYONGYANG	CHINA
TRIAL	ROUND	SEOUL	HEALTH
ANGELES	GOLF	SUNG	CHINESE
LOS	NOTED	NUCLEAR	KONG
PROSECUTION	PLAYERS	SOUTH	HONG
CASE	GEORGIA	COMMUNIST	DAUGHTERS

NMFDIV (and the other implementations of NMF in nmfpack) were not run on this dataset because they would exhaust all of the computer's memory. As noted earlier, R1D on text datasets is able to efficiently work with sparse matrices throughout its operation. R1D was able to compute 80 basis vectors of the TDT corpus in 171 seconds, whereas LSI required 269 seconds.

9. Conclusions

We have proposed an algorithm called R1D for nonnegative matrix factorization. It is based on greedy rank-one downdating according to an objective function, which is heuristically maximized. We have shown that the objective function is well suited for identifying topics in the ϵ-separable text model. Finally, we have shown that the algorithm performs well in practice.

This work raises several interesting open questions. First, the ϵ-separable text model seems rather too simple to describe real text, so it would be interesting to see if the results generalize to more realistic models.

A second arising question asks whether a result like Theorem 1 will hold for the R1D algorithm. In other words, if the heuristic subroutine `ApproxRankOneSubmatrix` is applied to an ϵ-separable corpus, does it successfully identify a topic? Here is an example of a difficulty. Suppose $n \to \infty$ much faster than L. In this case, the document j with the highest norm will be the one in which l_j is very close to L and in which one entry $A(i,j)$ is very close to L while the rest are mostly zeros. This is because the maximizer of $\|\mathbf{x}\|_2$ subject to the constraint that $\|\mathbf{x}\|_1 = C$ occurs when one entry of $\mathbf{x}$ is equal to C and the rest are zero. It is likely that at least one instance of such a document will occur regardless of the matrix $P(\cdot,\cdot)$ if n is sufficiently large. This document will then act as the seed for expanding M and N, but it may not be similar to any topic. This scenario can perhaps be prevented by a more intelligent selection of a starting vector for `ApproxRankOneSubmatrix`.

References

Asgarian, N., & Greiner, R. (2006). Using rank-1 biclusters to classify microarray data. Department of Computing Science, University of Alberta, Edmonton, AB, Canada.

Bergmann, S., Ihmels, J., & Barkai, N. (2003). Iterative signature algorithm for the analysis of large-scale gene expression data. *Physical Review E*, *67*, 031902.

Biggs, M., Ghodsi, A., & Vavasis, S. (2008). Nonnegative matrix factorization via rank-one downdate. Available online at http://www.arxiv.org/abs/0805.0120.

Boutsidis, C., & Gallopoulos, E. (2007). SVD based initialization: A head start for nonnegative matrix factorization. In press.

Cohen, J., & Rothblum, U. (1993). Nonnegative ranks, decompositions and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, *190*, 149–168.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*, 391–407.

Gillis, N. (2006). Approximation et sous-approximation de matrices par factorisation positive: algorithmes, complexité et applications. Master's thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium. In French.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix computations, 3rd edition*. Baltimore: Johns Hopkins University Press.

Gregory, D. A., & Pullman, N. J. (1983). Semiring rank: Boolean rank and nonnegative matrix rank. *J. Combin. Inform. System Sci*, *3*, 223–233.

Hofmann, T. (1999). Probabilistic latent semantic analysis. *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30-August 1, 1999* (pp. 289–296). Morgan Kaufmann.

Hoyer, P. (2000). nmfpack - matlab code for nmf. http://http://www.hiit.fi/node/70.

Hoyer, P. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, *5*, 1457–1469.

Kim, H., & Park, H. (2007). Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. Bioinformatics (to appear).

Lee, D., & Seung, H. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*, 788–791.

Paatero, P., & Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, *5*, 111–126.

Papadimitriou, C., Raghavan, P., Tamaki, H., & Vempala, S. (2000). Latent semantic indexing: A probabilistic analysis. *J. Comput. Syst. Sci.*, *61*, 217–235.

Peeters, R. (2003). The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, *131*, 651–654.

Stewart, G. W. (1993). On the early history of the singular value decomposition. *SIAM Review*, *35*, 551–566.

TDT Study (1997). Topic detection and tracking pilot study. http://projects.ldc.upenn.edu/TDT/.

Vavasis, S. (2007). On the complexity of nonnegative matrix factorization. arxiv.org, 0708.4149.

Strategy Evaluation in Extensive Games with Importance Sampling

Michael Bowling BOWLING@CS.UALBERTA.CA
Michael Johanson JOHANSON@CS.UALBERTA.CA
Neil Burch BURCH@CS.UALBERTA.CA
Duane Szafron DUANE@CS.UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2E8 Canada

Abstract

Typically agent evaluation is done through Monte Carlo estimation. However, stochastic agent decisions and stochastic outcomes can make this approach inefficient, requiring many samples for an accurate estimate. We present a new technique that can be used to simultaneously evaluate many strategies while playing a single strategy in the context of an extensive game. This technique is based on importance sampling, but utilizes two new mechanisms for significantly reducing variance in the estimates. We demonstrate its effectiveness in the domain of poker, where stochasticity makes traditional evaluation problematic.

1. Introduction

Evaluating an agent's performance is a component of nearly all research on sequential decision making. Typically, the agent's expected payoff is estimated through Monte Carlo samples of the (often stochastic) agent acting in an (often stochastic) environment. The degree of stochasticity in the environment or agent behavior determines how many samples are needed for an accurate estimate of performance. For results in synthetic domains with artificial agents, one can simply continue drawing samples until the estimate is accurate enough. For non-synthetic environments, domains that involve human participants, or when evaluation is part of an on-line algorithm, accurate estimates with a small number of samples are critical. This paper describes a new technique for tackling this problem in the context of extensive games.

An extensive game is a formal model of a sequential interaction between multiple, independent agents with imperfect information. It is a powerful yet compact framework for describing many strategic interactions between decision-makers, artificial and human[1]. Poker, for example, is a domain modeled very naturally as an extensive game. It involves independent and self-interested agents making sequential decisions based on both public and private information in a stochastic environment. Poker also demonstrates the challenge of evaluating agent performance. In one typical variant of poker, approximately 30,000 hands (or samples of playing the game) are sometimes needed to distinguish between professional and amateur levels of play. Matches between computer and human opponents typically involve far fewer hands, yet still need to draw similar statistical conclusions.

In this work, we present a new technique for deriving low variance estimators of agent performance in extensive games. We employ importance sampling while exploiting the fact that the strategy of the agent being evaluated is typically known. However, we reduce the variance that importance sampling normally incurs by selectively adding synthetic data that is derived from but consistent with the sample data. As a result we derive low-variance unbiased estimators for agent performance given samples of the outcome of the game. We further show that we can efficiently evaluate one strategy while only observing samples from another. Finally, we examine the important case where we only get partial information of the game outcome (e.g., if a player folds in poker, their private cards are not revealed during the match and so the sequence of game states is not fully known). All of our estimators are then evaluated empirically in the domain of poker in both full and partial information scenarios.

This paper is organized as follows. In Section 2 we introduce the extensive game model, formalize our problem, and describe previous work on variance reduction in agent evaluation. In Section 3 we present a general procedure for deriving unbiased estimators and give four examples of

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] In this work we use the words "agent", "player", and "decision-maker" interchangeably and, unless explicitly stated, aren't concerned if they are humans or computers.

these estimators. We then briefly introduce the domain of poker in Section 4 and describe how these estimators can be applied to this domain. In Section 5 we show empirical results of our approach in poker. Finally, we conclude in Section 6 with some directions for future work.

2. Background

We begin by describing extensive games and then we formalize the agent evaluation problem.

2.1. Extensive Games

Definition 1 *(Osborne & Rubenstein, 1994, p. 200) a finite extensive game with imperfect information has the following components:*

- *A finite set N of **players**.*
- *A finite set H of sequences, the possible **histories** of actions, such that the empty sequence is in H and every prefix of a sequence in H is also in H. $Z \subseteq H$ are the **terminal histories** (those which are not a prefix of any other sequences). $A(h) = \{a : (h, a) \in H\}$ are the actions available after a non-terminal history $h \in H$,*
- *A **player function** P that assigns to each non-terminal history (each member of $H \backslash Z$) a member of $N \cup \{c\}$, where c represents chance. $P(h)$ is the player who takes an action after the history h. If $P(h) = c$, then chance determines the action taken after history h.*
- *A function f_c that associates with every history h for which $P(h) = c$ a probability measure $f_c(\cdot|h)$ on $A(h)$ ($f_c(a|h)$ is the probability that a occurs given h), where each such probability measure is independent of every other such measure.*
- *For each player $i \in N$ a partition $\mathbf{I}_i$ of $\{h \in H : P(h) = i\}$ with the property that $A(h) = A(h')$ whenever h and h' are in the same member of the partition. $\mathbf{I}_i$ is the **information partition** of player i; a set $I_i \in \mathbf{I}_i$ is an **information set** of player i.*
- *For each player $i \in N$ a utility function u_i from the terminal states Z to the reals $\mathbf{R}$. If $N = \{1, 2\}$ and $u_1 = -u_2$, it is a **zero-sum extensive game**.*

A **strategy of player i** σ_i in an extensive game is a function that assigns a distribution over $A(I_i)$ to each $I_i \in \mathbf{I}_i$. A **strategy profile** σ consists of a strategy for each player, $\sigma_1, \sigma_2, \ldots$, with σ_{-i} referring to all the strategies in σ except σ_i.

Let $\pi^\sigma(h)$ be the probability of history h occurring if players choose actions according to σ. We can decompose $\pi^\sigma = \Pi_{i \in N \cup \{c\}} \pi_i^\sigma(h)$ into each player's contribution to this probability. Hence, $\pi_i^\sigma(h)$ is the probability that if player i plays according to σ then for all histories h' that are a proper prefix of h with $P(h') = i$, player i takes the subsequent action in h. Let $\pi_{-i}^\sigma(h)$ be the product of all players' contribution (including chance) except player i. The overall value to player i of a strategy profile is then the expected payoff of the resulting terminal node, i.e., $u_i(\sigma) = \sum_{z \in Z} u_i(z)\pi^\sigma(z)$. For $Y \subseteq Z$, a subset of possible terminal histories, define $\pi^\sigma(Y) = \sum_{z \in Y} \pi^\sigma(z)$, to be the probability of reaching any outcome in the set Y given σ, with $\pi_i^\sigma(Y)$ and $\pi_{-i}^\sigma(Y)$ defined similarly.

2.2. The Problem

Given some function on terminal histories $V : Z \to \Re$ we want to estimate $E_{z|\sigma}[V(z)]$. In most cases V is simply u_i, and the goal is to evaluate a particular player's expected payoff. We explore three different settings for this problem. In all three settings, we assume that σ_i (our player's strategy) is known, while $\sigma_{j \neq i}$ (the other players' strategies) are not known.

- *On-policy full-information*. In the simplest case, we get samples $z_{1...t} \in Z$ from the distribution π^σ.
- *Off-policy full-information*. In this case, we get samples $z_{1...t} \in Z$ from the distribution $\pi^{\hat{\sigma}}$ where $\hat{\sigma}$ differs from σ only in player i's strategy: $\pi_{-i}^\sigma = \pi_{-i}^{\hat{\sigma}}$. In this case we want to evaluate one strategy for player i from samples of playing a different one.
- *Off-policy partial-information*. In the hardest case, we don't get full samples of outcomes z_t, but rather just player i's view of the outcomes. For example, in poker, if a player folds, their cards are not revealed to the other players and so certain chance actions are not known. Formally, in this case we get samples of $K(z_t) \in \mathbf{K}$, where K is a many-to-one mapping and z_t comes from the distribution $\pi^{\hat{\sigma}}$ as above. K intuitively must satisfy the following conditions: for $z, z' \in Z$, if $K(z) = K(z')$ then,
  - $V(z) = V(z')$, and
  - $\forall \sigma \quad \pi_i^\sigma(z) = \pi_i^\sigma(z')$.

2.3. Monte Carlo Estimation

The typical approach to estimating $E_{z|\sigma}[V(z)]$ is through simple Monte Carlo estimation. Given independent samples $z_1, \ldots, z_t$ from the distribution π^σ, simply estimate the expectation as the sample mean of outcome values.

$$\frac{1}{t} \sum_{i=1}^{t} V(z_i) \tag{1}$$

As the estimator has zero bias, the mean squared error of the estimator is determined by its variance. If the variance of $V(z)$ given σ is large, the error in the estimate can be large and many samples are needed for accurate estimation.

Recently, we proposed a new technique for agent evaluation in extensive games (Zinkevich et al., 2006). We showed that value functions over non-terminal histories could be used to derive alternative unbiased estimators. If the chosen value function was close to the true expected value given the partial history and players' strategies, then the estimator would result in a reduction in variance. The approach essentially derives a real-valued function $\tilde{V}(z)$ that is used in place of V in the Monte Carlo estimator from Equation 1. The expectation of $\tilde{V}(z)$ matches the expectation of $V(z)$ for any choice of σ, and so the result is an unbiased estimator, but potentially with lower variance and thus lower mean-squared error. The specific application of this approach to poker, using an expert-defined value function, was named the DIVAT estimator and was shown to result in a dramatic reduction in variance. A simpler choice of value function, the expected value assuming the betting is "bet-call" for all remaining betting rounds, can even make a notable reduction. We refer to this conceptually and computationally simpler estimator as (Bet-Call) BC-DIVAT.

Both traditional Monte Carlo estimation and DIVAT are focused on the *on-policy* case, requiring outcomes sampled from the joint strategy that is being evaluated. Furthermore, DIVAT is restricted to *full-information*, where the exact outcome is known. Although limited in these regards, they also don't require any knowledge about any of the players' strategies.

3. General Approach

We now describe our new approach for deriving low-variance, unbiased estimators for agent evaluation. In this section we almost exclusively focus on the *off-policy full-information* case. Within this setting we observe a sampled outcome z from the distribution $\pi^{\hat{\sigma}}$, and the goal is to estimate $E_{z|\sigma}\left[V(z)\right]$. The outcomes are observed based on the strategy $\hat{\sigma}$ while we want to evaluate the expectation over σ, where they differ only in player i's strategy. This case subsumes the on-policy case, and we touch on the more difficult partial-information case at the end of this section. In order to handle this more challenging case, we require full knowledge of player i's strategies, both the strategy being observed $\hat{\sigma}_i$ and the one being evaluated σ_i.

At the core of our technique is the idea that synthetic histories derived from the sampled history can also be used in the estimator. For example, consider the unlikely case when σ is known entirely. Given an observed outcome $z \in Z$ (or even without an observed outcome) we can exactly compute the desired expectation by examining every outcome.

$$V_Z(z) \equiv \sum_{z' \in Z} V(z')\pi^{\sigma}(z') = E_{z|\sigma}\left[V(z)\right] \qquad (2)$$

Although impractical since we don't know σ, $V_Z(z)$ is an unbiased and zero variance estimator.

Instead of using every terminal history, we could restrict ourselves to a smaller set of terminal histories. Let $U(z' \in Z) \subseteq Z$ be a mapping of terminal histories to a set of terminal histories, where at least $z' \in U(z')$. We can construct an unbiased estimator that considers the history z' in the estimation whenever we observe a history from the set $U(z')$. Another way to consider things is to say that $U^{-1}(z)$ is the set of synthetic histories considered when we observe z. Specifically, we define the estimator $V_U(z)$ for the observed outcome z as,

$$V_U(z) \equiv \sum_{z' \in U^{-1}(z)} V(z')\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))} \qquad (3)$$

The estimator considers the value of every outcome z' where the observed history z is in the set $U(z')$. Each outcome though is weighted in a fashion akin to importance sampling. The weight term for z' is proportional to the probability of that history given σ, and inversely proportional to the probability that z' is one of the considered synthetic histories when observing sampled outcomes from $\hat{\sigma}$. Note that $V_U(z)$ is not an estimate of $V(z)$, but rather has the same expectation.

At first glance, V_U may seem just as impractical as V_Z since σ is not known. However, with a careful choice of U we can insure that the weight term depends only on the known strategies σ_i and $\hat{\sigma}_i$. Before presenting example choices of U, we first prove that V_U is unbiased.

Theorem 1 *If $\pi_i^{\hat{\sigma}}(z)$ is non-zero for all outcomes $z \in Z$, then,*

$$E_{z|\hat{\sigma}}\left[V_U(z)\right] = E_{z|\sigma}\left[V(z)\right],$$

i.e., V_U is an unbiased estimator.

Proof: First, let us consider the denominator in the weight term of V_U. Since $z' \in U(z')$ and $\pi_i^{\hat{\sigma}}$ is always positive, the denominator can only be zero if $\pi_{-i}^{\hat{\sigma}}(z')$ is zero. If this were true, $\pi_{-i}^{\sigma}(z')$ must also be zero, and as a consequence so must the numerator. As a result the terminal history z' is never reached and so it is correct to simply exclude such histories from the estimator's summation.

Define $\mathbf{1}(x)$ to be the indicator function that takes on the

value 1 if x is true and 0 if false.

$$E_{z|\hat{\sigma}}\left[V_U(z)\right]$$

$$= E_{z|\hat{\sigma}}\left[\sum_{z' \in U^{-1}(z)} V(z')\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))}\right] \tag{4}$$

$$= E_{z|\hat{\sigma}}\left[\sum_{z'} \mathbf{1}(z \in U(z'))V(z')\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))}\right] \tag{5}$$

$$= \sum_{z'} V(z')\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))}E_{z|\hat{\sigma}}\left[\mathbf{1}(z \in U(z'))\right] \tag{6}$$

$$= \sum_{z'} V(z')\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))}\pi^{\hat{\sigma}}(U(z')) \tag{7}$$

$$= \sum_{z'} V(z')\pi^{\sigma}(z') = E_{z|\sigma}\left[V(z)\right] \tag{8}$$

The derivation follows from the linearity of expectation, the definition of $\pi^{\hat{\sigma}}$, and the definition of expectation. ∎

We now look at four specific choices of U for which the weight term can be computed while only knowing player i's portion of the joint strategy σ.

Example 1: Basic Importance Sampling. The simplest choice of U for which V_U can be computed is $U(z) = \{z\}$. In other words, the estimator considers just the sampled history. In this case the weight term is:

$$\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))} = \frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(z')} \tag{9}$$

$$= \frac{\pi^{\sigma}_i(z')\pi^{\sigma}_{-i}(z')}{\pi^{\hat{\sigma}}_i(z')\pi^{\hat{\sigma}}_{-i}(z')} \tag{10}$$

$$= \frac{\pi^{\sigma}_i(z')}{\pi^{\hat{\sigma}}_i(z')} \tag{11}$$

The weight term only depends on σ_i and $\hat{\sigma}_i$ and so is a known quantity. When $\hat{\sigma}_i = \sigma_i$ the weight term is 1 and the result is simple Monte Carlo estimation. When $\hat{\sigma}_i$ is different, the estimator is a straightforward application of importance sampling.

Example 2: Game Ending Actions. A more interesting example is to consider all histories that differ from the sample history by only a single action by player i and that action must be the last action in the history. For example, in poker, the history where the player being evaluated chooses to fold at an earlier point in the betting sequence is considered in this estimator. Formally, define $S_{-i}(z) \in H$ to be the shortest prefix of z where the remaining actions in z are all made by player i or chance. Let $U(z) = \{z' \in Z : S_{-i}(z) \text{ is a prefix of } z'\}$. The weight term becomes,

$$\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))} = \frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(S_{-i}(z'))} \tag{12}$$

$$= \frac{\pi^{\sigma}_{-i}(z')\pi^{\sigma}_i(z')}{\pi^{\hat{\sigma}}_{-i}(S_{-i}(z'))\pi^{\hat{\sigma}}_i(S_{-i}(z'))} \tag{13}$$

$$= \frac{\pi^{\sigma}_{-i}(S_{-i}(z'))\pi^{\sigma}_i(z')}{\pi^{\hat{\sigma}}_{-i}(S_{-i}(z'))\pi^{\hat{\sigma}}_i(S_{-i}(z'))} \tag{14}$$

$$= \frac{\pi^{\sigma}_i(z')}{\pi^{\hat{\sigma}}_i(S_{-i}(z'))} \tag{15}$$

As this only depends on the strategies of player i, we can compute this quantity and therefore the estimator.

Example 3: Private Information. We can also use all histories in the update that differ only in player i's private information. In other words, any history that the other players wouldn't be able to distinguish from the sampled history is considered. For example, in poker, any history where player i receiving different private cards is considered in the estimator since the opponents' strategy cannot depend directly on this strictly private information. Formally, let $U(z) = \left\{z' \in Z : \forall \sigma\ \pi^{\sigma}_{-i}(z') = \pi^{\sigma}_{-i}(z)\right\}$. The weight term then becomes,

$$\frac{\pi^{\sigma}(z')}{\pi^{\hat{\sigma}}(U(z'))} = \frac{\pi^{\sigma}(z')}{\sum_{z'' \in U(z')}\pi^{\hat{\sigma}}(z'')} \tag{16}$$

$$= \frac{\pi^{\sigma}_{-i}(z')\pi^{\sigma}_i(z')}{\sum_{z'' \in U(z')}\pi^{\hat{\sigma}}_{-i}(z'')\pi^{\hat{\sigma}}_i(z'')} \tag{17}$$

$$= \frac{\pi^{\sigma}_{-i}(z')\pi^{\sigma}_i(z')}{\sum_{z'' \in U(z')}\pi^{\hat{\sigma}}_{-i}(z')\pi^{\hat{\sigma}}_i(z'')} \tag{18}$$

$$= \frac{\pi^{\sigma}_{-i}(z')\pi^{\sigma}_i(z')}{\pi^{\hat{\sigma}}_{-i}(z')\sum_{z'' \in U(z')}\pi^{\hat{\sigma}}_i(z'')} \tag{19}$$

$$= \frac{\pi^{\sigma}_i(z')}{\pi^{\hat{\sigma}}_i(U(z'))} \tag{20}$$

As this only depends on the strategies of player i, we can again compute this quantity and therefore the estimator as well.

Example 4: Combined. The past two examples show that we can consider histories that differ in the player's private information or by the player making an alternative game ending action. We can also combine these two ideas and consider any history that differs by both an alternative game ending action and the player's private information. Define $Q(z) = \left\{h \in H : |h| = |S_{-i}(z)| \text{ and } \forall \sigma \pi^{\sigma}_{-i}(h) = \pi^{\sigma}_{-i}(S_{-i}(z))\right\}$,

Let $U(z) = \{z' \in Z : \text{ a prefix of } z' \text{ is in } Q(z)\}$.

$$\frac{\pi^\sigma(z')}{\pi^{\hat{\sigma}}(U(z'))} = \frac{\pi^\sigma(z')}{\pi^{\hat{\sigma}}(Q(z'))} \tag{21}$$

$$= \frac{\pi^\sigma_{-i}(z')\pi^\sigma_i(z')}{\sum_{h \in Q(z')} \pi^{\hat{\sigma}}_{-i}(h)\pi^{\hat{\sigma}}_i(h)} \tag{22}$$

$$= \frac{\pi^\sigma_{-i}(z')\pi^\sigma_i(z')}{\sum_{h \in Q(z')} \pi^{\hat{\sigma}}_{-i}(S_{-i}(z))\pi^{\hat{\sigma}}_i(h)} \tag{23}$$

$$= \frac{\pi^\sigma_{-i}(S_{-i}(z'))\pi^\sigma_i(z')}{\pi^{\hat{\sigma}}_{-i}(S_{-i}(z')) \sum_{h \in Q(z')} \pi^{\hat{\sigma}}_i(h)} \tag{24}$$

$$= \frac{\pi^\sigma_i(z')}{\pi^{\hat{\sigma}}_i(Q(z'))} \tag{25}$$

Once again this quantity only depends on the strategies of player i and so we can compute this estimator as well.

We have presented four different estimators that try to extract additional information from a single observed game outcome. We can actually combine any of these estimators with other unbiased approaches for reducing variance. This can be done by replacing the V function in the above estimators with any unbiased estimate of V. In particular, these estimators can be combined with our previous DIVAT approach by choosing V to be the DIVAT (or BC-DIVAT) estimator instead of u_i.

3.1. Partial Information

The estimators above are provably unbiased for both the-policy and off-policy full-information case. We now briefly discuss the off-policy partial-information case. In this case we don't directly observe the actual terminal history z_t but only a many-to-one mapping $K(z_t)$ of the history. One simple adaptation of our estimators to this case is to use the history z' in the estimator whenever it is possible that the unknown terminal history could be in $U(z')$, while keeping the weight term unchanged. Although we lose the unbiased guarantee with these estimators, it is possible that the reduction in variance is more substantial than the error caused by the bias. We investigate empirically the magnitude of the bias and the resulting mean-squared error of such estimators in the domain of poker in Section 5.

4. Application to Poker

To analyze the effectiveness of these estimators, we will use the popular game of Texas Hold'em poker, as played in the AAAI Computer Poker Competition (Zinkevich & Littman, 2006). The game is two-player and zero-sum. Private cards are dealt to the players, and over four rounds, public cards are revealed. During each round, the players place bets that the combination of their public and private cards will be the strongest at the end of the game. The game has just under 10^{18} game states, and has the properties of imperfect information, stochastic outcomes, and observations of the game outcome during a match exhibit partial information.

Each of the situations described in Section 2, on-policy and off-policy as well as full-information and partial information, have relevance in the domain of poker. In particular, the *on-policy full-information* case is the situation where one is trying to evaluate a strategy from full-information descriptions of the hands, as might be available after a match is complete. For example, this could be used to more accurately determine the winner of a competition involving a small number of hands (which is always the case when humans are involved). In this situation it is critical, that the estimator is unbiased, i.e., it is an accurate reflection of the expected winnings and therefore does not incorrectly favor any playing style.

The *off-policy full-information* case is useful for examining past games against an opponent to determine which of many alternative strategies one might want to use against them in the future. The introduction of bias (depending on the strategy used when playing the past hands) is not problematic, as the goal in this case is an estimate with as little error as possible. Hence the introduction of bias is acceptable in exchange for significant decreases in variance.

Finally, the *off-policy partial-information* case corresponds to evaluating alternative strategies during an actual match. In this case, we want to evaluate a set of strategies, which aren't being played, to try and identify an effective choice for the current opponent. The player could then choose a strategy whose performance is estimated to be strong even for hands it wasn't playing.

The estimators from the previous section all have natural applications to the game of poker:

- **Basic Importance Sampling**. This is a straightforward application of importance sampling. The value of the observed outcome of the hand is weighted by the ratio of the probability that the strategy being evaluated (σ_i) takes the same sequence of actions to the probability that the playing strategy ($\hat{\sigma}_i$) takes the sequence of actions.

- **Game ending actions**. By selecting the *fold* betting action, a player surrenders the game in order to avoid matching an opponent's bet. Therefore, the game ending actions estimator can consider all histories in which the player could have folded during the observed history.[2] We call this the **Early Folds** (EF) estimator. The estimator sums over all possible prefixes

[2] In the full-information setting we can also consider situations where the player could have *called* on the final round of betting to end the hand.

of the betting sequence where the player could have chosen to fold. In the summation it weights the value of surrendering the pot at that point by the ratio of the probability of the observed betting up to that point and then folding given the player's cards (and σ_i) to the probability of the observed betting up to that point given the player's cards (and $\hat{\sigma}_i$).

- **Private information**. In Texas Hold'em, a player's private information is simply the two private cards they are dealt. Therefore, the private information estimator can consider all histories with the same betting sequence in which the player holds different private cards. We call this the **All Cards** (AC) estimator. The estimator sums over all possible two-card combinations (excepting those involving exposed board or opponent cards). In the summation it weights the value of the observed betting with the imagined cards by the ratio of the probability of the observed betting given those cards (and σ_i) to the probability of the observed betting (given $\hat{\sigma}_i$) summed over all cards.

5. Results

Over the past few years we have created a number of strong Texas Hold'em poker agents that have competed in the past two AAAI Computer Poker Competitions. To evaluate our new estimators, we consider games played between three of these poker agents: S2298 (Zinkevich et al., 2007), PsOpti4 (Billings et al., 2003), and CFR8 (Zinkevich et al., 2008). In addition, we also consider Orange, a competitor in the First Man-Machine Poker Championship.

To evaluate these estimators, we examined records of games played between each of three candidate strategies (S2298, CFR8, Orange) against the opponent PsOpti4. Each of these three records contains one million hands of poker, and can be viewed as full information (both players' private cards are always shown) or as partial information (when the opponent folds, their private cards are not revealed). We begin with the full-information experiments.

5.1. Full Information

We used the estimators described previously to find the value of each of the three candidate strategies, using full-information records of games played from just one of the candidate strategies. The strategy that actually played the hands in the record of games is called the on-policy strategy and the others are the off-policy strategies. The results of one these experiments is presented in Table 1. In this experiment, we examined one million full-information hands of S2298 playing against PsOpti4. S2298 (the on-policy strategy) and CFR8 and Orange (the off-policy strategies) are evaluated by our importance sampling estimators, as well as DIVAT, BC-DIVAT, and a few combination estimators. We present the empirical bias and standard deviation of the estimators in the first two columns. The third column, "RMSE", is the root-mean-squared error of the estimator if it were used as the method of evaluation for a 1000 hand match (a typical match length). All of the numbers are reported in millibets per hand played. A millibet is one thousandth of a small-bet, the fixed magnitude of bets used in the first two rounds of betting. To provide some intuition for these numbers, a player that always folds will lose 750 millibets per hand, and strong players aim to achieve an expected win rate over 50 millibets per hand.

Table 1. Full Information Case. Empirical bias, standard deviation, and root mean-squared-error over a 1000 hand match for various estimators. 1 million hands of poker between S2298 and PsOpti4 were observed. A bias of 0* indicates a provably unbiased estimator.

	Bias	StdDev	RMSE
S2298			
Basic	0*	5103	161
DIVAT	0*	1935	61
BC-DIVAT	0*	2891	91
Early Folds	0*	5126	162
All Cards	0*	4213	133
AC+BC-DIVAT	0*	2146	68
AC+EF+BC-DIVAT	0*	1778	56
CFR8			
Basic	200 ± 122	62543	1988
DIVAT	62 ± 104	53033	1678
BC-DIVAT	84 ± 45	22303	710
Early Folds	123 ± 120	61481	1948
All Cards	12 ± 16	8518	270
AC+BC-DIVAT	35 ± 13	3254	109
AC+EF+BC-DIVAT	2 ± 12	2514	80
Orange			
Basic	159 ± 40	20559	669
DIVAT	3 ± 25	11350	359
BC-DIVAT	103 ± 28	12862	420
Early Folds	82 ± 35	17923	572
All Cards	7 ± 16	8591	272
AC+BC-DIVAT	8 ± 13	3154	100
AC+EF+BC-DIVAT	6 ± 12	2421	77

In the on-policy case, where we are evaluating S2298, all of the estimators are provably unbiased, and so they only differ in variance. Note that the Basic estimator, in this case, is just the Monte-Carlo estimator over the actual money lost or won. The Early Folds estimator provides no variance reduction over the Monte-Carlo estimate, while the All Cards estimator provides only a slight reduction. However, this is not nearly as dramatic as the reduction provided by the DIVAT estimator. The importance sampling estimators, however, can be combined with the DIVAT es-

timator as described in Section . The combination of BC-DIVAT with All Cards ("AC+BC-DIVAT") results in lower variance than either of the estimators separately.[3] The addition of Early Folds ("AC+EF+BC-DIVAT") produces an even further reduction in variance, showing the best-performance of all the estimators, even though Early Folds on its own had little effect.

In the off-policy case, where we are evaluating CFR8 or Orange, we report the empirical bias (along with a 95% confidence bound) in addition to the variance. As DIVAT and BC-DIVAT were not designed for off-policy evaluation, we report numbers by combining them with the Basic estimator (i.e., using traditional importance sampling). Note that bias is possible in this case because our on-policy strategy (S2298) does not satisfy the assumption in Theorem 1, as there are some outcomes the strategy never plays. Basic importance sampling in this setting not only shows statistically significant bias, but also exhibits impractically large variance. DIVAT and BC-DIVAT, which caused considerable variance reduction on-policy, also should considerable variance reduction off-policy, but not enough to offset the extra variance from basic importance sampling. The All Cards estimator, on the other hand, shows dramatically lower variance with very little bias (in fact, the empirical bias is statistically insignificant). Combining the All Cards estimator with BC-DIVAT and Early Folds further reduces the variance, giving off-policy estimators that are almost as accurate as our best on-policy estimators.

The trends noted above continue in the other experiments, when CFR8 and Orange are being observed. For space considerations, we don't present the individual tables, but instead summarize these experiments in Table 2. The table shows the minimum and maximum empirically observed bias, standard deviation, and the root-mean-squared error of the estimator for a 1000 hand match. The strategies being evaluated are separated into the on-policy case, when the record involves data from that strategy, and the off-policy case, when it doesn't.

5.2. Partial Information

The same experiments were repeated for the case of partial information. The results of the experiment involving S2298 playing against PsOpti4 and evaluating our three candidate strategies under partial information is shown in Table 3. For DIVAT and BC-DIVAT, which require full information of the game outcome, we used a partial information variant where the full-information estimator was used when the game outcome was known (i.e., no player folded) and winnings was used when it was not. This variant can result in a biased estimator, as can be seen in the table of results. The All Cards estimator, although also without any guarantee of being unbiased, actually fares much better in practice, not displaying a statistically significant bias in either the off-policy or on-policy experiments. However, even though the DIVAT estimators are biased their low variance makes them preferred in terms of RMSE in the on-policy setting. In the off-policy setting, the variance caused by Basic importance sampling (as used with DIVAT and BC-DIVAT) makes the All Cards estimator the only practical choice. As in the full-information case we can combine the All Cards and BC-DIVAT for further variance reduction. The resulting estimator has lower RMSE than either All Cards or BC-DIVAT alone both in the on-policy and off-policy cases. The summary of the results of the other experiments, showing similar trends, are shown in Table 4.

Table 3. Partial-Information Case. Empirical bias, standard deviation, and root mean-squared-error over a 1000 hand match for various estimators. 1 million hands of poker between S2298 and PsOpti4 with partial information were observed. A bias of 0* indicates a provably unbiased estimator.

	Bias	StdDev	RMSE
S2298			
Basic	0*	5104	161
DIVAT	81±9	2762	119
BC-DIVAT	95±9	2759	129
Early Folds	47±1	5065	167
All Cards	5±13	4218	133
AC+BC-DIVAT	96±12	2650	127
CFR8			
Basic	202±80	40903	1309
DIVAT	175±47	23376	760
BC-DIVAT	183±47	23402	762
Early Folds	181±78	39877	1274
All Cards	13±19	7904	250
AC+BC-DIVAT	101±16	4014	162
Orange			
Basic	204±45	23314	765
DIVAT	218±22	10029	385
BC-DIVAT	244±21	10045	401
Early Folds	218±43	22379	741
All Cards	3±19	8092	256
AC+BC-DIVAT	203±16	3880	237

6. Conclusion

We introduced a new method for estimating agent performance in extensive games based on importance sampling. The technique exploits the fact that the agent's strategy is typically known to derive several low variance estima-

[3]The importance sampling estimators were combined with BC-DIVAT instead of DIVAT because the original DIVAT estimator is computationally burdensome, particularly when many evaluations are needed for every observation as is the case with the All Cards estimator.

Table 2. Summary of the Full-Information Case. Summary of empirical bias, standard deviation, and root-mean-squared error over a 1000 hand match for various estimators. The minimum and maximum encountered values for all combinations of observed and evaluated strategies is presented. A bias of 0* indicates a provably unbiased estimator.

	Bias			StdDev			RMSE		
	Min	–	Max	Min	–	Max	Min	–	Max
On Policy									
Basic	0*	–	0*	5102	–	5385	161	–	170
DIVAT	0*	–	0*	1935	–	2011	61	–	64
BC-DIVAT	0*	–	0*	2891	–	2930	91	–	92
AC+GE+BC-DIVAT	0*	–	0*	1701	–	1778	54	–	56
Off Policy									
Basic	49	–	200	20559	–	244469	669	–	7732
DIVAT	2	–	62	11350	–	138834	358	–	4390
BC-DIVAT	10	–	103	12862	–	173715	419	–	5493
AC+GE+BC-DIVAT	2	–	9	1816	–	2857	58	–	90

Table 4. Summary of the Partial-Information Case. Summary of empirical bias, standard deviation, and root-mean-squared error over a 1000 hand match for various estimators. The minimum and maximum encountered values for all combinations of observed and evaluated strategies is presented. A bias of 0* indicates a provably unbiased estimator.

	Bias			StdDev			RMSE		
	Min	–	Max	Min	–	Max	Min	–	Max
On Policy									
Basic	0*	–	0*	5104	–	5391	161	–	170
DIVAT	56	–	144	2762	–	2876	105	–	170
BC-DIVAT	78	–	199	2759	–	2859	118	–	219
AC+BC-DIVAT	78	–	206	2656	–	2766	115	–	224
Off Policy									
Basic	17	–	433	23314	–	238874	753	–	7566
DIVAT	103	–	282	10029	–	88791	384	–	2822
BC-DIVAT	35	–	243	10045	–	99287	400	–	3139
AC+BC-DIVAT	63	–	230	3055	–	6785	143	–	258

tors that can simultaneously evaluate many strategies while playing a single strategy. We prove that these estimators are unbiased in both the on-policy and off-policy case. We empirically evaluate the techniques in the domain of poker, showing significant improvements in terms of lower variance and lower bias. We show that the estimators can also be used even in the challenging problem of estimation with partial information observations.

Acknowledgments

We would like to thank Martin Zinkevich and Morgan Kan along with the entire University of Alberta Computer Poker Research Group for their valuable insights. This research was supported by NSERC and iCore.

References

Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., & Szafron, D. (2003). Approximating game-theoretic optimal strategies for full-scale poker. *International Joint Conference on Artificial Intelligence* (pp. 661–668).

Osborne, M., & Rubenstein, A. (1994). *A course in game theory*. Cambridge, Massachusetts: The MIT Press.

Zinkevich, M., Bowling, M., Bard, N., Kan, M., & Billings, D. (2006). Optimal unbiased estimators for evaluating agent performance. *American Association of Artificial Intelligence National Conference, AAAI'06* (pp. 573–578).

Zinkevich, M., Bowling, M., & Burch, N. (2007). A new algorithm for generating equilibria in massive zero-sum games. *Proceedings of the Twenty-Second Conference on Artificial Intelligence* (pp. 788–793).

Zinkevich, M., Johanson, M., Bowling, M., & Piccione, C. (2008). Regret minimization in games with incomplete information. *Advances in Neural Information Processing Systems 20*. To appear (8 pages).

Zinkevich, M., & Littman, M. (2006). The AAAI computer poker competition. *Journal of the International Computer Games Association*, *29*. News item.

Actively Learning Level-Sets of Composite Functions

Brent Bryan BRENT@GOOGLE.COM
Google Inc., 4720 Forbes Ave., Pittsburgh, PA 15213

Jeff Schneider SCHNEIDE@CS.CMU.EDU
Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213

Abstract

Scientists frequently have multiple types of experiments and data sets on which they can test the validity of their parameterized models and locate plausible regions for the model parameters. By examining multiple data sets, scientists can obtain inferences which typically are much more informative than the deductions derived from each of the data sources independently. Several standard data combination techniques result in target functions which are a weighted sum of the observed data sources. Thus, computing constraints on the plausible regions of the model parameter space can be formulated as finding a level set of a target function which is the sum of observable functions. We propose an active learning algorithm for this problem which selects both a sample (from the parameter space) and an observable function upon which to compute the next sample. Empirical tests on synthetic functions and on real data for an eight parameter cosmological model show that our algorithm significantly reduces the number of samples required to identify the desired level-set.

1. Introduction

Scientists frequently have multiple types of experiments and data sets on which they can test the validity of their parameterized models and the plausible or optimal regions for the model parameters. One task that can be considered is that of computing the parameter setting (from a pre-defined model parameter space) which maximizes the likelihood of all the observations given the models. However, this calculation does not determine whether or not the derived parameter setting is consistent with the data given the models. Instead, a more prudent approach is to compute the set of model parameters (from the parameter space) which cannot be statistically rejected by the combination of the observed data and theoretical models.

When given a single model and data set pair, computation of the feasible regions of parameter space can be done by performing a simple hypothesis test for all points in the space; that is, we are interested in the regions of parameter space where the null hypothesis — that the data was generated by the model — cannot be rejected at some specified confidence level. Extending this to the multiple model and data setting, we are interested in determining regions of parameter space where we cannot reject the hypothesis that each of the data sets was generated by its respective model at a given confidence level.

For example, when determining the spatial location of a disease outbreak, a researcher might use information derived from medical records (e.g. hospital admits), as well as sales of over the counter and prescription medications (Shmueli & Fienberg, 2006). Note that the presence (or lack thereof) of a single indicator may be enough to accept or reject a single hypothesis, resulting in increased data efficiency. Specifically, if there are many hospital admits from a single locality, the probability of disease is extremely high regardless of the over the counter and prescription drug sales. Moreover, while we believe that the underlying cause affects each of the signals we observe, we do not necessarily believe that the signals themselves are correlated. For instance, colds result in significant over the counter sales with few hospital visits or prescription sales. However, anthrax attacks will affect all three data streams.

There are many other examples of the multiple model setting. Here, we focus on finding $1 - \alpha$ confidence regions for statistical analyses involving multiple related data sets. Traditionally, the combination of statistical evidence has been achieved in the sciences in a somewhat ad-hoc fashion. For instance, a joint analysis can be performed by (loosely) intersecting the confidence regions of several studies. Additionally, results from one publication might be used to guide the selection of parameters in future experiments, possibly in the form of a prior.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

A more rigorous and efficient approach is to consider multiple experimental sources of evaluation simultaneously and choose samples in light of their contribution to the combined target function. This target function is the composition of the "observable" test functions: one for each data set and model pair. We assume that the observable functions share the same parameter space, but are functionally independent. As such, hierarchical models do not apply. Moreover, whereas multi-task learning problems are based on learning the commonality between the constituent models, the task of locating confidence regions benefits from the discrepancies between the models to efficiently accept or reject a parameter vector. While in theory we could check each point in the parameter space to determine whether or not it should be included within our $1{-}\alpha$ confidence region, in practice each experiment is too expensive.

As such, we develop active learning algorithms to learn the confidence regions. Active learning using informed choices of future experiments has long been known to drastically decrease a problem's sample complexity (Angluin, 1988). Many sampling heuristics have been developed to learn either the entire target function (e.g. MacKay (1992); Guestrin, C., et al. (2005)) or some feature of the target function, such as its level sets (e.g. Bryan, B., et al. (2005); Ramakrishnan, N., et al. (2005)). While we cannot directly observe the value of the target function, we can use the observable functions to infer its value. By measuring all observable functions at a particular parameter setting, we can compute the value of the target function, reducing the problem to a standard active learning problem. However, such an approach disregards any strong evidence provided by a single statistical test, and hence may result in extraneous sampling of the remaining statistical models.

Rather, we are interested in active learning algorithms which use information about each observable function to learn some composite target function. In Section 2, we propose a heuristic for actively learning level sets of composite functions of sums for continuous valued input spaces. In Section 3, we show that this heuristic performs the level-set discovery task more efficiently than both random and sequential sampling of the constituent functions using state of the art heuristics. In Section 4, we discuss how the task of finding joint confidence regions can be formulated as a level set problem, where the target function is the sum of several observable functions. Section 5 concludes by demonstrating the computation of 95% confidence regions for eight cosmological parameters using our algorithm.

2. Active Learning Algorithm

Let f be a target function we are interested in learning on the domain $\Theta \subseteq \mathbb{R}^d$. Suppose that f is the linear combination of m observable functions, f_i ($i = 1, \ldots, m$). Without loss of generality, we can drop the coefficients from the summation (as they can be included in the f_i's) and write $f(\theta) = \sum_{i=1}^{m} f_i(\theta)$ for all $\theta \in \Theta$. We are now interested in finding the level set, $\mathcal{S}$, of f at the threshold t:

$$\mathcal{S} = \left\{ \theta \in \Theta \,\middle|\, \sum_{i=1}^{m} f_i(\theta) = f(\theta) = t \right\}.$$

In general, computing the value of each f_i may not incur the same cost. However, we begin by assuming that the costs are similar, and hence try to minimize the total number of samples of observable functions required to accurately estimate $\mathcal{S}$. Moreover, we assume that f cannot be directly sampled, and that neither f nor any of the f_i's is invertible. That is, the only way to estimate a level-set of f is to sample points from the f_i's and infer f. As we will see in Section 4, this formulation accurately mimics combining p-values using Fisher's method, as the method for finding the individual p-values may be entirely unknown.

We must now determine how best to choose samples both among and within the f_i's. Ideally, we want to sample the observable function f_i at the point $\tilde{\theta} \in \Theta$ which best increases our prediction accuracy (e.g. whether another point is above or below the threshold) over f. Since the parameter space is continuous and multi-dimensional, we cannot afford to test all possible points and observable functions.

Instead, we model each of the observable functions independently given the current samples taken from that function, as illustrated in Figure 1. For each experiment, we randomly select a small subset of the parameter space (usually 1000 points drawn uniformly at random, although other distributions are possible based on domain knowledge) and choose the best point and observable function pair upon which to experiment from among these candidates. We find the value of the observable function at the selected point and add it to the data set used to model that function. The process is then repeated.

There are several methods one could use to model each of the f_i's, notably some form of parametric regression. However, we chose to approximate the f_i's using Gaussian process regression, as other forms of regression may over smooth the data, ignoring subtle features of the function that may become pronounced with more data. While much work has been done studying Gaussian processes, we only touch on the basic concepts here; we refer interested readers to Cressie (1991); Rasmussen and Williams (2006).

Gaussian processes are non-parametric forms of regression. Predictions for unobserved points are computed by using a weighted combination of the function values for those points which have already been observed, where a distance-based kernel function is used to determine the relative weights. These distance-based kernels generally weight

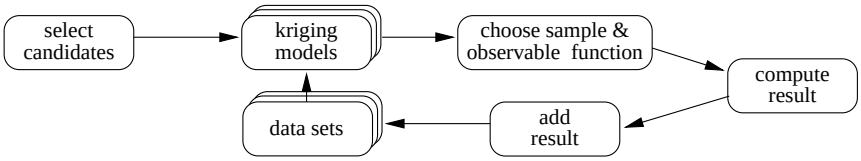


Figure 1. Outline of our sampling algorithm. Given an initial set of points (typically empty), we randomly select a set of candidates and score them using a set of Gaussian process models. The best scoring point and observable function pair is chosen, and we evaluate the selected observable function at the given point. This data is added to the corresponding data set.

nearby points significantly more than distant points. Thus, assuming the underlying function is continuous, Gaussian processes will perfectly describe the function given an infinite set of unique data points. While, in many applications the assumption of continuity is violated, Gaussian processes have been successfully used to model response surfaces in many domains with limited smoothness guarantees (Cressie, 1991; Santner et al., 2003).

In this work we use ordinary kriging (Cressie, 1991), which assumes a linear semivariance as a function of distance, as it is both data and computationally efficient. While other forms of Gaussian Processes could be used — most notably adaptive kernel methods (e.g. Kersting, K. et al. (2007)) — we find that a learned model based upon a simple kriging approximator performs well in practice and ensures that we do not spend more time computing the next sample than we do running the experiment.

Regardless of the kernel used, Gaussian processes predict that the value of a target point, $\tilde{\theta}$, will be Normally distributed with a mean and variance ($f_i(\tilde{\theta})$ and $\sigma_i^2(\tilde{\theta})$, respectively) given by:

$$f_i(\tilde{\theta}) = \bar{f}_i + \vec{\Sigma}_{i,\tilde{\theta}}^T \mathbf{\Sigma}_i^{-1} \vec{\mathcal{F}}_i \quad (1)$$

$$\sigma^2(\tilde{\theta}) = \vec{\Sigma}_{i,\tilde{\theta}}^T \mathbf{\Sigma}_i^{-1} \vec{\Sigma}_{i,\tilde{\theta}} \quad (2)$$

where $\mathcal{T}_i$ is the set of observed experiments of f_i,

$$\bar{f}_i = \frac{1}{|\mathcal{T}_i|} \sum_{j=1}^{|\mathcal{T}_i|} f_i(a_j),$$

$$\mathcal{F}_i[j] = f_i(\theta_j) - \bar{f}_i,$$

$\mathbf{\Sigma_i}$ denotes the covariance matrix between the elements of $\mathcal{T}_i$, and $\vec{\Sigma}_{i,\tilde{\theta}}$ is the covariance vector between elements of $\mathcal{T}_i$ and $\tilde{\theta}$.

For a set of n_i observed points ($|\mathcal{T}_i| = n_i$), prediction with a Gaussian process requires $O(n_i^3)$ time, as a $n_i \times n_i$ linear system of equations must be solved. However, for many Gaussian processes — and ordinary kriging in particular — the correlation between two points decreases as a function of distance. Thus, the full Gaussian process model is approximated well by a local Gaussian process in which only the k nearest neighbors of the query point are used, for some fixed constant k. This reduces the computation time to $O(k^3 + k \log(n_i))$ per prediction. Here, we let $k = 1000$.

2.1. Choosing Experiments

Given this active learning framework, we must now decide how to choose sample / observable function pairs. We consider the following heuristics:

Random One of the candidate points and an observable function pair is chosen uniformly at random. This method serves as a baseline for comparison of the other heuristics.

Variance The candidate point and observable function pair which has the highest predicted variance (out of all the candidate / observable function pairs) is selected. Using model variance to pick the next experiment is common for active learning methods whose goal is to map out the target function over a parameter space (MacKay, 1992; Guestrin, C., et al., 2005). In particular, (Guestrin, C., et al., 2005) showed that greedily picking experiments based upon model variance performs nearly as well as using a mutual information heuristic when learning the target over the entire parameter space; this is significant, as the mutual information heuristic can be shown to be $(1 - 1/e)$ optimal (Guestrin, C., et al., 2005). Since variance is closely related to distance for kriging models, this heuristic samples points which are far from their nearest neighbors. However, when searching for level-sets, we are less interested in the function away from the level-set boundary, and instead want to focus our sampling resources near the predicted boundary. In particular, sampling based solely on variance results in substantially worse performance than heuristics that concentrate on the function level-set (Bryan, B., et al., 2005).

Information Gain Information gain is a common myopic metric used in active learning. Computing the information gain over the whole state space for each observable function provides an optimal 1-step experiment choice. In some discrete or linear problems this can be done, but it is intractable for continuous non-linear spaces. As such we do not consider a traditional information gain heuristic, but rely on efficient point estimates which act as proxies for global information gain.

Sequential-Straddle As noted in Section 1, the problem can be simplified to a standard active learning problem if one sequentially samples each of the observable functions in order to directly compute f. (Bryan, B., et al., 2005) showed that in a setting where experiments yield the (approximately) true values of the target function, a good heuristic for level set identification is the straddle heuristic: $\text{straddle}(\tilde{\theta}) = 1.96\sigma^2(\tilde{\theta}) - |f(\tilde{\theta}) - t|$. This heuristic balances the need to explore uncertain parts of parameter space, with the desire to refine the model's estimate around those regions already known to be close to the level-set boundary; the constant 1.96 ensures that points with negative scores are far from the desired level set with at least a 95% probability. This heuristic leverages the straddle heuristic by choosing the candidate point with the highest combined straddle score,

$$\text{combined-straddle}(\tilde{\theta}) = 1.96 \sum_{i=1}^{m} \sigma_i^2(\tilde{\theta}) - \left| \sum_{i=1}^{m} f_i(\tilde{\theta}) - t \right|, \quad (3)$$

and then sequentially sampling all m observable functions at this point.

Variance-Straddle While (Bryan, B., et al., 2005) showed that the straddle heuristic works well when directly sampling the target function, we can hope to do better by considering the output from each observable function individually. For instance, if a sample point results in a very large value for one of the observable functions, it may be unlikely that the results of the other f_i's will be such that the resulting value of f is near the level-set. In particular, when dealing with χ^2 models (see Section 4), we know that $f_i \geq 0$ for all i. Thus, if a single f_i is greater than the level-set boundary, the target function will also be greater than the level-set boundary, and hence it may be more efficient to sample elsewhere. This heuristic simply chooses the next sample from among the candidates based on the combined-straddle score, and then selects the observable function with the largest variance at that point.

Variance-MaxVarStraddle Finally, we consider a variant of the straddle heuristic. This heuristic tries to mimic the information gain of choosing a particular point and observable function pair. Note that after observing a point, the variance of the kriging model is effectively zero at that point (since we have set c to be a very small positive value). The original straddle heuristic balances the expected gain in the model fit ($\sigma(\tilde{\theta})$) with the expected distance of the point to the level-set boundary.

However, with the multiple model formulation, we do not expect the model variance to decrease by $\sigma^2(\tilde{\theta}) = \sum_{i=1}^{m} \sigma_i^2(\tilde{\theta})$, but rather by $\sigma_i(\tilde{\theta})$ where f_i is the observable function we pick. Thus, a more accurate proxy for the information gain of a candidate point and observable function pair is:

$$\text{variance-maxvarstraddle}(\tilde{\theta}) = \max_i \left\{ 1.96\sigma_i^2(\tilde{\theta}) \right\} - \left| \sum_{i=1}^{m} f_i(\tilde{\theta}) - t \right|. \quad (4)$$

We choose the candidate point that maximizes this heuristic and the corresponding f_i.

3. Experiments

We now assess the accuracy with which our active learning model reproduces synthetic target functions for the sampling heuristics just described. This is done by computing the fraction of test points in which the predictive model (the sum of the kriging models associated with each observable function) agrees with the true target function about on which side of the threshold the test points lie. This process was repeated 20 times to account for variations due to the random nature of the candidate generation process. The first three target functions considered were sums of two observable functions, while the fourth was a sum of four observable functions. The kriging parameters for each model were computed *a priori* from the observable functions. The considered functions are:

Gaussian This problem consisted of determining the 95% acceptance region of two axis aligned perpendicular two dimensional Gaussian distributions centered at the origin. Both Gaussians had diagonal covariance matrices with on diagonal elements of 1 and 16. Since working in probability space results in many near-zero values, the problem was considered in log-space. As such, the target function was a 2 dimensional symmetric quadratic function, and the level-set was a circle centered at the origin. The range of the parameter space was ($\theta_1, \theta_2 \in [-3.4, 3.4]$)

Sin2D The second problem consists of finding where the two 2D sinusoidal observable functions

$$\begin{aligned} f_1(\theta_1, \theta_2) &= \sin(10\theta_1) + \cos(4\theta_2) - \cos(3\theta_1\theta_2) \\ f_2(\theta_1, \theta_2) &= \sin(10\theta_2) + \cos(4\theta_1) - \cos(3\theta_1\theta_2) \end{aligned}$$

sum to zero where $\theta_1, \theta_2 \in [0, 2]$. These observable functions were chosen because 1) the target threshold winds through the plot giving ample length to test the accuracy of the approximating model, 2) the boundary is discontinuous with several small pieces, 3) there is an ambiguous region around $(0.9, 1)$, where the true function is approximately equal to the threshold, and the gradient is small and 4) there are areas in the domain where the function is far from the threshold and hence we can see whether algorithms refrain from oversampling in these regions.

Table 1. Number of samples required to achieve a 99% accuracy on the Gaussian and SimpleSin2D tests, and a 90% accuracy on the Sin2D and 4-Sin2D tests based on 20 trials. The variance-maxvarstraddle heuristic consistently performs better than competitors.

	Gaussian	SimpleSin2D	Sin2D	4-Sin2D
random	> 1000	> 1000	> 1000	> 1000
variance	95.0 ± 11.0	> 500	105.0 ± 11.5	188.6 ± 32.2
variance-straddle	89.5 ± 5.0	157.9 ± 12.3	90.4 ± 9.0	72.5 ± 12.0
sequential-straddle	76.2 ± 3.5	150.3 ± 6.5	87.0 ± 7.3	98.1 ± 14.0
variance-maxvarstraddle	71.7 ± 3.3	127.3 ± 6.8	82.9 ± 10.2	54.9 ± 16.9

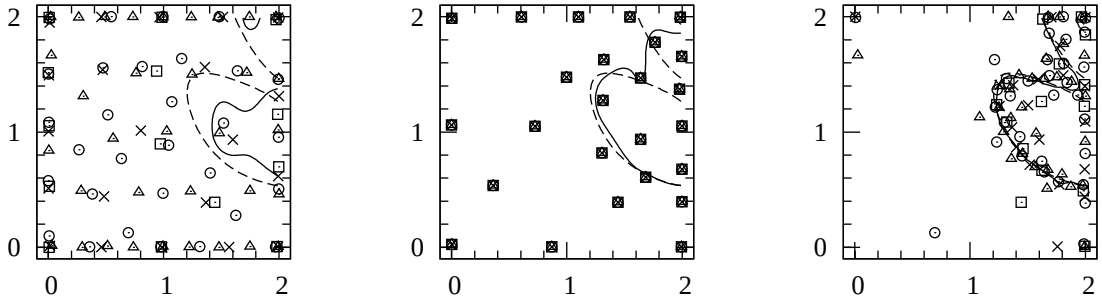


Figure 2. Predicted level-set (solid), true level-set (dashed) and experiments (squares, circle, triangles and x's) for the 4-Sin2D function after sampling 100 points using the Variance heuristic (left), the sequential-straddle heuristic (center), and the variance-maxvarstraddle heuristic (right).

SimpleSin2D This problem is a simplified version of the previous problem, where the observable functions

$$f_1(\theta_1, \theta_2) = \sin(4\theta_1) + \cos(4\theta_2) - \cos(\theta_1\theta_2)$$
$$f_2(\theta_1, \theta_2) = \sin(4\theta_2) + \cos(4\theta_1) - \cos(\theta_1\theta_2)$$

were chosen to reduce the problem's semi-variances (again $\theta_1, \theta_2 \in [0:2]$). Since problems with large semi-variances result in large model variance estimates in the kriging models, such problems require extensive sampling to correctly identify function level-sets. Performance on this function highlights an algorithm's ability to quickly rule out portions of the function.

4-Sin2D This task consisted of finding where four 2D sinusoids sum to -2. The sinusoids chosen for this problem were similar to those of the SimpleSin2D problem:

$$f_1(\theta_1, \theta_2) = \sin(4\theta_1) + \cos(2\theta_2) - \cos(3\theta_1)$$
$$f_2(\theta_1, \theta_2) = \sin(2\theta_2 - 2) + \cos(2\theta_1) - \cos(3\theta_1)$$
$$f_3(\theta_1, \theta_2) = \sin(3\theta_1\theta_2) + \cos(2\theta_1) + 1$$
$$f_4(\theta_1, \theta_2) = \cos(\theta_1\theta_2) - \sin(\theta_1\theta_2)$$

The resulting target function contains regions with both high and low derivatives near the specified threshold.

Classification accuracy results for the four tests are given in Table 1. variance-maxvarstraddle outperforms all of the other heuristics on each of the target functions. Unsurprisingly, the straddle-based heuristics beat the random and variance-weighted heuristics, as both the random and variance-weighted heuristics choose samples (roughly) uniformly throughout the parameter space, while the straddle-based heuristics focus on the level-set of interest. Additionally, the advantage of variance-maxvarstraddle over sequential-straddle grows as the number of observable functions increases, as the relative cost of a bad choice is increased. These results demonstrate that learning the models independently allows for better overall prediction.

One surprising result of our experimentation is that the sequential-straddle performs as well as the variance-straddle heuristic on the test functions which are sums of two observable functions. We believe that this result illustrates the fact that the variance-straddle heuristic is over estimating the importance of the variance component of the candidate points to the information gain of a point, while the fact that there are only two observable functions reduces the efficiency of the sequential-straddle heuristic only by a factor of two. The variance-straddle heuristic will be as likely to choose a candidate point where the predicted observable functions are moderate but equal, as it is to choose a point with a large predicted variance for one of the observable functions, and zero variance for the other observable functions. However, the second candidate has much more information than the first, as selecting the second candidate will give us the (approximately) exact value of the target function, while selecting the first will only reduce the overall variance by a moderate amount. On the 4-Sin2D task the variance-straddle heuristic is able to make use of the individual observable functions, but still does not

do as well as the variance-maxvarstraddle heuristic.

To illustrate the differences in sampling patterns between these heuristics, we plot the samples chosen for the observable functions (with squares, circles, triangles and x's, respectively) with the true (dashed) and predicted (solid) function level-sets for the 4-Sin2D task in Figure 2. The variance-maxvarstraddle heuristic is much better at picking points than the other two heuristics. Note that the variance-maxvarstraddle heuristic is able to learn that some regions of the space are poor by sampling just one of the observable functions; as such, its samples lie much closer to the target level-set. This reinforces our hypothesis that modeling the observable functions separately results in additional learning opportunities.

4. Joint Statistical Analyses

Now let us look at a concrete application of this sampling algorithm: joint statistical analyses. Let X_i be a random variable denoting a data source and x_i be a generic observation of X_i. For each data set, X_i, let m_i be a corresponding model of X_i given some $\theta \in \Theta$. We are interested in constructing a confidence region for the true value of the parameter, denoted $\theta^\star$, based on the observation that $X_i = x_i$ for each model / data set pair.

For a single data set, consider testing the hypothesis that $\theta^\star = \theta$ at level α for some arbitrary $\theta \in \Theta$. The associated acceptance region for the test, $\mathcal{A}_i(\theta)$, is the set of data values (model outputs) for which the test will not reject the hypothesis $\theta^\star = \theta$ for model m_i. Since we are interested in tests with significance level α, we require $P_\theta(X_i \in \mathcal{A}_i(\theta)) \geq 1 - \alpha$. We can then use $\mathcal{A}_i$ to construct a $1-\alpha$ confidence region, $\mathcal{C}_{\mathcal{A}_i}(x_i)$, for $\theta^\star$ based on the observed data x_i: $C_{\mathcal{A}_i}(x_i) = \{\theta \in \Theta | x_i \in \mathcal{A}_i(\theta)\}$.

We consider two approaches to combine the individual confidence tests above into joint confidence regions. In the first we create a statistical model which simultaneously considers all data sets. For instance, when performing an analysis on two data sets using χ^2 tests, we will have one χ^2 test for data set A and a second for data set B. Since the χ^2 test assumes that each of the data points have dependencies given by the covariance matrix, we can combine the two tests into a single χ^2 test of the form

$$\begin{bmatrix} \vec{x}_A - \vec{m}_A \\ \vec{x}_B - \vec{m}_B \end{bmatrix}^T \begin{bmatrix} \Sigma_A & \Sigma_{AB} \\ \Sigma_{AB} & \Sigma_B \end{bmatrix}^{-1} \begin{bmatrix} \vec{x}_A - \vec{m}_A \\ \vec{x}_B - \vec{m}_B \end{bmatrix} \sim \chi^2_{(a+b)}$$

where $m_\dagger, x_\dagger$ and $\Sigma_\dagger$ are the associated test model, observed data and observed covariance of data set $\dagger$ given some vector from the parameter space, a and b are the degrees of freedom of the tests associated with data sets A and B respectively, and Σ_{AB} is the covariance of the data points between data sets A and B. If data sets A and B are independent, then all elements of Σ_{AB} are zero and we can write the above expression as:

$$(\vec{x}_A - \vec{m}_A)^T \Sigma_A^{-1} (\vec{x}_A - \vec{m}_A) + (\vec{x}_B - \vec{m}_B)^T \Sigma_B^{-1} (\vec{x}_B - \vec{m}_B) \sim \chi^2_{(a+b)}.$$

That is, the target function is merely the sum of the two observable functions: the variance weighted sum of squares for both data sets.

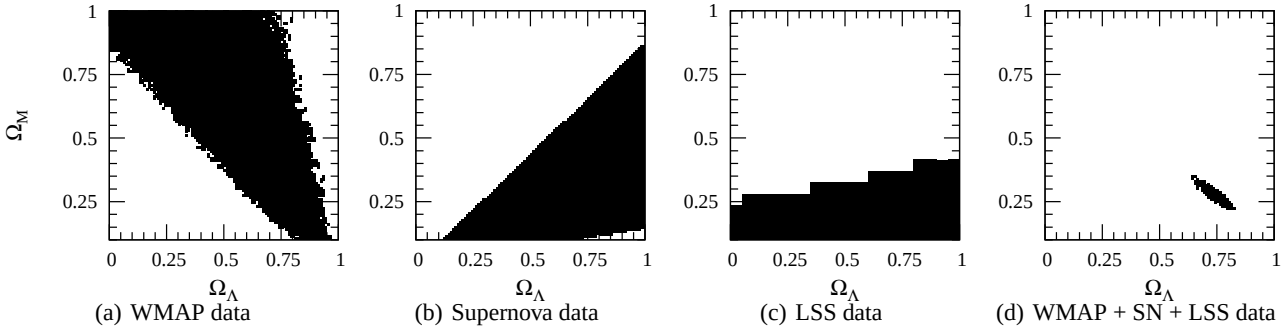
Another approach to performing simultaneous joint analysis is to combine the models' p-values. There are many ways to combine test procedures, including using Bonferroni corrections, the inverse normal method, and inverse logit methods (Hedges, 1985). A common method to combine p-values is Fisher's method (Fisher, 1932). Fisher noted that since a p-value, p_i, has a Uniform distribution, $-2\log(p_i)$ will have a $\chi^2_{(2)}$ distribution. Again, using the fact that the sum of independent χ^2 random variables has a χ^2 distribution, the test becomes: reject H_0 if and only if $-2\sum_{i=1}^{k} \log(p_i) \geq C$ where C is the critical value of a $\chi^2_{(2k)}$ distribution for some particular level α. Again, we see that the target function is the sum of observable functions.

Thus, given the models m_i and data sets X_i, we are interested in locating those $\theta \in \Theta$, such that the the resulting models m_i ($i = 1, \ldots, m$) are accepted by the chosen hypothesis test. This, in turn, reduces to testing whether the sum of a set of observable functions is below a specified threshold. Specifically, given a threshold t, we want to find the set of points, Θ', where the target function f is equal or less than the threshold: $\Theta' = \{\theta \in \Theta | f(\theta) \leq t\}$. However, note that we need only discover the boundary, $\mathcal{S} = \{\theta \in \Theta | f(\theta) = t\}$, as $\mathcal{S}$ implicitly defines Θ'. Therefore, using either χ^2 tests or Fisher's method, we can apply the algorithm described in Section 2 to locate the boundaries of the $1-\alpha$ confidence region.

5. Cosmological Data Example

To illustrate our algorithm and its application to joint statistical analyses, we show how it can be applied to an analysis of eight cosmological parameters that affect the formation and evolution of our universe using three data sets: the Comic Microwave Background (CMB) power spectrum as observed by Wilkinson Microwave Anisotropy Project (WMAP) (Bennett, C. L., et al., 2003), the Davis, T. M., et al. (2007) supernovae (SN) survey and a large scale structure survey (LSS) from Tegmark, M., et al. (2006).

While models for each of these data sets try to determine what the Universe is formed of and how it has evolved, they measure significantly different aspects of the Universe. The CMB data set records temperature fluctuations in the Uni-



(a) WMAP data (b) Supernova data (c) LSS data (d) WMAP + SN + LSS data

Figure 3. Comparison of the confidence regions derived for WMAP (a), supernova (b), and LSS (c) data sets with those derived using all three data sets together (d). Regions of solid color indicate values for $\Omega_{\rm M}$ and Ω_Λ for which some combination of the remaining parameters results in a model with probability greater than $1-\alpha$. The WMAP and LSS models are 7 parameter models, while the supernova is a 3 parameter model, and the combination model is an 8 parameter model.

verse just after the Big-Bang. The size and spatial proximity of these temperature fluctuations depict the types and rates of particle interactions in the early universe and hence characterize the formation of large scale structure (galaxies, clusters, walls and voids) in the current observable universe. Meanwhile, the supernovae data measures the expansion of the universe as a function of time, in order to constrain the total mass and eventual fate of the Universe. Finally, the large scale structure survey measures the degree of galaxy cluster clumping in order to determine the relative importance of dark matter and Baryonic (normal) matter. Combined, these data sets can be used to determine the age, composition and eventual fate of the Universe, as well as provide strong evidence for the presence of dark energy — a large-scale negative gravitational force.

In this analysis we look at an eight dimensional parameter space comprised of the optical depth (τ), dark energy mass fraction (Ω_Λ), total mass fraction ($\Omega_{\rm m}$), baryon density ($\omega_{\rm b}$), dark matter density ($\omega_{\rm dm}$), neutrino fraction (f_n), spectral index (n_s) and galaxy bias (b). The CMB model constrains the first seven parameters while the supernova model constrains $\omega_{\rm dm}$, $\omega_{\rm B}$, $\Omega_{\rm M}$ and Ω_Λ. The LSS model constrains all of the parameters except for τ.

Fisher's method was used to combine p-values from each of the three models. While for small p-values the log of the p-value goes to infinity, note that the algorithm is interested in determining where the sum of the p-values corresponds to the 95% quantile of a $\chi^2_{(6)}$ distribution. Since this results in $t \approx 12.6$, the algorithm has no incentive to select points which are expected to have near zero p-values.

Computing expected observations given parameter vectors is fast for the supernovae and large scale structure models, and hence we can quickly compute the p-values associated with these two models using χ^2 tests. However, computing the expected observations for the CMB data set is much more time consuming. Typically one employs a numerical solver, such as CMBFast to approximate the Boltzmann equation and yield the expected power spectrum.

To alleviate the problem posed by the computational costs of CMBFast, we initialize the Gaussian process model associated with the WMAP data using the one million p-values derived by Bryan, B., et al. (2005). Bryan, B., et al. (2005) uses confidence balls — a statistical procedure similar to χ^2 tests, generally with better inference properties — to map out the level-set associated with the 95% confidence region of the seven CMB parameters. Additional models were selected using the variance-maxvarstraddle heuristic with one small change: If the heuristic selects the observable function associated with the CMB data, we first compute the p-values associated with the supernova and large scale structure data sets to see if we can exclude the parameter vector without needing to run CMBFast. That is, we determine whether the sum of the log p-values from the supernovae and large scale structure data sets alone is larger than the threshold for the combined model. This modification allows us to reduce the number of CMBFast computations by about a factor of five. Using this modified variance-maxvarstraddle heuristic, we sampled roughly 1.5 million additional parameter vectors, about 300,000 of these points resulted in CMBFast runs. Note that 1.5 million parameter vectors corresponds to a grid with roughly six elements per side. Since the variance-based metrics sample the entire parameter space, their prediction performance is typically similar to this naive gird. Thus, using an active learning metric that focuses on the boundary that we are interested in (and ignores large parts of the parameter space which can be proved to be infeasible) significantly reduces the computational complexity of the algorithm.

In Figures 3(a)-3(c) we depict 95% confidence regions derived using only a single data set projected into the $\Omega_{\rm M}$ versus Ω_Λ space. Confidence regions are derived by binning the samples selected by the algorithm and including those bins in the confidence region which contain points where

$f \leq t$, resulting in the blockiness in the diagrams. The figures illistrate that the shapes of the 95% confidence regions for each of the data sources are quite different, validating our supposition that different observable functions can be used to efficiently reject parts of parameter space.

In Figure 3(d), depicts the 95% confidence region found using the joint analysis for all three data sets; one and two dimensional projections onto the other parameters can be found in Bryan (2007). It is clear that using the combination of all three data sets dramatically improves the inferences that can be made on the cosmological parameters' values. In particular, note that the derived confidence region is significantly smaller than what would have been obtained using a simple intersection. As a result, we cannot blindly combine the WMAP p-values of Bryan, B., et al. (2005) with p-values derived for the supernova and large scale structure data sets, as the surface of the combined target function is drastically different from the surfaces of each of the models independently. Specifically, all of the models in the Bryan, B., et al. (2005) data set can be rejected at the 95% confidence level by the supernova and large scale structure data. This is not surprising; the analysis of Bryan, B., et al. (2005) used only CMBFast one the WMAP data, and it is well known that CMBFast only loosly fits the WMAP data (Spergel, D. et al., 2003). Thus in order to accurately compute the 95% confidence regions of the joint model (using all three data sets), we must sample new models in the multiple model framework, as we did in Figure 3(d). Only then will we correctly learn the true level-set of the composite target function.

6. Conclusions

We have described the problem of learning a target function based on a set of related observable functions. This problem naturally arises in many situations including the joint analysis of multiple data sets which describe a single physical phenomenon. We have developed an algorithm for locating the level set of this target function while minimizing the number of experiments necessary. We described and showed how several different heuristics for choosing experiments from a set of candidates perform on synthetic target functions. Our experiments indicate that variance-maxvarstraddle outperforms both random and variance-weighted heuristics typically applied to active learning problems. Moreover, variance-maxvarstraddle is better than both the sequential- and variance-straddle heuristics, as it appears to better approximate the information gain of a candidate point.

Using the variance-maxvarstraddle heuristic, we were able to efficiently learn the level set of an eight dimensional surface. This level-set corresponds to the 95% confidence region of a joint analysis between three data sources. Using the CMB, supernovae and large scale structure data sets results in much tighter confidence regions than those obtained using only a single source of data, allowing for stronger scientific inferences. Standard ad hoc techniques for combining evidence, such as intersecting the data, or using weak priors do not result in such a significant reduction in the accepted parameter space.

References

Angluin, D. (1988). Queries and concept learning. *Machine Learning, 2*, 319–342.

Bennett, C. L., et al. (2003). First-Year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Foreground Emission. *Astrophysical Journal Supplemental, 148*, 97–117.

Bryan, B. (2007). *Actively learning specific function properties with application to statistical inference*. Doctoral dissertation, Carnegie Mellon University.

Bryan, B., et al. (2005). Active learning for identifying function threshold boundaries. In *Advances in neural information processing systems 18*. Cambridge, MA: MIT Press.

Cressie, N. (1991). *Statistics for spatial data*. New York: Wiley.

Davis, T. M., et al. (2007). Scrutinizing Exotic Cosmological Models Using ESSENCE Supernova Data Combined with Other Cosmological Probes. *Astrophysical Journal, 666*, 716.

Fisher, R. (1932). *Statistical methods for research workers*. London: Oliver and Boyd. 4 edition.

Guestrin, C., et al. (2005). Near-optimal sensor placements in gaussian processes. *ICML 2005: Proceedings of the 22nd International Conference on Machine learning*. ACM Press.

Hedges, L. V. (1985). *Statistical methods for meta-analysis*. Academic Press.

Kersting, K. et al. (2007). Most likely heteroscedastic gaussian process regression. *ICML '07: Proceedings of the 24th International Conference on Machine Learning*. ACM Press.

MacKay, D. (1992). Information-based objective functions for active data selection. *Neural Computation, 4*, 590.

Ramakrishnan, N., et al. (2005). Gaussian processes for active data mining of spatial aggregates. *Proceedings of the SIAM International Conference on Data Mining*.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Santner, T. J., Williams, B. J., & Notz, W. (2003). *The design and analyis of computer experiments*. Springer. 1 edition.

Shmueli, G., & Fienberg, S. E. (2006). *Statistical methods in counterterrorism*, chapter Current and Potential Statistical Methods for Monitoring Multiple Data Streams for Biosurveillance, 109. New York: Springer.

Spergel, D. et al. (2003). First-Year Wilkinson Microwave Anisotropy Probe Observations: Determination of Cosmological Parameters. *Astrophysical Journal Supplemental, 148*.

Tegmark, M., et al. (2006). Cosmological constraints from the SDSS luminous red galaxies. *Physical Review D, 74*, 123507.

Sparse Bayesian Nonparametric Regression

François Caron CARONFR@CS.UBC.CA
Arnaud Doucet ARNAUD@CS.UBC.CA
Departments of Computer Science and Statistics, University of British Columbia, Vancouver, Canada

Abstract

One of the most common problems in machine learning and statistics consists of estimating the mean response $X\beta$ from a vector of observations y assuming $y = X\beta + \varepsilon$ where X is known, β is a vector of parameters of interest and ε a vector of stochastic errors. We are particularly interested here in the case where the dimension K of β is much higher than the dimension of y. We propose some flexible Bayesian models which can yield sparse estimates of β. We show that as $K \to \infty$ these models are closely related to a class of Lévy processes. Simulations demonstrate that our models outperform significantly a range of popular alternatives.

1. Introduction

Consider the following linear regression model

$$y = X\beta + \varepsilon \tag{1}$$

where $y \in \mathbb{R}^L$ is the observation, $\beta = (\beta_1, \ldots, \beta_K) \in \mathbb{R}^K$ is the vector of unknown parameters, X is an known $L \times K$ matrix. We will assume that ε follows a zero-mean normal distribution $\varepsilon \sim \mathcal{N}\left(0, \sigma^2 I_L\right)$ where I_L is the identity matrix of dimension L.

We do not impose here any restriction on L and K but we are particularly interested in the case where $K >> L$. This scenario is very common in many application domains. In such cases, we are interested in obtaining a sparse estimate of β; that is an estimate $\widehat{\beta} = (\widehat{\beta}_1, \ldots, \widehat{\beta}_K)$ such that only a subset of the components $\widehat{\beta}_k$ differ from zero. This might be for sake of variable selection (Tibshirani, 1996; Figueiredo, 2003; Griffin & Brown, 2007) or to decompose a signal over an overcomplete basis (Lewicki & Sejnowski, 2000; Chen et al., 2001).

Numerous models and algorithms have been proposed in the machine learning and statistics literature to address this problem including Bayesian stochastic search methods based on the 'spike and slab' prior (West, 2003), Lasso (Tibshirani, 1996), projection pursuit or the Relevance Vector Machine (RVM) (Tipping, 2001). We follow here a Bayesian approach where we set a prior distribution on β and we will primarily focus on the case where $\widehat{\beta}$ is the resulting Maximum a Posteriori (MAP) estimate or equivalently the Penalized Maximum Likelihood (PML) estimate. Such MAP/PML approaches have been discussed many times in the literature and include the Lasso (the corresponding prior being the Laplace distribution) (Tibshirani, 1996; Lewicki & Sejnowski, 2000; Girolami, 2001), the normal-Jeffreys (NJ) prior (Figueiredo, 2003) or the normal-exponential gamma prior (Griffin & Brown, 2007). Asymptotic theoretical properties of such PML estimates are discussed in (Fan & Li, 2001).

We propose here a class of prior distributions based on scale mixture of Gaussians for β. For a finite K, our prior models correspond to normal-gamma (NG) and normal-inverse Gaussian (NIG) models. This class of models includes as limiting cases both the popular Laplace and normal-Jeffreys priors but is more flexible. As $K \to \infty$, we show that the proposed priors are closely related to the variance gamma and normal-inverse Gaussian processes which are Lévy processes (Applebaum, 2004). In this respect, our models are somehow complementary to two recently proposed Bayesian nonparametric models: the Indian buffet process (Ghahramani et al., 2006) and the infinite gamma-Poisson process (Titsias, 2007). Under given conditions, the normal-gamma prior yields sparse MAP estimates $\widehat{\beta}$. The log-posterior distributions associated to these prior distributions are not convex but we propose an Expectation-Maximization (EM) algorithm to find modes of the posteriors and

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

a Markov Chain Monte Carlo (MCMC) algorithm to sample from them. We demonstrate through simulations that these Bayesian models outperform significantly a range of established procedures on a variety of applications.

The rest of the paper is organized as follows. In Section 2, we propose the NG and NIG models for β. We establish some properties of these models for K finite and in the asymptotic case where $K \to \infty$. We also relate our model to the Indian buffet process (Ghahramani et al., 2006) and the infinite gamma-Poisson process (Titsias, 2007). In Section 3, we establish conditions under which the MAP/PML estimate $\widehat{\beta}$ can enjoy sparsity properties. Section 4 presents an EM algorithm to find modes of the posterior distributions and a Gibbs sampling algorithm to sample from them. We demonstrate the performance of our models and algorithms in Section 5. Finally we discuss some extensions in Section 6.

2. Sparse Bayesian Nonparametric Models

We will consider models where the components β are independent and identically distributed

$$p(\beta) = \prod_{k=1}^{K} p(\beta_k)$$

and $p(\beta_k)$ is a scale mixture of Gaussians; that is

$$p(\beta_k) = \int \mathcal{N}(\beta_k; 0, \sigma_k^2) p(\sigma_k^2)\, d\sigma_k^2 \qquad (2)$$

where $\mathcal{N}(x; \mu, \sigma^2)$ denotes the Gaussian distribution of argument x, mean μ and variance σ^2. We propose two conjugate distributions for σ_k^2; namely the gamma and the inverse Gaussian distributions. The resulting marginal distribution for β_k belongs in both cases to the class of generalized hyperbolic distributions.

In the models presented here, the unknown scale parameters are random and integrated out so that the marginal priors on the regression coefficients are not Gaussian. This differs from the RVM (Tipping, 2001) where these parameters are unknown and estimated through maximum likelihood.

2.1. Normal-Gamma Model

2.1.1. Definition

Consider the following gamma prior distribution

$$\sigma_k^2 \sim \mathcal{G}(\frac{\alpha}{K}, \frac{\gamma^2}{2})$$

whose probability density function (pdf) $\mathcal{G}(\sigma_k^2; \frac{\alpha}{K}, \frac{\gamma^2}{2})$ is given by

$$\frac{(\frac{\gamma^2}{2})^{\frac{\alpha}{K}}}{\Gamma(\frac{\alpha}{K})} (\sigma_k^2)^{\frac{\alpha}{K}-1} \exp(-\frac{\gamma^2}{2}\sigma_k^2).$$

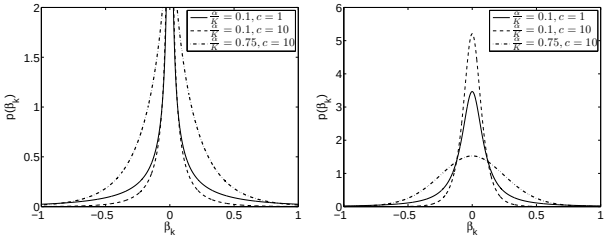
Following Eq. (2), the marginal pdf of β_k is given for $\beta_k \neq 0$ by

$$p(\beta_k) = \frac{\gamma^{\alpha/K+1/2}}{\sqrt{\pi} 2^{\alpha/K-1/2} \Gamma(\frac{\alpha}{K})} |\beta_k|^{\frac{\alpha}{K}-\frac{1}{2}} \mathcal{K}_{\frac{\alpha}{K}-\frac{1}{2}}(\gamma|\beta_k|) \qquad (3)$$

where $\mathcal{K}_\nu(\cdot)$ is the modified Bessel function of the second kind. We have

$$\lim_{\beta_k \to 0} p(\beta_k) = \begin{cases} \frac{\gamma}{2\sqrt{\pi}} \frac{\Gamma(\frac{\alpha}{K}-\frac{1}{2})}{\Gamma(\frac{\alpha}{K})} & \text{if } \frac{\alpha}{K} > \frac{1}{2} \\ \infty & \text{otherwise} \end{cases}$$

and the tails of this distribution decrease in $|\beta_k|^{\frac{\alpha}{K}-1} \exp(-\gamma|\beta_k|)$, see Figure 1(a). The parameters α and γ resp. control the shape and scale of the distribution. When $\alpha \to 0$, there is a high discrepancy between the values of σ_k^2, while when $\alpha \to \infty$, most of the values are equal.



(a) Normal-gamma (b) Normal-inverse Gaussian

Figure 1. Probability density functions of the NG and NIG for different values of the parameters.

This class of priors includes many standard priors. Indeed, Eq. (3) reduces to the Laplace prior when $\frac{\alpha}{K} = 1$ and we obtain the NJ prior when $\frac{\alpha}{K} \to 0$ and $\gamma \to 0$.

In Figure 2 some realizations of the process are given for different values $\alpha = 1, 5, 100$ and $\gamma^2/2 = \alpha$.

2.1.2. Properties

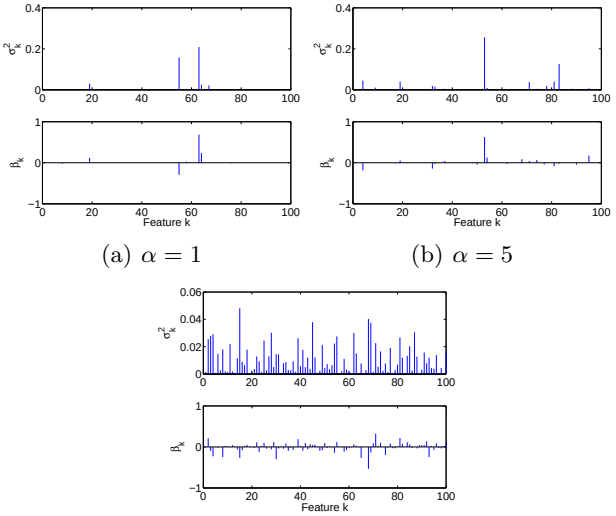
It follows from Eq. (3) that

$$\mathbb{E}[|\beta_k|] = \sqrt{\frac{4}{\pi\gamma^2}} \frac{\Gamma(\frac{\alpha}{K}+\frac{1}{2})}{\Gamma(\frac{\alpha}{K})}, \; \mathbb{E}[\beta_k^2] = \frac{2\alpha}{\gamma^2 K}$$

and we obtain

$$\lim_{K\to\infty} \mathbb{E}[\sum_{k=1}^{K} |\beta_k|] = \frac{2\alpha}{\gamma}, \; \mathbb{E}[\sum_{k=1}^{K} \beta_k^2] = \frac{2\alpha}{\gamma^2}.$$

Hence the sum of the terms remains bounded whatever being K.



(a) $\alpha = 1$ (b) $\alpha = 5$

(c) $\alpha = 100$

Figure 2. Realizations (top) $\left\{\sigma_k^2\right\}_{k=1,\ldots,K}$ and (bottom) $\{\beta_k\}_{k=1,\ldots,K}$ from the NG model for $\alpha = 1, 5, 100$.

Using properties of the gamma distribution, it is possible to relate β to a Lévy process known as the variance gamma process as $K \to \infty$. First consider a finite K. Let $\sigma^2_{(1)} \geq \sigma^2_{(2)} \geq \ldots \geq \sigma^2_{(K)}$ be the order statistics of the sequence $\sigma_1^2, \sigma_2^2, \ldots, \sigma_K^2$ and let $\pi_1, \ldots, \pi_K$ be random variables verifying the following (finite) stick-breaking construction

$$\pi_k = \zeta_k \prod_{j=1}^{k-1}(1-\zeta_j) \text{ with } \zeta_j \sim \mathcal{B}(1+\frac{\alpha}{K}, \alpha - \frac{k\alpha}{K}) \quad (4)$$

where $\mathcal{B}$ is the Beta distribution. Finally if $g \sim \mathcal{G}(\alpha, \frac{\gamma^2}{2})$ then we can check that the order statistics $\left(\sigma^2_{(k)}\right)$ follow the same distribution as the order statistics of $(g\pi_k)$. The characteristic function of β_k is given by

$$\Phi_{\beta_k}(u) = \frac{1}{\left(1 - \frac{iu}{\gamma}\right)^{\frac{\alpha}{K}}} \frac{1}{\left(1 + \frac{iu}{\gamma}\right)^{\frac{\alpha}{K}}}$$

and therefore

$$\beta_k \stackrel{d}{=} w_1 - w_2 \text{ where } w_1 \sim \mathcal{G}(\frac{\alpha}{K},\gamma) \text{ and } w_2 \sim \mathcal{G}(\frac{\alpha}{K},\gamma)$$

It follows that β_k can be written as the difference of two variables following a gamma distribution.

As $K \to \infty$, the order statistics $\left(\sigma^2_{(k)}\right)$ are the conic part of a gamma process with shape parameter α and scale parameter $\gamma^2/2$; see (Tsilevich et al., 2000) for details. In particular $\overline{\sigma}^2 = \left(\frac{\sigma^2_{(1)}}{\sum_k \sigma^2_{(k)}}, \frac{\sigma^2_{(2)}}{\sum_k \sigma^2_{(k)}}, \ldots\right)$ and $\sum_k \sigma^2_{(k)}$ are independent and respectively distributed according to $PD(\alpha)$ and $\mathcal{G}(\alpha, \gamma^2/2)$ where $PD(\alpha)$ is the Poisson-Dirichlet distribution of scale parameter α. It is well-known that this distribution can be recovered by the following (infinite) stick-breaking construction (Tsilevich et al., 2000) as if we set

$$\pi_k = \zeta_k \prod_{j=1}^{k-1}(1-\zeta_j) \text{ with } \zeta_j \sim \mathcal{B}(1, \alpha) \quad (5)$$

for any k then the order statistics $\left(\pi_{(k)}\right)$ are distributed from the Poisson-Dirichlet distribution.

The coefficients (β_k) are thus nothing but the weights (jumps) of the so-called variance gamma process which is a Brownian motion evaluated at times given by a gamma process (Applebaum, 2004; Madan & Seneta, 1990).

2.2. Normal-Inverse Gaussian Model

2.2.1. Definition

Consider the following inverse Gaussian prior distribution

$$\sigma_k^2 \sim \mathcal{IG}(\frac{\alpha}{K}, \gamma) \quad (6)$$

whose pdf $\mathcal{IG}(\sigma_k^2; \frac{\alpha}{K}, \gamma)$ is given by (Barndorff-Nielsen, 1997)

$$\frac{1}{\sqrt{2\pi}} \frac{\alpha}{K} \exp(\gamma \frac{\alpha}{K})(\sigma_k^2)^{-3/2} \exp(-\frac{1}{2}(\frac{\alpha^2}{K^2\sigma_k^2} + \gamma^2\sigma_k^2)) \quad (7)$$

Following Eq. (2), the marginal pdf of β_k is given

$$\frac{\alpha\gamma}{\pi K} \exp(\frac{\alpha\gamma}{K}) \left(\frac{\alpha^2}{K^2} + \beta_k^2\right)^{-\frac{1}{2}} \mathcal{K}_1\left(\gamma\sqrt{\frac{\alpha^2}{K^2} + \beta_k^2}\right) \quad (8)$$

and the tails of this distribution decrease in $|\beta_k|^{-3/2} \exp(-\gamma|\beta_k|)$. It is displayed in Figure 1(b). The parameters α and γ resp. control the shape and scale of the distribution. When $\alpha \to 0$, there is a high discrepancy between the values of σ_k^2, while when $\alpha \to \infty$, most of the values are equal. Some realizations of the model, for different values of α are represented in Figure 3.

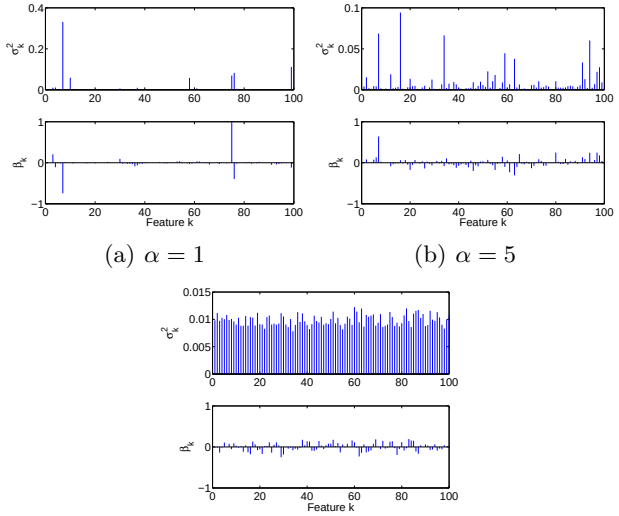
2.2.2. Properties

The moments are given

$$\mathbb{E}[|\beta_k|] = \frac{2\alpha}{K\pi} \exp(\frac{\gamma\alpha}{K}) \mathcal{K}_0(\frac{\alpha\gamma}{K}), \; \mathbb{E}[\beta_k^2] = \frac{\alpha}{K\gamma}$$

Therefore, as $K \to \infty$, the mean of sum of the absolute values is infinite while the sum of the square is $\frac{\alpha}{\gamma}$.

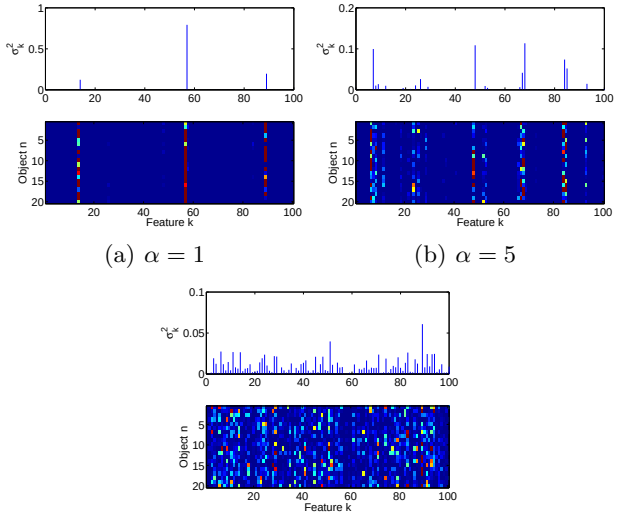
We can also establish in this case that the coefficients (β_k) tend to weights (jumps) of a normal-inverse Gaussian process (Barndorff-Nielsen, 1997).



(a) $\alpha = 1$ (b) $\alpha = 5$

(c) $\alpha = 100$

Figure 3. Realizations (top) $(\sigma_k^2)_{k=1,\dots,K}$ and (bottom) $(\beta_k)_{k=1,\dots,K}$ from the NIG model for $K = 100$, $N = 20$, $\alpha = 1, 10, 100$ and $\gamma = \alpha$.



(a) $\alpha = 1$ (b) $\alpha = 5$

(c) $\alpha = 100$

Figure 4. Realizations (top) $(\sigma_k^2)_{k=1,\dots,K}$ and (bottom) $(\beta_k^n)_{n=1,\dots,N,k=1,\dots,K}$ from the normal-gamma model for $K = 100$, $N = 20$, $\alpha = 1, 10, 100$ and $\gamma^2/2 = \alpha$. The lighter the colour, the larger $|\beta_k^n|$.

2.3. Extension

Consider now the case where we have N vectors of observations $\{y_n\}_{n=1}^N$ where $y_n \in \mathbb{R}^L$. We would like to model the fact that for a given k the random variables $\{\beta_k^n\}_{n=1}^N$ are statistically dependent and exchangeable. We consider the following hierarchical model

$$\sigma_k^2 \sim \mathcal{G}(\frac{\alpha}{K}, \frac{\gamma^2}{2}) \text{ or } \sigma_k^2 \sim \mathcal{IG}(\frac{\alpha}{K}, \gamma)$$

for $k = 1, \dots, K$ and

$$\beta_k^n \sim \mathcal{N}(0, \sigma_k^2)$$

for $n = 1, \dots, N$. Some realizations of the process for different values $\alpha = 1, 5, 100$ are represented in Figure 4.

In this respect, this work is complementary to two recently proposed Bayesian nonparametric models: the Indian buffet process (Ghahramani et al., 2006) and the infinite gamma-Poisson process (Titsias, 2007). In these two contributions, prior distributions over infinite matrices with integer-valued entries are defined. These models are constructed as the limits of finite-dimensional models based respectively on the beta-binomial and gamma-Poisson models. They enjoy the following property: while the number of non-zero entries of an (infinite) row is potentially infinite, the expected number of these entries is finite. These models are also closely related to the beta and gamma processes which are Lévy processes (Applebaum, 2004; Teh et al., 2007; Thibaux & Jordan, 2007). Our models could be interpreted as prior distributions over infinite matrices with real-valued entries. In our case, the number of non-zero entries of an (infinite) row is always infinite but we can have

$$\lim_{K\to\infty} \mathbb{E}\left[\sum_{k=1}^{K} |\beta_k^n|^\rho\right] < \infty \tag{9}$$

for $\rho = 1$ or $\rho = 2$. Morever for some values of $\frac{\alpha}{K}$ and γ we can also ensure that for any $x > 0$

$$\lim_{K\to\infty} \Pr\left(\exists k : |\beta_k^n| > x\right) > 0; \tag{10}$$

that is there is still a non-vanishing probability of having coefficients with large values as $K \to \infty$ despite Eq. (9).

The joint distribution is given by $p(\beta_{1:K}^{1:N}) = \prod_{k=1}^K p(\beta_k^{1:N})$ where for the NG model

$$p(\beta_k^{1:N}) \propto u_k^{\frac{\alpha}{K}-\frac{N}{2}} \mathcal{K}_{\frac{\alpha}{K}-\frac{N}{2}}(\gamma u_k)$$

and for the NIG model

$$p(\beta_k^{1:N}) \propto (q_k)^{-(N+1)/2} \mathcal{K}_{\frac{N+1}{2}}(\gamma q_k)$$

where

$$u_k = \sqrt{\sum\nolimits_{n=1}^{N} (\beta_k^n)^2}, \; q_k = \sqrt{\frac{\alpha^2}{K^2} + u_k^2} \tag{11}$$

3. Sparsity Properties

Further on we will also use the following notation for any random variable u

$$pen(u) \equiv \log(p(u))$$

Table 1. Penalizations and their derivatives for different prior distributions

	$pen(\beta_k^{1:N})$	$pen'(\beta_k^{1:N})$
Lasso ($N=1$)	$\gamma\lvert\beta_k\rvert$	γ
NJ	$N\log(u_k)$	N/u_k
NG	$(\frac{N}{2}-\frac{\alpha}{K})\log u_k$ $-\log \mathcal{K}_{\frac{\alpha}{K}-\frac{N}{2}}(\gamma u_k)$	$\frac{\gamma \mathcal{K}_{\frac{\alpha}{K}-\frac{N}{2}-1}(\gamma u_k)}{\mathcal{K}_{\frac{\alpha}{K}-\frac{N}{2}}(\gamma u_k)}$
NIG	$\frac{N+1}{2}\log(q_k)$ $-\log \mathcal{K}_{\frac{N+1}{2}}(\gamma q_k)$	$\frac{(N+1)u_k}{q_k^2}$ $+\frac{\gamma u_k}{q_k}\frac{\mathcal{K}_{\frac{N-1}{2}}(\gamma q_k)}{\mathcal{K}_{\frac{N+1}{2}}(\gamma q_k)}$

where '$\equiv$' denotes equal up to an additive constant independent of u. When computing the MAP/PML estimate for N data, we select

$$\widehat{\beta}^{1:N} = \arg\min_{\beta^{1:N}} \sum_{n=1}^{N} \frac{\|y_n - X\beta^n\|_2^2}{2\sigma^2} - \sum_{k=1}^{K} pen(\beta_k^{1:N}). \tag{12}$$

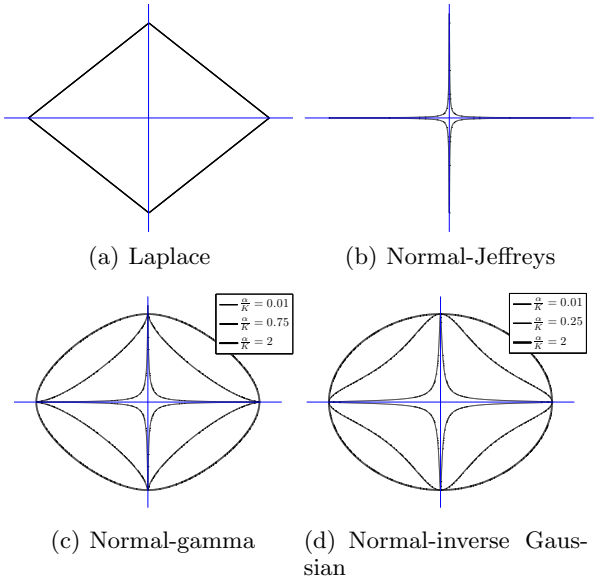
We give in Table 1 the penalizations $pen(\beta_k^{1:N})$ and their derivatives for different prior distributions as a function of u_k and q_k defined in Eq. (11).

When $\alpha/K = 1$, the NG prior is equal to the Laplace prior so its penalization reduces to the ℓ_1 penalization used in Lasso and basis pursuit (Tibshirani, 1996; Chen et al., 2001). When $\alpha/K \to 0$ and $c \to 0$ the prior is the NJ prior and the penalization reduces to $\log(|\beta_k|)$ which has been used in (Figueiredo, 2003). We display in Figure 5 the contours of constant value for various prior distributions when $N = 1$ and $K = 2$. For $\alpha/K < 1/2$, the MAP estimate (12) does not exist as the pdf (3) is unbounded. For other values of the parameters, a mode can dominate at zero whereas we are interested in the data driven turning point/local minimum (Griffin & Brown, 2007).

Consider now the case where the matrix X is orthogonal, $\sigma = 1$ and $N = 1$. The turning point and/or MAP/PML estimate is obtained by minimizing Eq. (12) which is equivalent to minimize componentwise

$$\frac{1}{2}(z_k - \beta_k)^2 + pen(\beta_k) \tag{13}$$

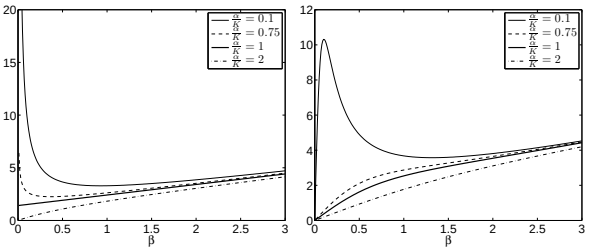
where $z = X^T y$. The first derivative of (13) is $sign(\beta_k)\left(|\beta_k| + pen'(|\beta_k|)\right) - z_k$. As stated in (Fan & Li, 2001, p. 1350), a sufficient condition for the estimate to be a thresholding rule is that the minimum of the function $|\beta_k| + pen'(|\beta_k|)$ is strictly positive. Plots of the function $|\beta_k| + pen'(|\beta_k|)$ are given in Figure 6 and the resulting thresholds corresponding to the argument minimizing (13) are presented in Figure 7. It follows that the normal-gamma prior is a thresholding rule for $\alpha/K \leq 1$ and yields sparse estimates. The normal-inverse Gaussian is not a thresholding rule as the derivative of the penalization is 0 when $\beta_k = 0$ whatever being the values of the parameters. However, from Figure 7(d), it is clear that it can yield "almost sparse" estimates; that is most components are such that $\left|\widehat{\beta}_k\right| \simeq 0$.



(a) Laplace (b) Normal-Jeffreys

(c) Normal-gamma (d) Normal-inverse Gaussian

Figure 5. Contour of constant value of $pen(\beta_1) + pen(\beta_2)$ for different prior distributions.



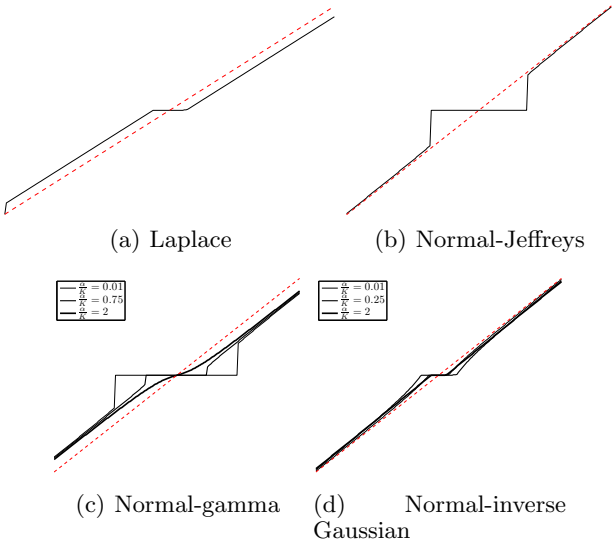
(a) Normal-gamma (b) Normal-inverse Gaussian

Figure 6. Plots of $|\beta_k| + pen'(|\beta_k|)$.

4. Algorithms

4.1. EM

The log-posterior in Eq. (12) is not concave but we can use the EM algorithm to find modes of it. The EM algorithm relies on the introduction of the missing data $\sigma_{1:K} = (\sigma_1, ..., \sigma_K)$. Conditional upon these missing data, the regression model is linear Gaussian and all the EM quantities can be easily computed in closed form; see for example (Figueiredo, 2003; Griffin & Brown, 2007). We have at iteration $i+1$ of the EM



(a) Laplace (b) Normal-Jeffreys

(c) Normal-gamma (d) Normal-inverse Gaussian

Figure 7. Thresholds for the different prior distributions.

$$\widehat{\beta}^{1:N}_{(i+1)} = \arg\max_{\beta_{1:N}} Q(\beta^{1:N}; \widehat{\beta}^{1:N}_{(i)})$$

where $Q(\beta^{1:N}; \widehat{\beta}^{1:N}_{(i)})$ is given by

$$\int \log(p(\beta^{1:N}|y_{1:N}, \sigma_{1:K})).p(\sigma_{1:K}|\widehat{\beta}^{1:N}_{(i)}, y_{1:N})d\sigma_{1:K}.$$

After a few calculations, we obtain

$$\widehat{\beta}^{n}_{(i+1)} = \left(\sigma^2 V_{(i)} + X^T X\right)^{-1} X^T y_n$$

with $V_{(i)} = diag(m_{1,(i)}, \ldots, m_{K,(i)})$ and $m_{k,(i)} = \left(\widehat{u}_{k,(i)}\right)^{-1} pen'(\widehat{u}_{k,(i)})$ where $\widehat{u}_{k,(i)} = \sqrt{\sum_{n=1}^{N}\left(\widehat{\beta}^{n}_{k,(i)}\right)^2}$, $pen'(\widehat{u}_{k,(i)}) = \frac{\partial pen(u_k)}{\partial u_k}\Big|_{\widehat{u}_{k,(i)}}$ (see Table 1).

4.2. MCMC

We can also easily sample from the posterior distribution $p(\beta^{1:N}|y_{1:N})$ by sampling from $p(\beta^{1:N}, \sigma^2_{1:K}|y_{1:N})$ using the Gibbs sampler. Indeed the full conditional distributions $p(\beta^{1:N}|\sigma_{1:K}, y_{1:N})$ and $p(\sigma^2_{1:K}|\beta^{1:N}, y_{1:N})$ are available in closed-form. The distribution $p(\beta^{1:N}|\sigma_{1:K}, y_{1:N})$ is a multivariate normal whereas we have $p(\sigma^2_{1:K}|\beta^{1:N}, y_{1:N}) = \prod_{k=1}^{K} p(\sigma^2_k|\beta^{1:N}_k)$. For the NG prior, we obtain

$$p(\sigma^2_k|\beta^{1:N}_k) = \frac{(\sigma^2_k)^{\frac{\alpha}{K}-\frac{N}{2}-1}\exp\left(-\frac{1}{2}\frac{u^2_k}{\sigma^2_k} - \gamma\sigma^2_k\right)}{2\left(\frac{u_k}{\gamma}\right)^{\frac{\alpha}{K}-\frac{N}{2}}\mathcal{K}_{\frac{\alpha}{K}-\frac{N}{2}}(\gamma u_k)}$$

which is a generalized inverse Gaussian distribution from which we can sample exactly. For the NIG distribution, we also obtain a generalized inverse Gaussian distribution.

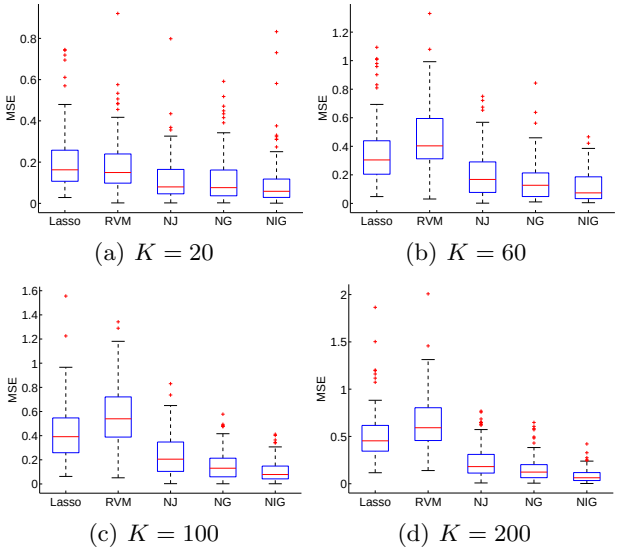
5. Applications

5.1. Simulated Data

In the following, we provide numerical comparisons between the Laplace (that is Lasso), the RVM, NJ, NG and NIG models. We simulate 100 datasets from (1) with $L = 50$ and $\sigma = 1$. The correlation between $X_{k,i}$ and $X_{k,j}$ is $\rho^{|i-j|}$ with $\rho = 0.5$. We set $\beta = (3\ 1.5\ 0\ 0\ 2\ 0\ 0\ldots)^T \in \mathbb{R}^K$ where the remaining components of the vector are set to zero. We consider the cases where $K = 20, 60, 100, 200$. Parameters of the Lasso, NG and NIG are estimated by 5-fold cross-validation, as described in (Tibshirani, 1996). The Lasso estimate is obtained with the Matlab implementation of the interior point method downloadable at http://www.stanford.edu/~boyd/l1_ls/. For the other priors, the estimate is obtained via 100 iterations of the EM algorithm. Box plots of the mean square error (MSE) are reported in Figure 8. These plots show that the performance of the estimators based on the NG and NIG priors outperform those of classical models in that case. In Figure 9 are represented the box plots of the number of estimated coefficients whose absolute value is below T, $T = 10^{-10}$ (the precision tuned for the Lasso estimate) and $T = 10^{-3}$, for $K = 200$. The true number of zeros in that case is 197. The NG outperforms the other models in identifying the zeros of the model. On the contrary, as the NIG estimate is not a thresholding rule, the median number of coefficients whose absolute value is below 10^{-10} for this model is zero. However, most of the coefficients have a very low absolute value, as the median of the coefficients with absolute value below 10^{-3} is equal to the true value 197 (see Figure 9(b)). Moreover, the estimator obtained by thresholding the coefficients whose absolute value is below 10^{-3} to zero yields very minor differences in terms of MSE.

5.2. Biscuit NIR Dataset

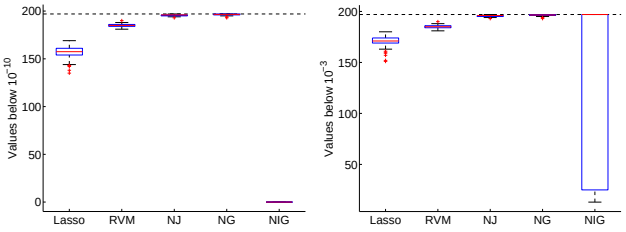
We consider the biscuits data which have been studied in (Griffin & Brown, 2007; West, 2003). The matrix X is composed of 300 (centered) NIR reflectance measurements from 70 biscuit dough pieces. The observations y are the percentage of fat, sucrose, flour and water associated to each piece. The objective here is to predict the level of each of the ingredients from the NIR reflectance measurements. The data are divided into a training dataset (39 measurements) and a test dataset (31 measurements). The fitted coefficients of fat and flour, using 5-fold cross-validation, are represented in Figure 10. The estimated spikes are consistent with the results obtained in (West, 2003; Griffin & Brown, 2007). In particular, both models detect



(a) $K = 20$ (b) $K = 60$

(c) $K = 100$ (d) $K = 200$

Figure 8. Box plots of the MSE associated to the simulated data.



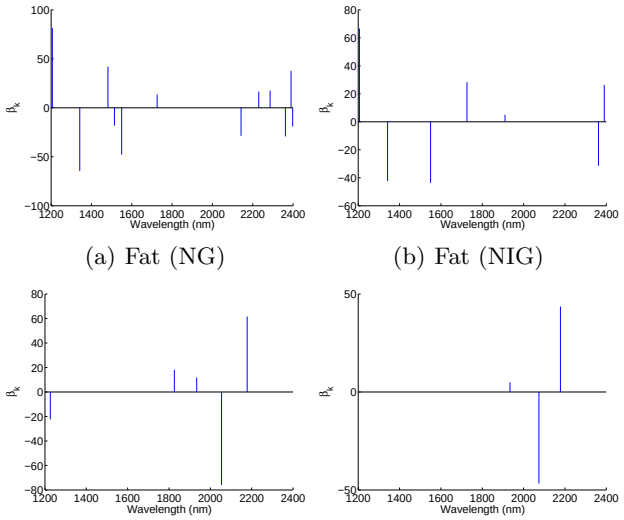
(a) $T = 10^{-10}$ (b) $T = 10^{-3}$

Figure 9. Box plots of the number of estimated coefficients whose absolute value is below a threshold T. Dash line represents the true value of zero coefficients (197).

a spike at 1726nm, which lies in a region known for fat absorbance. The predicted observations versus the true observations are given in Figure 11 for the training and test datasets. The test data are well fitted by the estimated coefficients. MSE errors for the test dataset are reported in Table 2. The proposed models show better performances for flour and similar performances for fat.

6. Discussion

We have presented some flexible priors for linear regression based on the NG and NIG models. The NG prior yields sparse local maxima of the posterior distribution whereas the NIG prior yields "almost sparse" estimates; that is most of the coefficients are extremely close to zero. We have shown that asymptotically these models are closely related to the variance gamma process and the normal-inverse Gaussian process. Contrary to the NJ model or the RVM,



(a) Fat (NG) (b) Fat (NIG)

(c) Flour (NG) (d) Flour (NIG)

Figure 10. Coefficients estimated with a normal-gamma (left) and normal-inverse Gaussian (right) prior for fat (top) and flour (bottom) ingredients.

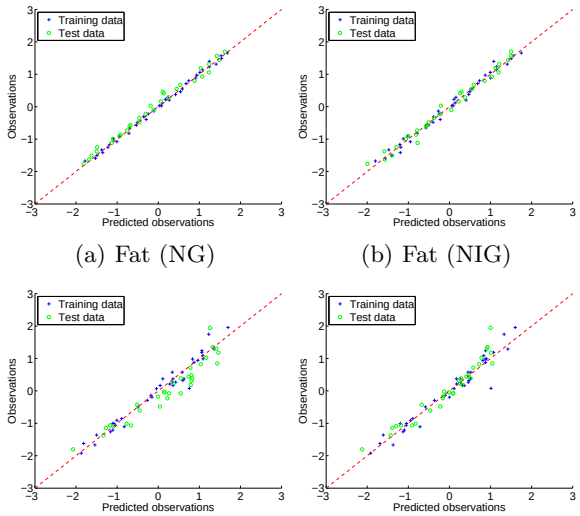
Table 2. MSE for biscuits NIR data

	Flour	Fat
NJ	9.93	0.56
RVM	6.48	0.56
NG	3.44	0.55
NIG	1.94	0.49

these models require specifying two hyperparameters. However, using a simple cross-validation procedure we have demonstrated that these models can perform significantly better that well-established procedures. In particular, the experimental performance of the NIG model are surprisingly good and deserve being further studied. The NG prior has been discussed in (Griffin & Brown, 2007). It was discarded because of its spike at zero and the flatness of the penalty for large values but no simulations were provided. They favour another model which relies on a cylinder parabolic function[1]. The NG prior has nonetheless interesting asymptotic properties in terms of Lévy processes and we have demonstrated its empirical performances. The NG, NIG and Laplace priors can also be considered as particular cases of generalized hyperbolic distributions. This class of distributions has been used in (Snoussi & Idier, 2006) for blind source separation.

The extension to (probit) classification is straightfor-

[1]The authors provide a link to a program to compute this function. Unfortunately, it is extremely slow. The resulting algorithm is at least one order of magnitude slower than our algorithms which rely on Bessel functions.



(a) Fat (NG) (b) Fat (NIG)

(c) Flour (NG) (d) Flour (NIG)

Figure 11. Observations versus predicted observations estimated with a normal-gamma (left) and normal-inverse Gaussian (right) prior for fat (top) and flour (bottom) ingredients.

ward by adding latent variables corresponding to the regression function plus some normal noise. Computationally it only requires adding one line in the EM algorithm and one simulation step in the Gibbs sampler.

References

Applebaum, D. (2004). *Lévy processes and stochastic calculus*. Cambridge University Press.

Barndorff-Nielsen, O. (1997). Normal inverse Gaussian distributions and stochastic volatility modelling. *Scandinavian Journal of Statistics*, *24*, 1–13.

Chen, S., Donoho, D., & Saunders, M. (2001). Atomic decomposition by basis pursuit. *SIAM review*, *43*, 129–159.

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, *96*, 1348–1360.

Figueiredo, M. (2003). Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 1150–1159.

Ghahramani, Z., Griffiths, T., & Sollich, P. (2006). Bayesian nonparametric latent feature models. *Proceedings Valencia/ISBA World meeting on Bayesian statistics*.

Girolami, M. (2001). A variational method for learning sparse and overcomplete representations. *Neural computation*, *13*, 2517–2532.

Griffin, J., & Brown, P. (2007). *Bayesian adaptive lasso with non-convex penalization* (Technical Report). Dept of Statistics, University of Warwick.

Lewicki, M. S., & Sejnowski, T. (2000). Learning overcomplete representations. *Neural computation*, *12*, 337–365.

Madan, D., & Seneta, E. (1990). The variance-gamma model for share market returns. *Journal of Business*, *63*, 511–524.

Snoussi, H., & Idier, J. (2006). Bayesian blind separation of generalized hyperbolic processes in noisy and underdeterminate mixtures. *IEEE Transactions on Signal Processing*, *54*, 3257–3269.

Teh, Y., Gorur, D., & Ghahramani, Z. (2007). Stick-breaking construction for the Indian buffet process. *International Conference on Artificial Intelligence and Statistics*.

Thibaux, R., & Jordan, M. (2007). Hierarchical beta processes and the Indian buffet process. *International Conference on Artificial Intelligence and Statistics*.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society B*, *58*, 267–288.

Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *211-244*, 211–244.

Titsias, M. (2007). The infinite gamma-Poisson feature model. *International Conference on Neural Information Processing Systems*.

Tsilevich, N., Vershik, A., & Yor, M. (2000). *Distinguished properties of the gamma process, and related topics* (Technical Report). Laboratoire de Probabilités et Modèles aléatoires, Paris.

West, M. (2003). Bayesian factor regression models in the "Large p, Small n" paradigm. In J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith and M. West (Eds.), *Bayesian statistics 7*, 723–732. Oxford University Press.

An Empirical Evaluation of Supervised Learning in High Dimensions

Rich Caruana CARUANA@CS.CORNELL.EDU
Nikos Karampatziakis NK@CS.CORNELL.EDU
Ainur Yessenalina AINUR@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

In this paper we perform an empirical evaluation of supervised learning on high-dimensional data. We evaluate performance on three metrics: accuracy, AUC, and squared loss and study the effect of increasing dimensionality on the performance of the learning algorithms. Our findings are consistent with previous studies for problems of relatively low dimension, but suggest that as dimensionality increases the relative performance of the learning algorithms changes. To our surprise, the method that performs consistently well across all dimensions is random forests, followed by neural nets, boosted trees, and SVMs.

1. Introduction

In the last decade, the dimensionality of many machine learning problems has increased substantially. Much of this results from increased interest in learning from text and images. Some of the increase in dimensionality, however, results from the development of techniques such as SVMs and L_1 regularization that are practical and effective in high dimensions. These advances may make it unnecessary to restrict the feature set and thus promote building and learning from data sets that include as many features as possible. At the same time, memory and computational power have increased to support computing with large data sets.

Perhaps the best known empirical studies to examine the performance of different learning methods are STATLOG (King et al., 1995) and (Caruana & Niculescu-Mizil, 2006). STATLOG was a very thorough study, but did not include test problems with high dimensions and could not evaluate newer learning methods such as bagging, boosting, and kernel methods. More recently, (Caruana & Niculescu-Mizil, 2006) includes a number of new learning algorithms that emerged after the STATLOG project, but only examined performance on problems with low-to-medium dimension. One must question if the results of either of these studies apply to text data, biomedical data, link analysis data etc. where many attributes are highly correlated and there may be insufficient data to learn complex interactions among attributes. This paper attempts to address that question.

There are several limitations to the empirical study in (Caruana & Niculescu-Mizil, 2006). First, they performed all experiments using only 5000 training cases, despite the fact that much more labeled data was available for many problems. For one of the problems (COVTYPE) more than 500,000 labeled cases were available. Intentionally training using far less data than is naturally available on each problem makes the results somewhat contrived. Second, although they evaluated learning performance on eight performance metrics, examination of their results shows that there are strong correlations among the performance measures and examining this many metrics probably added little to the empirical comparison and may have led to a false impression of statistical confidence. Third, and perhaps most important, all of the data sets examined had low to medium dimensionality. The average dimensionality of the 11 data sets in their study was about 50 and the maximum dimensionality was only 200. Many modern learning problems have orders of magnitude higher dimensionality. Clearly learning methods can behave very differently when learning from high-dimensional data than when learning from low-dimensional data.

In the empirical study performed for this paper we complement the prior work by: 1) using the natural size training data that is available for each problem; 2) using just three important performance metrics: ac-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

curacy, area under the ROC curve (AUC), and squared loss; and 3) performing experiments on data with high to very high dimensionality (750-700K dimensions).

2. Methodology

2.1. Learning Algorithms

This section summarizes the algorithms and parameter settings we used. The reader should bear in mind that some learning algorithms are more efficient at handling large training sets and high-dimensional data than others. For an efficient algorithm we can afford to explore the parameter space more exhaustively than for an algorithm that does not scale well. But that's not unrealistic; a practitioner may prefer an efficient algorithm that is regarded as weak but which can be tuned well over an algorithm that might be better but where careful tuning would be intractable. Below we describe the implementations and parameter settings we used. An algorithm marked with an asterisk (e.g. **ALG***) denotes our own custom implementation designed to handle high-dimensional sparse data.

SVMs: We train linear SVMs using SVM^{perf} (Joachims, 2006) with error rate as the loss function. We vary the value of C by factors of ten from 10^{-9} to 10^5. For kernel SVMs we used LaSVM, an approximate SVM solver that uses stochastic gradient descent (Bordes et al., 2005), since traditional kernel SVM implementations simply cannot handle the amounts of data in some of our experiments. To guarantee a reasonable running time we train the SVM for 30 minutes for each parameter setting and use the gradient based strategy for the selection of examples. We use polynomial kernels of degree 2 and 3 and RBF kernels with width $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2\}$. We vary the value of C by factors of ten from 10^{-7} to 10^5.

ANN*: We train neural nets with gradient descent backpropagation, early stopping and no momentum (cf. section 5). We vary the number of hidden units $\{8, 16, 32\}$ and learning rate $\{10^{-4}, 10^{-3}, 10^{-2}\}$.

Logistic Regression (LR): We use the BBR package (Genkin et al., 2006) to train models with either L_1 or L_2 regularization. The regularization parameter is varied by factors of ten from 10^{-7} to 10^5.

Naive Bayes (NB*): Continuous attributes are modeled as coming from a normal distribution. We use smoothing and vary the number of unobserved values {0,0.1,0.2,0.5,1,2,5,10,20,50,100}.

KNN*: We use distance weighted KNN. We use the 1000 nearest neighbors weighted by a Gaussian kernel with 40 different kernel widths in the range $[0.1, 820]$. The distance between two points is a weighted euclidean distance where the weight of each feature is determined by its information gain.

Random Forests (RF*): We grow 500 trees and the size of the feature set considered at each split is $s/2, s, 2s, 4s$ or $8s$ where s is the square root of the number of attributes, as suggested in (Breiman, 2001).

Bagged Decision Trees(BAGDT*): We bag 100 ID3 trees. The same implementation is used for **boosted stumps (BSTST*)** and **boosted trees (BSTDT*)** but because full-size ID3 trees can't easily be boosted, we stop splitting when a node contains less than 50 examples. We do 2^{10} and 2^{15} iterations of boosting for trees and stumps respectively and use the validation set to pick the number of iterations from the set $\{2^i | i = 0, 1, ...15\}$.

Perceptrons (PRC*): We train voted perceptrons (Freund & Schapire, 1999) with 1,5,10,20 and 30 passes through the data. We also average 100 perceptrons each of which is obtained by a single pass through a permutation of the data.

With SVM, ANN, LR, KNN and PRC and datasets with continuous attributes we train both on raw and standardized data. This preprocessing can be thought of as one more parameter for these algorithms. To preserve sparsity, which is crucial for the implementations we use, we treat the mean of each feature as zero, compute the standard deviation, and divide by it.

2.2. Performance Metrics

To evaluate performance we use three metrics: accuracy (ACC), a threshold metric, squared error (RMS), a probability metric, and area under the ROC curve (AUC), an ordering metric. To standardize scores across different problems and metrics, we divide performances by the median performance observed on each problem for that metric. For squared error we also invert the scale so that larger numbers indicate better performance as with accuracy and AUC.

2.3. Calibration

The output of some learning algorithms such as ANN, logistic regression, bagged trees and random forests can be interpreted as the conditional probability of the class given the input. The common implementation of other methods such as SVM and boosting, however, are not designed to predict probabilities (Niculescu-Mizil & Caruana, 2005).

To overcome this, we use two different methods to map the predictions from each learning algorithm to

calibrated probabilities. The first is Isotonic Regression (Zadrozny & Elkan, 2002), a method which fits a non-parametric non-decreasing function to the predictions. The second calibration method is Platt's method (Platt, 1999) which fits a sigmoid to the predictions. (Niculescu-Mizil & Caruana, 2005) suggests that Platt's method outperforms isotonic regression when there is less than about 1000 points available to learn the calibration function, and that calibration can hurt predictions from methods such as ANN and logistic regression (Caruana & Niculescu-Mizil, 2006). We will revisit those findings later in the discussion. Finally, note that calibrating can affect metrics other than probability metrics such as squared loss. It can affect accuracy by changing the optimal threshold (for calibrated predictions the optimum threshold will be near 0.5) and Isotonic Regression can affect AUC by creating ties on calibration plateaus where prior to calibration there was a definite ordering.

To summarize our methodology, we optimize for each dataset and metric individually. For each algorithm and parameter setting we calibrate the predictions using isotonic regression, Platt's method, and the identity function (no calibration) and choose the parameter settings and calibration method that optimizes the performance metric on a validation set.

2.4. Data Sets

We compare the methods on 11 binary classification problems whose dimensionality ranges from 761 to 685569. The datasets are summarized in Table 1.

TIS[1] is from the Kent Ridge Bio-medical Data Repository. The problem is to find Translation Initiation Sites (TIS) at which translation from mRNA to proteins initiates. CRYST[2] is a protein crystallography diffraction pattern analysis dataset from the X6A beamline at Brookhaven National Laboratory. STURN and CALAM are ornithology datasets.[3] The task is to predict the appearance of two bird species: sturnella neglecta and calamospiza melanocorys. KDD98 is from the 1998 KDD-Cup. The task is to predict if a person donates money. This is the only dataset with missing values. We impute the mean for continuous features and treat missing nominal and boolean features as new values. DIGITS[4] is the MNIST database of handwritten digits by Cortes and LeCun. It was converted from a 10 class problem to a hard binary problem by treating digits less than 5 as one class and the rest as the other class. IMDB and CITE are link prediction datasets.[5] For IMDB each attribute represents an actor, director, etc. For CITE attributes are the authors of a paper in the CiteSeer digital library. For IMDB the task is to predict if Mel Blanc was involved in the film or television program and for CITE the task is to predict if J. Lee was a coauthor of the paper. We created SPAM from the TREC 2005 Spam Public Corpora. Features take binary values showing if a word appears in the document or not. Words that appear less than three times in the whole corpus were removed. Real-Sim (R-S) is a compilation of Usenet articles from four discussion groups: simulated auto racing, simulated aviation, real autos and real aviation.[6] The task is to distinguish real from simulated. DSE[7] is newswire text with annotated opinion expressions. The task is to find Subjective Expressions i.e. if a particular word expresses an opinion.

Table 1. Description of problems

Problem	Attr	Train	Valid	Test	%Pos
Sturn	761	10K	2K	9K	33.65
Calam	761	10K	2K	9K	34.32
Digits	780	48K	12K	10K	49.01
Tis	927	5.2K	1.3K	6.9K	25.13
Cryst	1344	2.2K	1.1K	2.2K	45.61
KDD98	3848	76.3K	19K	96.3K	5.02
R-S	20958	35K	7K	30.3K	30.82
Cite	105354	81.5K	18.4K	81.5K	0.17
Dse	195203	120K	43.2K	107K	5.46
Spam	405333	36K	9K	42.7K	44.84
Imdb	685569	84K	18.4K	84K	0.44

To split data into training, validation and test sets, if the data came with original splits for train and test sets (i.e. DIGITS, KDD98) we preserved those splits and created validation sets as 10% of the train set. If the data originally was split into folds, we merged some folds to create a training set, a validation set and a test set. (We did this because running these experiments is so costly that we could not afford to perform N-fold cross validation as this would make the experiments about N times more costly.) DSE came in 10 folds plus a development fold twice as big as other folds. We used the development fold as the validation set and merged the first 5 folds for the train set and the rest for the test set. CRYST came in 5 folds. One fold became the validation set, 2 folds were merged for training and the rest became the test set.

For the rest of the datasets we tried to balance be-

[1] http://research.i2r.a-star.edu.sg/GEDatasets/Datasets.html
[2] http://ajbcentral.com/CrySis/dataset.html
[3] Art Munson, Personal Communication
[4] http://yann.lecun.com/exdb/mnist/
[5] http://komarix.org/ac/ds
[6] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets
[7] Eric Breck, Personal Communication

tween the following factors: (a) The test sets should be large enough so that differences between learning algorithms are apparent. (b) The test sets can be larger when the learning task is easy, but more data should be kept in the training set when the learning task is hard. (c) Some datasets would inevitably have more attributes than examples in the training set (IMDB, CITE, SPAM); for the rest we tried to put enough examples in the training set so that methods with small bias might learn something interesting. (d) The validation sets should be big enough so that parameter selection and calibration works well. In general we split the data so that we have around 50% in the training set and 50% in the test set. Validation data is drawn from the training set.

3. Results

Table 2 shows the performance of each learning method on each of the eleven problems. In the table, the problems are arranged left to right in order of increasing dimensionality. The table is broken into four sections. The top three sections report results for Accuracy (ACC), Squared Error (RMS), and Area under the ROC Curve (AUC). The bottom section is the average of the performance across these three metrics.

For each metric and problem, the performances have been standardized by dividing by the median performance observed for that problem and metric. Without standardization it is difficult to perform an unbiased comparison across different datasets and metrics. A score of one indicates that the method had typical performance on that problem and metric compared to the other learning methods. Scores above one indicate better than typical performance, while scores less than one indicate worse than typical performance. The scale for RMS has been reversed so that scores above one represent better than typical, i.e., lower, RMS.[8] The median performance for each problem and metric is included in the table to allow calculating raw performances from the standardized scores. The last column in the table is the average score across the eleven test problems. In each section of the table, learning algorithms are sorted by these average scores. The last column of the last section represents the average score across all problems and metrics.

Examining the results in the bottom section shows that on average across all problems and metrics, random forests have the highest overall performance. On average, they perform about 1% (1.0102) better than the typical model and about 0.6% (1.0102 vs. 1.0039) better than the next best method, ANN. The best methods overall are RF, ANN, boosted decision trees, and SVMs. The worst performing methods are Naive Bayes and perceptrons. On average, the top eight of ten methods fall within about 2% of each other. While it is not easy to achieve an additional 1% of performance at the top end of the scale, it is interesting that so many methods perform this similarly to each other on these high-dimensional problems.

If we examine the results for each of the metrics individually, we notice that the largest differences in performance among the different learning algorithms occur for AUC and the smallest differences occur for ACC. For accuracy, boosted decision trees are the best performing models followed by random forests. However, a closer examination of the table shows that boosted trees do better in accuracy mostly because of their excellent performance on the datasets with relatively low dimensionality. Comparing boosted trees with random forests in the left part of the table we see that random forests outperform boosted trees only on the TIS dataset. The situation is reversed on the right part of the table where boosted trees outperform random forests only on the CITE dataset. As dimensionality increases, we expect boosted trees to fall behind random forests.

In RMS, random forests are marginally better than boosted trees. This is confirmed by a bootstrap analysis (cf. Section 4): random forests have 33% and 35% chance of ranking 1st and 2nd respectively, while for boosted trees the corresponding probabilities are 31% and 21%. However, in AUC random forests are clear winners followed by, somewhat surprisingly, KNN.

Interestingly, although ANN is the 2nd best method overall in the bottom of the table, is does not perform 1st or 2nd for any of the individual metrics in the top of the table. It is 2nd overall only because ANNs consistently yield very good, though perhaps not exceptional, performance on all metrics.

A fact that is not apparent from the table is that calibration with isotonic regression works better than calibrating with Platt's method, or no calibration, on most problems and thus was used for almost all of the results reported in the table. Since our validation sets always are larger than 1000 examples, this confirms the findings in (Niculescu-Mizil & Caruana, 2005) that isotonic regression is preferred with large validation sets.

[8] This is different and simpler than the normalized scores used in (Caruana & Niculescu-Mizil, 2006). We have experimented with several ways of standardizing scores and the results change little with different methods. The learning methods that rank at the top (and the bottom) are least affected by the exact standardization method.

Table 2. Standardized scores of each learning algorithm

DIM	761	761	780	927	1344	3448	20958	105354	195203	405333	685569	—
ACC	STURN	CALAM	DIGITS	TIS	CRYST	KDD98	R-S	CITE	DSE	SPAM	IMDB	MEAN
MEDIAN	0.6901	0.7337	0.9681	0.9135	0.8820	0.9494	0.9599	0.9984	0.9585	0.9757	0.9980	—
BSTDT	0.9962	**1.0353**	1.0120	0.9993	**1.0178**	0.9998	0.9904	1.0000	0.9987	0.9992	1.0000	1.0044
RF	0.9943	1.0103	1.0076	1.0025	1.0162	**1.0000**	0.9995	0.9998	1.0013	**1.0044**	1.0000	1.0033
SVM	1.0044	1.0018	1.0024	1.0060	1.0028	0.9999	**1.0156**	**1.0008**	1.0004	1.0008	**1.0003**	1.0032
BAGDT	1.0001	1.0350	0.9976	1.0017	1.0111	**1.0000**	0.9827	1.0000	0.9996	0.9959	1.0000	1.0021
ANN	0.9999	0.9899	1.0051	1.0007	0.9869	**1.0000**	1.0109	1.0001	**1.0018**	1.0029	**1.0003**	0.9999
LR	1.0012	0.9896	0.8982	**1.0108**	1.0080	**1.0000**	1.0141	1.0001	1.0014	1.0026	0.9999	0.9932
BSTST	1.0077	1.0298	0.9017	0.9815	0.9930	**1.0000**	0.9925	0.9999	0.9948	0.9905	0.9989	0.9900
KNN	**1.0139**	0.9982	**1.0122**	0.9557	0.9972	0.9999	0.9224	1.0000	0.9987	0.9698	0.9996	0.9880
PRC	0.9972	0.9864	0.9010	0.9735	0.9930	**1.0000**	1.0119	0.9999	1.0007	1.0041	1.0001	0.9880
NB	0.9695	0.9347	0.8159	0.9230	0.9724	**1.0000**	1.0005	1.0000	0.9878	0.9509	0.9976	0.9593
RMS	STURN	CALAM	DIGITS	TIS	CRYST	KDD98	R-S	CITE	DSE	SPAM	IMDB	MEAN
MEDIAN	0.5472	0.5800	0.8449	0.7455	0.7051	0.7813	0.8257	0.9623	0.8154	0.8645	0.9597	—
RF	0.9980	1.0209	1.0186	1.0102	1.0277	1.0003	1.0011	0.9988	1.0072	**1.0118**	1.0006	1.0087
BSTDT	0.9993	1.0351	1.0363	0.9977	**1.0323**	0.9998	0.9781	1.0003	0.9983	1.0007	1.0003	1.0071
ANN	1.0042	0.9987	1.0088	1.0109	1.0014	1.0005	1.0315	1.0011	1.0068	1.0077	**1.0022**	1.0067
SVM	0.9979	0.9882	1.0076	**1.0149**	0.9972	0.9992	**1.0409**	**1.0091**	1.0067	0.9993	1.0004	1.0056
BAGDT	1.0007	**1.0357**	0.9924	1.0023	1.0218	0.9998	0.9587	1.0000	0.9994	0.9782	1.0012	0.9991
LR	1.0010	0.9963	0.8169	1.0232	0.9935	**1.0007**	1.0367	1.0009	**1.0082**	1.0073	0.9988	0.9894
PRC	0.9976	0.9841	0.8115	0.9537	0.9919	0.9998	1.0313	0.9979	1.0006	1.0071	0.9997	0.9796
BSTST	1.0078	1.0205	0.8202	0.9757	1.0021	**1.0007**	0.9861	1.0000	0.9900	0.9695	0.9952	0.9789
KNN	**1.0119**	1.0013	**1.0365**	0.9309	0.9986	1.0000	0.8468	0.9988	0.9983	0.9270	0.9941	0.9768
NB	0.9793	0.9509	0.7236	0.9031	0.9454	1.0000	0.9989	0.9981	0.9828	0.8984	0.9731	0.9412
AUC	STURN	CALAM	DIGITS	TIS	CRYST	KDD98	R-S	CITE	DSE	SPAM	IMDB	MEAN
MEDIAN	0.6700	0.7793	0.9945	0.9569	0.9490	0.5905	0.9913	0.7549	0.9008	0.9957	0.9654	—
RF	0.9892	1.0297	1.0017	1.0069	1.0134	1.0140	1.0009	1.0962	**1.0304**	**1.0022**	**1.0209**	1.0187
KNN	**1.0397**	0.9992	1.0024	0.9509	1.0007	1.0165	0.9905	**1.1581**	1.0027	0.9902	0.9648	1.0105
LR	1.0045	0.9903	0.9424	**1.0136**	1.0070	1.0492	**1.0041**	1.0272	1.0293	0.9999	1.0084	1.0069
ANN	1.0132	1.0008	1.0001	1.0042	0.9992	1.0461	1.0031	0.9779	1.0105	1.0001	1.0021	1.0052
BSTST	1.0199	1.0304	0.9468	0.9901	0.9993	**1.0512**	0.9991	0.9956	0.9973	0.9989	1.0036	1.0029
SVM	0.9870	0.9645	1.0002	1.0077	0.9909	0.9324	1.0032	1.1120	1.0100	1.0011	0.9979	1.0006
BSTDT	0.9991	1.0492	**1.0033**	0.9958	**1.0137**	0.9605	0.9962	0.9646	0.9881	1.0015	1.0041	0.9978
BAGDT	1.0009	**1.0551**	0.9999	1.0062	1.0116	0.9768	0.9890	0.9673	0.9691	0.9925	0.9809	0.9954
PRC	0.9973	0.9630	0.9372	0.9749	0.9937	0.9724	1.0036	0.9991	0.9777	1.0006	0.9477	0.9788
NB	0.9329	0.8936	0.8574	0.9407	0.9574	0.9860	0.9990	1.0009	0.9917	0.9798	0.8787	0.9471
AVG	STURN	CALAM	DIGITS	TIS	CRYST	KDD98	R-S	CITE	DSE	SPAM	IMDB	MEAN
RF	0.9938	1.0203	1.0093	1.0065	1.0191	1.0048	1.0005	1.0316	**1.0130**	**1.0061**	**1.0072**	1.0102
ANN	1.0058	0.9965	1.0047	1.0053	0.9958	1.0156	1.0152	0.9930	1.0064	1.0036	1.0015	1.0039
BSTDT	0.9982	1.0399	**1.0172**	0.9976	**1.0212**	0.9867	0.9882	0.9883	0.9950	1.0004	1.0014	1.0031
SVM	0.9965	0.9848	1.0034	1.0095	0.9970	0.9772	**1.0199**	1.0406	1.0057	1.0004	0.9995	1.0031
BAGDT	1.0006	**1.0419**	0.9966	1.0034	1.0148	0.9922	0.9768	0.9891	0.9894	0.9889	0.9940	0.9989
LR	1.0022	0.9921	0.8858	**1.0159**	1.0028	1.0166	1.0183	1.0094	1.0129	1.0033	1.0024	0.9965
KNN	**1.0219**	0.9996	1.0170	0.9458	0.9988	1.0055	0.9199	**1.0523**	0.9999	0.9623	0.9862	0.9917
BSTST	1.0118	1.0269	0.8896	0.9824	0.9982	**1.0173**	0.9926	0.9985	0.9941	0.9863	0.9992	0.9906
PRC	0.9974	0.9778	0.8832	0.9674	0.9929	0.9907	1.0156	0.9990	0.9930	1.0039	0.9825	0.9821
NB	0.9606	0.9264	0.7989	0.9223	0.9584	0.9953	0.9995	0.9997	0.9874	0.9430	0.9498	0.9492

3.1. Effect of Dimensionality

In this section we attempt to show the trends in performance as a function of dimensionality. In Figure 1 the x-axis shows dimensionality on a log scale. The y-axis is the *cumulative* score of each learning method on problems of increasing dimensionality. The score is the average across the three standardized performance metrics where standardization is done by subtracting the median performance on each problem.[9] Subtracting median performance means that scores above (below) zero indicate better (worse) than typical performance. The score accumulation is done left-to-right on problems of increasing dimensionality. A line that tends to slope upwards (downwards) signifies a method that performs better (worse) on average compared to other methods as dimensionality increases. A horizontal line suggests typical performance across problems of different dimensionality. Naive Bayes is excluded from the graph because it falls far below the other methods. Caution must be used when interpreting cumulative score plots. Due to the order in which scores are aggregated, vertical displacement through much of the graph is significantly affected by the performance on problems of lower dimensionality. The end of the graph on the right, however, accumulates across all problems and thus does not favor problems of any dimensionality, The slope roughly corresponds to the average relative performance across dimensions. From the plot it is clear that boosted trees do very

[9]Here we subtract the median instead of dividing by it because we are accumulating relative performance. Standardization by subtracting the median yields similar rankings as dividing by the median.

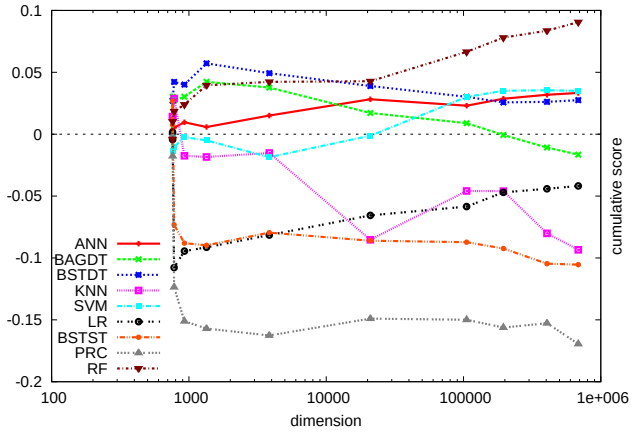


Figure 1. Cumulative standardized scores of each learning algorithm as a function of the dimension.

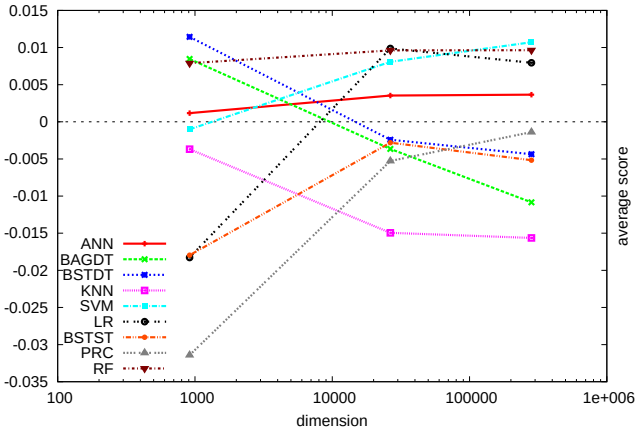


Figure 2. Moving average standardized scores of each learning algorithm as a function of the dimension.

well in modest dimensions, but lose ground to random forests, neural nets, and SVMs as dimensionality increases. Also, linear methods such as logistic regression begin to catch up as dimensionality increases.

Figure 2 shows the same results as Figure 1, but presented differently to avoid the complexity of accumulation. Here each point in the graph is the average performance of the 5 problems of lowest dimension (from 761 to 1344), the 5 problems of highest dimension (21K to 685K) and 5 problems of intermediate dimension (927 to 105K). Care must be used when interpreting this graph because each point averages over only 5 data sets. The results suggest that random forests overtake boosted trees. They are among the top performing methods for high-dimensional problems together with logistic regression and SVMs. Again we see that neural nets are consistently yielding above average performance even in very high dimension. Boosted trees, bagged trees, and KNN do not appear to cope well in very high dimensions. Boosted stumps, perceptrons, and Naive Bayes perform worse than the typical method regardless of dimension.

Figure 3 shows results similar to Figure 2 but only for different classes of SVMs: linear-only (L), kernel-only (K) and linear that can optimize accuracy or AUC (L+P) (Joachims, 2006). We also plot combinations of these (L+K and L+K+P) where the specific model that is best on the validation set is selected. The results suggest that the best overall performance with SVMs results from trying all possible SVMs (using the validation set to pick the best). Linear SVMs that can optimize accuracy or AUC outperform simple linear SVMs at modest dimensions, but have little effect when dimensionality is very high. Similarly, simple linear SVMs though not competitive with kernel SVMs at low dimensions, catch up as dimensionality increases.

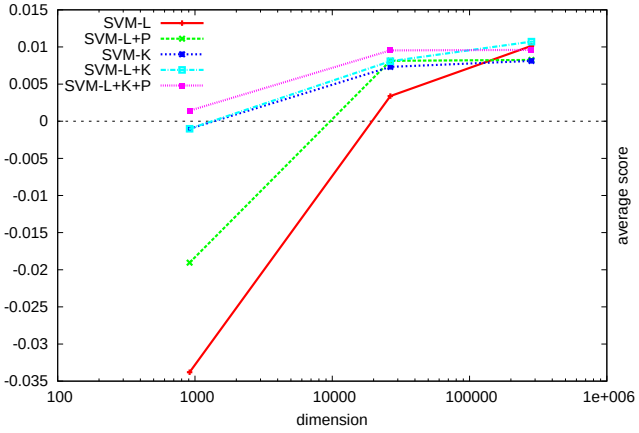


Figure 3. Moving average standardized scores for different SVM algorithms as a function of the dimension.

4. Bootstrap Analysis

We could not afford cross validation in these experiments because it would be too expensive. For some datasets and some methods, a single parameter setting can take days or weeks to run. Instead we used large test sets to make our estimates more reliable and adequately large validation sets to make sure that the parameters we select are good. However, without a statistical analysis, we cannot be sure that the differences we observe are not merely statistical fluctuation.

To help insure that our results would not change if we had selected datasets differently we did a bootstrap analysis similar to the one in (Caruana & Niculescu-Mizil, 2006). For a given metric we randomly select a bootstrap sample (sampling with replacement) from our 11 problems and then average the performance of each method across the problems in the bootstrap sample. Then we rank the methods. We repeat the bootstrap sampling 20,000 times and get 20,000 potentially different rankings of the learning methods.

Table 3 shows the results of the bootstrap analysis. Each entry in the table shows the percentage of time that each learning method ranks 1st, 2nd, 3rd, etc. on bootstrap samples of the datasets. Because of space limits, we only show results for average performance across the three metrics.

The bootstrap analysis suggests that random forests probably are the top performing method overall, with a 73% chance of ranking first, a 20% chance of ranking second, and less than a 8% chance of ranking below 2nd place. The ranks of other good methods, however, are less clear and there appears to be a three-way tie for 2nd place for boosted trees, ANNs, and SVMs.

5. Computational Challenges

Running this kind of experiment in high dimensions presents many computational challenges. In this section we outline a few of them.

In most high dimensional data features are sparse and the learning methods should take advantage of sparse vectors. For ANN, for example, when inputs are sparse, a lot of computation in the forward direction can be saved by using a matrix times sparse vector procedure. More savings happen when the weights are updated since the gradient of the error with respect to a weight going out of a unit with zero value vanishes. This is why our ANN implementation does not use momentum. If it did, all weights would have to be updated each iteration.

Another caveat is that for tree learning algorithms, indexing the data by feature instead of by example can speed up queries about which examples exhibit a particular feature. These queries are common during learning and one should consider this indexing scheme. Our random forest implementation indexes by feature.

Boosted decision trees on continuous data was the slowest of all methods. For bagged trees running times were better because we only grew 100 trees that can be grown in parallel. The same holds for random forests which have the added benefit that computation scales with the square root of dimensionality. Training ANNs was sometimes slow, mainly because applying some of the techniques in (Le Cun et al., 1998) would not preserve the sparsity of the data. For SVMs and logistic regression, we didn't have computational problems thanks to recent advances in scaling them (Genkin et al., 2006; Bordes et al., 2005; Joachims, 2006; Shalev-Shwartz et al., 2007). As a sanity check we compared the performance of the approximate kernel SVM solver with the exact SVM^{light} on some of our smallest problems and found no significant difference. Naive Bayes and perceptrons are among the fastest methods. KNN was sufficiently fast that we didn't have to use specialized data structures for nearest neighbor queries.

6. Related Work

Our work is most similar to (Caruana & Niculescu-Mizil, 2006). We already pointed out shortcomings in that study, but we also borrowed much from their methodology and tried to improve on it. STATLOG (King et al., 1995) was another comprehensive empirical study that was discussed in Section 1. A study by LeCun (LeCun et al., 1995) compares learning algorithms not only based on traditional performance metrics but also with respect to computational cost. Our study addresses this issue only qualitatively. Clearly, computational issues have to be taken into consideration in such large scale. A wide empirical comparison of voting algorithms such as bagging and boosting is conducted in (Bauer & Kohavi, 1999). The importance of evaluating performance on metrics such as AUC is discussed thoroughly in (Provost & Fawcett, 1997). The effect of different calibration methods is discussed in (Niculescu-Mizil & Caruana, 2005).

7. Discussion

Although there is substantial variability in performance across problems and metrics in our experiments, we can discern several interesting results. First, the results confirm the experiments in (Caruana & Niculescu-Mizil, 2006) where boosted decision trees perform exceptionally well when dimensionality is low. In this study boosted trees are the method of choice for up to about 4000 dimensions. Above that, random forests have the best overall performance. (Random forests were the 2nd best performing method in the previous study.) We suspect that the reason for this is that boosting trees is prone to overfitting and this becomes a serious problem in high dimensions. Random forests is better behaved in very high dimensions, it is easy to parallelize, scales efficiently to high dimensions and performs consistently well on all three metrics.

Non-linear methods do surprisingly well in high dimensions if model complexity can be controlled, e.g. by exploring the space of hypotheses from simple to complex (ANN), by margins (SVMs), or by basing some decisions on random projections (RF). Logistic regression and linear SVMs also gain in performance as dimensionality increases. Contrary to low dimensions, in high dimensions we have no evidence that linear SVMs can benefit from training procedures that directly op-

Table 3. Bootstrap analysis of rankings by average performance across problems

AVG	1ST	2ND	3RD	4TH	5TH	6TH	7TH	8TH	9TH	10TH
RF	0.727	0.207	0.054	0.011	0.001	0	0	0	0	0
ANN	0.053	0.172	0.299	0.256	0.119	0.072	0.019	0.011	0	0
BSTDT	0.059	0.228	0.18	0.222	0.18	0.075	0.044	0.012	0.001	0
SVM	0.043	0.195	0.213	0.193	0.156	0.088	0.08	0.031	0.001	0
LR	0.089	0.132	0.073	0.075	0.108	0.177	0.263	0.081	0	0
BAGDT	0.002	0.012	0.109	0.123	0.251	0.284	0.123	0.078	0.016	0
KNN	0.023	0.045	0.051	0.057	0.085	0.172	0.122	0.177	0.258	0.01
BSTST	0.004	0.009	0.021	0.063	0.086	0.109	0.3	0.387	0.02	0
PRC	0	0	0	0	0.013	0.024	0.047	0.222	0.695	0
NB	0	0	0	0	0	0	0	0	0.01	0.99

timize specific metrics such as AUC.

The results suggest that calibration never hurts and almost always helps on these problems. Even methods such as ANN and logistic regression benefit from calibration in most cases. We suspect that the reasons for this are the availability of more validation data for calibration than in previous studies and that high dimensional problems are harder in some sense.

Acknowledgments

We thank all the students who took CS678 at Cornell in the spring of 2007 and helped with this study. We especially thank Sergei Fotin, Michael Friedman, Myle Ott and Raghu Ramanujan who implemented KNN, ANNs, Random Forests and Boosted Trees respectively. We also thank Alec Berntson Eric Breck and Art Munson for providing the crystallography, DSE and ornithology datasets respectively. Art also put together the calibration procedures. Finally, we thank the 3 anonymous reviewers for their helpful comments. This work was supported in part by NSF Awards 0412930 and 0412894.

References

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *MLJ*, *36*, 105–139.

Bordes, A., Ertekin, S., Weston, J., & Bottou, L. (2005). Fast kernel classifiers with online and active learning. *JMLR*, *6*, 1579–1619.

Breiman, L. (2001). Random Forests. *MLJ*, *45*, 5–32.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *ICML '06*, 161–168.

Freund, Y., & Schapire, R. (1999). Large Margin Classification Using the Perceptron Algorithm. *MLJ*, *37*, 277–296.

Genkin, A., Lewis, D., & Madigan, D. (2006). Large-scale bayesian logistic regression for text categorization. *Technometrics*.

Joachims, T. (2006). Training linear SVMs in linear time. *SIGKDD*, 217–226.

King, R., Feng, C., & Shutherland, A. (1995). Statlog: comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, *9*, 259–287.

Le Cun, Y., Bottou, L., Orr, G. B., & Müller, K.-R. (1998). Efficient backprop. In *Neural networks, tricks of the trade*, LNCS 1524. Springer Verlag.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., et al. (1995). Comparison of learning algorithms for handwritten digit recognition. *International Conference on Artificial Neural Networks*, *60*.

Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. *ICML '05*, 625–632.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, *10*.

Provost, F. J., & Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. *KDD '97* (pp. 43–48).

Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. *ICML '07* (pp. 807–814).

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *KDD '02*, 694–699.

Fast Support Vector Machine Training and Classification on Graphics Processors

Bryan Catanzaro CATANZAR@EECS.BERKELEY.EDU
Narayanan Sundaram NARAYANS@EECS.BERKELEY.EDU
Kurt Keutzer KEUTZER@EECS.BERKELEY.EDU
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA

Abstract

Recent developments in programmable, highly parallel Graphics Processing Units (GPUs) have enabled high performance implementations of machine learning algorithms. We describe a solver for Support Vector Machine training running on a GPU, using the Sequential Minimal Optimization algorithm and an adaptive first and second order working set selection heuristic, which achieves speedups of 9-35× over LIBSVM running on a traditional processor. We also present a GPU-based system for SVM classification which achieves speedups of 81-138× over LIBSVM (5-24× over our own CPU based SVM classifier).

1. Introduction

Driven by the capabilities and limitations of modern semiconductor manufacturing, the computing industry is currently undergoing a massive shift towards parallel computing (Asanović et al., 2006). This shift brings dramatically enhanced performance to those algorithms which can be adapted to parallel computers.

One set of such algorithms are those used to implement Support Vector Machines (Cortes & Vapnik, 1995). Thanks to their robust generalization performance, SVMs have found use in diverse classification tasks, such as image recognition, bioinformatics, and text processing. Yet, training Support Vector Machines and using them for classification remains very computationally intensive. Much research has been done to accelerate training time, such as Osuna's decomposition approach (Osuna et al., 1997), Platt's Sequential Minimal Optimization (SMO) algorithm (Platt, 1999), Joachims' SVM^{light} (Joachims, 1999), which introduced shrinking and kernel caching, , and the working set selection heuristics used by LIBSVM (Fan et al., 2005). Despite this research, SVM training time is still significant for large training sets.

In this paper, we show how Support Vector Machine training and classification can be adapted to a highly parallel, yet widely available and affordable computing platform: the graphics processor, or more specifically, the Nvidia GeForce 8800 GTX, and detail the performance gains achieved.

The organization of the paper is as follows. Section 2 describes the SVM training and classification problems briefly. Section 3 gives an overview of the architectural and programming features of the GPU. Section 4 presents the details of implementation of the parallel SMO approach on the GPU. Section 5 explains the implementation details of the SVM classification problem. We present our results in Section 6 and conclude in Section 7.

2. Support Vector Machines

We consider the standard two-class soft-margin SVM classification problem (C-SVM), which classifies a given data point $x \in \mathbb{R}^n$ by assigning a label $y \in \{-1, 1\}$.

2.1. SVM Training

Given a labeled training set consisting of a set of data points $x_i, i \in \{1, ..., l\}$ with their accompanying labels $y_i, i \in \{1, ..., l\}$, the SVM training problem can be written as the following Quadratic Program, where α_i is a set of weights, one for each training point, which are being optimized to determine the SVM classifier, C is a parameter which trades classifier generality for accuracy on the training set, and $Q_{ij} = y_i y_j \Phi(x_i, x_j)$,

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

where $\Phi(x_i, x_j)$ is a kernel function.

$$\begin{aligned} \max_{\alpha} \; & F(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\alpha^T Q \alpha \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \; \forall i \in 1 \ldots l \\ & y^T \alpha = 0 \end{aligned} \tag{1}$$

We consider the standard kernel functions shown in table 1.

Table 1. Standard Kernel Functions

LINEAR	$\Phi(x_i, x_j) = x_i \cdot x_j$
POLYNOMIAL	$\Phi(x_i, x_j; a, r, d) = (a x_i \cdot x_j + r)^d$
GAUSSIAN	$\Phi(x_i, x_j; \gamma) = \exp\left\{-\gamma \lvert\lvert x_i - x_j \rvert\rvert^2\right\}$
SIGMOID	$\Phi(x_i, x_j; a, r) = \tanh(a x_i \cdot x_j + r)$

2.1.1. SMO Algorithm

The SVM Training problem can be solved by many methods, each with different parallelism implications. We have implemented the Sequential Minimal Optimization algorithm (Platt, 1999), with a hybrid working set selection heuristic making use of the first order heuristic proposed by (Keerthi et al., 2001) as well as the second order heuristic proposed by (Fan et al., 2005).

The SMO algorithm is a specialized optimization approach for the SVM quadratic program. It takes advantage of the sparse nature of the support vector problem and the simple nature of the constraints in the SVM QP to reduce each optimization step to its minimum form: updating two α_i weights. The bulk of the computation is then to update the Karush-Kuhn-Tucker optimality conditions for the remaining set of weights and then find the next two weights to update in the next iteration. This is repeated until convergence. We state this algorithm briefly, for reference purposes.

Algorithm 1 Sequential Minimal Optimization

Input: training data x_i, labels y_i, $\forall i \in \{1..l\}$
Initialize: $\alpha_i = 0$, $f_i = -y_i$, $\forall i \in \{1..l\}$,
Initialize: b_{high}, b_{low}, i_{high}, i_{low}
Update $\alpha_{i_{high}}$ and $\alpha_{i_{low}}$
repeat
  Update f_i, $\forall i \in \{1..l\}$
  Compute: b_{high}, i_{high}, b_{low}, i_{low}
  Update $\alpha_{i_{high}}$ and $\alpha_{i_{low}}$
until $b_{low} \leq b_{high} + 2\tau$

For the first iteration, we initialize $b_{high} = -1$, $i_{high} = \min\{i : y_i = 1\}$, $b_{low} = 1$, and $i_{low} = \min\{i : y_i = -1\}$.

During each iteration, once we have chosen i_{high} and i_{low}, we take the optimization step:

$$\alpha'_{i_{low}} = \alpha_{i_{low}} + y_{i_{low}}(b_{high} - b_{low})/\eta \tag{2}$$

$$\alpha'_{i_{high}} = \alpha_{i_{high}} + y_{i_{low}} y_{i_{high}} (\alpha_{i_{low}} - \alpha'_{i_{low}}) \tag{3}$$

where $\eta = \Phi(x_{i_{high}}, x_{i_{high}}) + \Phi(x_{i_{low}}, x_{i_{low}}) - 2\Phi(x_{i_{high}}, x_{i_{low}})$. To ensure that this update is feasible, $\alpha'_{i_{low}}$ and $\alpha'_{i_{high}}$ must be clipped to the valid range $0 \leq \alpha_i \leq C$.

The optimality conditions can be tracked through the vector $f_i = \sum_{j=1}^{l} \alpha_j y_j \Phi(x_i, x_j) - y_i$, which is constructed iteratively as the algorithm progresses. After each α update, f is updated for all points. This is one of the major computational steps of the algorithm, and is done as follows:

$$\begin{aligned} f'_i = f_i &+ (\alpha'_{i_{high}} - \alpha_{i_{high}}) y_{i_{high}} \Phi(x_{i_{high}}, x_i) \\ &+ (\alpha'_{i_{low}} - \alpha_{i_{low}}) y_{i_{low}} \Phi(x_{i_{low}}, x_i) \end{aligned} \tag{4}$$

In order to evaluate the optimality conditions, we define index sets:

$$\begin{aligned} I_{high} = &\{i : 0 < \alpha_i < C\} \cup \{i : y_i > 0, \alpha_i = 0\} \\ &\cup \{i : y_i < 0, \alpha_i = C\} \end{aligned} \tag{5}$$

$$\begin{aligned} I_{low} = &\{i : 0 < \alpha_i < C\} \cup \{i : y_i > 0, \alpha_i = C\} \\ &\cup \{i : y_i < 0, \alpha_i = 0\} \end{aligned} \tag{6}$$

Because of the approximate nature of the solution process, these index sets are computed to within a tolerance ϵ, e.g. $\{i : \epsilon < \alpha_i < (C - \epsilon)\}$.

We can then measure the optimality of our current solution by checking the optimality gap, which is the difference between $b_{high} = \min\{f_i : i \in I_{high}\}$, and $b_{low} = \max\{f_i : i \in I_{low}\}$. When $b_{low} \leq b_{high} + 2\tau$, we terminate the algorithm.

2.1.2. Working set selection

During each iteration, we need to choose i_{high} and i_{low}, which index the α weights which will be changed in the following optimization step. The first order heuristic from (Keerthi et al., 2001) chooses them as follows:

$$i_{high} = \arg\min\{f_i : i \in I_{high}\} \tag{7}$$

$$i_{low} = \arg\max\{f_i : i \in I_{low}\} \tag{8}$$

The second order heuristic from (Fan et al., 2005) chooses i_{high} and i_{low} to optimize the unconstrained SVM functional. An optimal approach to this problem would require examining $\binom{l}{2}$ candidate pairs, which would be computationally intractable. To simplify the problem, i_{high} is instead chosen as in the first order

heuristic, and then i_{low} is chosen to maximally improve the objective function while still guaranteeing progress towards the constrained optimum from problem (1). More explicitly:

$$i_{high} = \arg\min\{f_i : i \in I_{high}\} \quad (9)$$

$$i_{low} = \arg\max\{\Delta F_i(\alpha) : i \in I_{low}, f_{i_{high}} < f_i\} \quad (10)$$

After choosing i_{high}, we compute for all $i \in \{1..l\}$

$$\beta_i = f_{i_{high}} - f_i \quad (11)$$

$$\eta_i = \Phi(x_{i_{high}}, x_{i_{high}}) + \Phi(x_i, x_i) - 2\Phi(x_{i_{high}}, x_i) \quad (12)$$

$$\Delta F_i(\alpha) = \beta_i^2/\eta_i \quad (13)$$

We then find the maximum ΔF_i over all valid points ($i \in I_{low}$) for which we are guaranteed to progress towards the constrained optimum ($f_{i_{high}} < f_i$).

2.1.3. Adaptive heuristic

The second order heuristic utilizes more information from the SVM training problem, and so it generally reduces the number of iterations necessary during the solution process. However, it is more costly to compute. In our GPU implementation, the geometric mean of iteration time over our benchmark set using the second order heuristic increased by 1.9× compared to the first order heuristic. On some benchmarks, the total number of iterations decreased sufficiently to provide a significant speedup overall, but on others, the second order heuristic is counterproductive for our GPU implementation.

To overcome this problem, we implemented an adaptive heuristic that chooses between the two selection heuristics dynamically, with no input or tuning from the user. The adaptive heuristic periodically samples progress towards convergence as a function of wall-clock time using both heuristics, then chooses the more productive heuristic.

This sampling occurs every $l/10$ iterations, and during each sample, the heuristic under test is executed for two phases of 64 iterations each. The average optimality gap in each of these phases is computed, and then the rate of progress is estimated by dividing the change in the optimality gap over the two phases by the time it has taken to execute them. The same sampling process is then performed with the other heuristic, and the best heuristic is then used until the next sampling period.

2.2. SVM Classification

The SVM classification problem is as follows: for each data point z which should be classified, compute

$$\hat{z} = \operatorname{sgn}\left\{b + \sum_{i=1}^{l} y_i \alpha_i \Phi(x_i, z)\right\} \quad (14)$$

where $z \in \mathbb{R}^n$ is a point which needs to be classified, and all other variables remain as previously defined.

From the classification problem definition, it follows immediately that the decision surface is defined by referencing a subset of the training data, or more specifically, those training data points for which the corresponding $\alpha_i > 0$. Such points are called support vectors.

Generally, we classify not just one point, but a set of points. We exploit this for better performance, as explained in Section 5.

3. Graphics Processors

Graphics processors are currently transitioning from their initial role as specialized accelerators for triangle rasterization to general purpose engines for high throughput floating-point computation. Because they still service the large gaming industry, they are ubiquitous and relatively inexpensive.

GPU architectures are specialized for compute-intensive, memory-intensive, highly parallel computation, and therefore are designed such that more resources are devoted to data processing than caching or control flow. State of the art GPUs provide up to an order of magnitude more peak IEEE single-precision floating-point than their CPU counterparts. Additionally, GPUs have much more aggressive memory subsystems, typically endowed with more than 10x higher memory bandwidth than a CPU. Peak performance is usually impossible to achieve on general purpose applications, yet capturing even a fraction of peak performance yields significant speedup.

GPU performance is dependent on finding high degrees of parallelism: a typical computation running on the GPU must express thousands of threads in order to effectively use the hardware capabilities. As such, we consider it an example of future "many-core" processing (Asanović et al., 2006). Algorithms for machine learning applications will need to consider such parallelism in order to utilize many-core processors. Applications which do not express parallelism will not continue improving their performance when run on newer computing platforms at the rates we have enjoyed in the past. Therefore, finding large scale par-

allelism is important for compute performance in the future. Programming for GPUs is then indicative of the future many-core programming experience.

3.1. Nvidia GeForce 8800 GTX

In this project, we employ the NVIDIA GeForce 8800 GTX GPU, which is an instance of the G80 GPU architecture, and is a standard GPU widely available on the market. Pertinent facts about the GPU platform can be found in table 2. We refer the reader to the Nvidia CUDA reference manual for more details (Nvidia, 2007).

Table 2. Nvidia GeForce 8800 GTX Characteristics

# of stream processors	128
Peak general purpose IEEE SP	346 GFlops
Multiprocessor local store size	16 kB
Clock rate	1.35 GHz
Memory capacity	768 MB
Memory bandwidth	86.4 GB/s
CPU⟷GPU bandwidth	3.2 Gbit/s

3.2. CUDA

Nvidia provides a programming environment for its GPUs called the Compute Unified Device Architecture (CUDA). The user codes in annotated C++, accelerating compute intensive portions of the application by executing them on the GPU.

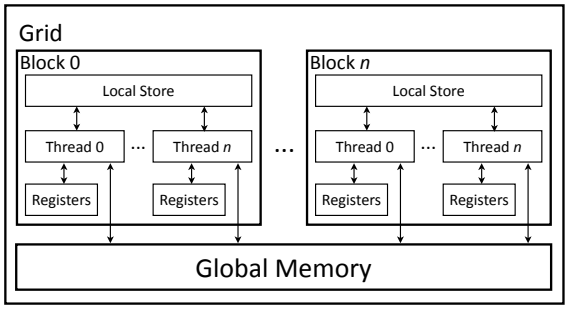


Figure 1. Logical organization of the GeForce 8800

Figure 1 illustrates how the GPU appears to the programmer. The programmer organizes the computation into grids, which are organized as a set of thread blocks. The grids run sequentially on the GPU, meaning that all computation in the grid must finish before another grid is invoked. As mentioned, grids contain thread blocks, which are batches of threads that execute together, sharing local memories and synchronizing at programmer specified barriers. A maximum of 512 threads can comprise a thread block, which puts a limit on the scope of synchronization and communication in the computation. However, enormous numbers of blocks can be launched in parallel in the grid, so that the total number of threads that can be launched in parallel is very high. In practice, we need a large number of thread blocks to ensure that the compute power of the GPU is efficiently utilized.

4. SVM Training Implementation

Since GPUs need a large number of threads to efficiently exploit parallelism, we create one thread for every data point in the training set. For the first phase of the computation, each thread computes f_i' from equation (4). We then apply a working set selection heuristic to select the next points which will be optimized. The details are explained in the following section.

4.1. Map Reduce

At least since the LISP programming language, programmers have been mapping independent computations onto partitioned data sets, using reduce operations to summarize the results. Recently, Google proposed a Map Reduce variant for processing large datasets on compute clusters (Dean & Ghemawat, 2004). This algorithmic pattern is very useful for extracting parallelism, since it is simple to understand, and maps well to parallel hardware, given the inherent parallelism in the map stage of the computation.

The Map Reduce pattern has been shown to be useful for many machine learning applications (Chu et al., 2007), and is a natural fit for our SVM training algorithm. For the first order heuristic, the computation of f_i' for all points is the map function, and the search for b_{low}, b_{high}, i_{low} and i_{high} is the reduction operation. For the second order heuristic, there are two Map Reduce stages: one to compute f_i', b_{high} and i_{high}, and another where the map stage computes ΔF_i for all points, while the reduce stage computes b_{low} and i_{low}.

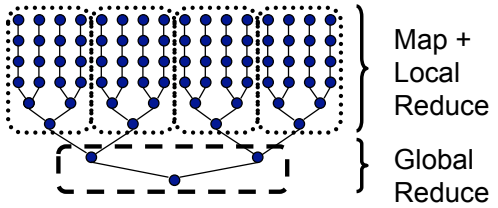


Figure 2. Structuring the Map Reduce

Because the CUDA programming model has strict lim-

itations on synchronization and communication between thread blocks, we organize the reductions in two phases, as shown in figure 2. The first phase does the map computation, as well as a local reduce within a thread block. The second phase finishes the global reduction. Each phase of this process is implemented as a separate call to the GPU.

4.2. Implementation Details

4.2.1. Caching

Since evaluating the kernel function $\Phi(\cdot)$ is the dominant part of the computation, it is useful to cache as much as possible from the matrix of kernel function evaluations $K_{ij} = \Phi(x_i, x_j)$ (Joachims, 1999). We compute rows of this matrix on the fly, as needed by the algorithm, and cache them in the available memory on the GPU.

When updating the vector f, we need access to two rows of K, since we have changed exactly two entries in α. In our system, the CPU checks to see which of these two rows, if any, are present in the cache. If a row is not present, the CPU voids the least recently used row of the cache, and assigns it to the new row which is needed. For the rows which hit in the cache, the GPU avoids doing the kernel evaluations. Otherwise, the GPU writes out the appropriate row or rows after computing the kernel values. When using the second order heuristic, the computation of ΔF references the row of K corresponding to i_{high}, which guarantees that the next update of f will have a cache hit for its access to the same row.

4.2.2. Data Movement

Programming the GPU requires manually copying data from the host computer to the GPU and vice versa, and it also requires manually copying data from the GPU's global memory to the fast local stores. As mentioned previously, if the cache does not contain a particular row of K corresponding to the point x_j, that row will need to be generated, which means that we need to compute $\Phi(x_i, x_j)\ \forall i \in 1..l$. Since the vector x_j is shared between all computations, we load it into the GPU's local store. This is key to performance, since accessing the local store is orders of magnitude faster than accessing the global memory.

4.3. Related Work

There have been previous attempts to parallelize the SVM training problem. The most similar to ours is (Cao et al., 2006), which parallelizes the SMO algorithm on a cluster of computers using MPI. Both our approach and their approach use the concurrency inherent in the KKT condition updates as the major source of parallelism. However, in terms of implementation, GPUs present a completely different model than clusters, and hence the amount of parallelism exploited, such as the number of threads, granularity of computation per thread, memory access patterns, and data partitioning are very different. We also implement more sophisticated working set selection heuristics.

Many other approaches for parallelizing SVM training have been presented. The cascade SVM (Graf et al., 2005) is another proposed method for parallelizing SVM training on clusters. It uses a method of divide and conquer to solve large SVM problems. (Zanni et al., 2006) parallelize the underlying QP solver using Parallel Gradient Projection Technique. Work has been done on using a parallel Interior Point Method for solving the SVM training problem (Wu et al., 2006). (Collobert et al., 2002) proposes a method where the several smaller SVMs are trained in a parallel fashion and their outputs weighted using a Artificial Neural Network. (Ferreira et al., 2006) implement a gradient based solution for SVM training, which relies on data parallelism in computing the gradient of the objective function for an unconstrained QP optimization at its core. Some of these techniques, for example, the training set decomposition approaches like the Cascade SVM are orthogonal to the work we describe, and could be applied to our solver. (Bottou et al., 2007) give an extensive overview of parallel SVM implementations. We implemented the parallel SMO training algorithm because of its relative simplicity, yet high performance and robust convergence characteristics.

5. SVM Classification Implementation

We approached the SVM classification problem by making use of Map Reduce computations as well as vendor supplied Basic Linear Algebra Subroutines - specifically, the Matrix Matrix Multiplication routine (SGEMM), which calculates $C' = \alpha AB + \beta C$, for matrices A, B, and C and scalars α and β. For the Linear, Polynomial, and Sigmoid kernels, calculating the classification value involves finding the dot product between all test points and the support vectors, which is done through SGEMM. For the Gaussian kernel, we use the simple identity $||x - y||^2 = x \cdot x + y \cdot y - 2x \cdot y$ to recast the computation into a Matrix Matrix multiplication, where the SGEMM computes $D_{ij} = -\gamma||z_i - x_j||^2 = 2\gamma(z_i \cdot x_j) - \gamma(z_i \cdot z_i + x_j \cdot x_j)$, for a set of unknown points z and a set of support vectors x. We then apply a map reduce computation to

combine the computed D values to get the final result.

Continuing the Gaussian example, the map function exponentiates D_{ij} element wise, multiplies each column of the resulting matrix by the appropriate $y_j\alpha_j$. The reduce function sums the rows of the matrix and adds b to obtain the final classification for each data point as given by equation (14). Other kernels require similar Map Reduce calculations to finish the classification.

6. Results

The SMO implementation on the GPU is compared with LIBSVM, as LIBSVM uses Sequential Minimal Optimization for SVM training. We used the Gaussian kernel in all of our experiments, since it is widely employed.

6.1. Training

We tested the performance of our GPU implementation versus LIBSVM on the datasets detailed in tables 3 and 4.

Table 3. Datasets - References and training parameters

Dataset	C	γ
Adult (Asuncion & Newman, 2007)	100	0.5
Web (Platt, 1999)	64	7.8125
MNIST (LeCun et al., 1998)	10	0.125
USPS (Hull, 1994)	10	2^{-8}
Forest (Asuncion & Newman, 2007)	10	0.125
Face (Rowley et al., 1998)	10	0.125

Table 4. Dataset Size

Dataset	# Points	# Dimensions
Adult	32,561	123
Web	49,749	300
MNIST	60,000	784
USPS	7,291	256
Forest	561,012	54
Face	6,977	381

The sizes of the datasets are given in table 4. References for the datasets used and the (C, γ) values used for SVM training are provided in table 3.

We ran LIBSVM on an Intel Core 2 Duo 2.66 GHz processor, and gave LIBSVM a cache size of 650 MB, which is larger than our GPU implementation was allowed. CPU-GPU communication overhead was included in the solver runtime, but file I/O time was excluded for both our solver and LIBSVM. Table 5 shows results from our solver. File I/O varies from 1.2 seconds for USPS to about 12 seconds for Forest dataset. The CPU - GPU data transfer overhead was also very low. The time taken to transfer the training data to the GPU and copy the results back was less than 0.6 seconds, even for our largest dataset (Forest).

Since any two solvers give slightly different answers on the same optimization problem, due to the inexact nature of the optimization process, we show the number of support vectors returned by the two solvers as well as how close the final values of b were for the GPU solver and LIBSVM, which were both run with the same tolerance value $\tau = 0.001$. As shown in the table, the deviation in number of support vectors between the two solvers is less than 2%, and the deviation in the offset b is always less than 0.1%. Our solver provides equivalent accuracy to the LIBSVM solver, which will be shown again in the classification results section.

Table 5. SVM Training Convergence Comparison

Dataset	Number of SVs		Difference in b (%)
	GPU Adaptive	LIBSVM	
Adult	18,674	19,058	-0.004
Web	35,220	35,232	-0.01
MNIST	43,730	43,756	-0.04
USPS	684	684	0.07
Forest	270,351	270,311	0.07
Face	3,313	3,322	0.01

Table 6 contains performance results for the two solvers. We see speedups in all cases from 9× to 35×. For reference, we have shown results for the solvers using both heuristics statically. Examining the data shows that the adaptive heuristic performs robustly, surpassing or coming close to the performance of the best static heuristic on all benchmarks.

6.2. Classification

Results for our classifier are presented in table 8. We achieve $81 - 138\times$ speedup over LibSVM on the datasets shown. As with the solver, file I/O times were excluded from overall runtime. File I/O times vary from 0.4 seconds for Adult dataset to about 6 seconds for MNIST dataset.

6.2.1. Optimizations to CPU based classifier

LIBSVM classifies data points serially. This effectively precludes data locality optimizations and produces significant slowdown. It also represents data in a sparse format, which can cause overhead as well.

To optimize the CPU classifier, we performed the following:

1. We changed the data structure used for storing

Table 6. SVM Training Results

Dataset	GPU 1st Order		GPU 2nd Order		GPU Adaptive		LIBSVM		Speedup (×)
	Iter.	Time (s)	Iter.	Time (s)	Iter.	Time (s)	Iter.	Time (s)	(Adaptive)
Adult	114,985	30.15	40,044	30.46	64,446	**26.92**	43,735	550.2	20.4
Web	79,749	174.17	81,498	290.23	70,686	**163.89**	85,299	2422.46	14.8
MNIST	68,055	**475.42**	67,731	864.46	68,113	483.07	76,385	16965.79	35.1
USPS	6,949	0.596	3,730	**0.546**	4,734	0.576	4,614	5.092	8.8
Forest	2,070,867	4571.17	236,601	**1441.08**	450,506	2023.24	275,516	66523.53	32.9
Face	6,044	**1.30**	4,876	**1.30**	5,535	1.32	5,342	27.61	20.8

the support vectors and test vectors from a sparse indexed set to a dense matrix.

2. To maximize performance, we used BLAS routines from the Intel Math Kernel Library to perform operations similar to those mentioned in Section 5.

3. Wherever possible, loops were parallelized (2-way for the dual-core machine) using OpenMP.

These optimizations improved the classification speed on the CPU by a factor of $3.4 - 28.3\times$. The speedup numbers for the different datasets are shown in table 8. It should be noted that the GPU version is better than the optimized CPU versions by a factor of $4.9 - 23.9\times$.

For some insight into these results, we note that the optimized CPU classifier performs best on problems with a large number of input space dimensions, which helps make the SVM classification process compute bound. For problems with a small number of input space dimensions, the SVM classification process is memory bound, meaning it is limited by memory bandwidth. Since the GPU has much higher memory bandwidth, as noted in section 3, it is even more attractive for such problems.

We tested the combined SVM training and classification process for accuracy by using the SVM classifier produced by the GPU solver with the GPU classification routine, and used the SVM classifier provided by LIBSVM's solver to perform classification with LIBSVM. Thus, the accuracy of the classification results presented in table 7 reflect the overall accuracy of the GPU solver and GPU classifier system. The results are identical, which shows that our GPU based SVM system is as accurate as traditional CPU based methods.

Table 7. Accuracy of GPU SVM classification vs. LIBSVM

Dataset	GPU Accuracy	LIBSVM Accuracy
Adult	6619/8000	6619/8000
Web	3920/4000	3920/4000
MNIST	2400/2500	2400/2500
USPS	1948/2007	1948/2007
Face	23665/24045	23665/24045

7. Conclusion

This work has demonstrated the utility of graphics processors for SVM classification and training. Training time is reduced by $9 - 35\times$, and classification time is reduced by $81 - 138\times$ compared to LIBSVM, or $5 - 24\times$ over our own CPU based SVM classifier. These kinds of performance improvements can change the scope of SVM problems which are routinely solved, increasing the applicability of SVMs to difficult classification problems. For example, training a classifier for an input data set with almost 600000 data points and 50 dimensions takes only 34 minutes on the GPU, compared with over 18 hours on the CPU.

The GPU is a very low cost way to achieve such high performance: the GeForce 8800 GTX fits into any modern desktop machine, and currently costs $300. Problems which used to require a compute cluster can now be solved on one's own desktop. New machine learning algorithms that can take advantage of this kind of performance, by expressing parallelism widely, will provide compelling benefits on future many-core platforms.

Acknowledgements

The authors acknowledge the support of the Gigascale Systems Research Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program. Bryan Catanzaro is also supported by a National Science Foundation Graduate Research Fellowship. The authors thank the anonymous reviewers for their com-

Table 8. Performance of GPU SVM classifier compared to LIBSVM and Optimized CPU classifier

Dataset	LibSVM	CPU Optimized classifier		GPU Classifier		
	Time (s)	Time (s)	Speedup (×) compared to LIBSVM	Time (s)	Speedup (×) compared to LibSVM	Speedup (×) compared to CPU optimized code
Adult	61.307	7.476	8.2	0.575	106.6	13.0
Web	106.835	15.733	6.8	1.063	100.5	14.8
MNIST	269.880	9.522	28.3	1.951	138.3	4.9
USPS	0.777	0.229	3.4	0.00958	81.1	23.9
Face	88.835	5.191	17.1	0.705	126.0	7.4

ments and suggestions.

References

Asanović, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W., & Yelick, K. A. (2006). *The Landscape of Parallel Computing Research: A View from Berkeley* (Technical Report UCB/EECS-2006-183). EECS Department, University of California, Berkeley.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository.

Bottou, L., Chapelle, O., DeCoste, D., & Weston, J. (2007). *Large-scale kernel machines*. The MIT Press.

Cao, L., Keerthi, S., Ong, C.-J., Zhang, J., Periyathamby, U., Fu, X. J., & Lee, H. (2006). Parallel sequential minimal optimization for the training of support vector machines. *IEEE Transactions on Neural Networks*, *17*, 1039–1049.

Chu, C.-T., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotun, K. (2007). Map-reduce for machine learning on multicore. In B. Schölkopf, J. Platt and T. Hoffman (Eds.), *Advances in neural information processing systems 19*, 281–288. Cambridge, MA: MIT Press.

Collobert, R., Bengio, S., & Bengio, Y. (2002). A parallel mixture of svms for very large scale problems. *Neural Computation*, *14*, 1105–1114.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, *20*, 273–297.

Dean, J., & Ghemawat, S. (2004). Mapreduce: simplified data processing on large clusters. *OSDI'04: Proceedings of the 6th Symposium on Operating Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association.

Fan, R.-E., Chen, P.-H., & Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *J. Mach. Learn. Res.*, *6*, 1889–1918.

Ferreira, L. V., Kaskurewicz, E., & Bhaya, A. (2006). Parallel implementation of gradient-based neural networks for svm training. *International Joint Conference on Neural Networks*.

Graf, H. P., Cosatto, E., Bottou, L., Dourdanovic, I., & Vapnik, V. (2005). Parallel support vector machines: The cascade svm. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*, 521–528. Cambridge, MA: MIT Press.

Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, *16*, 550–554.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In *Advances in kernel methods: support vector learning*. Cambridge, MA, USA: MIT Press.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Comput.*, *13*, 637–649.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

Nvidia (2007). Nvidia CUDA. `http://nvidia.com/cuda`.

Osuna, E., Freund, R., & Girosi, F. (1997). An improved training algorithm for support vector machines. *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, 276–285.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: support vector learning*, 185–208. Cambridge, MA, USA: MIT Press.

Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 23–38.

Wu, G., Chang, E., Chen, Y. K., & Hughes, C. (2006). Incremental approximate matrix factorization for speeding up support vector machines. *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 760–766). New York, NY, USA: ACM Press.

Zanni, L., Serafini, T., & Zanghirati, G. (2006). Parallel software for training large scale support vector machines on multiprocessor systems. *J. Mach. Learn. Res.*, *7*, 1467–1492.

Fast Nearest Neighbor Retrieval for Bregman Divergences

Lawrence Cayton LCAYTON@CS.UCSD.EDU
Department of Computer Science and Engineering, University of California, San Diego, CA 92093

Abstract

We present a data structure enabling efficient nearest neighbor (NN) retrieval for bregman divergences. The family of bregman divergences includes many popular dissimilarity measures including KL-divergence (relative entropy), Mahalanobis distance, and Itakura-Saito divergence. These divergences present a challenge for efficient NN retrieval because they are not, in general, metrics, for which most NN data structures are designed. The data structure introduced in this work shares the same basic structure as the popular metric ball tree, but employs convexity properties of bregman divergences in place of the triangle inequality. Experiments demonstrate speedups over brute-force search of up to several orders of magnitude.

1. Introduction

Nearest neighbor (NN) search is a core primitive in machine learning, vision, signal processing, and elsewhere. Given a database X, a dissimilarity measure d, and a query q, the goal is to find the $x \in X$ minimizing $d(x, q)$. Brute-force search is often impractical given the size and dimensionality of modern data sets, so many data structures have been developed to accelerate NN retrieval.

Most retrieval data structures are for the ℓ_2 norm and, more generally, metrics. Though many dissimilarity measures are metrics, many are not. For example, the natural notion of dissimilarity between probability distributions is the KL-divergence (relative entropy), which is not a metric. It has been used to compare histograms in a wide variety of applications, including text analysis, image classification, and content-based image retrieval (Pereira et al., 1993; Puzicha et al., 1999; Rasiwasia et al., 2007). Because the KL-divergence does not satisfy the triangle inequality, very little of the research on NN retrieval structures applies.

The KL-divergence belongs to a broad family of dissimilarities called *bregman divergences*. Other examples include Mahalanobis distance, used *e.g.* in classification (Weinberger et al., 2006); the Itakura-Saito divergence, used in sound processing (Gray et al., 1980); and ℓ_2^2 distance. Bregman divergences present a challenge for fast NN retrieval since they need not be symmetric or satisfy the triangle inequality.

This paper introduces *bregman ball trees* (bbtrees), the first NN retrieval data structure for general bregman divergences. The data structure is a relative of the popular metric ball tree (Omohundro, 1989; Uhlmann, 1991; Moore, 2000). Since this data structure is built on the triangle inequality, the extension to bregman divergences is non-trivial.

A bbtree defines a hierarchical space decomposition based on bregman balls; retrieving a NN with the tree requires computing bounds on the bregman divergence from a query to these balls. We show that this divergence can be computed exactly with a simple bisection search that is very efficient. Since only bounds on the divergence are needed, we can often stop the search early using primal and dual function evaluations.

In the experiments, we show that the bbtree provides a substantial speedup—often orders of magnitude—over brute-force search.

2. Background

This section provides background on bregman divergences and nearest neighbor search.

2.1. Bregman Divergences

First we briefly overview bregman divergences.

Definition 1 (Bregman, 1967)**.** *Let f be a strictly convex differentiable function.* [1] *The bregman diver-*

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] Additional technical restrictions are typically put on

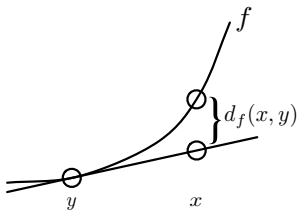


Figure 1. The bregman divergence between x and y.

gence based on f is

$$d_f(x,y) \equiv f(x) - f(y) - \langle \nabla f(y), x - y \rangle.$$

One can interpret the bregman divergence as the distance between a function and its first-order taylor expansion. In particular, $d_f(x,y)$ is the difference between $f(x)$ and the linear approximation of $f(x)$ centered at y; see figure 1. Since f is convex, $d_f(x,y)$ is always nonnegative.

Some standard bregman divergences and their base functions are listed in table 1.

A bregman divergence is typically used to assess similarity between two objects, much like a metric. But though metrics and bregman divergences are both used for similarity assessment, they do not share the same fundamental properties. Metrics satisfy three basic properties: non-negativity: $d(x,y) \geq 0$; symmetry: $d(x,y) = d(y,x)$; and, perhaps most importantly, the triangle inequality: $d(x,z) \leq d(x,y) + d(y,z)$. Bregman divergences are nonnegative, however they do not satisfy the triangle inequality (in general) and can be asymmetric.

Bregman divergences do satisfy a variety of geometric properties, a couple of which we will need later. The bregman divergence $d_f(x,y)$ is convex in x, but not necessarily in y. Define the bregman ball of radius R around μ as

$$B(\mu, R) \equiv \{x \ : \ d_f(x,\mu) \leq R\}.$$

Since $d_f(x,\mu)$ is convex in x, $B(\mu,R)$ is a convex set.

Another interesting property concerns means. For a set of points, the mean under a bregman divergence is well defined and, interestingly, is independent of the choice of divergence:

$$\mu_X \equiv \text{argmin}_\mu \sum_{x \in X} d_f(x,\mu) = \frac{1}{|X|} \sum_{x \in X} x.$$

This fact can be used to extend k-means to the family of bregman divergences (Banerjee et al., 2005).

f. In particular, f is assumed to be *Legendre*.

Table 1. Some standard bregman divergences.

	$f(x)$	$d_f(x,y)$
ℓ_2^2	$\frac{1}{2}\|x\|_2^2$	$\frac{1}{2}\|x-y\|_2^2$
KL	$\sum x_i \log x_i$	$\sum x_i \log \frac{x_i}{y_i}$
Mahalanobis	$\frac{1}{2}x^\top Q x$	$\frac{1}{2}(x-y)^\top Q (x-y)$
Itakura-Saito	$-\sum \log x_i$	$\sum \left(\frac{x_i}{y_i} - \log \frac{x_i}{y_i} - 1 \right)$

2.2. NN Search

Because of the tremendous practical and theoretical importance of nearest neighbor search in machine learning, computational geometry, databases, and elsewhere, many retrieval schemes have been developed to reduce the computational cost of finding NNs.

KD-trees (Friedman et al., 1977) are one of the earliest and most popular data structures for NN retrieval. The data structure and accompanying search algorithm provide a blueprint for a huge body of future work (including the present one). The tree defines a hierarchical space partition where each node defines an axis-aligned rectangle. The search algorithm is a simple branch and bound exploration of the tree. Though KD-trees are useful in many applications, their performance has been widely observed to degrade badly with the dimensionality of the database.

Metric ball trees (Omohundro, 1989; Uhlmann, 1991; Yianilos, 1993; Moore, 2000) extend the basic methodology behind KD-trees to metric spaces by using metric balls in place of rectangles. The search algorithm uses the triangle inequality to prune out nodes. They seem to scale with dimensionality better than KD-trees (Moore, 2000), though high-dimensional data remains very challenging. Some high-dimensional datasets are intrinsically low-dimensional; various retrieval schemes have been developed that scale with a notion of intrinsic dimensionality (Beygelzimer et al., 2006).

In many applications, an exact NN is not required; something nearby is good enough. This is especially true in machine learning applications, where there is typically a lot of noise and uncertainty. Thus many researchers have switched to the problem of *approximate* NN search. This relaxation led to some significant breakthroughs, perhaps the most important being locality sensitive hashing (Datar et al., 2004). Spill trees (Liu et al., 2004) are another data structure for approximate NN search and have exhibited very strong performance empirically.

The present paper appears to be the first to describe a general method for efficiently finding bregman NNs; however, some related problems have been examined. (Nielsen et al., 2007) explores the geometric properties of bregman voronoi diagrams. Voronoi diagrams are of course closely related to NN search, but do not lead to an efficient NN data structure beyond dimension 2. (Guha et al., 2007) contains results on *sketching* bregman (and other) divergences. Sketching is related to dimensionality reduction, which is the basis for many NN schemes.

We are aware of only one NN speedup scheme for KL-divergences (Spellman & Vemuri, 2005). The results in this paper are quite limited: experiments were conducted on only one dataset and the speedup is less than 3x. Moreover, there appears to be a significant technical flaw in the derivation of their data structure. In particular, they cite the pythagorean theorem as an *equality* for projection onto an arbitrary convex set, whereas it is actually an *inequality*.

3. Bregman Ball Trees

This section describes the bregman ball tree data structure. The data structure and search algorithms follow the same basic program used in KD-trees and metric trees; in place of rectangular cells or metric balls, the fundamental geometric object is a bregman ball.

A bbtree defines a hierarchical space partition based on bregman balls. The data structure is a binary tree where each node i is associated with a subset of the database $X_i \subset X$. Node i additionally defines a bregman ball $B(\mu_i, R_i)$ with center μ_i and radius R_i such that $X_i \subset B(\mu_i, R_i)$. Interior (non-leaf) nodes of tree have two child nodes l and r. The database points belonging to node i are split between child l and r; each point in X_i appears in exactly one of X_l or X_r.[2] Though X_l and X_r are disjoint, the balls $B(\mu_l, R_l)$ and $B(\mu_r, R_r)$ may overlap. The root node of the tree encapsulates the entire database. Each leaf covers a small fraction of the database; the set of all leaves cover the entirety.

3.1. Searching

This subsection describes how to retrieve a query's nearest neighbor with a bbtree. Throughout, $X = \{x_1, \ldots, x_n\}$ is the database, q is a query, and $d_f(\cdot, \cdot)$ is a (fixed) bregman divergence. The point we are searching for is the *left* NN

$$x_q \equiv \text{argmin}_{x \in X} d_f(x, q).$$

Finding the *right* NN ($\text{argmin}_{x \in X} d_f(q, x)$) is considered in section 5.

Branch and bound search locates x_q in the bbtree. First, the tree is descended; at each node, the search algorithm chooses the child for which $d_f(\mu, q)$ is smallest and *ignores* the sibling node (temporarily). Upon arriving at a leaf node i, the algorithm calculates $d_f(x, q)$ for all $x \in X_i$. The closest point is the candidate NN; call it x_c. Now the algorithm must traverse back up the tree and consider the previously ignored siblings. An ignored sibling j must be explored if

$$d_f(x_c, q) > \min_{x \in B(\mu_j, R_j)} d(x, q). \quad (1)$$

The algorithm computes the right side of (1); we come back that in a moment. If (1) holds, then node j and all of its children can be ignored since the NN cannot be found in that subtree. Otherwise, the subtree rooted at j must be explored. This algorithm is easily adjusted to return the k-nearest neighbors.

The algorithm hinges on the computation of (1)—the *bregman projection onto a bregman ball*. In the ℓ_2^2 (or arbitrary metric) case, the projection can be computed analytically with the triangle inequality. Since general bregman divergences do not satisfy this inequality, we need a different way to compute—or at least bound—the right side of (1). Computing this projection is the main technical contribution of this paper, so we discuss it separately in section 4.

3.2. Approximate Search

As we mentioned in section 2.2, many practical applications do not require an exact NN. This is especially true in machine learning applications, where there is typically a lot of noise and even the representation of points used is heuristic (*e.g.* selecting an appropriate kernel for an SVM often involves guesswork). This flexibility is fortunate, since exact NN retrieval methods rarely work well on high-dimensional data.

Following (Liu et al., 2004), a simple way to speed up the retrieval time of the bbtree is to simply stop after only a few leaves have been examined. This idea originates from the empirical observation that metric and KD-trees often locate a point very close to the NN quickly, then spend most of the execution time backtracking. We show empirically that the quality of the NN degrades gracefully as the number of leaves examined decreases. Even when the search procedure is stopped very early, it returns a solution that is among the nearest neighbors.

[2] The disjointedness of the two point sets is not essential.

3.3. Building

The performance of the search algorithm depends on how many nodes can be pruned; the more, the better. Intuitively, the balls of two siblings should be well-separated and compact. If the balls are well-separated, a query is likely to be much closer to one than the other. If the balls are compact, then the distance from a query to a ball will be a good approximation to the distance from a query to the nearest point within the ball. Thus at each level, we'd like to divide the points into two well-separated sets, each of which is compact. A natural way to do this is to use k-means, which has already been extended to bregman divergences (Banerjee et al., 2005).

The build algorithm proceeds from top down. Starting at the top, the algorithm runs k-means to partition the points into two clusters. This process is repeated recursively. The total build time is $O(n \log n)$. Clustering from the bottom-up might yield better results, but the $O(n^2 \log n)$ build time is impractical for large datasets.

4. Computing the Bound

Recall that the search procedure needs to determine if the bound

$$d_f(x_c, q) > \min_{x \in B(\mu, R)} d_f(x, q) \tag{2}$$

holds, where x_c is the current candidate NN. We first show that the right side can be computed to accuracy ϵ in only $O(\log \frac{1}{\epsilon})$ steps with a simple bisection search. Since we only actually need upper and lower bounds on the quantity, we then present a procedure that augments the bisection search with primal and dual bounds so that it can stop early.

The right of (2) is a convex program:

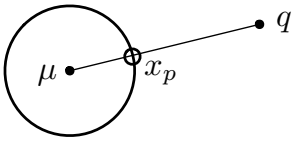
$$\begin{aligned} \min_{x} \quad & d_f(x, q) \\ \text{subject to:} \quad & d_f(x, \mu) \leq R. \end{aligned} \tag{P}$$

The search algorithm will need to solve (P) many times in the course of locating q's NN, so we need to be able to compute a solution very quickly.

Before considering the general case, let us pause to examine the ℓ_2^2 case. In this case, we can compute the projection x_p analytically:

$$x_p = \theta\mu + (1 - \theta)q,$$

where $\theta = \frac{\sqrt{2R}}{\|q - \mu\|}$.



What properties of this projection might extend to all of bregman divergences?

1. First, x_p lies on the line between q and μ; this drastically reduces the search space from a D-dimensional convex set to a one-dimensional line.
2. Second, x_p lies on the boundary of $B(\mu, R)$—*i.e* $d_f(x_p, \mu) = R$. Combined with property 1, this fact completely determines x_p: it is the point where the line between μ and q intersects the shell of $B(\mu, R)$.
3. Finally, since the ℓ_2^2 ball is spherically symmetric, we can compute this intersection analytically.

We prove that the first property is a special case of a fact that holds for all bregman divergences. Additionally, the second property generalizes to bregman divergences without change. The final property does not go through, so we will not be able to find a solution to (P) analytically.

Throughout, we use $q' \equiv \nabla f(q)$, $\mu' \equiv \nabla f(\mu)$, *etc.* to simplify notation. x_p denotes the optimal solution to (P).

Claim 2. *x_p' lies on the line between q' and μ'.*

Proof. The lagrange dual function of (P) is

$$\inf_x d_f(x, q) + \lambda(d_f(x, \mu) - R), \tag{3}$$

where $\lambda \geq 0$. Differentiating (3) with respect to x and setting it equal to 0, we get

$$\nabla f(x_p) - \nabla f(q) + \lambda \nabla f(x_p) - \lambda \nabla f(\mu) = 0.$$

We use the change of variable $\theta \equiv \frac{\lambda}{1+\lambda}$ and rearrange to arrive at

$$\nabla f(x_p) = \theta\mu' + (1 - \theta)q',$$

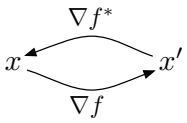
where $\theta \in [0, 1)$. □

Thus we see that property 1 of the ℓ_2^2 projection is a special case of a relationship between the gradients; it follows from claim 2 because $\nabla f(x) = x$ for the ℓ_2^2 divergence.

Since f is strictly convex, the gradient mapping is one-to-one. Moreover, the inverse mapping is given by the gradient of the *convex conjugate*, defined as

$$f^*(y) \equiv \sup_x \{\langle x, y\rangle - f(x)\}. \quad (4)$$

Symbolically:



Thus to solve (P), we can look for the optimal x' along $\theta\mu' + (1-\theta)q'$, and then apply ∇f^* to recover x_p.[3] To keep notation simple, we define

$$x'_\theta \equiv \theta\mu' + (1-\theta)q' \quad \text{and} \quad (5)$$
$$x_\theta \equiv \nabla f^*(x'_\theta). \quad (6)$$

Now onto the second property.

Claim 3. *$d_f(x_p, \mu) = R$—i.e. the projection lies on the boundary of $B(\mu, R)$.*

The claim follows from complementary slackness applied to (3). Claims 2 and 3 imply that finding the projection of q onto $B(\mu, R)$ is equivalent to

$$\begin{aligned} \text{find} \quad & \theta \\ \text{subject to:} \quad & d_f(x_\theta, \mu) = R \\ & \theta \in (0, 1] \\ & x_\theta = \nabla f^*(\theta\mu' + (1-\theta)q'). \end{aligned}$$

Fortunately, solving this program is simple.

Claim 4. *$d_f(x_\theta, \mu)$ is monotonic in θ.*

This claim follows from the convexity of f^*. Since $d_f(x_\theta, \mu)$ is monotonic, we can efficiently search for θ_p satisfying $d_f(x_{\theta_p}, \mu) = R$ using bisection search on θ. We summarize the result in the following theorem.

Theorem 5. *Suppose $\|\nabla^2 f^*\|_2$ is bounded around x'_p. Then a point x satisfying*

$$|d_f(x, q) - d_f(x_p, q)| \le \epsilon + O(\epsilon^2)$$

can be found in $O(\log 1/\epsilon)$ iterations. Each iteration requires one divergence evaluation and one gradient evaluation.

4.1. Stopping Early

Recall that the point of all this analysis is to evaluate whether

$$d_f(x_c, q) > \min_{x \in B(\mu,R)} d_f(x, q), \quad (7)$$

where x_c is the current candidate NN. If (7) holds, the node in question must be searched; otherwise it can be pruned. We can evaluate the right side of (7) exactly using the bisection method described previously, but an exact solution is not needed. Suppose we have bounds a and A satisfying

$$A \ge \min_{x \in B(\mu,R)} d_f(x, q) \ge a.$$

If $d_f(x_c, q) < a$, the node can be pruned; if $d_f(x_c, q) > A$, the node must be explored. We now describe upper and lower bounds that are computed at each step of the bisection search; the search proceeds until one of the two stopping conditions is met.

A lower bound is given by weak duality. The lagrange dual function is

$$\mathcal{L}(\theta) \equiv d_f(x_\theta, q) + \frac{\theta}{1-\theta}\Big(d_f(x_\theta, \mu) - R\Big). \quad (8)$$

By weak duality, for any $\theta \in [0, 1)$,

$$\mathcal{L}(\theta) \le \min_{x \in B(\mu,R)} d_f(x, q). \quad (9)$$

For the upper bound, we use the primal. At any θ satisfying $d_f(x_\theta, \mu) \le R$, we have

$$d_f(x_\theta, q) \ge \min_{x \in B(\mu,R)} d_f(x, q). \quad (10)$$

Let us now put all of the pieces together. We wish to evaluate whether (7) holds. The algorithm performs bisection search on θ, attempting to locate the θ satisfying $d_f(x_\theta, \mu) = R$. At step i the algorithm evaluates θ_i on two functions. First, it checks the lower bound bound given by the dual function $\mathcal{L}(\theta_i)$ defined in (8). If $\mathcal{L}(\theta_i) > d_f(x_c, q)$, then the node can be pruned. Otherwise, if $x_{\theta_i} \in B(\mu, R)$, we can update the upper bound. If $d_f(x_{\theta_i}, q) < d_f(x_c, q)$, then the node must be searched. Otherwise, neither bound holds, so the bisection search continues. See Algorithm 1 for pseudocode.

5. Left and Right NN

Since a bregman divergence can be asymmetric, it defines two NN problems:

- (lNN) return $\text{argmin}_{x \in X} d_f(x, q)$ and
- (rNN) return $\text{argmin}_{x \in X} d_f(q, x)$.

The bbtree data structure finds the left NN . We show that it can also be used to find the right NN.

[3] All of the base functions in table 1 have closed form conjugates.

Algorithm 1 CanPrune

Input: $\theta_l, \theta_r \in (0,1]$, $q, x_c, \mu \in \mathbb{R}^D$, $R \in \mathbb{R}$.
Set $\theta = \frac{\theta_l + \theta_r}{2}$.
Set $x_\theta = \nabla f^*(\theta\mu' + (1-\theta)q')$
if $\mathcal{L}(\theta) > d_f(x_c, q)$ **then**
  return YES
else if $x_\theta \in B(\mu, R)$ and $d_f(x_\theta, q) < d_f(x_c, q)$ **then**
  return NO
else if $d_f(x_\theta, \mu) > R$ **then**
  return CanPrune$(\theta_l, \theta, q, x_c, \mu)$
else if $d_f(x_\theta, \mu) < R$ **then**
  return CanPrune$(\theta, \theta_r, q, x_c, \mu)$
end if

Recall that the convex conjugate of f is defined as $f^*(y) \equiv \sup_x\{\langle x, y\rangle - f(x)\}$. The supremum is realized at a point x satisfying $\nabla f(x) = y$; thus

$$f^*(y') = \langle y, y'\rangle - f(y).$$

We use this identity to rewrite $d_f(\cdot,\cdot)$:

$$\begin{aligned} d_f(x,y) &= f(x) - f(y) - \langle y', x - y\rangle \\ &= f(x) + f^*(y') - \langle y', x\rangle \\ &= d_{f^*}(y', x'). \end{aligned}$$

This relationship provides a simple prescription for adapting the bbtree to the rNN problem: build a bbtree for the divergence d_{f^*} and the database $X' \equiv \{\nabla f(x_1), \ldots, \nabla f(x_n)\}$. On query q, $q' \equiv \nabla f(q)$ is computed and the bbtree finds $x' \in X'$ minimizing $d_{f^*}(x', q')$. The point x whose gradient is x' is then the rNN to q.

6. Experiments

We examine the performance benefit of using bbtrees for approximate and exact NN search. All experiments were conducted with a simple C implementation that is available from the author's website.

The results are for the KL-divergence. We chose to evaluate the bbtree for the KL-divergence because it is used widely in machine learning, text mining, and computer vision; moreover, very little is known about efficient NN retrieval for it. In contrast, there has been a tremendous amount of work for speeding up the ℓ_2^2 and Mahalanobis divergences—they both may be handled by standard metric trees and many other methods. Other bregman divergences appear much less often in applications. Still, examining the practical performance of bbtrees for these other bregman divergences is an interesting direction for future work.

We ran experiments on several challenging datasets.

- **rcv-D**. We used latent dirichlet allocation (LDA) (Blei et al., 2003) to generate topic histograms for 500k documents in the rcv1 corpus (Lewis et al., 2004). These histograms were generated by building a LDA model on a training set and then performing inference on 500k documents to generate their posterior dirichlet parameters. Suitably scaled, these parameters give a representation of the documents in the topic simplex (Blei et al., 2003). We generated data using this process for $D = 8, 16, \ldots, 256$ topics.
- **Corel histograms**. This dataset contains 60k color histograms generated from the Corel image dataset. Each histogram is 64-dimensional.
- **Semantic space**. This dataset is a 371-dimensional representation of 5000 images from the Corel Stock photo collection. Each image is represented as a distribution over 371 description keywords (Rasiwasia et al., 2007).
- **SIFT signatures**. This dataset contains 1111-dimensional representations of 10k images from the PASCAL 2007 dataset (Everingham et al., 2007). Each point is a histogram of quantized SIFT features as suggested in (Nowak et al., 2006).

Notice that most of these datasets are fairly high-dimensional.

We are mostly interested in approximate NN retrieval, since that is likely sufficient for machine learning applications. If the bbtree is stopped early, it is not guaranteed to return an exact NN, so we need a way to evaluate the quality of the point it returns. One natural evaluation metric is this: How many points from the database are closer to the query than the returned point? Call this value NC for "number closer". If NC is small compared to the size of the database, say 10 versus 100k, then it will likely share many properties with the true NN (*e.g.* class label).[4]

The results are shown in figure 2. These are strong results; it is shown that the bbtree is often orders of magnitude faster than brute-force search without a substantial degradation of quality. More analysis appears in the caption.

[4] A different evaluation criteria is the approximation ratio ϵ satisfying $d_f(x,q) \leq (1+\epsilon)d_f(x_q, q)$, where x_q is q's true NN. We did not use this measure because it is difficult to interpret. For example, suppose we find $\epsilon = .3$ approximate NNs from two different databases A and B. It could easily be the case that *all* points in A are 1.3-approximate NNs, whereas only the exact NN in database B is 1.3-approximate.

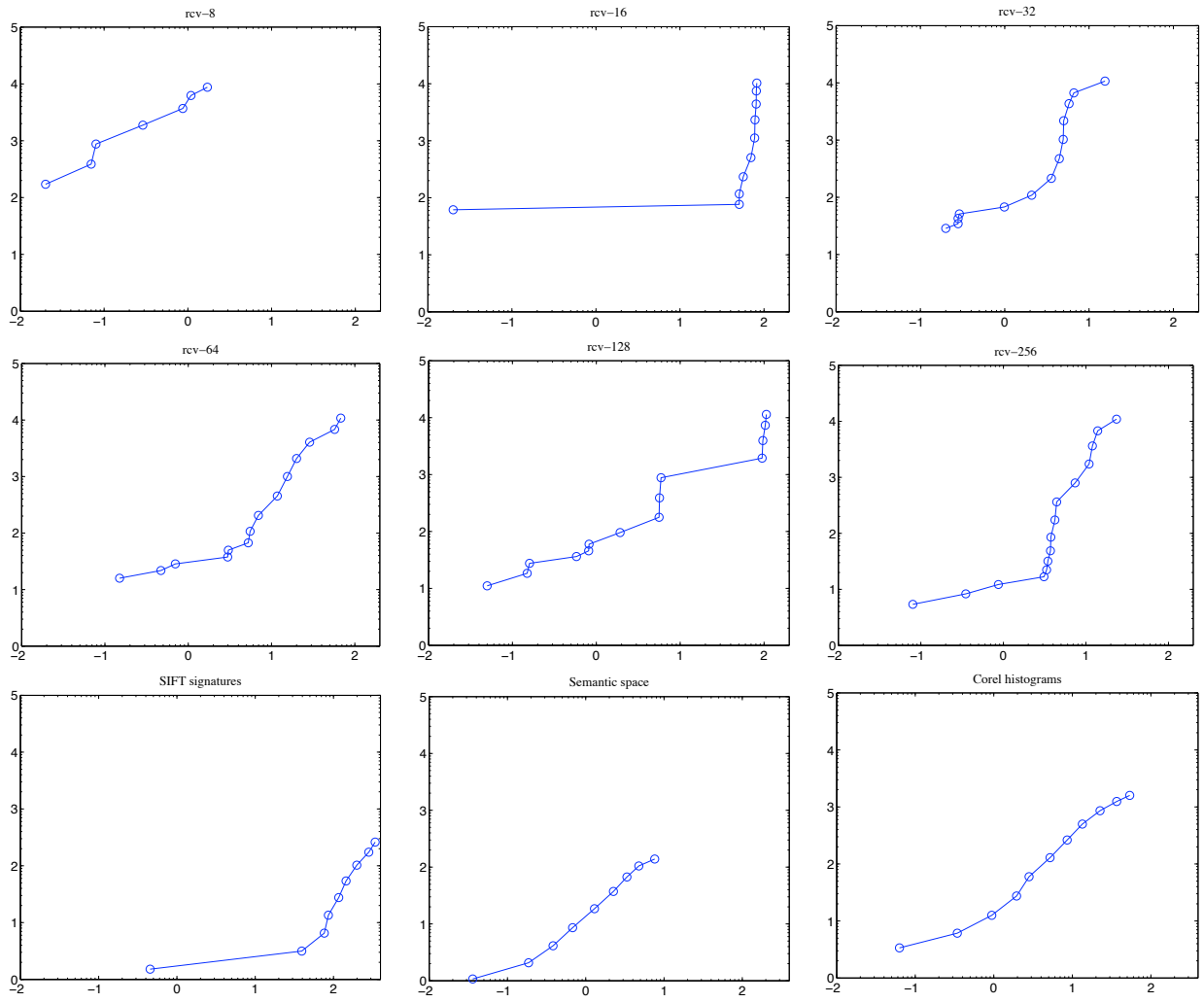


Figure 2. Log-log plots (base 10): y-axis is the exponent of the speedup over brute force search, x-axis is the exponent of the number of database points closer to the query than the reported NN. The y-axis ranges from 10^0 (no speedup) to 10^5. The x-axis ranges from 10^{-2} to 10^2. All results are averages over queries not in the database.

Consider the plot for rcv-128 (center). At $x = 10^0$, the bbtree is returning one of the two nearest neighbors (on average) out of 500k points at a 100x speedup over brute force search. At $x = 10^1$, the bbtree is returning one of the eleven nearest neighbors (again, out of 500k points) and yields three orders of magnitude speedup over brute force search.

The best results are achieved on the rcv-D datasets and the Corel histogram dataset. The improvements are less pronounced for the SIFT signature and Semantic space data, which may be a result of both the high dimensionality and small size of these two datasets. Even so, we are getting useful speedups on the semantic space dataset (10-100x speedup with small error). For the SIFT signatures, we are getting a 10x speedup while receiving NNs in the top 1%.

Table 2. Exact search

dataset	dimensionality	speedup
rcv-8	8	64.5
rcv-16	16	36.7
rcv-32	32	21.9
rcv-64	64	12.0
corel histograms	64	2.4
rcv-128	128	5.3
rcv-256	256	3.3
semantic space	371	1.0
SIFT signatures	1111	0.9

Finally, we consider exact NN retrieval. It is well known that finding a (guaranteed) exact NN in moderate to high-dimensional databases is very challenging. In particular, metric trees, KD-trees, and relatives typically afford a reasonable speedup in moderate dimensions, but the speedup diminishes with increasing dimensionality (Moore, 2000; Liu et al., 2004). When used for exact search, the bbtree reflects this basic pattern. Table 2 shows the results. The bbtree provides a substantial speedup on the moderate-dimensional databases (up through $D = 256$), but no speedup on the two databases of highest dimensionality.

7. Conclusion

In this paper, we introduced bregman ball trees and demonstrated their efficacy in NN search. The experiments demonstrated that bbtrees can speed up approximate NN retrieval for the KL-divergence by orders of magnitude over brute force search. There are many possible directions for future research. On the practical side, which ideas behind the many variants of metric trees might be useful for bbtrees? On the theoretical side, what is a good notion of intrinsic dimensionality for bregman divergences and can a practical data structure be designed around it?

Acknowledgements

Thanks to Serge Belongie, Sanjoy Dasgupta, Charles Elkan, Carolina Galleguillos, Daniel Hsu, Nikhil Rasiwasia, and Lawrence Saul. Support was provided by the NSF under grants IIS-0347646 and IIS-0713540.

References

Banerjee, A., Merugu, S., Dhillon, I. S., & Ghosh, J. (2005). Clustering with bregman divergences. *JMLR*.

Beygelzimer, A., Kakade, S., & Langford, J. (2006). Cover trees for nearest neighbor. *ICML*.

Blei, D., Ng, A., & Jordan, M. (2003). Latent dirichlet allocation. *JMLR*.

Bregman, L. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, *7*, 200–217.

Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. *SCG 2004*.

Everingham, M., Gool, L. V., Williams, C. K., Winn, J., & Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 Results.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, *3(3)*, 209–226.

Gray, R. M., Buzo, A., Gray, A. H., & Matsuyama, Y. (1980). Distortion measures for speech processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*.

Guha, S., Indyk, P., & McGregor, A. (2007). Sketching information divergences. *COLT*.

Lewis, D. D., Yang, Y., Rose, T., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *JMLR*.

Liu, T., Moore, A. W., Gray, A., & Yang, K. (2004). An investigation of practical approximate neighbor algorithms. *NIPS*.

Moore, A. W. (2000). Using the triangle inequality to survive high-dimensional data. *UAI*.

Nielsen, F., Boissonnat, J.-D., & Nock, R. (2007). On bregman voronoi diagrams. *SODA* (pp. 746–755).

Nowak, E., Jurie, F., & Triggs, B. (2006). Sampling strategies for bag-of-features image classification. *ECCV*.

Omohundro, S. (1989). *Five balltree construction algorithms* (Technical Report). ICSI.

Pereira, F., Tishby, N., & Lee, L. (1993). Distributional clustering of English words. *31st Annual Meeting of the ACL* (pp. 183–190).

Puzicha, J., Buhmann, J., Rubner, Y., & Tomasi, C. (1999). Empirical evaluation of dissimilarity measures for color and texture. *ICCV*.

Rasiwasia, N., Moreno, P., & Vasconcelos, N. (2007). Bridging the gap: query by semantic example. *IEEE Transactions on Multimedia*.

Spellman, E., & Vemuri, B. (2005). Efficient shape indexing using an information theoretic representation. *International Conference on Image and Video Retrieval*.

Uhlmann, J. K. (1991). Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, *40*, 175–179.

Weinberger, K., Blitzer, J., & Saul, L. (2006). Distance metric learning for large margin nearest neighbor classification. *NIPS*.

Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. *SODA*.

Nearest Hyperdisk Methods for High-Dimensional Classification

Hakan Cevikalp HAKAN.CEVIKALP@GMAIL.COM
Eskisehir Osmangazi University, Eskisehir, Turkey

Bill Triggs BILL.TRIGGS@IMAG.FR
Laboratoire Jean Kuntzmann, Grenoble, France

Robi Polikar POLIKAR@ROWAN.EDU
Rowan University, Glassboro, NJ USA

Abstract

In high-dimensional classification problems it is infeasible to include enough training samples to cover the class regions densely. Irregularities in the resulting sparse sample distributions cause local classifiers such as Nearest Neighbors (NN) and kernel methods to have irregular decision boundaries. One solution is to "fill in the holes" by building a convex model of the region spanned by the training samples of each class and classifying examples based on their distances to these approximate models. Methods of this kind based on affine and convex hulls and bounding hyperspheres have already been studied. Here we propose a method based on the *bounding hyperdisk* of each class – the intersection of the affine hull and the smallest bounding hypersphere of its training samples. We argue that in many cases hyperdisks are preferable to affine and convex hulls and hyperspheres: they bound the classes more tightly than affine hulls or hyperspheres while avoiding much of the sample overfitting and computational complexity that is inherent in high-dimensional convex hulls. We show that the hyperdisk method can be kernelized to provide nonlinear classifiers based on non-Euclidean distance metrics. Experiments on several classification problems show promising results.

1. Introduction

Nearest neighbours (NN) – assigning the query to the class with the nearest training sample(s) under some suitable distance metric – is one of the simplest methods for multi-class classification. Asymptotically it makes at most twice as many errors as the optimal Bayes rule classifier, but this result assumes dense sampling which requires training sets that are exponentially large in the dimensionality of the underlying feature space class distributions. In high-dimensional problems such as text, gene or visual object classification, tractable training sets are necessarily much smaller than this, and the performance of NN can often be poor. The main problem is the sparse and irregular distribution of the training samples, which often leaves "holes" in the input space – regions that have few or no nearby training samples from the relevant class. Equivalently, local density estimates in high dimensions are intrinsically noisy because any region with a radius significantly smaller than that of the class has such a small volume relative to that of the class that it typically contains few or no samples. These effects make the inter-class decision boundaries of high dimensional NN and local kernel based methods erratic, thus leading to classification errors.

One way to circumvent this problem is to approximate each class with a point set that "fills in the holes" between the examples. In particular, any convex set containing the examples has this property. Several approximations of this kind have already been studied including the affine hulls, convex hulls, bounding hyperspheres and bounding hyperellipsoids of the examples (Gulmezoglu et al., 2001, Laaksonen, 1997, Nalbantov et al., 2007, Vincent & Bengio, 2001). Despite the simplicity of their geometry, such approximations are useful in high dimensions because in any case fine local details can not be resolved with practical numbers of samples. Queries are classified to the class whose convex approximation is closest to the query point – a convex nearest-point problem that can be solved reasonably efficiently with standard methods. This is equivalent to NN in which additional points are fantasized to fill in the set of each class.

Affine hulls (*i.e.* spanning linear subspaces that have

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

been shifted to pass through the centroid of the class) were first used for global classifiers of isolated words and hand-written digits in (Gulmezoglu et al., 2001, Laaksonen, 1997), giving good classification performance. Similarly, (Nalbantov et al., 2007) used convex hulls for global classifiers on some of the UCI and SlatLog problems, comparing these to Support Vector Machines (SVMs) both theoretically and empirically. Such global convex approximations may fail to capture the decision boundaries of classes with nonlinear boundaries and one can also build more local approximations, or even build a separate approximation for each query sample based on convex approximations of its k nearest neighbours from each class. Again the query is classified to the (locally) nearest hull. Although this is not immune to the hole problem, (Vincent & Bengio, 2001) reported significant improvements over traditional NN for affine and convex hull methods of this kind in handwritten digit classification. Another way to handle complex boundaries is via nonlinear mapping to a high-dimensional feature space (*e.g*. via a kernel) followed by a global convex set approximation of the kind described below.

Besides classification, approximations based on affine or convex hulls have also been used for dimensionality reduction. Mixtures of Principal Component Analyzers can be used to approximate nonlinear data manifolds under local linearity assumptions (Hinton et al., 1997). Locally Linear Embedding (Roweis & Saul, 2000) approximates the nonlinear structure of high-dimensional data by exploiting local affine/convex reconstructions. (Verbeek, 2006) combined several locally valid linear manifolds to obtain a global nonlinear mapping between the high-dimensional sample space and a low-dimensional manifold. In (Cevikalp et al., 2008), we proposed a margin based discriminative dimensionality reduction method based on convex models of classes.

The current paper presents a new convex approximation based classifier that models each class with its bounding hyperdisk – the intersection of the affine hull and the minimal bounding hypersphere of its training examples. Hyperdisks are attractive primitives because they maintain the stability of the affine hull and hypersphere methods while providing better localization of the training samples and hence potentially better discrimination. Convex hull approximations tend to be unrealistically tight (for practical training set sizes, classes typically extend considerably beyond the convex hull of the training samples) while affine hull and hypersphere ones tend to be too loose in complementary senses (one too "broad", the other too "deep"). The hyperdisk approach to some extent captures the best aspects of each method. It can be applied both globally and locally and it is simple enough to be expressible in terms of dot products and hence to allow kernelization.

The paper is organized as follows. In section 2 we recall the affine and convex hull based methods. Section 3 introduces the hyperdisk method. Section 4 describes our experiments and data sets. Finally, section 5 presents conclusions and future directions.

2. Background on Related Methods

2.1. Nearest Affine Hull (NAH) Classification

Let the training samples be $\mathbf{x}_{ci} \in \mathbb{R}^d$, where $c = 1, \ldots, C$ indexes the C classes and $i = 1, \ldots, N_c$ indexes the N_c samples of class c. We suppose that the affine hull of the samples from each class is a proper subset of $\mathbb{R}^d$ of dimension less than d (which certainly holds when $N_c \ll d$). The affine hull is the affine span of the training samples, *i.e*. the smallest affine subspace containing them

$$H_c^{\text{aff}} = \left\{ \mathbf{x} = \textstyle\sum_{i=1}^{N_c} \alpha_i \, \mathbf{x}_{ci} \;\middle|\; \sum_i \alpha_i = 1 \right\}. \qquad (1)$$

The affine hull gives a rather loose approximation to the class region because it does not constrain the position of the training points within the affine subspace. The distance from a query point $\mathbf{x}_q$ to an affine hull H_c^{aff} is the norm of the displacement from $\mathbf{x}_q$ to the closest point on the hull, which can be expressed as the orthogonal projection of $\mathbf{x}_q$ normal to the subspace (see, *e.g*., (Cevikalp et al., 2007) for derivations):

$$d(\mathbf{x}_q, H_c^{\text{aff}}) = \|(\mathbf{I} - \mathbf{P}_c)(\mathbf{x}_q - \boldsymbol{\mu}_c)\| = \|\mathbf{P}_c^{\perp} \mathbf{x}_q - \boldsymbol{\mu}_c^{\perp}\|. \quad (2)$$

Here: $\mathbf{I}$ is the identity matrix, $\mathbf{P}_c$ is the orthogonal projection onto the spanning subspace (the range of the covariance matrix) of the class-c training samples, and $\mathbf{P}_c^{\perp} = \mathbf{I} - \mathbf{P}_c$ is the orthogonal projection onto the null space of the covariance – *i.e*. the orthogonal complement of the spanning subspace, called the *indifference subspace* in (Gulmezoglu et al., 2001, Cevikalp et al., 2005). $\boldsymbol{\mu}_c$ can be any reference point in H_c^{aff} – *e.g*. one of the samples $\mathbf{x}_{ci}$, or their mean – and $\boldsymbol{\mu}_c^{\perp} = \mathbf{P}_c^{\perp} \boldsymbol{\mu}_c$, the residual of $\boldsymbol{\mu}_c$ under the projection, encodes the orthogonal displacement of H_c^{aff} from the origin.

As its name suggests, the NAH classifier assigns the query to the class whose affine hull is the closest:

$$g(\mathbf{x}_q) = \min_{c=1,\ldots,C} (d(\mathbf{x}_q, H_c^{\text{aff}})). \qquad (3)$$

Equivalently, NAH chooses the class that provides the best (smallest $\|\text{error}\|$) reconstruction of the query using an affine combination of training samples. The decision boundaries of NAH are piecewise quadratic. Numerically, point projections can be computed on the fly without explicitly evaluating and storing the $d \times d$ projection matrices $\mathbf{P}_c$ and $\mathbf{P}_c^{\perp}$ by using $\mathbf{P}_c = \mathbf{Q}_c \, \mathbf{Q}_c^{\top}$ where $\mathbf{Q}_c$ is the U matrix of the thin SVD (or equivalently the Q matrix of the

thin QR decomposition) of the matrix of centred class-c training examples $[\mathbf{x}_{c1} - \boldsymbol{\mu}_c, \ldots, \mathbf{x}_{cN_c} - \boldsymbol{\mu}_c]$.

In practice the training data is often somewhat noisy. This can harm the classification performance owing to the inclusion of spurious 'noise' dimensions in the affine hulls. To reduce this we suppress dimensions of the SVD (and hence of $\mathbf{Q}_c$) that correspond to overly small singular values.

For nonlinear classes that lie on smooth manifolds, NAH can also be applied locally by finding the k-nearest samples to the query from each class, building local affine hulls using these nearest neighbors, and assigning the query to the class with the closest hull (Vincent & Bengio, 2001). This can reproduce complex nonlinear decision boundaries.

2.2. Nearest Convex Hull (NCH) Classification

The affine hull gives a rather loose approximation to the class region. Alternatively, we can take a maximally tight bound by approximating the class with the convex hull of its training samples. For this, we include non-negativity constraints $\alpha_i \geq 0$, $i = 1, \ldots, N_c$ in (1) and replace all of the affine hull distance computations with convex hull ones. The distance from a query $\mathbf{x}_q$ to the convex hull of class c is the norm of the displacement from $\mathbf{x}_q$ to the closest point on the hull. This reduces to solving the following quadratic programming problem

$$\begin{aligned} &\min_{\boldsymbol{\alpha}_c} \ \frac{1}{2}||\mathbf{x}_q - \mathbf{X}_c\,\boldsymbol{\alpha}_c||^2 \\ &s.t. \ \sum_{i=1}^{N_c} \alpha_{ci} = 1, \ \ \alpha_{ci} \geq 0, \ \ i = 1, \ldots, N_c, \end{aligned} \tag{4}$$

where $\mathbf{X}_c$ is a matrix whose columns are the class-c training samples. Given the optimal α^*_{ci} coefficients, the distance from $\mathbf{x}_q$ to the convex hull of the class c is $||\mathbf{x}_q - \mathbf{X}_c\,\boldsymbol{\alpha}^*_c||$. This is repeated for each class and the query is assigned to the class with the closest convex hull.

Finding the maximum margin between two classes is equivalent to finding the closest points on their convex hulls (Bennett & Bredensteiner, 2000) so convex distances can also be computed by using a classical hard-margin SVM algorithm to find the margin (convex distance) separating each class from the given query point.

NAH and NCH are "one class" methods in the sense that we do not explicitly calculate the decision boundaries during the training phase. Instead they remain implicit and the decisions are made on-line for each test sample. However both approaches can be viewed as large margin classifiers closely related to hard-margin linear SVM's. In particular, the piecewise linear/quadratic decision boundary of NCH contains the SVM boundary as one facet, and generalizes it to use distance to the convex hull rather than linear separation as the decision criterion.

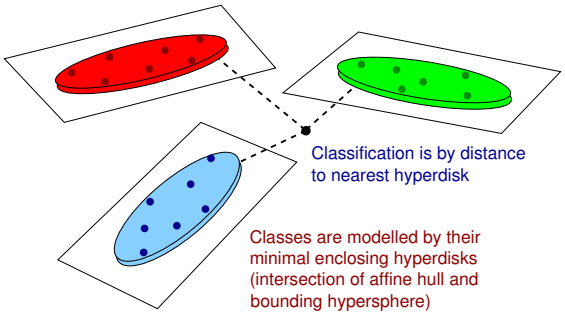


Figure 1. The principle of the proposed nearest bounding hyperdisk method. Classes are modelled by the bounding hyperdisk of their training examples and new examples are classified to the class with the closest hyperdisk.

3. Nearest Bounding Hyperdisk (NHD) Classification

In high-dimensional spaces, classes often extend well beyond the convex hulls of their training samples. For example, any individual simplex spanned by points sampled from a high-dimensional hypersphere can include only a negligible fraction of the volume of the sphere even if the vertices themselves are well spaced and close to the surface of the sphere. Conversely, affine hulls often give a rather loose approximation to the class as they do not constrain the positions of the training points within the affine subspace. This is problematic if the classes have similar or intersecting affine hulls but very different distributions of samples within their hulls. In such cases the classification performance will be poor if the affine projections of the queries onto the affine hulls are too far from training samples (*e.g.* as indicated by large values of the α_i coefficients for the constructed affine projections). The "soft margin" approach to handling this is to allow negative weights in (4) but to penalize over-large values by including upper and lower bounds in the quadratic program. However this deteriorates the run-time efficiency of NAH because the affine hull parameters of classes can no longer be computed in advance.

Instead, we can keep both a simpler geometric interpretation and good run-time efficiency by approximating the class samples with their bounding hyperdisk, *i.e.* the intersection of their affine hull and their minimal bounding hypersphere.

3.1. Global Nearest Hyperdisk Method

We will only describe the basic global Nearest Hyperdisk (NHD) classifier, but local application is also possible in the same way as for NAH and NCH. NHD approximates each class with the smallest bounding hyperdisk of its training samples – the set formed by inter-

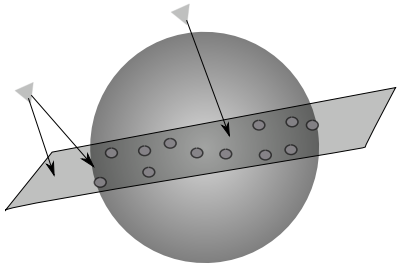


Figure 2. Computing distances from queries to a hyperdisk. The affine projection of the query on the left lies outside the hypersphere so it needs to be projected along the affine hull onto the hypersphere before the query-hyperdisk distance can be calculated. The affine projection of the query at the top already lies within the hypersphere so no adjustment is necessary.

secting their affine hull and their smallest bounding hypersphere. Such hyperdisks can be computed economically and they support rapid nearest point computations. There are already a number of methods based on affine hulls or bounding hyperspheres – for example hyperspheres have been used for outlier detection (Tax & Duin, 2004, Shawe-Taylor & Cristianini, 2004) and binary classification (Wang et al., 2005) – but we are not aware of any previous machine learning method based on hyperdisks. The bounding hypersphere of class c is characterized by its center $\mathbf{s}_c$ and radius r_c. These can be found by solving the following quadratic program

$$\begin{aligned} &\min_{\gamma, r_c, \xi} \; r_c^2 + \gamma \sum_{i=1}^{N_c} \xi_i \\ &\quad s.t. \;\; ||\mathbf{x}_{ci} - \mathbf{s}_c||^2 \leq r_c^2 + \xi_i, \;\; i = 1, \ldots, N_c, \end{aligned} \tag{5}$$

or its dual

$$\begin{aligned} &\min_{\alpha} \;\; \sum_{i,j} \alpha_i \alpha_j \langle \mathbf{x}_{ci}, \mathbf{x}_{cj} \rangle - \sum_i \alpha_i \langle \mathbf{x}_{ci}, \mathbf{x}_{ci} \rangle \\ &s.t. \;\; \sum_{i=1}^{N_c} \alpha_i = 1, \;\; 0 \leq \alpha_i \leq \gamma, \;\; i = 1, \ldots, N_c. \end{aligned} \tag{6}$$

Here α_i are Lagrange multipliers and $\gamma \in [0, 1]$ is a ceiling parameter that can be set to a finite value to eliminate overdistant points as outliers. Given the solution, the center of the hypersphere is $\mathbf{s}_c = \sum_{i=1}^{N_c} \alpha_i \mathbf{x}_{ci}$ and the radius is $r_c = ||\mathbf{x}_{ci} - \mathbf{s}_c||$ for any $\mathbf{x}_{ci}$ with $0 < \alpha_i < \gamma$.

To compute the distance from a query to the hyperdisk of a class, we find the affine projection of the query onto the affine hull by $\mathbf{x}_q^{\text{aff}} = \mathbf{P}_c(\mathbf{x}_q - \boldsymbol{\mu}_c) + \boldsymbol{\mu}_c = \mathbf{P}_c \mathbf{x}_q + \boldsymbol{\mu}_c^{\perp}$. If the projection lies outside the bounding hypersphere we move it along the line joining it to the center of the sphere until it touches the sphere. The distance from the query to the disk is the distance from it to the (possibly moved) projection – see fig. 2. Formally, the distance is

$$d(\mathbf{x}_q, H_c^{\text{disk}}) = \sqrt{\max(\|\mathbf{x}_q^{\text{aff}} - \mathbf{s}_c\| - r_c, 0)^2 + \|\mathbf{x}_q - \mathbf{x}_q^{\text{aff}}\|^2}.$$

3.2. Kernelization of the Hyperdisk Method

We now show that the hyperdisk method can be kernelized, allowing it to be used in implicit high dimensional feature spaces induced by Mercer kernels. This brings all of the usual advantages and disadvantages of kernelization, notably scope for a richer choice of distance functions and highly nonlinear decision boundaries that can aid data separability in return for the need to work with an implicit model defined by a large set of training samples.

The kernel trick can be used to map the data into an implicit feature space as in Kernel PCA (Schölkopf et al., 1998). Let $\boldsymbol{\phi}(\cdot)$ be the implicit feature space embedding and $k(\mathbf{x}, \mathbf{y}) = \boldsymbol{\phi}^\top(\mathbf{x})\boldsymbol{\phi}(\mathbf{y})$ be the corresponding kernel function. Suppose that we want to project a sample $\mathbf{x}$ onto the affine hull of a given set of samples $\{\mathbf{x}_i | i = 1, \ldots, m\}$. Let $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_m)]$ be their feature space embedding matrix, $\mathbf{K} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} = [k(\mathbf{x}_i, \mathbf{x}_j)]$ be their $m \times m$ kernel matrix and $\mathbf{k}_{\mathbf{x}} = \boldsymbol{\Phi}^\top \boldsymbol{\phi}(\mathbf{x}) = [k(\mathbf{x}_i, \mathbf{x})]$ be the $m \times 1$ kernel vector of $\mathbf{x}$ against the samples. The feature space mean of the samples is $\boldsymbol{\mu} = \frac{1}{m} \boldsymbol{\Phi}\, \mathbf{1}_m$ where $\mathbf{1}_m$ is an m-vector of 1's. The explicit approach detailed below (3) is based on the thin SVD $\mathbf{U}\mathbf{D}\mathbf{V}^\top$ of the matrix of centered sample features $[\boldsymbol{\phi}(\mathbf{x}_1) - \boldsymbol{\mu}, \ldots, \boldsymbol{\phi}(\mathbf{x}_m) - \boldsymbol{\mu}] = \boldsymbol{\Phi}\,\boldsymbol{\Pi}$, where $\boldsymbol{\Pi} = \mathbf{I} - \frac{1}{m} \mathbf{1}_m \mathbf{1}_m^\top$ is an orthogonal projection in sample space that implements subtraction of the mean on $\boldsymbol{\Phi}$. Given this, the projection of $\mathbf{x}$ onto the affine hull of the mapped samples is then $\mathbf{U}\,\mathbf{U}^\top(\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\mu}) + \boldsymbol{\mu}$, and the squared residual of this projection is $\|\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\mu}\|^2 - \|\mathbf{U}^\top(\boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\mu})\|^2$. Also, we are free to use any origin and linear basis that we choose for computations within the affine hull so long as we do so consistently. In particular, if we choose the orthogonal basis given by $\mathbf{U}$ centred at $\boldsymbol{\mu}^\perp = (\mathbf{I} - \mathbf{U}\mathbf{U}^\top)\boldsymbol{\mu}$, the projection of $\mathbf{x}$ onto the affine hull is represented simply by $\mathbf{U}^\top \boldsymbol{\phi}(\mathbf{x})$.

Noting that the $\mathbf{D}$ matrices of thin SVDs (*i.e.* taking only the significantly non-zero singular values) are invertible, we have $\mathbf{U} = \boldsymbol{\Phi}\,\boldsymbol{\Pi}\,\mathbf{V}\,\mathbf{D}^{-1} = \boldsymbol{\Phi}\,\mathbf{A}^\top$ where $\mathbf{A} = \mathbf{D}^{-1}\mathbf{V}^\top\boldsymbol{\Pi}$. In the kernelized case we can not evaluate the SVD of $\boldsymbol{\Phi}\,\boldsymbol{\Pi}$ explicitly because this would require numerical computations in feature space, but we can work implicitly in sample-space in terms of the eigendecomposition $\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ of the centred kernel matrix $\tilde{\mathbf{K}} = (\boldsymbol{\Phi}\,\boldsymbol{\Pi})^\top(\boldsymbol{\Phi}\,\boldsymbol{\Pi}) = \boldsymbol{\Pi}\,\mathbf{K}\,\boldsymbol{\Pi}$. Here, $\mathbf{V}$ is the same matrix as in the SVD of $\boldsymbol{\Phi}\,\boldsymbol{\Pi}$ and $\boldsymbol{\Lambda} = \mathbf{D}^2$ so that $\mathbf{A} = \boldsymbol{\Lambda}^{-1/2}\,\mathbf{V}^\top\,\boldsymbol{\Pi}$.

Putting all of these pieces together and noting that $\|\boldsymbol{\phi}(\mathbf{x})\|^2 = k(\mathbf{x}, \mathbf{x})$, we find that the squared residual error of the projection of $\mathbf{x}$ onto the affine hull of the examples

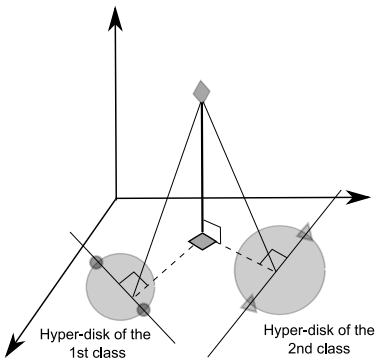


Figure 3. The kernelized NAH/NCH/NHD classifiers are based on distances between query samples and affine, *etc*., hulls of classes within the subspace spanned by the complete training data. These distances produce the same class assignments as the original classifiers.

is

$$k(\mathbf{x},\mathbf{x}) - \mathbf{k}_\mathbf{x}^\top \mathbf{A}^\top \mathbf{A} \mathbf{k}_\mathbf{x} - (2\,\mathbf{k}_\mathbf{x} - \mathbf{K}\tfrac{\mathbf{1}_m}{m})^\top (\mathbf{I} - \mathbf{A}^\top \mathbf{A} \mathbf{K}) \tfrac{\mathbf{1}_m}{m} \quad (7)$$

and the sample-space representative of the feature-space projection of $\mathbf{x}$ into the affine hull of the samples is simply $\mathbf{A}\,\mathbf{k}_\mathbf{x}$. We can use the representation vectors $\mathbf{A}\,\mathbf{k}_\mathbf{x}$ for any affine computation within the feature space affine hull, including calculations of hyperspheres and convex hulls, projections of new samples onto these, and within-hull distance computations. To calculate the overall squared distance from the example to the desired convex set within the hull, the squared residual error of the projection onto the hull (7) needs to be added to the squared within-hull distance.

In retrospect the obvious way to perform the above computations would be to use a separate feature subspace ($\mathbf{\Phi}$, $\mathbf{K}$, $\mathbf{k}_\mathbf{x}$, $\mathbf{A}$, *etc*.) for each class, but in the experiments below we actually worked in a global feature subspace based on the combined training samples of all classes. This subspace contains the affine hulls of all of the classes so the projections of test samples onto classes can be done in two stages, first projecting the sample onto the global affine hull, then projecting the result onto the class hull within the global one. The first projection is class-independent so it simply adds a sample-dependent constant residual to all of the sample-class distances. For decisions based on relative sample-class distances, these constants can be ignored. As a result, it suffices to perform all computations with the global $\mathbf{A}\,\mathbf{k}_\mathbf{x}$ vectors as though they were the original affine input points. In particular, the kernelized versions of NAH, NHD and NCH simply apply the corresponding linear method to the $\mathbf{A}\,\mathbf{k}_\mathbf{x}$ vectors of the global feature subspace. This process is illustrated in fig. 3. It only provides relative distances, so $k(\mathbf{x},\mathbf{x})$ is never needed.

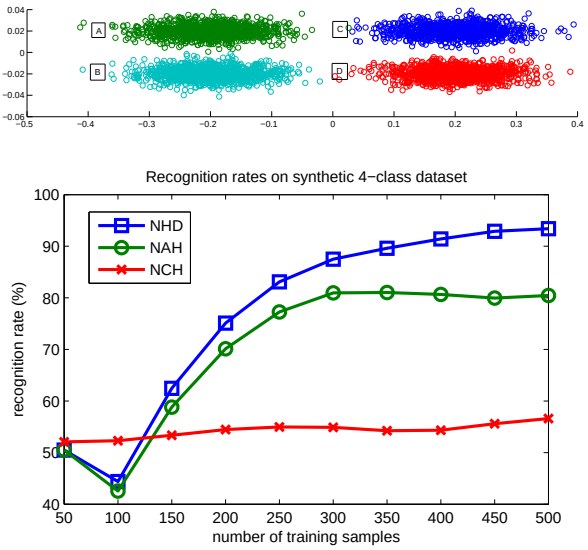


Figure 4. Top: the first two dimensions of the four disk dataset. Bottom: overall test-set recognition rates for NHD, NAH and NCH on this dataset for varying numbers of training points.

4. Experiments

We compared the proposed hyperdisk method (NHD) to Nearest Neighbour (NN), Nearest Affine Hull (NAH), Nearest Convex Hull (NCH) and Nearest Sphere Center[1] (NSC) classifiers in two regimes: high-dimensional problems where the dimensionality of the input space is much larger than the size of the training sets and the native (unkernelized) classifier is used; and low-dimensional problems where the training sets are larger than the dimensionality of the input space and a kernelized classifier is needed. We tested the methods on three tasks from multi-class visual recognition in the high-dimensional regime, and on five tasks from the UCI collection in the low-dimensional one. In each case, we optimized the algorithm parameters using global coarse-to-fine search, with random partitions of the training data into training and validation sets.

4.1. Experiments on Synthetic Data

Before starting, we illustrate some properties of the methods on a simple synthetic data set with four classes. This was produced by creating four unit-radius spheres (one for each class) in 300 dimensions with centres $(\pm 0.2, \pm 0.2, 0, \ldots, 0)$, sampling test and training points uniformly within each sphere, then compressing 200 of the dimensions including the second one by a factor of 10 – see

[1]NSC computes bounding hyperspheres for each class and assigns the query to the class whose sphere *center* is nearest.

fig. 4 (top). This produces a high dimensional data set with 100D-disk like classes and many irrelevant variables. The classes are fairly well separable but the data has somewhat suboptimal scaling. For the NAH and NHD methods we estimated the affine dimension using an eigenvalue gap detector that reliably gave the correct result (100) for all runs with more than about 120 training points.

Fig. 4 (bottom) shows the resulting recognition rates for NAH, NHD and NCH with varying numbers of training samples. The hyperdisk method predominates, particularly for larger numbers of training samples. The example is somewhat idealized – the data is quite clean and the classes have a form that is well adapted to the hyperdisk model – but it illustrates several advantages of the hyperdisk method. Firstly, NCH performs poorly. It separates classes $\{A, B\}$ from $\{C, D\}$ almost perfectly, but it is not much better than random (around 55-60% correct) at separating A from B and C from D. This happens because the interclass spacing is small and the convex hulls of the training samples fill so little of the volume of the 100-D class disks that test samples are almost as likely to lie close to the hull of the wrong class as to that of the right one – *i.e.* even though the hulls "fill in the gaps" between the training samples, they are still very poor estimates of the actual class boundaries. NCH is also much slower than NAH and NHD at run time because it needs to solve a quadratic program for each test sample to find the nearest point on the hull. Both problems are endemic to the convex hull formulation.

Secondly, NAH does surprisingly well, especially when one considers that it has an asymptotic error rate of 50%: for exact estimates of the 100-D affine hulls of the classes, A and C (and similarly, B and D) are indistinguishable because they have identical affine hulls. Empirically NAH does much better than this because the estimates of the affine hulls are noisy: being estimated from examples of class A, the hull for class A always passes close to the centre of class A, but its random tilt typically makes it pass somewhat further from the centre of class C, and vice versa. Hence, empirical NAH estimates indirectly incorporate some information about the relative positions of the classes within their affine hyperplanes. This may explain why the performances of NAH and NHD are often similar in the below experiments on real data. However, as the above results suggest, it is often advisable to incorporate the position information explicitly by using NHD.

4.2. Experiments on Image Datasets

ORL Face Dataset.[2] The Olivetti-Oracle Research Lab face dataset contains 10 upright 92×112 frontal face images per person of $C = 40$ individuals, taken at different times with slightly different lighting conditions, image positions, facial expressions and facial details. For this experiment we used the raw image pixels as input features without applying any visual preprocessing. For training we randomly selected $N = 3, 5, 7$ images of each individual, keeping the remaining $10 - N$ for testing. The results are summarized in table 4.1 (top left). The NHD and NAH classifiers were equal best among the methods tested, followed by NCH, then NN, with NSC coming last.

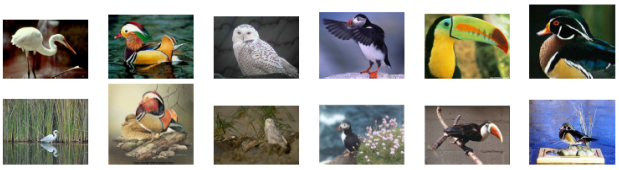


Figure 5. Some examples from the Birds dataset.

Coil100 Objects Dataset.[3] The Coil100 dataset includes 72 views each of 100 different objects taken on a turntable at orientations spaced at 5 degree intervals. We chose 40 objects randomly for the experiments. We used the raw grayscale pixels of the 128×128 images as input features, without applying any further visual preprocessing. For training we randomly selected $N = 18, 36, 54$ images of each object, keeping the remaining $72 - N$ for testing. The results are given in table 4.1 (top right). NHD and NAH again give very similar results with NHD having a slight edge. NHD achieves the best accuracy for $N = 18, 54$ while for $N = 36$ NCH is preferred to NHD and NAH. NSC again produced the worst results.

Birds Dataset. This contains six categories, each with 100 images (Lazebnik et al., 2005). It is a challenging visual object recognition task with the birds appearing against highly cluttered backgrounds and the images having large intra-class, scale, and viewpoint variability. Some example images are shown in fig. 5. We use a "bag of features" representation for the images as they are too diverse to allow simple geometric alignment of their objects. In this method, patches are sampled from the image at many different positions and scales, either densely, randomly or based on the output of some kind of salient region detector. Here we used a dense grid of patches. Each patch was described using the robust visual descriptor SIFT (Lowe, 2004) and vector quantized using nearest neighbor assignment against a 2000 word visual dictionary learned from the complete set of training patches. For training we randomly selected $N = 25, 50, 75$ images of each class, keeping the remaining $100 - N$ for testing.

The results are given in table 4.1 (bottom left). For $N = 50, 75$, NCH achieves the best recognition rates whereas

[2] *www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html*

[3] *www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php*

Table 1. Classification Rates (%) and their standard deviations on respectively the ORL Face data set (top left), the COIL data set (top right), and the Birds data set (bottom left). The recognition rates are averages over 15 random training/test splits. (Bottom right) Classification Rates (%) on selected UCI data sets.

ORL	$N = 3$	$N = 5$	$N = 7$
NHD	**88.50** $\pm$ 2.2	95.30 $\pm$ 1.5	**97.00** $\pm$ 1.8
NAH	**88.50** $\pm$ 2.2	**95.30** $\pm$ 1.5	**97.00** $\pm$ 1.8
NCH	88.47 $\pm$ 2.2	94.97 $\pm$ 1.5	96.72 $\pm$ 1.6
NSC	86.50 $\pm$ 2.8	91.77 $\pm$ 1.5	93.61 $\pm$ 2.0
NN	87.74 $\pm$ 2.3	94.30 $\pm$ 1.5	96.11 $\pm$ 1.7

COIL	$N = 18$	$N = 36$	$N = 54$
NHD	**97.38** $\pm$ 0.3	99.35 $\pm$ 0.4	**99.93** $\pm$ 0.1
NAH	97.33 $\pm$ 0.3	99.32 $\pm$ 0.5	**99.93** $\pm$ 0.1
NCH	97.35 $\pm$ 0.3	**99.41** $\pm$ 0.3	99.41 $\pm$ 0.4
NSC	82.81 $\pm$ 3.3	82.94 $\pm$ 1.2	83.98 $\pm$ 1.2
NN	96.56 $\pm$ 0.5	98.84 $\pm$ 0.4	99.72 $\pm$ 0.3

Birds	$N = 25$	$N = 50$	$N = 75$
NHD	**86.62** $\pm$ 1.6	90.51 $\pm$ 1.2	92.14 $\pm$ 1.8
NAH	**86.62** $\pm$ 1.6	90.51 $\pm$ 1.2	92.14 $\pm$ 1.8
NCH	86.60 $\pm$ 1.6	**90.91** $\pm$ 1.4	**92.67** $\pm$ 1.7
NSC	84.43 $\pm$ 2.3	87.82 $\pm$ 1.8	87.85 $\pm$ 1.8
NN	53.38 $\pm$ 4.1	60.51 $\pm$ 8.3	64.05 $\pm$ 2.6

UCI	Iris	IS	MF	Wine	WDBC
NHD	**96.7**	96.0	**98.4**	96.7	96.3
NAH	**96.7**	95.7	**98.4**	96.7	95.3
NCH	96.0	95.7	98.2	**97.8**	**97.7**
NSC	96.0	93.5	97.9	96.1	95.1
NN	96.0	**96.3**	97.6	94.5	96.0

Table 2. The key parameters of the low-dimensional datasets selected from the UCI Repository.

Data set	# Classes	# Examples	Dim.
Iris	3	150	4
IS	7	2310	19
MF	10	2000	256
Wine	3	178	13
WDBC	2	569	30

NHD and NAH are equal best for $N = 25$. All of the convex approximation based methods significantly outperform Nearest Neighbours.

4.3. Experiments with UCI Datasets

In the second group of experiments we tested the kernelized versions of the methods on five lower-dimensional datasets from the UCI repository: Iris, Image Segmentation (IS), Multiple Features (MF) - pixel averages, Wine, and Wisconsin Diagnostic Breast Cancer (WDBC). The key parameters of the datasets are summarized in table 2 and the results are presented in table 4.1 (bottom right).

In each case the dimensionality of the input space is smaller than the number of samples in each class. It follows that the native NAH classifier cannot be used directly because the affine hull of each class typically spans the entire input space. However kernelized versions of all of the classifiers can still be applied. The NCH and NSC formulations directly support kernelization while for NAH and NHD we used the Kernel PCA projection method described in section 3.2. We used Gaussian kernels and 5-fold cross-validation for all experiments.

NHD and NAH were the equal best classifiers for the Iris and MF databases while NCH came first for Wine and WDBC, and NN for IS. In all of the cases tested the proposed NHD classifier either matches or outperforms the NAH classifier. The convex approximation based approaches typically outperformed NN, but the difference was not as high as in the Birds database.

4.4. Discussion

The NHD and NAH classifiers often had almost identical performance but when there were differences NHD usually dominated. This suggests that NHD's tighter bounds on the classes are sometimes useful, but that they are often inactive, either because the affine hull projections of most queries already lie within the class hyperspheres or because the additional projections onto the hyperspheres do not add useful new discriminant information.

NHD and NAH often outperformed NCH in both the high-dimensional native experiments and the low-dimensional kernelized ones. As mentioned above, in high dimensions the convex hulls of the training samples typically significantly underestimate the extents of the classes unless the number of samples is exponential in the dimension of the class. Thus, despite the simplicity of their underlying approximations, the affine hulls and hyperdisks may often turn out to be better guides to the region spanned by the class than the convex hulls.

In the low-dimensional problems, NN (and related kernel methods) often perform relatively well, perhaps because hole artifacts are not so prevalent in low dimensions. Similarly, as the dimension decreases, NCH progressively improves relative to NAH because it provides tighter bounds on the class regions. NHD seems to offer a useful compro-

mise here.

In terms of run-time efficiency NAH and NHD are to be preferred as the affine hull and bounding hyperdisk parameters can be computed off-line. When there are large numbers of training samples, NCH often becomes prohibitively slow at run-time because it needs to solve a quadratic program for each sample-hull distance computation.

5. Summary and Conclusions

We have introduced a new method for high-dimensional classification based on approximating each class with the minimal bounding hyperdisk of its training samples – the intersection of their affine hull and their bounding hypersphere – and assigning test samples to the class with the nearest hyperdisk. For robustness, the algorithm uses PCA to suppress over-small "noise" dimensions in the affine hull and it removes outliers from the hypersphere calculation by bounding their Lagrange multipliers. In practice the hyperdisk approximation offers a useful middle ground between the loose approximation provided by the affine hull of the samples and the over-tight one given by their convex hull. It can also be kernelized to allow it to be used in lower-dimensional problems that require complex decision boundaries.

Future work. We are currently working on large-margin classifiers that calculate explicit decision boundaries during the training phase by maximizing the separation between the affine hull or hyperdisk approximations of the classes. These may be useful alternatives to SVMs, which maximize the separation between the convex hulls of the classes. Given that the affine hull and hyperdisk methods were often more accurate than the convex hull ones in the experiments, the new methods may yield more efficient classifiers than SVM in terms of both accuracy and computational complexity.

References

Bennett, K. P., & Bredensteiner, E. J. (2000). Duality and geometry in svm classifiers. *ICML*.

Cevikalp, H., Larlus, D., Douze, M., & Jurie, F. (2007). Local subspace classifiers: Linear and nonlinear approaches. *IEEE Workshop on Machine Learning for Signal Processing*. Thessaloniki, Greece.

Cevikalp, H., Neamtu, M., Wilkes, M., & Barkana, A. (2005). Discriminative common vectors for face recognition. *IEEE Transactions on PAMI*, *27*, 4–13.

Cevikalp, H., Triggs, B., Jurie, F., & Polikar, R. (2008). Margin-based discriminant dimensionality reduction for visual recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Gulmezoglu, M. B., Dzhafarov, V., & Barkana, A. (2001). The common vector approach and its relation to principal component analysis. *IEEE Trans. Speech Audio Proc.*, *9*, 655–662.

Hinton, G. E., Dayan, P., & Revow, M. (1997). Modeling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, *18*, 65–74.

Laaksonen, J. (1997). *Subspace classifiers in recognition of handwritten digits*. Doctoral dissertation, Helsinki University of Technology.

Lazebnik, S., Schmid, C., & Ponce, J. (2005). A maximum entropy framework for part-based texture and object recognition. *International Conference on Computer Vision*.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

Nalbantov, G. I., Groenen, P. J. F., & Bioch, J. C. (2007). *Nearest convex hull classification* (Technical Report). Econometric Institute and Erasmus Research Institute of Management.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Schölkopf, B., Smola, A. J., & Muller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.

Tax, D. M. J., & Duin, R. P. W. (2004). Support vector data description. *Machine Learning*, *54*, 45–66.

Verbeek, J. (2006). Learning non-linear image manifolds by global alignment of local linear models. *IEEE Transactions on PAMI*, *28*, 1236–1250.

Vincent, P., & Bengio, Y. (2001). K-local hyperplane and convex distance nearest neighbor algorithms. *NIPS*.

Wang, J., Neskovic, P., & Cooper, L. N. (2005). Pattern classification via single spheres. *Discovery Science* (pp. 241–252).

Learning to Sportscast: A Test of Grounded Language Acquisition

David L. Chen DLCC@CS.UTEXAS.EDU
Raymond J. Mooney MOONEY@CS.UTEXAS.EDU
Dept. of Computer Sciences, The University of Texas at Austin, 1 University Station C0500, Austin TX 78712, USA

Abstract

We present a novel commentator system that learns language from sportscasts of simulated soccer games. The system learns to parse and generate commentaries without any engineered knowledge about the English language. Training is done using only ambiguous supervision in the form of textual human commentaries and simulation states of the soccer games. The system simultaneously tries to establish correspondences between the commentaries and the simulation states as well as build a translation model. We also present a novel algorithm, Iterative Generation Strategy Learning (IGSL), for deciding which events to comment on. Human evaluations of the generated commentaries indicate they are of reasonable quality compared to human commentaries.

Figure 1. Screenshot of our commentator system

1. Introduction

Children acquire language through exposure to linguistic input in the context of a rich, relevant, perceptual environment. By connecting words and phrases to objects and events in the world, the semantics of language is grounded in perceptual experience (Harnad, 1990). Ideally, a machine learning system would be able to acquire language in a similar manner without human supervision. As a step in this direction, we present a commentator system that can describe events in a simulated soccer game by learning from sample human commentaries paired with the simulation states. A screenshot of our system with generated commentaries is shown in Figure 1.

Although there has been some interesting computational work in grounded language learning (Roy, 2002; Bailey et al., 1997; Yu & Ballard, 2004), most of the focus has been on dealing with raw perceptual data and the complexity of the language involved has been very modest. To help make progress, we study the problem in a simulated environment that retains many of the important properties of a dynamic world with multiple agents and actions while avoiding many of the complexities of robotics and vision. Specifically, we use the Robocup simulator (Chen et al., 2003) which provides a fairly detailed physical simulation of robot soccer. While several groups have constructed Robocup commentator systems (André et al., 2000) that provide a textual natural-language (NL) transcript of the simulated game, their systems use manually-developed templates and are incapable of learning.

Our commentator learns to semantically interpret and generate language in the Robocup soccer domain by observing an on-going commentary of the game paired with the dynamic simulator state. By exploiting existing techniques for abstracting a symbolic description of the activity on the field from the detailed state of the physical simulator (André et al., 2000), we obtain a pairing of natural language with a symbolic description of the perceptual context in which it was uttered. However, such training data is highly ambiguous because each comment usually co-occurs with several events in the game. We integrate and enhance existing methods for learning semantic parsers and NL gen-

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

erators (Kate & Mooney, 2007; Wong & Mooney, 2007a) in order to learn to understand and produce grounded language from such ambiguous training data.

2. Background

Systems for learning semantic parsers induce a function that maps NL sentences to *meaning representations* (MRs) in some formal logical language. Existing work has focused on learning from a supervised corpus in which each sentence is manually annotated with its correct MR (Mooney, 2007). Such human annotated corpora are expensive and difficult to produce, limiting the utility of this approach. The systems described below assume they have access to a formal context-free grammar, called the *meaning representation grammar* (MRG), that defines the MR language (MRL).

2.1. KRISP and KRISPER

KRISP (Kate & Mooney, 2006) uses SVMs with string kernels (Lodhi et al., 2002) to learn semantic parsers. For each production in the MRG, the system learns an SVM string classifier that recognizes the associated NL words or phrases. The resulting suite of classifiers is then used to construct the most probable MR for a complete NL sentence. Given the partial matching provided by string kernels and the over-fitting prevention provided by SVMs, KRISP has been experimentally shown to be robust to noisy training data.

KRISPER (Kate & Mooney, 2007) is an extension to KRISP that handles ambiguous training data, in which each sentence is annotated only with a *set* of potential MRs, only one of which is correct. It employs an iterative approach analogous to EM that improves upon the selection of the correct NL–MR pairs in each iteration. In the first iteration, it assumes that all of the MRs paired with a sentence are correct and trains KRISP with the resulting noisy supervision. In subsequent iterations, KRISPER uses the currently trained parser to score each potential NL–MR pair, selects the most likely MR for each sentence, and retrains the parser. In this manner, KRISPER is able to learn from the type of weak supervision expected for a grounded language learner exposed only to sentences in ambiguous contexts. However, the system has previously only been tested on artificially corrupted or generated data.

2.2. WASP

WASP learns semantic parsers using *statistical machine translation* (SMT) techniques (we use the Wong & Mooney (2007b) version). It induces a *probabilistic synchronous context-free grammar* (PSCFG) (Wu, 1997) to translate NL sentences into logical MRs using a modification of recent

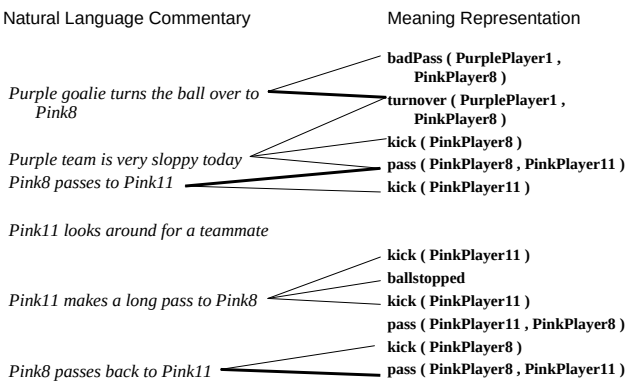


Figure 2. Sample trace of ambiguous training data

methods in syntax-based SMT (Chiang, 2005). Since a PSCFG is symmetric with respect to input/output, the same learned model can also be used to *generate* NL sentences from formal MRs. Thus, WASP learns a PSCFG that supports both semantic parsing and natural language generation. Since it does not have a formal grammar for the NL, the generator also learns an n-gram language model for the NL and uses it to choose the overall most probable NL translation of a given MR using a noisy-channel model (Wong & Mooney, 2007a).

3. Sportscasting Data

To train and test our system, we assembled human-commentated soccer games from the Robocup simulation league (www.robocup.org). Since our focus is language learning not computer vision, we chose to use simulated games instead of real game video to simplify the extraction of perceptual information. Symbolic representations of game events were automatically extracted from the simulator traces by a rule-based system. The extracted events mainly involve actions with the ball, such as kicking and passing, but also include other game information such as whether the current playmode is kickoff, offside, or corner kick. The events are represented as atomic formulas in predicate logic with timestamps. These logical facts constitute the requisite MRs, and we manually developed a simple MRG for this formal semantic language.

For the NL portion of the data, we had humans commentate games while watching them on the simulator. The commentators typed their comments into a text box, which were recorded with a timestamp. To construct the final ambiguous training data, we paired each comment with all of the events that occurred five seconds or less before the comment was made. A sample set of ambiguous training data is shown in Figure 2. Note that the use of English words for predicates and constants in the MR is for human read-

	Number of events	Number of comments			Events per comment		
		Total	Have MRs	Have Correct MR	Max	Average	Std. Dev.
2001 final	3992	722	671	520	9	2.235	1.641
2002 final	2125	514	458	376	10	2.403	1.653
2003 final	2112	410	397	320	12	2.849	2.051
2004 final	2223	390	342	323	9	2.729	1.697

Table 1. Statistics about the dataset

ability only, the system treats these as arbitrary conceptual tokens and must learn their connection to English words.

We annotated a total of four games, namely, the finals for the Robocup simulation league for each year from 2001 to 2004. Summary statistics about the data are shown in Table 1. The 2001 final has almost twice the number of events as the other games because it went into double overtime. For evaluation purposes only, a gold-standard matching was produced by examining each comment manually and selecting the correct MR if it exists. The bold lines in Figure 2 indicate the correct matches. Notice some sentences do not have correct matches (about one fifth of our data). For example, the sentence "Purple team is very sloppy today" cannot be represented in our MRL and consequently does not have a corresponding correct MR. On the other hand, in the case of the sentence "Pink11 makes a long pass to Pink8", the correct MR falls outside the 5-second window. For each game, Table 1 shows the total number of NL sentences, the number of these that have at least one recent extracted event to which it *could* refer, and the number of these that actually *do* refer to one of these recent extracted events. The maximum, average, and standard deviation for the number of recent events paired with each comment is also given.

4. New Algorithms

While existing systems are capable of solving parts of the sportscasting problem, none of them are able to perform the whole task on their own. We introduce three new end-to-end systems below which are able to learn from the ambiguous supervision in our training data and generate commentaries on unseen games.

4.1. WASPER

Since our primary goal is to learn a sportscaster rather than a parser, we use WASP to learn a system that can also generate NL from MRs produced by the perceptual system. However, WASP requires unambiguous training data which is not available for our domain. Therefore, we extend WASP using EM-like retraining similar to KRISPER to handle ambiguously annotated data, resulting in a system we call WASPER. In general, any system that learns semantic parsers can be extended to handle ambiguous data as long as it can produce confidence levels for given NL–MR pairs.

4.2. KRISPER-WASP

KRISP has been shown to be superior to WASP at handling noisy training data (Kate & Mooney, 2006). Consequently, we can expect KRISPER's parser to outperform WASPER's because EM-like training on ambiguous data initially creates a lot of noisy, incorrect supervision. Even if the average number of possible MRs per sentence is only 2, it still results in at least 50% noise in the training data in the first iteration. However, KRISPER cannot learn a language generator, which is necessary for our sportscasting task. As a result, we create a new system called KRISPER-WASP that is both good at disambiguating the training data and capable of generation. We first use KRISPER to train on the ambiguous data and produce a disambiguated training set by using its prediction for the most likely MR for each sentence. This unambiguous training set is then used to train WASP to produce both a parser and a generator.

4.3. WASPER-GEN

In both KRISPER and WASPER, the criterion for selecting the best NL–MR pairs during retraining is based on maximizing the probability of parsing a sentence into a particular MR. However, since WASPER is capable of both parsing and generation, we could alternatively select the best NL–MR pairs by evaluating how likely it is to *generate* the sentence from a particular MR. Thus, we built another version of WASPER (WASPER-GEN) that disambiguates the training data in order to maximize the performance of *generation* rather than parsing. It uses a generation-based score rather than a parsing-based score to select the best NL–MR pairs. Specifically, an NL–MR pair (n, m) is scored by using the current trained generator to generate an NL sentence for m and then comparing the generated sentence to n to compute the NIST score. NIST score is a machine translation (MT) metric that measures the precision of a translation in terms of the proportion of n-grams it shares with a human translation (Doddington, 2002). It is also used to evaluate NL generation. Another popular MT metric is BLEU score (Papineni et al., 2002) but we found it inadequate for our domain because it overly penalizes

translations shorter than the target sentences. Most of our generated commentaries are shorter than the human commentaries due to the fact that humans are more verbose and many details of the human descriptions are not represented by our MRL.

4.4. Learning for Strategic Generation

A language generator alone is not enough to produce a sportscast. In addition to knowing *how* to say something, one must also know *what* to say. A sportscaster must also choose which events to describe. In NLP, deciding what to say is called *strategic generation*.

We developed a simple method for learning which events to describe. For each event type (i.e. for each predicate like `pass`, or `goal`), the system uses the training data to estimate a probability that it is mentioned by the sportscaster. Given the gold-standard NL–MR matches, this probability is easy to estimate; however, the learner does not know the correct matching. Instead, the system must estimate the probabilities from the ambiguous training data. We compare two basic methods for estimating these probabilities.

The first method uses the *inferred* NL–MR matching produced by the language-learning system. The probability of commenting on each event type, E_i, is estimated as the percentage of events of type E_i that have been matched to *some* NL sentence.

The second method, which we call Iterative Generation Strategy Learning (IGSL), uses a variant of EM, treating the matching assignments as hidden variables, initializing each match with a prior probability, and iterating to improve the probability estimates of commenting on each event type. Unlike the first method, IGSL uses MRs not associated with any sentences explicitly in training. Algorithm 1 shows the pseudocode. Each sentence accounts for at most one occurrence of an event being commented (some comments do not correspond to any MRs), so we enforce that the counts associated with a sentence add up to exactly one. In the initial iteration, every possible match gets assigned a weight inversely proportional to its amount of ambiguity. Thus, a sentence associated with five possible MRs will assign each match a weight of $\frac{1}{5}$. In the subsequent iterations, we use the learned estimates for each event type to assign weights to the edges, again normalizing to make sure that the weights of the edges coming out of each sentence sum to one.

To generate a sportscast, we first use the learned probabilities to determine which events to describe. For each time step, we only consider commenting on the event with the highest probability. The system then generates a comment for this event stochastically based on the estimated probability for its event type.

Algorithm 1 Iterative Generation Strategy Learning

input event types $E = \{E_1, ..., E_i, ..., E_n\}$, the number of occurrences of each event type $totalCount(E_i)$, sentences S and their associated sets of meaning representations $MR(s)$,
output probabilities of commenting on each event type $Pr(E_i)$

for event type $E_i \in E$ **do**
  Initialize $count = 0$
  for sentence $s \in S$ and $E_i \in MR(s)$ **do**
    $count = count + \frac{1}{|(MR(s))|}$
  end for
  $Pr(E_i) = \frac{count}{totalCount(E_i)}$
end for

repeat
  for event type $E_i \in E$ **do**
    Initialize $count = 0$
    for sentence $s \in S$ and $E_i \in MR(s)$ **do**
      $totalProb = 0$
      for event $E_j \in MR(s)$ **do**
        $totalProb = totalProb + Pr(E_j)$
      end for
      $count = count + \frac{Pr(E_i)}{totalProb}$
    end for
    $Pr(E_i) = \frac{count}{totalCount(E_i)}$
  end for
until Convergence or MAX_ITER reached

5. Experimental Evaluation

This section presents experimental results on the Robocup data for four systems: KRISPER, WASPER, KRISPER-WASP, and WASPER-GEN. To better gauge the effect of accurate ambiguity resolution, we also include results of unmodified WASP. Since WASP requires unambiguous training data, we randomly pick a meaning for each sentence from its set of potential MRs. Finally, we also include the result of WASP trained using *gold matching* which consists of the correct NL–MR pairs annotated by a human. This represents an upper-bound on what our systems could achieve if they disambiguated the training data perfectly.

We evaluate each system on three tasks: matching, parsing, and generation. The matching task measures how well the systems can disambiguate the training data. The parsing and generation tasks measure how well the systems can translate from NL to MR, and from MR to NL, respectively.

Since there are four games in total, we trained using all possible combinations of one to three games, and in each case, tested on the games not used for training. Results were averaged over all train/test combinations. We evalu-

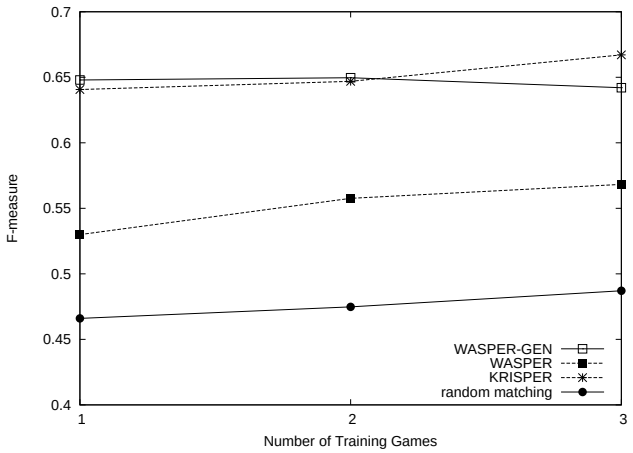


Figure 3. Matching Results

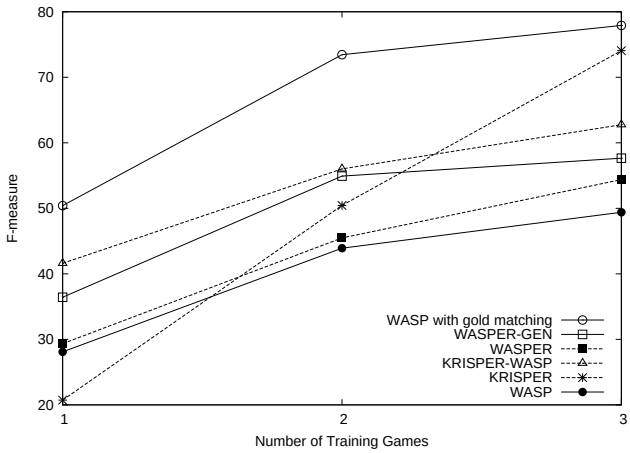


Figure 4. Semantic Parsing Results

ated matching and parsing using F-measure, the harmonic mean of recall and precision. Precision is the fraction of the system's annotations that are correct. Recall is the fraction of the annotations from the gold-standard that the system correctly produces. Generation is evaluated using NIST scores which roughly estimates how well the produced sentences match with the target sentences.

5.1. Matching NL and MR

Since handling ambiguous training data is an important aspect of grounded language learning, we first evaluate how well the various systems pick the correct NL–MR pairs. Figure 3 shows the F-measure for identifying the correct set of pairs for the various systems. WASPER does better than random matching, but worse than the other two systems. While we expected KRISPER to perform better since it is more adept at handling noisy data, it is somewhat surprising that WASPER-GEN does about the same. A potential explanation is that WASPER-GEN avoids making certain systematic errors typical of the other systems. This is discussed further in section 5.3.

5.2. Semantic Parsing

Next, we present results on the accuracy of the learned semantic parsers. Each trained system is used to parse and produce an MR for each sentence in the test set that has a correct MR in the gold-standard matching. A parse is considered correct if and only if it matches the gold standard exactly. Parsing is a fairly difficult task because there is usually more than one way to describe the same event. For example, "Player1 passes to player2" can refer to the same event as "Player1 kicks to player2." Thus, accurate parsing requires learning all the different ways people describe an event. Synonymy is not limited to verbs. In our data, "Pink1", "PinkG" and "pink goalie" all refer to player1 on the pink team. Since we are not providing the systems with any prior knowledge, parsers have to learn all these different ways of referring to the same entity.

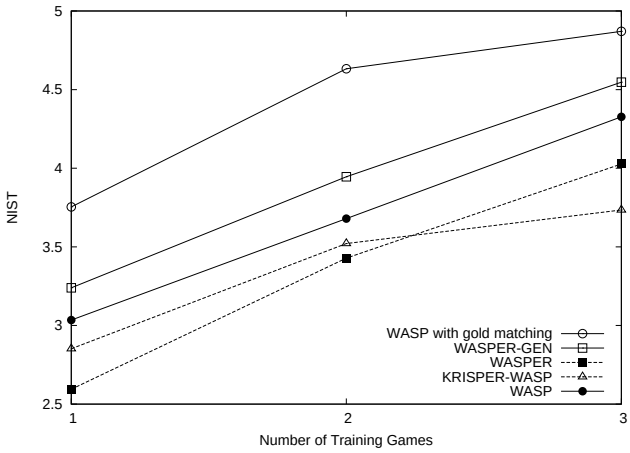


Figure 5. Generation results

Results are shown in Figure 4, and, as expected, follow the matching results. Systems that did better at disambiguating the training data also did better on parsing since their supervised training data is less noisy. When trained on 3 games, KRISPER does the best since it is most effective at handling the noise in the final supervised data. However, it tends to do worse than the other systems when given less training data.

5.3. Generation

The third evaluation task is generation. All of the WASP-based systems are given each MR in the test set that has a gold-standard matching NL sentence and asked to generate an NL description. The quality of the generated sentence is measured by comparing it to the gold-standard using NIST scoring.

This task is easier than parsing because the system only needs to learn *one way* to accurately describe an event. This property is reflected in the results, shown in Figure 5, where even the baseline system WASP does fairly well, outperforming WASPER and KRISPER-WASP. As the number of event types is fairly small, only a relatively small number of correct matchings is required to perform this task well as long as each event type is associated with a correct sentence pattern more often than any other sentence patterns. Consequently, it is far more costly to make systematic errors as is the case for WASPER and KRISPER-WASP.

Even though systems such as WASPER and KRISPER-WASP do fairly well at disambiguating the training data, the mistakes they make in selecting the NL–MR pairs often repeat the same basic error. For example, a bad pass event is often followed by a turnover event. If initially the system incorrectly determines that the comment "Player1 turns the ball over to the other team" refers to a bad pass, it will parse the sentence "Player2 turns the ball over to the other team" as a bad pass as well since it just reinforced that connection. Even if the system trains on a correct example where a bad pass is paired with the linguistic input "Player1 made a bad pass", it does not affect the parsing of the first two sentences and does not correct the mistakes. As a result, a bad pass becomes incorrectly associated with the sentence pattern "Someone turns the ball over to the other team."

On the other hand, WASPER-GEN does the best due to the imbalance between the variability of natural language comments and the MRs. While the same MR will typically occur many times in a game, the exact same comments are almost never uttered again. This leads to two performance advantages for WASPER-GEN.

WASPER-GEN avoids making the same kind of systematic mistakes as WASPER and KRISPER-WASP. Following the previous example, when WASPER-GEN encounters the correct matching for bad pass, it learns to associate bad passes with the correct sentence pattern. When it goes back to those first two incorrect pairings, it will likely correct its mistakes. This is because the same MR bad pass is present in all three examples. Thus, it will slowly move away from the incorrect connections. Of course, parsing and generation are symmetrical processes, so using generation to disambiguate data has its own problems. Namely, it is possible to converge to a point where many events generates the same natural language description. However, since there is much more variability in natural language, it is very unlikely that the same sentence pattern will occur repeatedly, each time associated with different events.

Another performance advantage of WASPER-GEN can be found by looking at the objective differences. Systems such as WASPER and KRISPER-WASP which use parsing

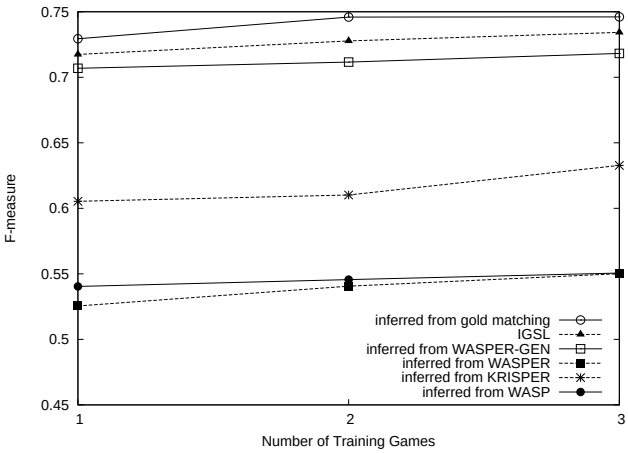


Figure 6. Strategic Generation Results

IGSL		WASPER-GEN	
corner_kick	1	pass	1
pass	0.983	badPass	0.708
badPass	0.970	corner_kick	0.438
goal	0.970	block	0.429
block	0.955	turnover	0.377

Table 2. Top scoring predicates with their estimated probabilities of being described

scores, try to learn a good translation model for each sentence pattern. On the other hand, WASPER-GEN only tries to learn a good translation model for each MR pattern. Thus, WASPER-GEN is more likely to converge on a good model as there are fewer MR patterns than sentence patterns. However, it can be argued that learning good translation models for each sentence pattern will help in producing more varied commentaries, a quality that is not captured by the NIST score.

5.4. Strategic Generation

The different methods for learning strategic generation are evaluated based on how often the events they describe coincide with those the human decided to describe in the test data. For the first method, results from using the inferred matchings produced by KRISPER, WASPER, KRISPER-WASP, and WASPER-GEN as well as the gold and random matching for establishing baselines are all presented in Figure 6. From the graph, it is clear that IGSL outperforms learning from the inferred matchings and actually performs at a level close to using the gold matching. However, it is important to note that we are limiting the potential of learning from the gold matching by using only the predicates to decide whether to talk about an event.

The top scoring predicates from IGSL as well as the best result from using inferred matchings, WASPER-GEN, are

	English Fluency	*Semantic Correctness*	*Sportscasting Ability*
Human	3.938	4.25	3.625
Machine	3.438	3.563	2.938

Table 3. Human evaluation of overall sportscast

shown in Table 2. While both systems learn to talk about frequent events such as passing, WASPER-GEN does poorly on rare, but significant events such as goal scoring. This is because WASPER-GEN saw those events very rarely in training and did not learn to correctly match them to sentences. It is worth noting that IGSL learns a higher probability for events in general. This improves its recall and hurts its precision. However, since many of its top-ranked events such as goals are rare, the overall quality is maintained without becoming overly verbose. Therefore, we used IGSL for the human evaluations below.

5.5. Human Evaluation

Automatic evaluation of generation is an imperfect approximation of human assessment at best. Moreover, automatically evaluating the quality of an entire generated sportscast is even more difficult. Consequently, we recruited four fluent English speakers with no previous experience with Robocup or any of our systems to serve as human judges. We compared their subjective evaluations of human and machine generated sportscasts. Each judge was given 8 clips of simulated game video along with subtitled commentaries. The 8 clips use 4 game segments of 2 minutes each, one from each of the four games. Each of the 4 game segments is shown twice, once with human commentary and once with generated commentary. We use IGSL to determine the events to comment on and use WASPER-GEN (our best performing system for generation) to produce the commentaries. The system was always trained on three games, leaving out the one from which the test segment was extracted. The videos are shown in random order with the human and machine commentaries of a segment flipped between judges to ensure no consistent bias toward segments being shown earlier or later. We asked the judges to score the commentaries using the following metrics:

Score	*English Fluency*	*Semantic Correctness*	*Sportscasting Ability*
5	Flawless	Always	Excellent
4	Good	Usually	Good
3	Non-native	Sometimes	Average
2	Disfluent	Rarely	Bad
1	Gibberish	Never	Terrible

Fluency and semantic correctness, or adequacy, are standard metrics in human evaluations of NL translations and generations. Fluency measures how well the commentaries are structured, including syntax and grammar. Semantic correctness indicates whether the commentaries actually describe what is happening in the game. Finally, sportscasting ability measures the overall quality of the sportscast. This includes whether the sportscasts are interesting and flow well. The scores are averaged over all four games and across all the judges. Table 3 shows the results.

While human commentaries are clearly superior to the machine's, the largest difference between the average scores is only 0.7. Moreover, the judges indicated that they were able to understand and follow the generated commentaries without trouble. Part of the reason for the lower scores actually result from our impoverished MRL. Semantic correctness scores were deducted when the machine misses commenting on certain facts not represented in our MRL such as the location of the ball and the players. The lack of temporal or locality information also results in dry and repetitive comments which hurt the sportscasting score. This is an important point that is not captured by the NIST score. In our NIST score evaluation, each sentence is treated separately and no attempt was made at measuring how well the individual comments fit together. However, it is clear from the human evaluations that variability of sentence pattern is vital to a good sportscast. The machine can correctly comment on all the factual events in a game and still produce a bad sportscast that no one wants to listen to.

6. Related Work

Robotics and vision researchers have worked on inferring a grounded meaning of individual words or short referring expressions from visual perceptual context, e.g. (Roy, 2002; Bailey et al., 1997; Barnard et al., 2003; Yu & Ballard, 2004). However, the complexity of the natural language used in this existing work is very restrictive, many of the systems use pre-coded knowledge of the language, and almost all use static images to learn language describing objects and their relations, and cannot use dynamic video to learn language describing actions. Some recent work on video retrieval has focused on learning to recognize events in sports videos and connect them to English words (Fleischman & Roy, 2007). There has also been recent work on grounded language learning in simulated computer-game environments (Gorniak & Roy, 2005). However, none of this prior work makes use of modern statistical-NLP parsing techniques, learns to build formal meaning representations for complete sentences, or learns to generate natural language.

There has been some recent work on learning generation strategies using reinforcement learning (Zaragoza & Li, 2005). In contrast, our domain does not include interaction with the users and no feedback is available.

7. Future Work

The current system is limited by its simple MRL. For example, the location of players or the ball is not represented. Moreover, we do not keep contextual information which makes it difficult to generate interesting, non-repetitive sportscasts. Contextual information would also help us provide comments not directly induced by the events happening now, such as the current score. Finally, it is clear that we need a more hierarchical representation that captures the relationships between events in order to avoid making systematic matching errors on frequently co-occurring events.

With respect to algorithms, using learned strategic-generation knowledge (information about what events are likely to illicit comments) could improve the resolution of ambiguities. We would also like to eventually apply our methods to real captioned video input using the latest methods in computer vision.

8. Conclusion

We have presented an end-to-end system that learns from sample commentaries and generates sportscasts for novel games. Dealing with the ambiguity inherent in the training environment is a critical issue in learning language from perceptual context. We have evaluated various methods for disambiguating the training data in order to build a language generator. Using a generation evaluation metric as the criterion for selecting the best NL–MR pairs produced the best results overall. Our system also learns a simple model of strategic generation from the ambiguous training data by estimating the probability that each event type invokes a comment. Experimental evaluation verified that the system learns to accurately parse and generate comments and to generate sportscasts that are competitive with those produced by humans.

Acknowledgement

We thank Adam Bossy for his work on simulating perception for the Robocup games. This work was funded by the NSF grant IIS–0712907X. Most of the experiments were run on the Mastodon Cluster, provided by NSF Grant EIA-0303609.

References

André, E., Binsted, K., Tanaka-Ishii, K., Luke, S., Herzog, G., & Rist, T. (2000). Three RoboCup simulation league commentator systems. *AI Magazine*, *21*, 57–66.

Bailey, D., Feldman, J., Narayanan, S., & Lakoff, G. (1997). Modeling embodied lexical development. *COGSC-97*.

Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D. M., & Jordan, M. I. (2003). Matching words and pictures. *Journal of Machine Learning Research*, *3*, 1107–1135.

Chen, M., Foroughi, E., Heintz, F., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Noda, I., Obst, O., Riley, P., Steffens, T., Wang, Y., & Yin, X. (2003). Users manual: RoboCup soccer server manual for soccer server version 7.07 and later. Available at `http://sourceforge.net/projects/sserver/`.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. *ACL-05* (pp. 263–270). Ann Arbor, MI.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. *Proc. of ARPA Workshop on Human Language Technology* (pp. 128–132). San Diego, CA.

Fleischman, M., & Roy, D. (2007). Situated models of meaning for sports video retrieval. *NAACL-HLT-07*. Rochester, NY.

Gorniak, P., & Roy, D. (2005). Speaking with your sidekick: Understanding situated speech in computer role playing games. *AIIDE-05*. Stanford, CA.

Harnad, S. (1990). The symbol grounding problem. *Physica D*, *42*, 335–346.

Kate, R. J., & Mooney, R. J. (2006). Using string-kernels for learning semantic parsers. *ACL-06* (pp. 913–920). Sydney, Australia.

Kate, R. J., & Mooney, R. J. (2007). Learning language semantics from ambiguous supervision. *AAAI-2007* (pp. 895–900).

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, *2*, 419–444.

Mooney, R. J. (2007). Learning for semantic parsing. In A. Gelbukh (Ed.), *Computational linguistics and intelligent text processing: Proc. of the 8th Intl. Conference, CICLing 2007, Mexico City*, 311–324. Berlin: Springer Verlag.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. *ACL-02* (pp. 311–318). Philadelphia, PA.

Roy, D. (2002). Learning visually grounded words and syntax for a scene description task. *Computer Speech and Language*, *16*, 353–385.

Wong, Y. W., & Mooney, R. (2007a). Generation by inverting a semantic parser that uses statistical machine translation. *NAACL-HLT-07* (pp. 172–179).

Wong, Y. W., & Mooney, R. J. (2007b). Learning synchronous grammars for semantic parsing with lambda calculus. *ACL-07* (pp. 960–967).

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, *23*, 377–403.

Yu, C., & Ballard, D. H. (2004). On the integration of grounding language and learning objects. *AAAI-2004* (pp. 488–493).

Zaragoza, H., & Li, C.-H. (2005). Learning what to talk about in descriptive games. *HLT/EMNLP-05* (pp. 291–298). Vancouver, Canada.

Training SVM with Indefinite Kernels

Jianhui Chen JIANHUI.CHEN@ASU.EDU
Jieping Ye JIEPING.YE@ASU.EDU
Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287, USA

Abstract

Similarity matrices generated from many applications may not be positive semidefinite, and hence can't fit into the kernel machine framework. In this paper, we study the problem of training support vector machines with an indefinite kernel. We consider a regularized SVM formulation, in which the indefinite kernel matrix is treated as a noisy observation of some unknown positive semidefinite one (proxy kernel) and the support vectors and the proxy kernel can be computed simultaneously. We propose a semi-infinite quadratically constrained linear program formulation for the optimization, which can be solved iteratively to find a global optimum solution. We further propose to employ an additional pruning strategy, which significantly improves the efficiency of the algorithm, while retaining the convergence property of the algorithm. In addition, we show the close relationship between the proposed formulation and multiple kernel learning. Experiments on a collection of benchmark data sets demonstrate the efficiency and effectiveness of the proposed algorithm.

1. Introduction

Kernel methods work by embedding the data into a high-dimensional (possibly infinite-dimensional) feature space, where the embedding is defined implicitly through a kernel function. Evaluating the kernel function on all pairs of data points produces a symmetric and positive semidefinite (PSD) kernel matrix. Support Vector Machine (SVM) with a positive semidefinite kernel matrix has been applied successfully in numerous classification tasks including face recognition, image retrieval, and micro-array gene expression data analysis (Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2001; Tong & Chang, 2001). The PSD property of the kernel matrix ensures the existence of a Reproducing Kernel Hilbert Space (RKHS) and results in a convex formulation for SVM. Thus, a global optimal solution exists.

In practice, however, similarity matrices generated from many applications may not be PSD (Qamra et al., 2005; Roth et al., 2003a; Shimodaira et al., 2001). The problem of learning with a non-PSD similarity matrix (indefinite kernel) has been addressed by many researchers (Wu et al., 2005; Haasdonk, 2005; Lin & Lin, 2003). One simple and popular approach is to generate a PSD kernel matrix by transforming the spectrum of the indefinite kernel matrix (Wu et al., 2005). Several representative transformation methods include *denoise* which neglects the negative eigenvalues (Graepel et al., 1998; Pekalska et al., 2002), *flip* which flips the sign of the negative eigenvalues (Graepel et al., 1998), *diffusion* which applies matrix diffusion on the indefinite kernel (Kondor & Lafferty, 2002), and *shift* which shifts all the eigenvalues by a positive constant (Roth et al., 2003b). One common limitation of these approaches is that the transformation may lead to the loss of valuable information in the data.

Several other works use the non-PSD similarity matrix as a kernel, but they change the formulation of SVM. In (Lin & Lin, 2003), an SMO-type method is proposed to find stationary points for the non-convex dual formulation of SVM with a non-PSD sigmoid kernel. However, this method is based on the assumption that a corresponding RKHS still exists such that SVM formulations are valid. Haasdonk (2005) interprets learning with an indefinite kernel as the minimization of distance between two convex hulls in some pseudo-Euclidean (pE) space. However, it assumes that the representer theorem holds in such a pE space. Ong et al. (2004) associate the indefinite kernels with a Reproducing Kernel Kreĭn Space (RKKS), in which a general representer theorem exists and a regularized risk functional can be defined.

Recently, Luss and d'Aspremont (2007) propose a regularized SVM formulation, in which the indefinite kernel matrix is considered as a noisy observation of some unknown PSD one (proxy kernel). One attractive property

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

of this formulation is that the support vectors as well as the proxy kernel can be found simultaneously. However, the convex reformulation in (Luss & d'Aspremont, 2007) involves a nondifferentiable objective function. To facilitate the calculation of the gradient, Luss and d'Aspremont (2007) quadratically smoothed the objective function, resulting in two algorithms including the projected gradient method and the analytic center cutting plan method.

In this paper, we study the problem of training SVM with an indefinite kernel matrix following the formulation in (Luss & d'Aspremont, 2007). We show that this problem can be reformulated as a semi-infinite quadratically constrained linear program (SIQCLP), which includes a finite number of optimization variables with an infinite number of constraints. We then propose an iterative algorithm to solve this SIQCLP problem, which consists of two key steps: computing an intermediate SVM solution by solving a quadratically constraint linear program with a restricted subset of constraints, and updating the subset of constraints based on the obtained intermediate SVM solution. We further show the convergence property of the proposed iterative algorithm.

One limitation of the proposed algorithm is that the computational cost for solving the quadratically constraint linear program depends on the number of constraints, which gradually increases during the iteration. We propose to improve the efficiency of the iterative algorithm by pruning inactive constraints at each iteration. We show that such pruning will not affect the convergence property of the algorithm. In addition, we show the close relationship between the proposed SIQCLP formulation and multiple kernel learning (MKL). More specifically, the intermediate quadratically constraint linear program with a restricted subset of constraints is shown to be equivalent to a regularized version of the multiple kernel learning formulation in (Lanckriet et al., 2004). Thus, efficient algorithms for MKL (Lanckriet et al., 2004; Rakotomamonjy et al., 2007; Sonnenburg et al., 2006) can be applied to solve the SIQCLP problem. We have performed experiments on a collection of benchmark data sets. The presented experimental results demonstrate the efficiency and effectiveness of the proposed algorithm.

2. Background

Assume $K \in \mathbb{R}^{n\times n}$ is a valid kernel matrix, that is, K is positive semidefinite (PSD). Let $y = [y_1, \cdots, y_n] \in \mathbb{R}^n$ be the vector of class labels, where $y_i \in \{-1, +1\}$. The dual formulation of 1-norm soft margin SVM classification is given by (Schölkopf & Smola, 2001):

$$\begin{aligned} \max_{\alpha\in\mathbb{R}^d} \quad & \alpha^T e - \tfrac{1}{2}\alpha^T Y K Y \alpha \\ \text{subject to} \quad & \alpha^T y = 0,\ 0 \le \alpha \le C, \end{aligned} \tag{1}$$

where α is the vector of Lagrange dual variables, $Y = \text{diag}(y)$, C is a pre-specified parameter, and e is a vector of all ones of length n.

Since K is PSD, the optimization problem in Eq. (1) is a convex Quadratic Program (QP) (Boyd & Vandenberghe, 2004); hence a global optimal solution can be found via standard optimization techniques such as primal-dual interior point methods (Nocedal & Wright, 1999). In practice, however, many similarity matrices may be non-PSD (indefinite kernels), including sigmoid kernels (Vapnik, 1995) for various values of its parameters and hyperbolic tangent kernels (Smola et al., 2000). Additional examples include the protein sequence similarity measures based on Smith-Waterman and BLAST scores.

In (Luss & d'Aspremont, 2007), the indefinite kernel is considered as a noisy observation of some unknown PSD kernel (proxy kernel), and the following max-min optimization problem is proposed for simultaneous proxy kernel learning and SVM classification:

$$\begin{aligned} \max_{\alpha\in\mathbb{R}^d} \min_{K\in\mathbb{R}^{d\times d}} \quad & \alpha^T e - \tfrac{1}{2}\alpha^T Y K Y \alpha + \rho\|K - K_0\|_F^2 \\ \text{subject to} \quad & \alpha^T y = 0,\ 0 \le \alpha \le C,\ K \succeq 0, \end{aligned} \tag{2}$$

where K_0 is a pre-specified indefinite kernel matrix, K is the unknown proxy kernel matrix, and $\rho > 0$ is the pre-specified parameter, and $||\cdot||_F$ denotes the Frobenius norm of a matrix (Golub & Van Loan, 1996).

The objective function in Eq. (2) is convex in K and concave in α, thus a global optimal solution exists. However, direct optimization of Eq. (2) in terms of both α and K leads to a complex optimization problem involving nondifferentiable objective function (Luss & d'Aspremont, 2007). To facilitate the calculation of the gradient, Luss and d'Aspremont (2007) quadratically smoothed the objective function. Two algorithms including the projected gradient method and the analytic center cutting plan method are proposed for the proposed formulation.

3. Problem Formulation

We propose to solve the optimization problem in Eq. (2) by first reformulating it as a semi-infinite program (SIP) (Hettich & Kortanek, 1993a). The SIP problem refers to optimization problems that maximizes the functional $F(z)$ subject to a system of constraints on z, expressed as $g(z, t) \le 0$ for all t in some set B. When the objective is linear and the constraints are quadratic, the optimization problem is known as semi-infinite quadratically constrained linear program (SIQCLP).

For notational simplicity, we denote the objective function

in Eq. (2) as:

$$S(\alpha, K) = \alpha^T e - \frac{1}{2}\alpha^T YKY\alpha + \rho\|K - K_0\|_F^2. \quad (3)$$

The optimal solution to the max-min problem in Eq. (2) is a saddle-point for the function $S(\alpha, K)$ subject to the constraints in Eq. (2). Let (α^*, K^*) be optimal to Eq. (2). For any feasible α and K in Eq. (2), we have

$$S(\alpha, K^*) \le S(\alpha^*, K^*) \le S(\alpha^*, K); \quad (4)$$

moreover, it can be verified that

$$S(\alpha, K^*) = \min_{\tilde{K}\succeq 0} S(\alpha, \tilde{K}) \le S(\alpha, K). \quad (5)$$

By adding an additional variable $t \in \mathbb{R}$, the max-min optimization problem in Eq. (2) can be reformulated into a SIQCLP problem as follows:

$$\begin{aligned} \max_{\alpha\in\mathbb{R}^d,\, t\in\mathbb{R}} \quad & t \\ \text{subject to} \quad & \alpha^T y = 0,\ 0 \le \alpha \le C \\ & t \le S(\alpha, K),\ \forall\ K \succeq 0. \end{aligned} \quad (6)$$

The optimization problem in Eq. (6) has two optimization variables (t and α) with an infinite number of (quadratic) constraints, i.e., one quadratic constraint $t \le S(\alpha, K)$ for each kernel matrix K. When there is only one (fixed) kernel matrix K involved in Eq. (6), this optimization problem reduces to a standard SVM problem.

4. Algorithm

We propose an iterative algorithm to solve Eq. (6), which is guaranteed to converge to a global optimum. The algorithm is closely related to the *bundle method* (Hiriart-Urruty & Lemarechal, 1993; Teo et al., 2007).

The optimization problem in Eq. (6) maximizes its objective function with respect to two variables t and α with an infinite number of (quadratic) constraints. We approach the optimum by optimizing the variables t and α with a restricted subset of the infinite number of constraints, and then updating the constraint subset based on the obtained suboptimal t and α in an iterative manner. It is similar to the strategy presented in (Sonnenburg et al., 2006). The algorithm belongs to a family of algorithms for solving general SIP problems called the *exchange methods*, in which the constraints are exchanged at each iteration. The global optimality property of the final solution after convergence is guaranteed (Hettich & Kortanek, 1993b).

For a restricted subset of constraints, called a *localization* set of kernel matrices $\mathbf{K} = \{K_i\}_{i=1}^p$, the intermediate suboptimal t and α can be computed by solving the following optimization problem:

$$\begin{aligned} \max_{\alpha\in\mathbb{R}^d,\, t\in\mathbb{R}} \quad & t \\ \text{subject to} \quad & \alpha^T y = 0,\ 0 \le \alpha \le C \\ & t \le S(\alpha, K_i),\ i = 1, \cdots, p. \end{aligned} \quad (7)$$

This corresponds to a quadratically constrained linear program (QCLP) with p quadratic constraints. The optimization problem is often called the *restricted master problem*, and the obtained suboptimal solution pair (α, t) is called *intermediate solution*. Note that this QCLP problem can be solved efficiently using general optimization solvers.

To approach the optimum of the SIQCLP problem from a given intermediate solution pair (t, α), we find the next constraint with the maximum violation, i.e., the kernel matrix K that minimizes $S(\alpha, K)$. The optimal K can be computed by solving the following minimization problem:

$$\min_K S(\alpha, K) = \min_K \rho\|K - K_0\|_F^2 - \frac{1}{2}\alpha^T YKY\alpha. \quad (8)$$

If the optimal K^* to Eq. (8) satisfies $t \le S(\alpha, K^*)$, then the current intermediate solution pair (t, α) is optimal for the optimization problem in Eq. (6). Otherwise, K^* is added into the localization set $\mathbf{K}$. The intermediate solution pair (t, α) is updated by solving the restricted master problem based on the updated $\mathbf{K}$. We repeat this iterative process until convergence. The final solution is guaranteed to be globally optimal (Sonnenburg et al., 2006).

It can be shown (Luss & d'Aspremont, 2007) that the optimal K^* to the optimization problem in Eq. (8) for a fixed α is given by:

$$K^* = \left(K_0 + Y\alpha\alpha^T Y/(4\rho)\right)_+. \quad (9)$$

Here X_+ refers to the positive part of a symmetric matrix X, i.e., $X_+ = \sum_i \max(0, \lambda_i) x_i x_i^T$, where λ_i and x_i are the i-th eigenvalue and eigenvector of X.

Based on the discussions above, we propose an iterative algorithm to solve the optimization in Eq. (6). The pseudocode is presented in Algorithm 1. Note that the algorithm searches for a quadratic constraint (specified by K^*) with the maximum violation in step 1, then updates the intermediate solution (t, α), which is repeated iteratively until convergence. When no more constraints with violation can be found, i.e., all constraints are satisfied, Algorithm 1 converges. In practice, we determine the convergence by comparing an upper and lower bound of the objective as described in the next section.

4.1. Convergence Analysis

We analyze the convergence property of Algorithm 1. Recall that Algorithm 1 alternates between the updating of the

Algorithm 1 Proposed Algorithm

Input: Indefinite kernel K_0 and class label vector y;
Output: α and $\mathbf{K}$;
Initialization: $(t_0, \alpha_0) \leftarrow \max_\alpha S(\alpha, (K_0)_+)$;
Initialization: $i \leftarrow 1, \mathbf{K} = \varnothing$;
Do
  Step 1: compute K^* from Eq. (9) and $K_i \leftarrow K^*$;
  if $S(\alpha_{i-1}, K_i) \geq t_{i-1}$ **then** exit the loop;
    else update localization set $\mathbf{K} \leftarrow \mathbf{K} \cup \{K_i\}$;
  end if
  Step 2: compute (t_i, α_i) by solving Eq. (7);
  $i \leftarrow i + 1$;
until convergence

localization set $\mathbf{K}$ (step 1) and the updating of the intermediate solution pair (t, α) (step 2). At the i-th iteration, a new kernel matrix K_i is computed from Eq. (9) based on α_{i-1}. That is,

$$S(\alpha_{i-1}, K_i) = \min_{K \succeq 0} S(\alpha_{i-1}, K). \tag{10}$$

With the addition of K_i, the localization set is updated as $\mathbf{K} = \{K_j\}_{j=1}^i$. We denote

$$l_i^- = \max_{j \leq i} S(\alpha_{j-1}, K_j). \tag{11}$$

Let (t_i, α_i) be the solution pair to the optimization problem in Eq. (7) after the i-th iteration. Denote $u_i^+ = t_i$. That is,

$$\alpha_i = \arg\max_\alpha \left(\min_{K \in \mathbf{K}} S(\alpha, K) \right), \tag{12}$$

$$u_i^+ \equiv t_i = \min_{K \in \mathbf{K}} S(\alpha_i, K) = \max_\alpha \min_{K \in \mathbf{K}} S(\alpha, K), \tag{13}$$

where $\mathbf{K}$ is the updated restricted localization set.

The following theorem shows that Algorithm 1 makes continuous progress towards the optimal solution:

Theorem 4.1. *Let l_i^- and u_i^+ be defined in Eq. (11) and Eq. (13), respectively. Let (α^*, t^*) be the optimal solution pair to the optimization problem in Eq. (6). Then*

$$u_i^+ \geq t^* \geq l_i^-. \tag{14}$$

Moreover, the sequence $\{u_i^+\}$ is monotonically decreasing, and the sequences $\{l_i^-\}$ is monotonically increasing.

Proof. For any feasible α in Eq. (6), we have

$$\min_{K \in \mathbf{K}} S(\alpha, K) \geq \min_{K \succeq 0} S(\alpha, K). \tag{15}$$

It follows that the inequality above also holds for their corresponding pointwise maximum with respect to α. From Eq. (13) and the equality below

$$t^* = \max_{\alpha \in \mathbb{R}^d} \min_{K \succeq 0} S(\alpha, K), \tag{16}$$

we have $u_i^+ \geq t^*$. On the other hand, it follows from Eq. (10) that

$$K_j = \arg\min_{K \succeq 0} S(\alpha_{j-1}, K), \;\; j = 1, \cdots, i. \tag{17}$$

Thus $\{(\alpha_{j-1}, S(\alpha_{j-1}, K_j)), j = 1, \cdots, i\}$ is a set of feasible solution pairs to the optimization problem in Eq. (6). Since (α^*, t^*) is the optimal solution pair to Eq. (6), we have $t^* \geq S(\alpha_{j-1}, K_j)$, for $j = 1, \cdots, i$. If follows from Eq. (11) that $t^* \geq l_i^-$.

From Eq. (13), we have

$$u_i^+ = \max_\alpha \min_{K \in \mathbf{K}} S(\alpha, K). \tag{18}$$

Thus, the sequence $\{u_i^+\}$ is monotonically decreasing, as the size of localization set $\mathbf{K}$ monotonically increases. It follows from Eq. (11) that $\{l_i^-\}$ is monotonically increasing. This completes the proof of the theorem. □

Based on the result in Theorem 4.1, we can use the gap between u_i^+ and l_i^- to trace the convergence of Algorithm 1. When this gap is smaller that a pre-specified tolerance, we stop the algorithm.

4.2. Pruning Inactive Constraints

In Algorithm 1, a quadratically constraint linear program (QCLP) is involved at each iteration (step 2). The computational cost for solving QCLP grows with the number of quadratic constraints, which increases by one after each iteration. We show that at each iteration, many (inactive) quadratic constraints can be pruned, while retaining the convergence property of the algorithm.

Assume that (α_i, t_i) is the optimal solution pair at the i-th iteration with $\mathbf{K}^i = \{K_j\}_{j=1}^p$ as the localization set. We further partition $\mathbf{K}^i$ into two subsets as $\mathbf{K}^i = \mathbf{K}_{act}^i \cup \mathbf{K}_{ina}^i$ such that

$$t_i = S(\alpha_i, K), \;\; \forall \; K \in \mathbf{K}_{act}^i, \tag{19}$$

and

$$t_i < S(\alpha_i, K), \;\; \forall \; K \in \mathbf{K}_{ina}^i, \tag{20}$$

where the equalities in Eq. (19) and inequalities in Eq. (20) are called *active* and *inactive* constraints (Nocedal & Wright, 1999), respectively. Let K^* be the optimal matrix given in Eq. (9) with $\alpha = \alpha_i$. In Algorithm 1, we use $\mathbf{K}^i \cup K^*$ as the new localization set. We propose to improve the efficiency by removing the inactive constraints from the optimization and updating the new localization set $\mathbf{K}^{i+1}$ as $\mathbf{K}^{i+1} = \mathbf{K}_{act}^i \cup K^*$. Let (α_{i+1}, t_{i+1}) be the optimal solution pair at the $(i+1)$-th iteration with the updated $\mathbf{K}^{i+1}$ as the localization set. To show the convergence, we need to prove $t_{i+1} \leq t_i$, as summarized below:

Lemma 4.1. *Let t_i and t_{i+1} be defined as above. Then $t_{i+1} \leq t_i$.*

Proof. Prove by contradiction. Assume that $t_{i+1} \leq t_i$ doesn't hold, i.e., $t_{i+1} > t_i$.

Let $(\tilde{\alpha}, \tilde{t})$ be the optimal solution pair to the optimization problem in Eq. (7) with $\mathbf{K}^i_{act}$ as the localization set. It is clear that $t_{i+1} \leq \tilde{t}$, as $\mathbf{K}^i_{act} \subset \mathbf{K}^{i+1} = \mathbf{K}^i_{act} \cup K^*$. Thus $\tilde{t} \geq t_{i+1} > t_i$.

For any $K \in \mathbf{K}^i_{act}$, we have $S(\tilde{\alpha}, K) \geq \tilde{t}$. Since $\tilde{t} > t_i$, the following holds for any $K \in \mathbf{K}^i_{act}$:

$$S(\tilde{\alpha}, K) \geq \tilde{t} > t_i = S(\alpha_i, K). \tag{21}$$

For any $\eta \in (0, 1)$, let $\beta = \eta\tilde{\alpha} + (1-\eta)\alpha_i$. Since $S(\alpha, K)$ is concave on α and $t_i = S(\alpha_i, K)$ for any $K \in \mathbf{K}^i_{act}$ from Eq. (19), the following holds for any $K \in \mathbf{K}^i_{act}$:

$$S(\beta, K) \geq \eta S(\tilde{\alpha}, K) + (1-\eta) S(\alpha_i, K) > t_i. \tag{22}$$

Recall that for any $K \in \mathbf{K}^i_{ina}$, we have $S(\alpha_i, K) > t_i$. Since $S(\alpha, K)$ is continuous on α, and $\mathbf{K}^i_{ina}$ is a finite set, there exists an $\epsilon \in (0, 1)$ sufficiently close to zero such that

$$S(\beta_\epsilon, K) > t_i, \ \ \forall\ K \in \mathbf{K}^i_{ina}, \tag{23}$$

where $\beta_\epsilon = \epsilon\tilde{\alpha} + (1-\epsilon)\alpha_i$.

It follows from Eqs. (22) and (23) that

$$\hat{t} = \min_{K \in \mathbf{K}^i} S(\beta_\epsilon, K) > t_i. \tag{24}$$

Since $(\beta_\epsilon, \hat{t})$ is a feasible solution pair to Eq. (7) with $\mathbf{K}^i$ as the localization set, Eq. (24) contradicts with our assumption that (α_i, t_i) is the optimal solution pair to Eq. (7). This completes the proof of the lemma. □

Lemma 4.1 shows that the upper bound defined in Eq. (13) with the inactive constraints pruned as above decreases monotonically. As the lower bound defined in Eq. (11) always increases monotonically, the proposed pruning strategy retains the convergence property in Theorem 4.1.

5. Relationship with Multiple Kernel Learning

We show the close relationship between the proposed SIQCLP formulation in Eq. (6) and the multiple kernel learning formulation in (Lanckriet et al., 2004).

For a given set of kernel matrices $\{K_i\}_{i=1}^p$ and a class label vector y, Lanckriet et al. (2004) propose to learn an optimal convex combination of the p pre-specified kernel matrices by solving the following optimization problem:

$$\begin{aligned} \min_{\{\theta_i\}} \max_{\alpha \in \mathbb{R}^d} \quad & \alpha^T e - \frac{1}{2}\alpha^T Y \left(\sum_{i=1}^p \theta_i K_i\right) Y\alpha \\ \text{subject to} \quad & \sum_{i=1}^p \theta_i \ \text{tr}(K_i) = 1, \\ & \alpha^T y = 0, \ \ 0 \leq \alpha \leq C, \end{aligned} \tag{25}$$

where $Y = \text{diag}(y)$, and C is the pre-specified parameter.

Recall that K_0 is a indefinite kernel matrix. For each of the given PSD kernel matrix K_i, we denote

$$\mu_i = \|K_i - K_0\|_F^2, \ \ i = 1, \cdots, p, \tag{26}$$

where μ_i measures the distance between K_i and K_0 in terms of Frobenius norm. Consider a regularized version of the optimization problem in Eq. (25) given by:

$$\begin{aligned} \min_{\{\theta_i\}} \max_{\alpha \in \mathbb{R}^d} \quad & \alpha^T e - \frac{1}{2}\alpha^T Y \left(\sum_{i=1}^p \theta_i K_i\right) Y\alpha + \rho \sum_{i=1}^p \theta_i \mu_i \\ \text{subject to} \quad & \sum_{i=1}^p \theta_i = 1, \ \ \alpha^T y = 0, \ \ 0 \leq \alpha \leq C, \end{aligned} \tag{27}$$

where ρ is the pre-specified parameter as in Eq. (6). The optimization problem in Eq. (27) computes an optimal linear combination of the p pre-specified kernel matrices by maximizing the margin for SVM classification, while penalizing kernels with a large deviation from K_0.

The following theorem shows the equivalence relationship between the regularized MKL problem in Eq. (27) and the SIQCLP formulation in Eq. (7).

Theorem 5.1. *Let $\{K_i\}_{i=1}^p$ be a set of pre-specified PSD kernel matrices. Then the optimization problem in Eq. (7) is equivalent to the one in Eq. (27).*

Proof. Since all constraints in Eq. (27) are linear and the objective is convex on $\{\theta_i\}$ and concave on α, the minimization and the maximization in Eq. (27) can be exchanges. This leads to the following optimization problem:

$$\begin{aligned} & \max_{\alpha \in \mathbb{R}^d} \min_{K \succeq 0} \alpha^T e - \frac{1}{2}\alpha^T YKY\alpha + \rho \sum_{i=1}^p \theta_i \mu_i \\ = \ & \max_{\alpha \in \mathbb{R}^d} \min_{\sum \theta_i = 1} \alpha^T e - \frac{1}{2}\alpha^T Y \sum_{i=1}^p \theta_i K_i Y\alpha + \rho \sum_{i=1}^p \theta_i \mu_i \\ = \ & \max_{\alpha \in \mathbb{R}^d} \min_{\sum \theta_i = 1} \left(\sum_{i=1}^p \theta_i t_i\right), \end{aligned} \tag{28}$$

where t_i is defined as

$$t_i = \left(\alpha^T e - \frac{1}{2}\alpha^T Y K_i Y\alpha + \rho\mu_i\right). \tag{29}$$

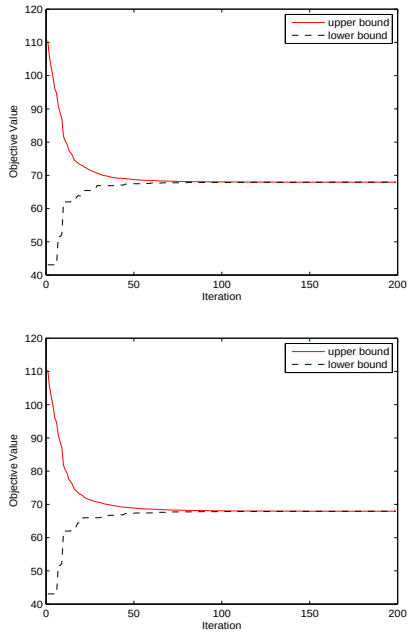


Figure 1. Convergence of the proposed algorithms without the pruning strategy applied (top graph) and with the pruning strategy applied (bottom graph).

From Eq. (28) and Eq. (29), the optimization problem in Eq. (27) can be reformulated as:

$$\begin{aligned} \max_{\alpha\in\mathbb{R}^d,\ t\in\mathbb{R}} \quad & t \\ \text{subject to} \quad & \alpha^T y = 0,\ \ 0 \le \alpha \le C, \\ & t \le \alpha^T e - \frac{1}{2}\alpha^T Y K_i Y \alpha + \rho\mu_i, \\ & i = 1, \cdots, p, \end{aligned}$$

which is equivalent to the SIQCLP formulation given in Eq. (7). We complete the proof of this theorem. □

The equivalent result in Theorem 5.1 implies that our proposed SIQCLP formulation in Eq. (6) can be solved by recycling existing efficient MKL implementations (Rakotomamonjy et al., 2007; Sonnenburg et al., 2006).

6. Experiments

We experimentally evaluate the convergence property of the proposed algorithms. We also compare the proposed algorithms with other representative ones using a collection of benchmark data sets.

6.1. Experimental Setup

We use several benchmark data sets from the UCI repository (Newman et al., 1998) including Sonar, Ionosphere, Breast Cancer, and Diabetes, as well as USPS (Hull, 1994) and Heart[1]. For USPS, we select two classes corresponding to two digits 3 and 5, and randomly select 600 samples for each digit.

In our simulation study, we first generate Gaussian kernels from the data with the parameter value estimated via cross-validation and then construct indefinite kernels through perturbation. More specifically, we randomly generate a matrix E with zero mean and identity covariance matrix, and then apply $\xi\hat{E}$ as the perturbation, where $\hat{E} = (E + E^T)/2$ and $\xi > 0$ is small constant. We set $C = 1$ in SVM. The value of ρ is estimated via cross-validation.

6.2. Convergence

In this experiment, we empirically evaluate the convergence property of the proposed algorithms with and without pruning. We also investigate the number of kernel matrices involved when the pruning strategy is employed. We use the sonar data set for this study, and the perturbation matrix is set to be $0.1\hat{E}$.

The results are presented in Figure 1. The top graph in Figure 1 shows the convergence of the upper bound as well as the lower bound of the objective value when the algorithm without the pruning strategy is applied. The bottom graph shows the convergence result for the case when the pruning strategy is applied. We can observe from the figure that the upper bound and lower bound curves approach each other gradually during the iteration. More specifically, the upper bound monotonically decreases, while the lower bound monotonically increases, both approaching the optimal objective value. This is consistent with our convergence results in Section 4.1. Interestingly, our results show that the proposed algorithms with or without the pruning strategy applied result in a similar convergent rate. We further observe that the gap between the upper and lower bound is less that 10^{-2} after about 150 iterations, and it takes about 600 iterations to attain a gap smaller than 10^{-5}.

Figure 2 shows the number of kernels (the size of the localization set) involved at each iteration. We can observe from the figure that with the pruning strategy, the number of kernels involved in the algorithm stabilizes around a small constant. In contrast, this number increases gradually when the pruning strategy is not applied. These results demonstrate the advantage of the proposed pruning strategy.

We show in Figure 3 the generalization performance (measured by classification accuracy) of the proposed algorithm with pruning at each of the first 70 iterations. We observe a large variation at the first few iterations, while the accuracy becomes more stable after about 40 iterations. We further run the algorithm until convergence and the resulting accu-

[1] http://www.is.umk.pl/projects/datasets-stat.html#heart

Data Set	Size	λ^- num.	λ^+ num.	λ_{min}	λ_{max}	$\lvert\lambda_{max}/\lambda_{min}\rvert$	Denoise	Flip	Shift	SVM	Indefinite SVM
Sonar	208	57.41	150.62	-1.36	18.42	13.55	78.57	79.52	78.10	72.86	80.95
Ionosphere	351	169.62	181.45	-25.50	94.49	3.71	75.57	71.43	71.41	68.00	77.43
Breast Cancer	683	323.21	359.82	-3.51	390.52	111.26	95.38	95.62	95.38	89.54	95.36
Heart	270	125.57	144.71	-10.96	42.93	3.92	71.02	67.28	65.42	65.43	72.22
USPS-3-5	1200	520.12	680.31	-3.54	81.99	23.16	96.25	96.88	95.63	96.11	96.81
Diabetes	768	381.22	385.18	-3.93	8.13	2.06	68.83	64.28	62.98	66.23	70.08

Table 1. Comparison of the proposed algorithm with other representative algorithms in terms of classification accuracy (in percentage). All values shown in the table are the averaged ones over 10 partitions of the data into training and test sets with a ratio $4:1$. λ^- num (λ^+ num) denotes the number of negative (positive) eigenvalues; λ_{min} (λ_{max}) denotes the minimum (maximum) eigenvalue; $|\lambda_{max}/\lambda_{min}|$ denotes the ratio of the absolute values of λ_{max} and λ_{min}. SVM refers to applying indefinite kernel in the SVM formulation directly, while indefinite SVM refers to our proposed algorithm with the pruning strategy applied.

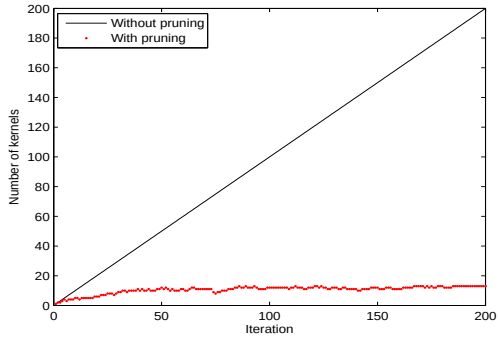


Figure 2. The number of kernel matrices involved for the proposed algorithms with and without the pruning.

racy is about 76%. We obtain a similar observation from other data sets. This implies that an early-stopping strategy could be employed for the proposed algorithm.

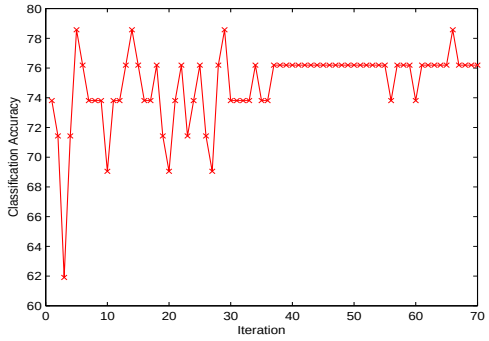


Figure 3. Generalization performance (measured by classification accuracy in percentage) of the proposed algorithm with pruning.

6.3. Classification Performance

In this experiment, we compare our proposed algorithms (Indefinite SVM) with other representative ones including Denoise, Flip, Shift, and SVM using indefinite kernels in terms of classification accuracy. The presented experimental results are averaged over 10 random partitions of the data into a training and a test set with a ratio $4:1$.

The experimental results are summarized in Table 1. We also report the maximum and minimum eigenvalues of the indefinite kernel matrix in the table. We can observe from the table that Indefinite SVM is competitive with all other algorithms in most cases. It outperforms all other algorithms on the Sonar, Ionosphere, Heart, and Diabetes data sets, where the perturbed kernel matrix has a relatively small ratio $|\lambda_{max}/\lambda_{min}|$. For the other two data sets including Breast Cancer and USPS-3-5, where the perturbed kernel matrix has a relatively large ratio $|\lambda_{max}/\lambda_{min}|$, Indefinite SVM is comparable to the best among all other algorithms. These results demonstrate the effectiveness of the proposed learning algorithm, especially when the indefinite kernel matrix is highly non-PSD. A similar trend has been observed in (Luss & d'Aspremont, 2007).

7. Conclusion

In this paper, we study the problem of training SVM with an indefinite kernel matrix following the formulation in (Luss & d'Aspremont, 2007). We propose a semi-infinite quadratically constrained linear program formulation, which can be solved iteratively. The algorithm alternates between the computation of an intermediate SVM solution by solving a quadratically constraint linear program with a subset of constraints, and the computation of the new constraint set based on the obtained intermediate SVM solution. We further propose to improve the efficiency of the iterative algorithm by pruning inactive constraints at each iteration. We show that such pruning will not affect the convergence property of the algorithm. In addition, we show the close relationship between the proposed SIQCLP formulation and multiple kernel learning. The presented analysis provides new insights into the nature of this learning formulation.

We have performed a simulation study using a collection of benchmark data sets. Our results verify the convergence property of the proposed algorithms. Our empirical results show that the proposed algorithms with or with-

out the pruning strategy applied result in a similar convergent rate, while a much smaller number of kernel matrices are involved when the pruning strategy is applied. Our results also demonstrate the favorable performance of the proposed algorithms in terms of classification accuracy in comparison with several other representative algorithms. Our future works include the analysis of the convergence rate of the proposed algorithms similar to the analysis conducted in (Teo et al., 2007), the estimation of the regularization parameter ρ, and the application of the proposed algorithms to real-world applications involving indefinite kernels such as protein sequence and structure analysis based on various sequence/structure alignment measures.

Acknowledgments

This research is sponsored in part by funds from the Arizona State University and the National Science Foundation under Grant No. IIS-0612069.

References

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix computations*. Johns Hopkins University Press. 3 edition.

Graepel, T., Herbrich, R., Bollmann-Sdorra, P., & Obermayer, K. (1998). Classification on pairwise proximity data. *NIPS* (pp. 438–444).

Haasdonk, B. (2005). Feature space interpretation of svms with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 482–492.

Hettich, R., & Kortanek, K. O. (1993a). Semi-infinite programming: theory, methods, and applications. *SIAM Review*, *35*, 380–429.

Hettich, R., & Kortanek, K. O. (1993b). Semi-infinite programming: Theory, methods, and applications. *SIAM Review*, *35*, 380–429.

Hiriart-Urruty, J.-B., & Lemarechal, C. (1993). *Convex analysis and minimization algorithms II*. Springer.

Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*, 550–554.

Kondor, R. I., & Lafferty, J. D. (2002). Diffusion kernels on graphs and other discrete input spaces. *ICML* (pp. 315–322).

Lanckriet, G. R. G., Cristianini, N., Bartlett, P. L., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, *5*, 27–72.

Lin, H.-T., & Lin, C.-J. (2003). A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. *Technical Report, National Taiwan University*.

Luss, R., & d'Aspremont, A. (2007). Support vector machine classification with indefinite kernels. *NIPS*.

Newman, D., Hettich, S., Blake, C., & Merz, C. (1998). UCI repository of machine learning databases.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization springer series in operations research*. Springer.

Ong, C. S., Mary, X., Canu, S., & Smola, A. J. (2004). Learning with non-positive kernels. *ICML*.

Pekalska, E., Paclik, P., & Duin, R. P. W. (2002). A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, *2*, 175–211.

Qamra, A., Meng, Y., & Chang, E. Y. (2005). Enhanced perceptual distance functions and indexing for image replica recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 379–391.

Rakotomamonjy, A., Bach, F., Canu, S., & Grandvalet, Y. (2007). More efficiency in multiple kernel learning. *ICML*.

Roth, V., Laub, J., Kawanabe, M., & Buhmann, J. M. (2003a). Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 1540–1551.

Roth, V., Laub, J., Kawanabe, M., & Buhmann, J. M. (2003b). Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 1540–1551.

Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. The MIT Press.

Shimodaira, H., Noma, K.-I., Nakai, M., & Sagayama, S. (2001). Dynamic time-alignment kernel in support vector machine. *NIPS* (pp. 921–928).

Smola, A. J., Óvári, Z. L., & Williamson, R. C. (2000). Regularization with dot-product kernels. *NIPS* (pp. 308–314).

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, *7*, 1531–1565.

Teo, C. H., Smola, A., Vishwanathan, S. V., & Le, Q. V. (2007). A scalable modular convex solver for regularized risk minimization. *KDD* (pp. 727–736).

Tong, S., & Chang, E. (2001). Support vector machine active learning for image retrieval. *ACM Multimedia* (pp. 107–118).

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc.

Wu, G., Zhang, Z., & Chang, E. Y. (2005). An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. *Technical Report, UCSB*.

Learning for Control from Multiple Demonstrations

Adam Coates ACOATES@CS.STANFORD.EDU
Pieter Abbeel PABBEEL@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Stanford University CS Department, 353 Serra Mall, Stanford, CA 94305 USA

Abstract

We consider the problem of learning to follow a desired trajectory when given a small number of demonstrations from a sub-optimal expert. We present an algorithm that (i) extracts the—initially unknown—desired trajectory from the sub-optimal expert's demonstrations and (ii) learns a local model suitable for control along the learned trajectory. We apply our algorithm to the problem of autonomous helicopter flight. In all cases, the autonomous helicopter's performance exceeds that of our expert helicopter pilot's demonstrations. Even stronger, our results significantly extend the state-of-the-art in autonomous helicopter aerobatics. In particular, our results include the first autonomous tic-tocs, loops and hurricane, vastly superior performance on previously performed aerobatic maneuvers (such as in-place flips and rolls), and a complete airshow, which requires autonomous transitions between these and various other maneuvers.

1. Introduction

Many tasks in robotics can be described as a trajectory that the robot should follow. Unfortunately, specifying the desired trajectory and building an appropriate model for the robot dynamics along that trajectory are often non-trivial tasks. For example, when asked to describe the trajectory that a helicopter should follow to perform an aerobatic flip, one would have to specify a trajectory that (i) corresponds to the aerobatic flip task, and (ii) is consistent with the helicopter's dynamics. The latter requires (iii) an accurate helicopter dynamics model for all of the flight regimes encountered in the vicinity of the trajectory. These coupled tasks are non-trivial for systems with complex dynamics, such as helicopters. Failing to adequately address these points leads to a significantly more difficult control problem.

In the apprenticeship learning setting, where an expert is available, rather than relying on a hand-engineered target trajectory, one can instead have the expert demonstrate the desired trajectory. The expert demonstration yields both a desired trajectory for the robot to follow, as well as data to build a dynamics model in the vicinity of this trajectory. Unfortunately, perfect demonstrations can be hard (if not impossible) to obtain. However, repeated expert demonstrations are often suboptimal in different ways, suggesting that a large number of suboptimal expert demonstrations could implicitly encode the ideal trajectory the suboptimal expert is trying to demonstrate.

In this paper we propose an algorithm that approximately extracts this implicitly encoded optimal demonstration from multiple suboptimal expert demonstrations, and then builds a model of the dynamics in the vicinity of this trajectory suitable for high-performance control. In doing so, the algorithm learns a target trajectory and a model that allows the robot to not only mimic the behavior of the expert but even perform significantly better.

Properly extracting the underlying ideal trajectory from a set of suboptimal trajectories requires a significantly more sophisticated approach than merely averaging the states observed at each time-step. A simple arithmetic average of the states would result in a trajectory that does not even obey the constraints of the dynamics model. Also, in practice, each of the demonstrations will occur at different rates so that attempting to combine states from the same time-step in each trajectory will not work properly.

We propose a generative model that describes the expert demonstrations as noisy observations of the unobserved, intended target trajectory, where each demonstration is possibly warped along the time axis. We present an EM algorithm—which uses a (extended) Kalman smoother and an efficient dynamic programming algorithm to perform the E-step—to both infer the unobserved, intended target trajectory and a time-alignment of all the demonstrations. The time-aligned demonstrations provide the appropriate data to learn

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

good local models in the vicinity of the trajectory—such trajectory-specific local models tend to greatly improve control performance.

Our algorithm allows one to easily incorporate prior knowledge to further improve the quality of the learned trajectory. For example, for a helicopter performing in-place flips, it is known that the helicopter can be roughly centered around the same position over the entire sequence of flips. Our algorithm incorporates this prior knowledge, and successfully factors out the position drift in the expert demonstrations.

We apply our algorithm to learn trajectories and dynamics models for aerobatic flight with a remote controlled helicopter. Our experimental results show that (i) our algorithm successfully extracts a good trajectory from the multiple sub-optimal demonstrations, and (ii) the resulting flight performance significantly extends the state of the art in aerobatic helicopter flight (Abbeel et al., 2007; Gavrilets et al., 2002). Most importantly, our resulting controllers are the first to perform as well, and often even better, than our expert pilot.

We posted movies of our autonomous helicopter flights at:

`http://heli.stanford.edu`

The remainder of this paper is organized as follows: Section 2 presents our generative model for (multiple) suboptimal demonstrations; Section 3 describes our trajectory learning algorithm in detail; Section 4 describes our local model learning algorithm; Section 5 describes our helicopter platform and experimental results; Section 6 discusses related work.

2. Generative Model

2.1. Basic Generative Model

We are given M demonstration trajectories of length N^k, for $k = 0..M-1$. Each trajectory is a sequence of states, s_j^k, and control inputs, u_j^k, composed into a single state vector:

$$y_j^k = \begin{bmatrix} s_j^k \\ u_j^k \end{bmatrix}, \text{ for } j = 0..N^k - 1, k = 0..M-1.$$

Our goal is to estimate a "hidden" target trajectory of length T, denoted similarly:

$$z_t = \begin{bmatrix} s_t^\star \\ u_t^\star \end{bmatrix}, \text{ for } t = 0..T-1.$$

We use the following notation: $\mathbf{y} = \{y_j^k \mid j = 0..N^k - 1, k = 0..M-1\}$, $\mathbf{z} = \{z_t \mid t = 0..T-1\}$, and similarly for other indexed variables.

The generative model for the ideal trajectory is given by an initial state distribution $z_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ and an approximate model of the dynamics

$$z_{t+1} = f(z_t) + \omega_t^{(z)}, \quad \omega_t^{(z)} \sim \mathcal{N}(0, \Sigma^{(z)}). \tag{1}$$

The dynamics model does not need to be particularly accurate—in our experiments, we use a single generic model learned from a large corpus of data that is not specific to the trajectory we want to perform. In our experiments (Section 5) we provide some concrete examples showing how accurately the generic model captures the true dynamics for our helicopter.[1]

Our generative model represents each demonstration as a set of independent "observations" of the hidden, ideal trajectory $\mathbf{z}$. Specifically, our model assumes

$$y_j^k = z_{\tau_j^k} + \omega_j^{(y)}, \quad \omega_j^{(y)} \sim \mathcal{N}(0, \Sigma^{(y)}). \tag{2}$$

Here τ_j^k is the time index in the hidden trajectory to which the observation y_j^k is mapped. The noise term in the observation equation captures both inaccuracy in estimating the observed trajectories from sensor data, as well as errors in the maneuver that are the result of the human pilot's imperfect demonstration.[2]

The time indices τ_j^k are unobserved, and our model assumes the following distribution with parameters d_i^k:

$$\mathbb{P}(\tau_{j+1}^k | \tau_j^k) = \begin{cases} d_1^k & \text{if } \tau_{j+1}^k - \tau_j^k = 1 \\ d_2^k & \text{if } \tau_{j+1}^k - \tau_j^k = 2 \\ d_3^k & \text{if } \tau_{j+1}^k - \tau_j^k = 3 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\tau_0^k \equiv 0. \tag{4}$$

To accommodate small, gradual shifts in time between the hidden and observed trajectories, our model assumes the observed trajectories are subsampled versions of the hidden trajectory. We found that having a hidden trajectory length equal to twice the average length of the demonstrations, i.e., $T = 2(\frac{1}{M}\sum_{k=1}^{M} N^k)$, gives sufficient resolution.

Figure 1 depicts the graphical model corresponding to our basic generative model. Note that each observation y_j^k depends on the hidden trajectory's state at time τ_j^k, which means that for τ_j^k unobserved, y_j^k depends on all states in the hidden trajectory that it could be associated with.

2.2. Extensions to the Generative Model

Thus far we have assumed that the expert demonstrations are misaligned copies of the ideal trajectory

[1] The state transition model also predicts the controls as a function of the previous state and controls. In our experiments we predict $u_{t+1}^\star$ as $u_t^\star$ plus Gaussian noise.

[2] Even though our observations, $\mathbf{y}$, are correlated over time with each other due to the dynamics governing the observed trajectory, our model assumes that the observations y_j^k are independent for all $j = 0..N^k - 1$ and $k = 0..M-1$.

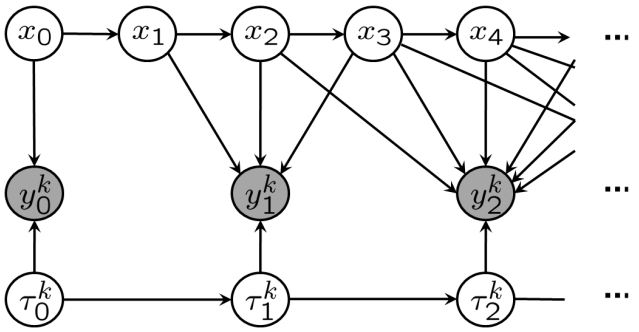


Figure 1. Graphical model representing our trajectory assumptions. (Shaded nodes are observed.)

merely corrupted by Gaussian noise. Listgarten et al. have used this same basic generative model (for the case where $f(\cdot)$ is the identity function) to align speech signals and biological data (Listgarten, 2006; Listgarten et al., 2005). We now augment the basic model to account for other sources of error which are important for modeling and control.

2.2.1. Learning Local Model Parameters

For many systems, we can substantially improve our modeling accuracy by using a time-varying model $f_t(\cdot)$ that is specific to the vicinity of the intended trajectory at each time t. We express f_t as our "crude" model, f, augmented with a bias term[3], $\beta_t^\star$:

$$z_{t+1} = f_t(z_t) + \omega_t^{(z)} \equiv f(z_t) + \beta_t^\star + \omega_t^{(z)}.$$

To regularize our model, we assume that $\beta_t^\star$ changes only slowly over time. We have $\beta_{t+1}^\star \sim \mathcal{N}(\beta_t^\star, \Sigma^{(\beta)})$.

We incorporate the bias into our observation model by computing the observed bias $\beta_j^k = y_j^k - f(y_{j-1}^k)$ for each of the observed state transitions, and modeling this as a direct observation of the "true" model bias corrupted by Gaussian noise. The result of this modification is that the ideal trajectory must not only look similar to the demonstration trajectories, but it must also obey a dynamics model which includes those errors consistently observed in the demonstrations.

2.2.2. Factoring out Demonstration Drift

It is often difficult, even for an expert pilot, during aerobatic maneuvers to keep the helicopter centered around a fixed position. The recorded position trajectory will often drift around unintentionally. Since these position errors are highly correlated, they are not explained well by the Gaussian noise term in our observation model.

To capture such slow drift in the demonstrated trajectories, we augment the latent trajectory's state with a "drift" vector δ_t^k for each time t and each demonstrated trajectory k. We model the drift as a zero-mean random walk with (relatively) small variance. The state observations are now noisy measurements of $z_t + \delta_t^k$ rather than merely z_t.

2.2.3. Incorporating Prior Knowledge

Even though it might be hard to specify the complete ideal trajectory in state space, we might still have prior knowledge about the trajectory. Hence, we introduce additional observations $\rho_t = \rho(z_t)$ corresponding to our prior knowledge about the ideal trajectory at time t. The function $\rho(z_t)$ computes some features of the hidden state z_t and our expert supplies the value ρ_t that this feature should take. For example, for the case of a helicopter performing an in-place flip, we use an observation that corresponds to our expert pilot's knowledge that the helicopter should stay at a fixed position while it is flipping. We assume that these observations may be corrupted by Gaussian noise, where the variance of the noise expresses our confidence in the accuracy of the expert's advice. In the case of the flip, the variance expresses our knowledge that it is, in fact, impossible to flip perfectly in-place and that the actual position of the helicopter may vary slightly from the position given by the expert.

Incorporating prior knowledge of this kind can greatly enhance the learned ideal trajectory. We give more detailed examples in Section 5.

2.2.4. Model Summary

In summary, we have the following generative model:

$$\begin{aligned}
z_{t+1} &= f(z_t) + \beta_t^\star + \omega_t^{(z)}, && (5)\\
\beta_{t+1}^\star &= \beta_t^\star + \omega_t^{(\beta)}, && (6)\\
\delta_{t+1}^k &= \delta_t^k + \omega_t^{(\delta)}, && (7)\\
\rho_t &= \rho(z_t) + \omega_t^{(\rho)}, && (8)\\
y_j^k &= z_{\tau_j^k} + \delta_j^k + \omega_j^{(y)}, && (9)\\
\tau_j^k &\sim \mathbb{P}(\tau_{j+1}^k | \tau_j^k) && (10)
\end{aligned}$$

Here $\omega_t^{(z)}, \omega_t^{(\beta)}, \omega_t^{(\delta)}, \omega_t^{(\rho)}, \omega_j^{(y)}$ are zero mean Gaussian random variables with respective covariance matrices $\Sigma^{(z)}, \Sigma^{(\beta)}, \Sigma^{(\delta)}, \Sigma^{(\rho)}, \Sigma^{(y)}$. The transition probabilities for τ_j^k are defined by Eqs. (3, 4) with parameters d_1^k, d_2^k, d_3^k (collectively denoted $\mathbf{d}$).

3. Trajectory Learning Algorithm

Our learning algorithm automatically finds the time-alignment indexes $\boldsymbol{\tau}$, the time-index transition probabilities $\mathbf{d}$, and the covariance matrices $\Sigma^{(\cdot)}$ by (approximately) maximizing the joint likelihood of the observed trajectories $\mathbf{y}$ and the observed prior knowl-

[3]Our generative model can incorporate richer local models. We discuss our choice of merely using biases in our generative trajectory model in more detail in Section 4.

edge about the ideal trajectory $\boldsymbol{\rho}$, while marginalizing out over the unobserved, intended trajectory $\mathbf{z}$. Concretely, our algorithm (approximately) solves

$$\max_{\boldsymbol{\tau},\Sigma^{(\cdot)},\mathbf{d}} \log \mathbb{P}(\mathbf{y}, \boldsymbol{\rho}, \boldsymbol{\tau} \; ; \; \Sigma^{(\cdot)}, \mathbf{d}). \tag{11}$$

Then, once our algorithm has found $\boldsymbol{\tau}, \mathbf{d}, \Sigma^{(\cdot)}$, it finds the most likely hidden trajectory, namely the trajectory $\mathbf{z}$ that maximizes the joint likelihood of the observed trajectories $\mathbf{y}$ and the observed prior knowledge about the ideal trajectory $\boldsymbol{\rho}$ for the learned parameters $\boldsymbol{\tau}, \mathbf{d}, \Sigma^{(\cdot)}$.[4]

The joint optimization in Eq. (11) is difficult because (as can be seen in Figure 1) the lack of knowledge of the time-alignment index variables $\boldsymbol{\tau}$ introduces a very large set of dependencies between all the variables. However, when $\boldsymbol{\tau}$ is known, the optimization problem in Eq. (11) greatly simplifies thanks to context specific independencies (Boutilier et al., 1996). When $\boldsymbol{\tau}$ is fixed, we obtain a model such as the one shown in Figure 2. In this model we can directly estimate the multinomial parameters $\mathbf{d}$ in closed form; and we have a standard HMM parameter learning problem for the covariances $\Sigma^{(\cdot)}$, which can be solved using the EM algorithm (Dempster et al., 1977)—often referred to as Baum-Welch in the context of HMMs. Concretely, for our setting, the EM algorithm's E-step computes the pairwise marginals over sequential hidden state variables by running a (extended) Kalman smoother; the M-step then uses these marginals to update the covariances $\Sigma^{(\cdot)}$.

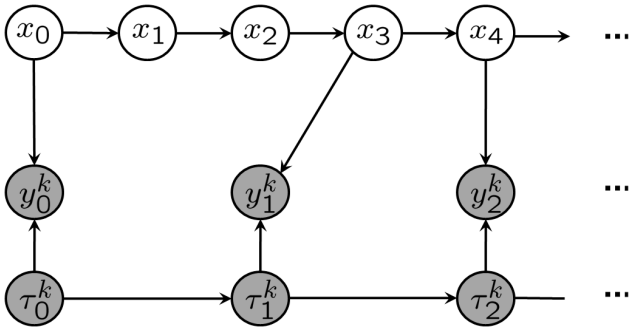


Figure 2. Example of graphical model when $\boldsymbol{\tau}$ is known. (Shaded nodes are observed.)

To also optimize over the time-indexing variables $\boldsymbol{\tau}$, we propose an alternating optimization procedure. For fixed $\Sigma^{(\cdot)}$ and $\mathbf{d}$, and for fixed $\mathbf{z}$, we can find the optimal time-indexing variables $\boldsymbol{\tau}$ using dynamic programming over the time-index assignments for each demonstration independently. The dynamic programming algorithm to find $\boldsymbol{\tau}$ is known in the speech recognition literature as dynamic time warping (Sakoe & Chiba, 1978) and in the biological sequence alignment literature as the Needleman-Wunsch algorithm (Needleman & Wunsch, 1970). The fixed $\mathbf{z}$ we use, is the one that maximizes the likelihood of the observations for the current setting of parameters $\boldsymbol{\tau}, \mathbf{d}, \Sigma^{(\cdot)}$.[5]

In practice, rather than alternating between complete optimizations over $\Sigma^{(\cdot)}, \mathbf{d}$ and $\boldsymbol{\tau}$, we only partially optimize over $\Sigma^{(\cdot)}$, running only one iteration of the EM algorithm.

We provide the complete details of our algorithm in the full paper (Coates et al., 2008).

4. Local Model Learning

For complex dynamical systems, the state z_t used in the dynamics model often does not correspond to the "complete state" of the system, since the latter could involve large numbers of previous states or unobserved variables that make modeling difficult.[6] However, when we only seek to model the system dynamics along a specific trajectory, knowledge of both z_t and how far we are along that trajectory is often sufficient to accurately predict the next state z_{t+1}.

Once the alignments between the demonstrations are computed by our trajectory learning algorithm, we can use the time aligned demonstration data to learn a sequence of trajectory-specific models. The time indices of the aligned demonstrations now accurately associate the demonstration data points with locations along the learned trajectory, allowing us to build models for the state at time t using the appropriate corresponding data from the demonstration trajectories.[7]

[4]Note maximizing over the hidden trajectory and the covariance parameters simultaneously introduces undesirable local maxima: the likelihood score would be highest (namely infinity) for a hidden trajectory with a sequence of states exactly corresponding to the (crude) dynamics model $f(\cdot)$ and state-transition covariance matrices equal to all-zeros as long as the observation covariances are nonzero. Hence we marginalize out the hidden trajectory to find $\boldsymbol{\tau}, \mathbf{d}, \Sigma^{(\cdot)}$.

[5]Fixing $\mathbf{z}$ means the dynamic time warping step only approximately optimizes the original objective. Unfortunately, without fixing $\mathbf{z}$, the independencies required to obtain an efficient dynamic programming algorithm do not hold. In practice we find our approximation works very well.

[6]This is particularly true for helicopters. Whereas the state of the helicopter is very crudely captured by the 12D rigid-body state representation we use for our controllers, the "true" physical state of the system includes, among others, the airflow around the helicopter, the rotor head speed, and the actuator dynamics.

[7]We could learn the richer local model within the trajectory alignment algorithm, updating the dynamics model during the M-step. We chose not to do so since these models are more computationally expensive to estimate. The richer local models have minimal influence on the alignment because the biases capture the average model error—the richer models capture the derivatives around it. Given the limited influence on the alignment, we chose to save computational time and only estimate the richer models

Figure 3. Our XCell Tempest autonomous helicopter.

To construct an accurate nonlinear model to predict z_{t+1} from z_t, using the aligned data, one could use locally weighted linear regression (Atkeson et al., 1997), where a linear model is learned based on a weighted dataset. Data points from our aligned demonstrations that are nearer to the current time index along the trajectory, t, and nearer the current state, z_t, would be weighted more highly than data far away. While this allows us to build a more accurate model from our time-aligned data, the weighted regression must be done online, since the weights depend on the current state, z_t. For performance reasons[8] this may often be impractical. Thus, we weight data only based on the time index, and learn a parametric model in the remaining variables (which, in our experiments, has the same form as the global "crude" model, $f(\cdot)$). Concretely, when estimating the model for the dynamics at time t, we weight a data point at time t' by:[9]

$$W(t') = \exp\left(-\frac{(t-t')^2}{\sigma^2}\right),$$

where σ is a bandwidth parameter. Typical values for σ are between one and two seconds in our experiments. Since the weights for the data points now only depend on the time index, we can precompute all models $f_t(\cdot)$ along the entire trajectory. The ability to precompute the models is a feature crucial to our control algorithm, which relies heavily on fast simulation.

5. Experimental Results

5.1. Experimental Setup

To test our algorithm, we had our expert helicopter pilot fly our XCell Tempest helicopter (Figure 3), which can perform professional, competition-level maneuvers.[10]

We collected multiple demonstrations from our expert for a variety of aerobatic trajectories: continuous in-place flips and rolls, a continuous tail-down "tic toc," and an airshow, which consists of the following maneuvers in rapid sequence: split-S, snap roll, stall-turn, loop, loop with pirouette, stall-turn with pirouette, "hurricane" (fast backward funnel), knife-edge, flips and rolls, tic-toc and inverted hover.

The (crude) helicopter dynamics $f(\cdot)$ is constructed using the method of Abbeel et al. (2006a).[11] The helicopter dynamics model predicts linear and angular accelerations as a function of current state and inputs. The next state is then obtained by integrating forward in time using the standard rigid-body equations.

In the trajectory learning algorithm, we have bias terms $\beta_t^\star$ for each of the predicted accelerations. We use the state-drift variables, δ_t^k, for position only.

For the flips, rolls, and tic-tocs we incorporated our prior knowledge that the helicopter should stay in place. We added a measurement of the form:

$$0 = p(z_t) + \omega^{(\rho_0)}, \quad \omega^{(\rho_0)} \sim \mathcal{N}(0, \Sigma^{(\rho_0)})$$

where $p(\cdot)$ is a function that returns the position coordinates of z_t, and $\Sigma^{(\rho_0)}$ is a diagonal covariance matrix. This measurement—which is a direct observation of the pilot's intended trajectory—is similar to advice given to a novice human pilot to describe the desired maneuver: A good flip, roll, or tic-toc trajectory stays close to the same position.

We also used additional advice in the airshow to indicate that the vertical loops, stall-turns and split-S should all lie in a single vertical plane; that the hurricanes should lie in a horizontal plane and that a good knife-edge stays in a vertical plane. These measurements take the form:

$$c = N^\top p(z_t) + \omega^{(\rho_1)}, \quad \omega^{(\rho_1)} \sim \mathcal{N}(0, \Sigma^{(\rho_1)})$$

where, again, $p(z_t)$ returns the position coordinates of z_t. N is a vector normal to the plane of the maneuver, c is a constant, and $\Sigma^{(\rho_1)}$ is a diagonal covariance matrix.

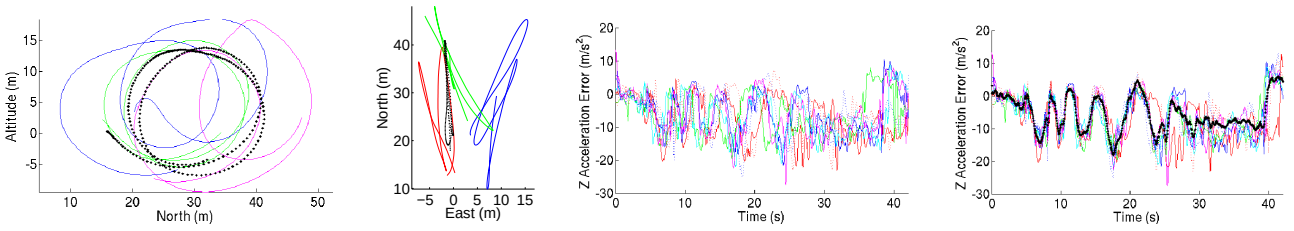
after alignment.

[8]During real-time control execution, our model is queried roughly 52000 times per second. Even with KD-tree or cover-tree data structures a full locally weighted model would be much too slow.

[9]In practice, the data points along a short segment of the trajectory lie in a low-dimensional subspace of the state space. This sometimes leads to an ill-conditioned parameter estimation problem. To mitigate this problem, we regularize our models toward the "crude" model $f(\cdot)$.

[10]We instrumented the helicopter with a Microstrain 3DM-GX1 orientation sensor. A ground-based camera system measures the helicopter's position. A Kalman filter uses these measurements to track the helicopter's position, velocity, orientation and angular rate.

[11]The model of Abbeel et al. (2006a) naturally generalizes to any orientation of the helicopter regardless of the flight regime from which data is collected. Hence, even without collecting data from aerobatic flight, we can reasonably attempt to use such a model for aerobatic flying, though we expect it to be relatively inaccurate.



(a) (b) (c) (d)

Figure 4. Colored lines: demonstrations. Black dotted line: trajectory inferred by our algorithm. (See text for details.)

5.2. Trajectory Learning Results

Figure 4(a) shows the horizontal and vertical position of the helicopter during the two loops flown during the airshow. The colored lines show the expert pilot's demonstrations. The black dotted line shows the inferred ideal path produced by our algorithm. The loops are more rounded and more consistent in the inferred ideal path. We did not incorporate any prior knowledge to this extent. Figure 4(b) shows a top-down view of the same demonstrations and inferred trajectory. The prior successfully encouraged the inferred trajectory to lie in a vertical plane, while obeying the system dynamics.

Figure 4(c) shows one of the bias terms, namely the model prediction errors for the Z-axis acceleration of the helicopter computed from the demonstrations, before time-alignment. Figure 4(d) shows the result after alignment (in color) as well as the inferred acceleration error (black dotted). We see that the unaligned bias measurements allude to errors approximately in the -1G to -2G range for the first 40 seconds of the airshow (a period that involves high-G maneuvering that is not predicted accurately by the "crude" model). However, only the aligned biases precisely show the magnitudes and locations of these errors along the trajectory. The alignment allows us to build our ideal trajectory based upon a much more accurate model that is tailored to match the dynamics observed in the demonstrations.

Results for other maneuvers and state variables are similar. At the URL provided in the introduction we posted movies which simultaneously replay the different demonstrations, before alignment and after alignment. The movies visualize the alignment results in many state dimensions simultaneously.

5.3. Flight Results

After constructing the idealized trajectory and models using our algorithm, we attempted to fly the trajectory on the actual helicopter.

Our helicopter uses a receding-horizon differential dynamic programming (DDP) controller (Jacobson & Mayne, 1970). DDP approximately solves general continuous state-space optimal control problems by taking advantage of the fact that optimal control problems with linear dynamics and a quadratic reward function (known as linear quadratic regulator (LQR) problems) can be solved efficiently. It is well-known that the solution to the (time-varying, finite horizon) LQR problem is a sequence of linear feedback controllers. In short, DDP iteratively approximates the general control problem with LQR problems until convergence, resulting in a sequence of linear feedback controllers that are approximately optimal. In the receding-horizon algorithm, we not only run DDP initially to design the sequence of controllers, but also re-run DDP during control execution at every time step and recompute the optimal controller over a fixed-length time interval (the horizon), assuming the precomputed controller and cost-to-go are correct after this horizon.

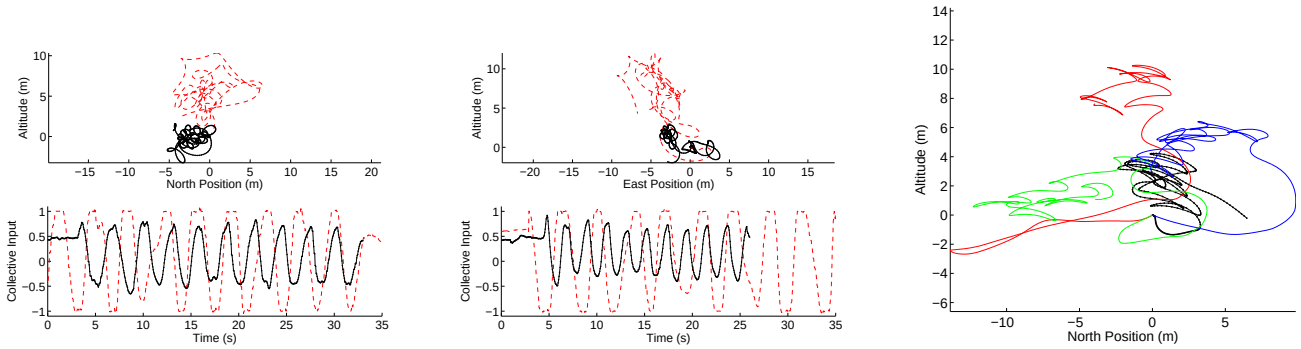
As described in Section 4, our algorithm outputs a sequence of learned local parametric models, each of the form described by Abbeel et al. (2006a). Our implementation linearizes these models on the fly with a 2 second horizon (at 20Hz). Our reward function penalizes error from the target trajectory, $s_t^\star$, as well as deviation from the desired controls, $u_t^\star$, and the desired control velocities, $u_{t+1}^\star - u_t^\star$.

First we compare our results with the previous state-of-the-art in aerobatic helicopter flight, namely the in-place rolls and flips of Abbeel et al. (2007). That work used hand-specified target trajectories and a single nonlinear model for the entire trajectory.

Figure 5(a) shows the Y-Z position[12] and the collective (thrust) control inputs for the in-place rolls for both their controller and ours. Our controller achieves (i) better position performance (standard deviation of approximately 2.3 meters in the Y-Z plane, compared to about 4.6 meters and (ii) lower overall collective control values (which roughly represents the amount of energy being used to fly the maneuver).

Similarly, Figure 5(b) shows the X-Z position and the collective control inputs for the in-place flips for both controllers. Like for the rolls, we see that our controller significantly outperforms that of Abbeel et al. (2007), both in position accuracy and in control energy expended.

[12]These are the position coordinates projected into a plane orthogonal to the axis of rotation.



(a) (b) (c)

Figure 5. Flight results. (a),(b) Solid black: our results. Dashed red: Abbeel et al. (2007). (c) Dotted black: autonomous tic-toc. Solid colored: expert demonstrations. (See text for details.)

Besides flips and rolls, we also performed autonomous "tic tocs"—widely considered to be an even more challenging aerobatic maneuver. During the (tail-down) tic-toc maneuver the helicopter pitches quickly backward and forward in-place with the tail pointed toward the ground (resembling an inverted clock pendulum). The complex relationship between pitch angle, horizontal motion, vertical motion, and thrust makes it extremely difficult to create a feasible tic-toc trajectory by hand. Our attempts to use such a hand-coded trajectory with the DDP algorithm from (Abbeel et al., 2007) failed repeatedly. By contrast, our algorithm readily yields an excellent feasible trajectory that was successfully flown on the first attempt. Figure 5(c) shows the expert trajectories (in color), and the autonomously flown tic-toc (black dotted). Our controller significantly outperforms the expert's demonstrations.

We also applied our algorithm to successfully fly a complete aerobatic airshow, which consists of the following maneuvers in rapid sequence: split-S, snap roll, stall-turn, loop, loop with pirouette, stall-turn with pirouette, "hurricane" (fast backward funnel), knife-edge, flips and rolls, tic-toc and inverted hover.

The trajectory-specific local model learning typically captures the dynamics well enough to fly all the aforementioned maneuvers reliably. Since our computer controller flies the trajectory very consistently, however, this allows us to repeatedly acquire data from the same vicinity of the target trajectory on the real helicopter. Similar to Abbeel et al. (2007), we incorporate this flight data into our model learning, allowing us to improve flight accuracy even further. For example, during the first autonomous airshow our controller achieves an RMS position error of 3.29 meters, and this procedure improved performance to 1.75 meters RMS position error.

Videos of all our flights are available at:

`http://heli.stanford.edu`

6. Related Work

Although no prior works span our entire setting of learning for control from multiple demonstrations, there are separate pieces of work that relate to various components of our approach.

Atkeson and Schaal (1997) use multiple demonstrations to learn a model for a robot arm, and then find an optimal controller in their simulator, initializing their optimal control algorithm with one of the demonstrations.

The work of Calinon et al. (2007) considered learning trajectories and constraints from demonstrations for robotic tasks. There, they do not consider the system's dynamics or provide a clear mechanism for the inclusion of prior knowledge. Our formulation presents a principled, joint optimization which takes into account the multiple demonstrations, as well as the (complex) system dynamics and prior knowledge. While Calinon et al. (2007) also use some form of dynamic time warping, they do not try to optimize a joint objective capturing both the system dynamics and time-warping.

Among others, An et al. (1988) and, more recently, Abbeel et al. (2006b) have exploited the idea of trajectory-indexed model learning for control. However, contrary to our setting, their algorithms do not time align nor coherently integrate data from multiple trajectories.

While the work by Listgarten et al. (Listgarten, 2006; Listgarten et al., 2005) does not consider robotic control and model learning, they also consider the problem of multiple continuous time series alignment with a hidden time series.

Our work also has strong similarities with recent work on inverse reinforcement learning, which extracts a reward function (rather than a trajectory) from the expert demonstrations. See, e.g., Ng and Russell (2000); Abbeel and Ng (2004); Ratliff et al. (2006); Neu and Szepesvari (2007); Ramachandran and Amir (2007); Syed and Schapire (2008).

Most prior work on autonomous helicopter flight only considers the flight-regime close to hover. There are three notable exceptions. The aerobatic work of Gavrilets et al. (2002) comprises three maneuvers: split-S, snap-roll, and stall-turn, which we also include during the first 10 seconds of our airshow for comparison. They record pilot demonstrations, and then hand-engineer a sequence of desired angular rates and velocities, as well as transition points. Ng et al. (2004) have their autonomous helicopter perform sustained inverted hover. We compared the performance of our system with the work of Abbeel et al. (2007), by far the most advanced autonomous aerobatics results to date, in Section 5.

7. Conclusion

We presented an algorithm that takes advantage of multiple suboptimal trajectory demonstrations to (i) extract (an estimate of) the ideal demonstration, (ii) learn a local model along this trajectory. Our algorithm is generally applicable for learning trajectories and dynamics models along trajectories from multiple demonstrations. We showed the effectiveness of our algorithm for control by applying it to the challenging problem of autonomous helicopter aerobatics. The ideal target trajectory and the local models output by our trajectory learning algorithm enable our controllers to significantly outperform the prior state of the art.

Acknowledgments

We thank Garett Oku for piloting and building our helicopter. Adam Coates is supported by a Stanford Graduate Fellowship. This work was also supported in part by the DARPA Learning Locomotion program under contract number FA8650-05-C-7261.

References

Abbeel, P., Coates, A., Quigley, M., & Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. *NIPS 19*.

Abbeel, P., Ganapathi, V., & Ng, A. Y. (2006a). Learning vehicular dynamics with application to modeling helicopters. *NIPS 18*.

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. *Proc. ICML*.

Abbeel, P., Quigley, M., & Ng, A. Y. (2006b). Using inaccurate models in reinforcement learning. *Proc. ICML*.

An, C. H., Atkeson, C. G., & Hollerbach, J. M. (1988). *Model-based control of a robot manipulator*. MIT Press.

Atkeson, C., & Schaal, S. (1997). Robot learning from demonstration. *Proc. ICML*. g

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally weighted learning for control. *Artificial Intelligence Review, 11*.

Boutilier, C., Friedman, N., Goldszmidt, M., & Koller, D. (1996). Context-specific independence in Bayesian networks. *Proc. UAI*.

Calinon, S., Guenter, F., & Billard, A. (2007). On learning, representing and generalizing a task in a humanoid robot. *IEEE Trans. on Systems, Man and Cybernetics, Part B*.

Coates, A., Abbeel, P., & Ng, A. Y. (2008). Learning for control from multiple demonstrations (Full version). http://heli.stanford.edu/icml2008.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society*.

Gavrilets, V., Martinos, I., Mettler, B., & Feron, E. (2002). Control logic for automated aerobatic flight of miniature helicopter. *AIAA Guidance, Navigation and Control Conference*.

Jacobson, D. H., & Mayne, D. Q. (1970). *Differential dynamic programming*. Elsevier.

Listgarten, J. (2006). *Analysis of sibling time series data: alignment and difference detection*. Doctoral dissertation, University of Toronto.

Listgarten, J., Neal, R. M., Roweis, S. T., & Emili, A. (2005). Multiple alignment of continuous time series. *NIPS 17*.

Needleman, S., & Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*

Neu, G., & Szepesvari, C. (2007). Apprenticeship learning using inverse reinforcement learning and gradient methods. *Proc. UAI*.

Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., & Liang, E. (2004). Autonomous inverted helicopter flight via reinforcement learning. *ISER*.

Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. *Proc. ICML*.

Ramachandran, D., & Amir, E. (2007). Bayesian inverse reinforcement learning. *Proc. IJCAI*.

Ratliff, N., Bagnell, J., & Zinkevich, M. (2006). Maximum margin planning. *Proc. ICML*.

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*.

Syed, U., & Schapire, R. E. (2008). A game-theoretic approach to apprenticeship learning. *NIPS 20*.

Spectral Clustering with Inconsistent Advice

Tom Coleman COLEMANT@CSSE.UNIMELB.EDU.AU
James Saunderson J.SAUNDERSON@UGRAD.UNIMELB.EDU.AU
Anthony Wirth AWIRTH@CSSE.UNIMELB.EDU.AU
The University of Melbourne, Victoria 3010 Australia

Abstract

Clustering with advice (often known as constrained clustering) has been a recent focus of the data mining community. Success has been achieved incorporating advice into the k-means and spectral clustering frameworks. Although the theory community has explored inconsistent advice, it has not yet been incorporated into spectral clustering. Extending work of De Bie and Cristianini, we set out a framework for finding minimum normalised cuts, subject to inconsistent advice.

1. Introduction

Clustering is an exploratory data analysis problem which asks us to form groups of related objects. Although humans have a good intuition for clustering in two dimensions, if the data is in a higher dimensional space, it can be hard to visualise. In this paper, we will focus on the problem of clustering data into *two* clusters, subject to a balance criterion and *advice*.

1.1. Clustering with advice

It is sensible for clustering algorithms to be able to incorporate *must-link* and *cannot-link* advice[1], as it is known in the constrained clustering community. For example, in biology, when experimentally clustering proteins (or genes etc), it is often practical to test associations of individual pairs. However, there is no guarantee that the advice we generate in this way will be correct. Additionally, it is well known that human and biological 'experiments' are often subject to noise. If we have enough noisy advice, that advice will be inconsistent—that is, there is no way to cluster the data which agrees with all the advice.

In that case, the objective naturally becomes to respect as much advice as possible. In fact, if we ignore all the data apart from the advice, we have the the *2-correlation clustering* problem (2CC), known to be NP-hard (Bansal et al., 2004). In this context, we can think of the *advice graph*—which has an edge for each piece of advice, labelled with a + for a must-link, and a − otherwise.

1.2. Balanced clustering

A natural problem to solve when clustering in general is the *normalised cut* (Shi & Malik, 2000). Normalised cut (NCUT) asks us to find a cut which minimises inter-cluster affinity whilst maximising intra-cluster affinity in a sensible way. In the two cluster case, this is to minimise the quantity $\frac{\text{cut}(S,\bar{S})}{\text{vol}(S)\text{vol}(\bar{S})}$, where $\text{cut}(S,\bar{S})$ measures the affinity across the cut and $\text{vol}(S)$ is the total degree (out-affinity) of all the nodes within S.

Our aim for the paper is to attempt to optimise both the NCUT and 2CC criteria simultaneously. To do so we will need to relax the problems—we cannot hope to solve them combinatorially.

1.3. Relaxed versions of the clustering problems

Traditionally, spectral approaches were used for the NCUT problem, as they led to fast algorithms. Recently there has been a trend towards tighter, semidefinite programming (SDP)-based, relaxations. We will demonstrate how to alter the basic spectral clustering algorithm, and the SDP techniques, to integrate and deal sensibly with inconsistent advice.

Suppose that the advice *is* consistent (it is simple to

[1]Traditionally in the literature, what we call advice is referred to as *constraints*. We use the term advice here to avoid confusion with constraints that are introduced into the problem in later sections.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

check this fact). Once we know that this is the case, it is sensible to constrain the space of the solutions that we explore only to contain clusterings that are consistent with this advice.

1.4. Existing approaches

Spectral Clustering Spectral Clustering appeared first in the literature in the 1970s. Much of the recent popularity of the technique was instigated by the connection to NCUT as shown by Shi and Malik (2000).

Advice Advice (instance-level constraints) for clustering problems was introduced to the machine learning community in the work of Wagstaff and Cardie (2000) who developed a variant of the classic k-means algorithms to incorporate advice.

Kamvar, Klein and Manning (2003) integrated advice into the spectral formulation by directly changing entries of the Laplacian matrix. Xing et al. (2003) improve on this idea by essentially changing the Laplacian in a more consistent way. They do this by finding a metric that best agrees with the advice. These methods do not directly exploit the nature of the spectral algorithm.

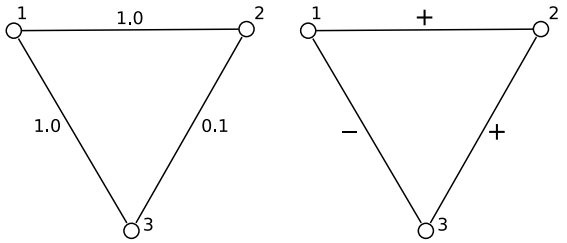
SDP relaxations for N-cut Xing and Jordan (2003) outline a SDP formulation for the NCUT problem for multiple clusters and highlight the connection to spectral clustering. De Bie and Cristianini (2006) provide an SDP which is easier to deal with, and demonstrate that the subspace trick can also be used to introduce advice to this problem.

The subspace trick De Bie, Suykens and De Moor (2004) outline a subspace trick to integrate advice into spectral clustering by constraining a solution to be within the subspace of solutions which agree with the advice. This approach was also previously mentioned in the work of Yu and Shi (2001). This technique leaves the spectral algorithm essentially unchanged; it now just searches for eigenvectors in a different subspace. However it is not necessarily apparent from their work how to extend this technique to inconsistent advice. *This is the key issue addressed in this paper.*

1.5. Addressing Inconsistency

So how can we apply the subspace trick when the advice we have is no longer consistent?

As a first approach (METHOD ONE), we could simply try to solve 2CC defined by the advice, and reject any advice that this solution fails to respect. A good solution to 2CC will ensure that we minimise the number of such edges that we will have to ignore. Then the advice that remains will be consistent, and we can then use the subspace trick. Or indeed, we could use any other constraint-based clustering algorithm in this way.



(a) N-cut problem (b) CC problem

Figure 1. A problem for which not all optimal solutions to 2CC are optimal for the accompanying NCUT problem.

However, this idea has some problems. A toy example of inconsistent advice in Figure 1 shows that deleting any one of the three edges will result in an optimal solution to 2CC. However, one specific cut (namely separating node 3 from nodes 1 and 2) has a much better NCUT cost. So, in forcing a particular optimal solution to 2CC, we are constraining our NCUT solution too much. A second approach (METHOD TWO) that solves this problem is to calculate the *cost* of an approximately optimal solution to 2CC. Rather than force our NCUT solution to be consistent with this 2CC solution, instead we simply require that our NCUT solution has the same correlation clustering cost. This approach will give the NCUT side of our algorithm some room to move, avoiding situations like Figure 1. This technique will be outlined in section 4.1.

A third approach (METHOD THREE) is to allow the algorithm to differ from the optimum correlation clustering cost, but only by some a given factor. So now the NCUT side of the problem has some breathing space in which to find a good solution, whilst we are still forcing a very good solution to 2CC. This approach is developed in section 4.2.

2. Relaxing the Problem

2.1. Normalised Cut

To define the NCUT problem, we begin with an edge-weighted affinity graph with associated affinity matrix A. Distant edges may have zero affinity—we can represent this by deleting the connecting edge, which will speed up the computation.

We represent a 2-clustering by a vector v, where each coordinate represents a datapoint, and is either $+1$ or -1 depending on cluster assignment. The NCUT value becomes:

$$\text{NCUT}(v) = \frac{v^T \mathcal{L}(A) v}{v^T \mathcal{L}(dd^T) v} \tag{1}$$

where d is the vector of vertex degrees, and $\mathcal{L}(X)$ is the *Laplacian* of matrix X. Recall that if e is the vector consisting of all ones and $\text{diag}(x)$ is the matrix with vector x on the main diagonal and zeros elsewhere, then $\mathcal{L}(X) = \text{diag}(Xe) - X$.

2.2. Spectral Clustering

In the spectral relaxation, instead of assigning ± 1 to each vertex, we assign a real number v_i. If $v = (v_1, \ldots, v_n)$ then NCUT relaxes to

P1. *Spectral clustering*

$$\min \quad \frac{v^T \mathcal{L}(A) v}{v^T D v} \qquad \text{s.t.} \qquad d^T v = 0$$

where $D = \text{diag}(d)$. This relaxation is correct because $v^T \mathcal{L}(A) v$ is invariant under translations of v so we can add the constraint $d^T v = 0$ without changing the optimum cost. With this constraint, the denominator of (1) can be simplified as in **P1**.

It turns out that $v = D^{\frac{1}{2}} u$ is an optimum solution of **P1**, where u is the eigenvector of $D^{-\frac{1}{2}} \mathcal{L}(A) D^{-\frac{1}{2}}$ corresponding to the smallest non-zero eigenvalue.

2.3. The SDP formulation

In the two cluster case, De Bie and Cristianini (2006) devised an efficient relaxation of NCUT to a semidefinite program. In this case, instead of assigning ± 1 to the vertices we assign *vectors* v_i of some common length. If X is the Gram matrix of these vectors (i.e. $X_{ij} = v_i^T v_j$) then the relaxation is

P2. *De Bie and Cristianini SDP*

$$\min_{X,q} \quad \mathcal{L}(A) \bullet X$$

$$\text{s.t.} \quad \begin{aligned} \mathcal{L}(dd^T) \bullet X &= 1 && (2) \\ \forall i \in [n] \quad X_{ii} &= q && (3) \\ X &\succeq 0 && (4) \end{aligned}$$

where $A \bullet B = \text{trace}(AB)$ for matrices of appropriate dimension. Here the free variable q is the common (squared) length of the v_i and (2) is a scaling constraint corresponding to the denominator of the NCUT objective function (1).

Importantly, given any $X \succeq 0$ we can find $v_1, \ldots, v_n$ such that $X_{ij} = v_i^T v_j$; we can thus convert a solution to **P2** to an assignment of vectors to the vertices of the graph.

Spectral clustering can be recovered from **P2** by removing the constraints of (3) (see Goemans (1997)).

2.4. The 'subspace trick'

The subspace trick of De Bie, Suykens and De Moor (2004) gives a method for incorporating consistent advice into spectral and SDP relaxations of NCUT. As an example, consider spectral clustering and suppose we have two 'blocks' of independent advice. The first that two vertices, say v_1 and v_2, should be in the same cluster and both should be in a different cluster to v_3, the second that vertices v_4 and v_5 should be in the same cluster. Then it makes sense to constrain the solution vector v so that $v_1 = v_2$ and $v_4 = v_5$ guaranteeing that these pairs of vertices end up in the same cluster after rounding. It also makes sense to constrain v so that $v_3 = -v_2 = -v_1$. This can be done by assuming v has the form

$$v = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & I_{n-5} \end{bmatrix} u = Yu$$

where $u \in \mathbb{R}^{n-3}$. The identity matrix corresponds to vertices for which we have no advice and so should not constrain.

3. Correlation Clustering

Given an advice graph, in the form of $+$ (must-link) or $-$ (cannot-link) edges between datapoints, the correlation clustering problem asks us to cluster the datapoints so that the number of pieces of advice (i.e. edge labels) that are disobeyed is minimised.

3.1. 2CC — the combinatorial problem

In general, unlike the affinity graph, the advice graph is not connected. So we can solve correlation clustering independently on each connected component. We call the vertices in a connected component an *advice block*. We assume without loss of generality that the order on the vertices ensures that the vertices within each block are consecutive. Here we will deal with the problem of solving 2CC for a single advice block $\mathcal{B}$ with m vertices. In later sections, we will consider multiple advice blocks.

If e is an edge within $\mathcal{B}$ let $w_e \in \pm 1$ correspond to the the sign of e. As for NCUT, we assign $v_i = \pm 1$ to each

vertex depending on the cluster in which we place that vertex. For convenience, let E_{ij} be the matrix with a 1 in the (i,j) entry and zeros everywhere else.

Our immediate aim is to find, in terms of v, a simple expression for the number of constraints violated by the labelling.

Consider a single edge $e = \{i,j\}$ of $\mathcal{B}$ with label w_e. Define

$$M_e = (E_{ii} + E_{jj}) - w_e(E_{ij} + E_{ji})$$

and note that $M_e \succeq 0$ because its eigenvalues are 0 and 2. Now

$$\begin{aligned} v^T M_e v &= v_i^2 - 2w_e v_i v_j + v_j^2 && (5)\\ &= |v_i - w_e v_j|^2 && (6)\\ &= \begin{cases} 0 & \text{if } v \text{ respects the advice on } e \\ 4 & \text{otherwise.} \end{cases} && (7) \end{aligned}$$

So if we define $M_{\mathcal{B}} = \sum_e M_e$ it follows that

$$v^T M_{\mathcal{B}} v = 4 \times (\#\text{ pieces of advice violated by } v).$$

Thus 2CC is essentially

$$\min_{v \in \{-1,1\}^m} v^T M_{\mathcal{B}} v. \qquad (8)$$

Note that a clustering that satisfies all the advice in a block will have cost zero. Also observe that $v^T v = m$ is a constant so we could replace the objective function with $(v^T M_{\mathcal{B}} v)/(v^T v)$ without changing the optimum vector.

3.2. Relaxations of 2CC

Recall that our overall aim is to constrain any algorithm we have for (approximately) solving NCUT to produce clusterings which are, in terms of 2CC cost, not much worse than the optimum.

Since we cannot hope to solve (8) exactly, we will instead solve a relaxed version of it. In this paper we consider two relaxations which arise in much the same way as the spectral and SDP relaxations of NCUT.

P3. *Spectral relaxation of correlation clustering*

$$\min_v \quad \frac{v^T M_{\mathcal{B}} v}{v^T v}$$

Observe that the solution of **P3** is given by any non-zero vector in the λ_{min}-eigenspace of $M_{\mathcal{B}}$.

P4. *SDP relaxation of correlation clustering*

$$\begin{aligned} \min_X \quad & M_{\mathcal{B}} \bullet X \\ \text{s.t.} \quad \forall i \in [m] \quad X_{ii} &= 1 \qquad (9)\\ X &\succeq 0 \end{aligned}$$

For either relaxation, if the advice is consistent, the relaxation produces a solution of the same cost (zero) as the optimal solution to the combinatorial problem (8). This is because any solution of the original problem is a feasible point of the relaxed problem, and the relaxed problem has non-negative cost as $M_{\mathcal{B}} \succeq 0$.

4. Clustering with inconsistent advice

In this section, we give the details of METHOD TWO and METHOD THREE, introduced in Section 1.5, for both the spectral and SDP relaxations of NCUT.

Throughout, let $\mathcal{B}_1, \ldots, \mathcal{B}_p$ be the advice blocks of the advice graph. Let $v_{\mathcal{B}}$ denote the projection of v onto the coordinates involved in advice block $\mathcal{B}$. Along similar lines, if $u_{\mathcal{B}}$ is a vector of length $|\mathcal{B}| \le n$ associated with the advice block $\mathcal{B}$, define $\widetilde{u_{\mathcal{B}}}$ to be the length n vector that agrees with $u_{\mathcal{B}}$ in the appropriate coordinates and has zeros elsewhere. For a $|\mathcal{B}| \times |\mathcal{B}|$ matrix $M_{\mathcal{B}}$ we also define $\widetilde{M_{\mathcal{B}}}$ in a similar fashion.

4.1. Combining 2CC and Ncut: Method Two

Let OPT_j denote the optimum cost of the SDP relaxation of 2CC (**P4**) for block j. For the SDP relaxation, we can add the constraint that for each advice block, the 2CC cost of point X is at most $q \cdot \text{OPT}_j$. (The scaling by q is necessary because in **P4** the variables satisfy $X_{ii} = 1$ whereas in **P2** the variables satisfy $X_{ii} = q$.) This forces the new SDP (**P5**) only to consider points of minimum SDP-relaxed 2CC cost.

P5. METHOD TWO *(SDP version)*

$$\begin{aligned} \min_{X,q} \quad & \mathcal{L}(A) \bullet X \\ & \mathcal{L}(dd^T) \bullet X = \text{vol}(\mathcal{V}) \\ \text{s.t.} \quad & \forall i \in [n] \quad X_{ii} = q \\ & \forall j \in [p] \quad \widetilde{M_{\mathcal{B}_j}} \bullet X \le q \cdot \text{OPT}_j \qquad (10)\\ & X \succeq 0 \end{aligned}$$

In the spectral case, the analogous thing to do would be to add the following constraints to the spectral relaxation of NCUT.

$$\forall j \in [p] \quad \frac{v_{\mathcal{B}_j}^T M_{B_j} v_{\mathcal{B}_j}}{v_{\mathcal{B}_j}^T v_{\mathcal{B}_j}} \le \lambda_{min}(M_{\mathcal{B}_j}) \qquad (11)$$

But doing so would mean the problem would no longer be an eigenvalue problem—in fact it would be an SDP—which would undermine the main strength of spectral clustering, its speed.

Luckily, the condition in (11) is equivalent to the condition that each $v_{\mathcal{B}_j}$ is in

the λ_{min}-eigenspace of $M_{\mathcal{B}_j}$, resulting in **P6**.

P6. METHOD TWO *(spectral version)*

$$\min_{v} \quad \frac{v^T \mathcal{L}(A) v}{v^T D v}$$

$$\text{s.t.} \qquad d^T v = 0 \tag{12}$$

$$\forall j \in [p] \quad v_{\mathcal{B}_j} \in \lambda_{min}\text{-eigenspace of } M_{\mathcal{B}_j} \tag{13}$$

The constraints (12) and (13) are forcing v to be in some linear subspace of $\mathbb{R}^n$. So the problem can then be solved using the subspace trick. Details of how to do this are in Appendix A.

4.2. Combining 2CC and Ncut: Method Three

The main drawback of METHOD TWO is that it does not give the algorithm much freedom to balance the trade-off between the 2CC and the NCUT problem. If the advice is quite inconsistent, then forcing the algorithm to follow solutions of a relaxation of 2CC too closely will result in poor performance.

Above, we forced the algorithm to produce a (relaxation of) a clustering that had cost at most the minimum cost of the appropriate relaxation of 2CC. Now we introduce a parameter $f \geq 1$ which tells us the factor by which we are willing to exceed the 2CC cost.

This is straightforward to introduce to the SDP formulation. We simply replace the constraints (11) of **P5** with

$$\forall j \in [p] \quad \widetilde{M_{\mathcal{B}_j}} \bullet X \leq f \cdot q \cdot \text{OPT}_j. \tag{14}$$

In the spectral formulation, the constraints we actually want to add are

$$\forall j \in [p] \quad \frac{v_{\mathcal{B}_j}^T M_{B_j} v_{\mathcal{B}_j}}{v_{\mathcal{B}_j}^T v_{\mathcal{B}_j}} \leq f \cdot \lambda_{min}(M_{\mathcal{B}_j}) \tag{15}$$

but, again we cannot add these and still have an eigenvalue problem. Unfortunately in this case we cannot get a constraint equivalent to (15) by the subspace trick. So, in the interests of producing a practical algorithm, we approximate (15) by

$$\forall j \in [p] \quad v_{\mathcal{B}_j} \in (\leq f \cdot \lambda_{min})\text{-eigenspace of } M_{B_j} \tag{16}$$

where the $(\leq f \cdot \lambda_{min})$-eigenspace of M_{B_j} is the span of all eigenvectors of $M_{\mathcal{B}_j}$ with eigenvalue at most $f \cdot \lambda_{min}$. If v satisfies (16) then it satisfies (15), but the converse does not necessarily hold.

Replacing the constraints in (13) of **P6** with the constraints in (16) gives our final spectral algorithm for clustering with inconsistent advice. It can again be solved with the subspace trick, using the techniques outlined in Appendix A, because all the constraints simply force v to be in some linear subspace of $\mathbb{R}^n$.

5. Experimental Investigations

5.1. Experiment Setup

In order to test the performance of the algorithms on real world datasets, we used six of the UCI repository datasets (Asuncion & Newman, 2007). All datasets are multi-dimensional binary classification problems.

Both datasets were stripped of incomplete records, and in one case (the SPAMBASE dataset), sampled down to 500 datapoints. In each case, the two clusters were of different sizes. This contributed to the mediocre performance of the pure spectral algorithm. This gives us reason to believe that adding advice will help the situation.

For reasons of speed, our experiments primarily use the spectral version of each of the algorithms. Relaxed solutions are rounded to clusterings by cutting at zero. This ensures that advice respected in the relaxed solution is respected in the final clustering.

Advice We generated two different 'types' of synthetic advice for these problems to get a sense of how the algorithms perform. The first we call DENSE—here we are generating around n pieces of advice. We are generating that advice in a dense fashion—we concentrate all advice within 5 separate groups of 20 datapoints. This simulates a few sets of experiments done on some small subset of the total dataspace. Each piece of advice agrees with the actual classification independently with some probability p.

The second type of advice is the COMPLETE case—here we are simulating pairwise comparisons that are relatively cheap, but quite noisy. So we generate a piece of advice for each pair of datapoints, and thus our advice graph is complete.

2CC Algorithms In order to test METHOD ONE, we need to solve 2CC on each advice block. In the DENSE case, we use a tight, strongly performing SDP relaxation (Agarwal et al., 2005). In the complete case, we use the simple 3-approximation algorithm of Bansal, Blum and Chalwa (2004) with a final local search step (see also our other paper (Coleman et al., 2008)).

For each advice type on each dataset, we ran spectral clustering with no advice (as a baseline), METHOD ONE (as a second baseline), and then spectral clustering with every different meaningful f value from 1 upwards. That is, every increment in f that added one additional eigenvector to a single block, until all eigenvectors were added (which is exactly the same as the no advice case).

5.2. Results

Dense advice Figure 2 displays the results of the DENSE advice problem on the HEART DISEASE dataset with $p = 0.75$. We can see that the advice here is sufficiently inconsistent that algorithms which follow it closely (i.e. METHOD ONE and METHOD TWO) perform far worse than the algorithm that ignores it completely (that is, spectral alone). But we can see that by increasing f and striking a balance between ignoring advice and respecting it too strongly, we can achieve results that outperform either extreme. We also note that two other datasets, CONGRESSIONAL VOTING RECORDS and AUSTRALIAN, perform similarly.

Figure 3 shows the results of running very similar advice (again $p = 0.75$) on the SPAMBASE dataset. Here we can see that algorithms that strictly follow the advice outperform algorithms that ignore it, quite significantly. It is perhaps unsurprising then that when we allow the algorithm more and more freedom to ignore the advice we move toward the baseline no advice score. This highlights the fact that these algorithms are not always of use—there needs to be enough inaccuracy in the advice that attempting to follow it is not a great idea.

However, if we lower p to be 0.65, the situation changes, and we get a scenario as demonstrated by Figure 4. Here as for the HEART DISEASE case, using only the 2CC solution is worse than using no advice at all, and for a large range of f values, the compromise of using some advice is better than either extreme. Here the HABERMAN dataset performs similarly. The difference in this case is that for high f values, very poor performance is exhibited. We will discuss this in the next section.

Finally, we consider the HEPATITIS dataset (Figure 5). Here we see new behaviour, as our algorithms only begin to perform well for high f values.

Complete Advice Figures 6 and 7 show the results of the experiments on the two datasets with COMPLETE advice. We first notice that in order to get meaningful experiments, we needed to set p extraordinarily low—all the way down to $p = 0.53$. If p is much higher than this, advice is *so* complete that any incorrect edges will be vastly overshadowed by correct ones, and simply solving 2CC on the instance will give 100% accuracy.

However, with $p = 0.53$ and the problem interesting, we can see that things are similar to the DENSE case. Again, when f is low, we start at the 2CC-baseline,

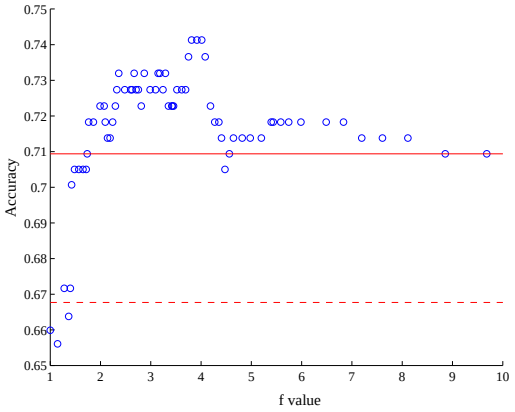


Figure 2. HEART DISEASE dataset, DENSE advice, $p = 0.75$. The unbroken line is the baseline no advice accuracy; the dashed line is the correction clustering based algorithm (METHOD ONE).

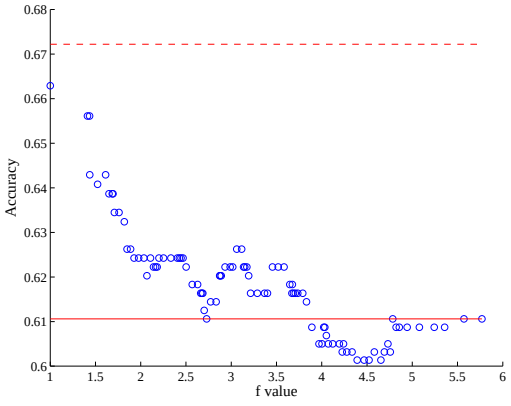


Figure 3. SPAMBASE dataset, DENSE Advice, $p = 0.75$

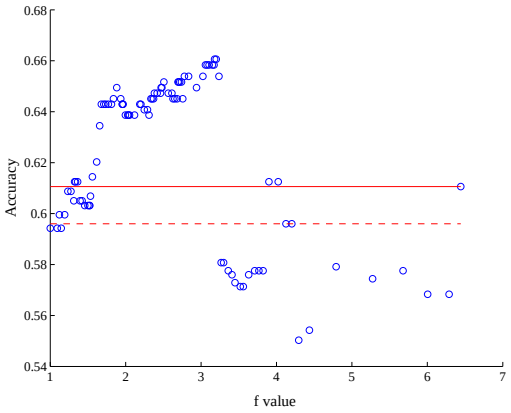


Figure 4. SPAMBASE dataset, DENSE advice, $p = 0.65$

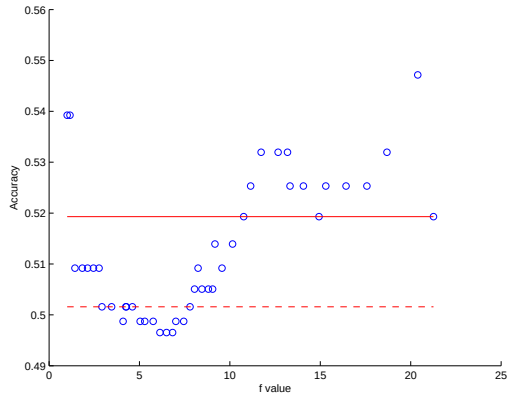


Figure 5. HEPATITIS dataset, DENSE advice, $p = 0.6$

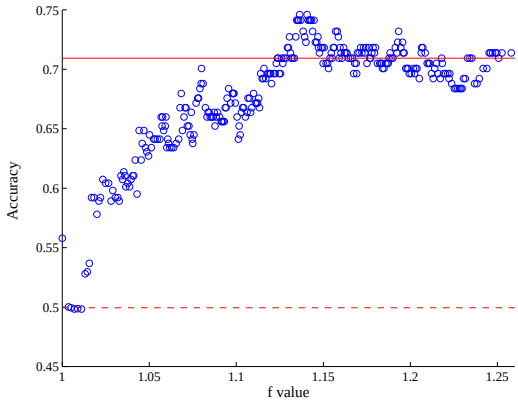


Figure 6. HEART DISEASE dataset, COMPLETE advice, $p = 0.53$

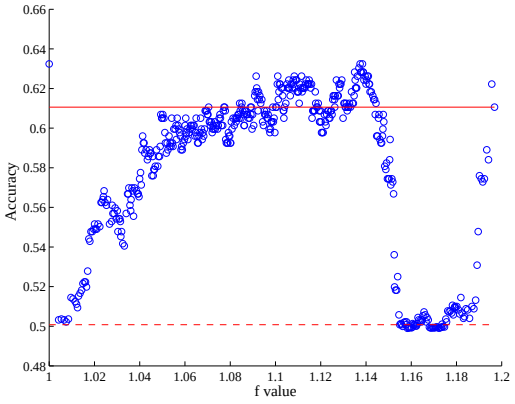


Figure 7. SPAMBASE dataset, COMPLETE advice, $p = 0.53$

and as f increases we move towards and above the no-advice baseline. An interesting point is that the 2CC baseline is around 0.5 in both cases. Note that this is an extremely low score—the advice alone is useless for solving the problem, yet it is still a useful addendum for the spectral method.

As we saw in the DENSE case, one interesting difference between the two datasets is the way the performance drops off as f increases. For HEART DISEASE, the performance seems to asymptote to the no-advice case as we increase f (as we would expect). However, in both cases for the SPAMBASE data, there is a huge dropoff in performance for high end f values. We have no explanation currently for this phenomenon.

6. Conclusions and further work

We have presented a new algorithm that uses inconsistent advice in spectral clustering. This paper is the first to do so. Initial experiments indicate that in many situations our methods are successful, however further theoretical and experimental work is needed. For example, given a clustering problem with inconsistent advice, how do we know when to use METHOD THREE rather than METHOD TWO? And if we are to use METHOD THREE, how do we decide which value of f to choose?

This paper was intended as a largely theoretical work—experiments were performed to give preliminary evidence that the techniques work. Certainly a more thorough comparison to existing work is needed—a technique similar to METHOD ONE could be used in order to compare our algorithms to other approaches that can only deal with consistent constraints. These will be tested in the full version of the paper.

Additionally, in this paper we focused on clustering into two clusters. This is for two reasons. First, there is no obvious way to express cannot-link advice when we have more than two clusters—the approach used here does not generalise nicely. Furthermore, we do not know of an SDP relaxation of the problem which fits into the framework of this paper for the case of more than two clusters. Future work will try to address these problems.

Acknowledgments

Thanks to Ian Davidson for encouraging us to pursue this problem, and to the anonymous reviewers of the draft version. This work was supported by the Australian Research Council through Discovery Project

Grant DP0663979.

References

Agarwal, A., Charikar, M., Makarychev, K., & Makarychev, Y. (2005). $O(\sqrt{\log n})$ approximation algorithms for MIN UNCUT, MIN 2CNF DELETION, and DIRECTED CUT problems. *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, 573–581.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository.

Bansal, N., Blum, A., & Chawla, S. (2004). Correlation clustering. *Machine Learning*, *56*, 89–113.

Coleman, T., Saunderson, J., & Wirth, A. (2008). A local-search 2-approximation for 2-correlation clustering. Submitted.

De Bie, T., & Cristianini, N. (2006). Fast SDP Relaxations of Graph Cut Clustering, Transduction, and Other Combinatorial Problems. *The Journal of Machine Learning Research*, *7*, 1409–1436.

De Bie, T., Suykens, J., & De Moor, B. (2004). Learning from general label constraints. *Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition*, 671–679.

Goemans, M. (1997). Semidefinite programming in combinatorial optimization. *Mathematical Programming*, *79*, 143–161.

Kamvar, S., Klein, D., & Manning, C. (2003). Spectral learning. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2003)*, 561–566.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 888–905.

Wagstaff, K., & Cardie, C. (2000). Clustering with instance-level constraints. *Proceedings of the Seventeenth International Conference on Machine Learning*, *1103*, 1110.

Xing, E., & Jordan, M. (2003). *On Semidefinite Relaxation for Normalized K-cut and Connections to Spectral Clustering*. Computer Science Division, University of California.

Xing, E., Ng, A., Jordan, M., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems*, *15*, 505–512.

Yu, S., & Shi, J. (2001). *Grouping with Bias*. Carnegie Mellon University, the Robotics Institute.

A. Implementing spectral clustering with inconsistent advice

In this section we explain how to implement the spectral version of METHOD TWO and METHOD THREE using the subspace trick.

Let $W_{\mathcal{B}_j}$ be a matrix whose columns are a basis for the $(\leq f \cdot \lambda_{min})$-eigenspace of $M_{\mathcal{B}_j}$. Let $\mathcal{N}(X)$ and $\mathcal{R}(X)$ respectively denote the nullspace and range space of a matrix X. Then the problem can be written as follows:

P7. *Spectral clustering with inconsistent advice*

$$\begin{aligned} \min_{v} \quad & v^T \mathcal{L}(A) v \\ \text{s.t.} \quad & v^T D v = 1 \\ & v \in \mathcal{N}(d^T) \qquad (17) \\ \forall j \in [p] \quad & v_{\mathcal{B}_j} \in \mathcal{R}(W_{\mathcal{B}_j}) \qquad (18) \end{aligned}$$

Let

$$W = \begin{bmatrix} W_{\mathcal{B}_1} & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & W_{\mathcal{B}_p} & 0 \\ 0 & \cdots & 0 & I \end{bmatrix}$$

where the dimension of I is the number of vertices not involved in any advice. Then we can replace the constraints (17) and (18) with

$$v \in \mathcal{N}(d^T) \cap \mathcal{R}(W) = \mathcal{C}$$

Suppose Y is a matrix satisfying $\mathcal{R}(Y) = \mathcal{C}$ and $Y^T D Y = I$. Then if we let $v = Yu$ it is clear that $v \in \mathcal{R}(Y)$ which is what we want. So **P7** becomes

$$\min\ (u^T Y^T)\mathcal{L}(A)(Yu) \quad \text{s.t.} \quad u^T u = 1 \qquad (19)$$

A solution of (19) is given by taking u to be an eigenvector corresponding to the smallest eigenvalue of $Y^T \mathcal{L}(A) Y$. The solution to the original problem is then $v = Yu$.

Elementary linear algebra shows Y generated thus is satisfactory:

1. Let N be a matrix whose columns are an orthonormal basis for $\mathcal{N}(d^T W)$ with respect to the inner product $\langle x, y \rangle = y^T x$.
2. Let R be a matrix whose columns are an orthonormal basis for $\mathcal{R}(W)$ with respect to the inner product $\langle x, y \rangle_D = y^T D x$.
3. Set $Y = RN$.

A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning

Ronan Collobert COLLOBER@NEC-LABS.COM
Jason Weston JASONW@NEC-LABS.COM
NEC Labs America, 4 Independence Way, Princeton, NJ 08540 USA

Abstract

We describe a single convolutional neural network architecture that, given a sentence, outputs a host of language processing predictions: part-of-speech tags, chunks, named entity tags, semantic roles, semantically similar words and the likelihood that the sentence makes sense (grammatically and semantically) using a language model. The entire network is trained *jointly* on all these tasks using weight-sharing, an instance of *multitask learning*. All the tasks use labeled data except the language model which is learnt from unlabeled text and represents a novel form of *semi-supervised learning* for the shared tasks. We show how both *multitask learning* and *semi-supervised learning* improve the generalization of the shared tasks, resulting in state-of-the-art performance.

1. Introduction

The field of Natural Language Processing (NLP) aims to convert human language into a formal representation that is easy for computers to manipulate. Current end applications include information extraction, machine translation, summarization, search and human-computer interfaces.

While complete semantic understanding is still a far-distant goal, researchers have taken a divide and conquer approach and identified several sub-tasks useful for application development and analysis. These range from the syntactic, such as part-of-speech tagging, chunking and parsing, to the semantic, such as word-sense disambiguation, semantic-role labeling, named entity extraction and anaphora resolution.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Currently, most research analyzes those tasks *separately*. Many systems possess few characteristics that would help develop a unified architecture which would presumably be necessary for deeper semantic tasks. In particular, many systems possess three failings in this regard: (i) they are *shallow* in the sense that the classifier is often linear, (ii) for good performance with a linear classifier they must incorporate many hand-engineered features specific for the task; and (iii) they cascade features learnt separately from other tasks, thus propagating errors.

In this work we attempt to define a unified architecture for Natural Language Processing that *learns features* that are relevant to the tasks at hand given very limited prior knowledge. This is achieved by training a *deep neural network*, building upon work by (Bengio & Ducharme, 2001) and (Collobert & Weston, 2007). We define a rather general convolutional network architecture and describe its application to many well known NLP tasks including part-of-speech tagging, chunking, named-entity recognition, learning a language model and the task of semantic role-labeling.

All of these tasks are integrated into a single system which is trained *jointly*. All the tasks except the language model are supervised tasks with labeled training data. The language model is trained in an unsupervised fashion on the entire Wikipedia website. Training this task jointly with the other tasks comprises a novel form of *semi-supervised learning*.

We focus on, in our opinion, the most difficult of these tasks: the semantic role-labeling problem. We show that both (i) multitask learning and (ii) semi-supervised learning significantly improve performance on this task *in the absence of hand-engineered features*.

We also show how the combined tasks, and in particular the unsupervised task, learn powerful features with clear semantic information given no human supervision other than the (labeled) data from the tasks (see Table 1).

The article is structured as follows. In Section 2 we describe each of the NLP tasks we consider, and in Section 3 we define the general architecture that we use to solve all the tasks. Section 4 describes how this architecture is employed for *multitask* learning on all the *labeled* tasks we consider, and Section 5 describes the *unlabeled* task of building a language model in some detail. Section 6 gives experimental results of our system, and Section 7 concludes with a discussion of our results and possible directions for future research.

2. NLP Tasks

We consider six standard NLP tasks in this paper.

Part-Of-Speech Tagging (POS) aims at labeling each word with a unique tag that indicates its syntactic role, e.g. plural noun, adverb, ...

Chunking, also called shallow parsing, aims at labeling segments of a sentence with syntactic constituents such as noun or verb phrase (NP or VP). Each word is assigned only one unique tag, often encoded as a begin-chunk (e.g. B-NP) or inside-chunk tag (e.g. I-NP).

Named Entity Recognition (NER) labels atomic elements in the sentence into categories such as "PERSON", "COMPANY", or "LOCATION".

Semantic Role Labeling (SRL) aims at giving a semantic role to a syntactic constituent of a sentence. In the PropBank (Palmer et al., 2005) formalism one assigns roles ARG0-5 to words that are arguments of a predicate in the sentence, e.g. the following sentence might be tagged "[John]$_{ARG0}$ [ate]$_{REL}$ [the apple]$_{ARG1}$ ", where "ate" is the predicate. The precise arguments depend on a verb's *frame* and if there are multiple verbs in a sentence some words might have multiple tags. In addition to the ARG0-5 tags, there there are 13 modifier tags such as ARGM-LOC (locational) and ARGM-TMP (temporal) that operate in a similar way for all verbs.

Language Models A language model traditionally estimates the probability of the next word being w in a sequence. We consider a different setting: predict whether the given sequence exists in nature, or not, following the methodology of (Okanohara & Tsujii, 2007). This is achieved by labeling real texts as positive examples, and generating "fake" negative text.

Semantically Related Words ("Synonyms") This is the task of predicting whether two words are semantically related (synonyms, holonyms, hypernyms...) which is measured using the WordNet database (`http://wordnet.princeton.edu`) as ground truth.

Our main interest is SRL, as it is, in our opinion, the most complex of these tasks. We use all these tasks to: (i) show the generality of our proposed architecture; and (ii) improve SRL through multitask learning.

3. General Deep Architecture for NLP

All the NLP tasks above can be seen as tasks assigning labels to words. The traditional NLP approach is: extract from the sentence a rich set of hand-designed features which are then fed to a classical *shallow* classification algorithm, e.g. a Support Vector Machine (SVM), often with a linear kernel. The choice of features is a completely empirical process, mainly based on trial and error, and the feature selection is task dependent, implying additional research for each new NLP task. Complex tasks like SRL then require a large number of possibly complex features (e.g., extracted from a parse tree) which makes such systems slow and intractable for large-scale applications.

Instead we advocate a deep neural network (NN) architecture, trained in an end-to-end fashion. The input sentence is processed by several layers of feature extraction. The features in deep layers of the network are *automatically trained* by backpropagation to be relevant to the task. We describe in this section a general deep architecture suitable for all our NLP tasks, and easily generalizable to other NLP tasks.

Our architecture is summarized in Figure 1. The first layer extracts features for each word. The second layer extracts features from the sentence treating it as a *sequence* with local and global structure (i.e., it is not treated like a bag of words). The following layers are classical NN layers.

3.1. Transforming Indices into Vectors

As our architecture deals with raw words and not engineered features, the first layer has to map words into real-valued vectors for processing by subsequent layers of the NN. For simplicity (and efficiency) we consider words as indices in a finite dictionary of words $\mathcal{D} \subset \mathbb{N}$.

Lookup-Table Layer Each word $i \in \mathcal{D}$ is *embedded* into a d-dimensional space using a *lookup table* $LT_W(\cdot)$:

$$LT_W(i) = W_i \,,$$

where $W \in \mathbb{R}^{d \times |\mathcal{D}|}$ is a matrix of parameters to be learnt, $W_i \in \mathbb{R}^d$ is the i^{th} column of W and d is the word vector size (wsz) to be chosen by the user. In the first layer of our architecture an input sentence $\{s_1, s_2, \ldots s_n\}$ of n words in $\mathcal{D}$ is thus transformed into a series of vectors $\{W_{s_1}, W_{s_2}, \ldots W_{s_n}\}$ by apply-

ing the lookup-table to each of its words.

It is important to note that the parameters W of the layer are automatically *trained* during the learning process using backpropagation.

Variations on Word Representations In practice, one may want to introduce some basic pre-processing, such as word-stemming or dealing with upper and lower case. In our experiments, we limited ourselves to converting all words to lower case, and represent the capitalization as a separate feature (yes or no).

When a word is decomposed into K elements (features), it can be represented as a tuple $\boldsymbol{i} = \{i^1, i^2, \dots i^K\} \in \mathcal{D}^1 \times \cdots \times \mathcal{D}^K$, where $\mathcal{D}^k$ is the dictionary for the k^{th}-element. We associate to each element a lookup-table $LT_{W^k}(\cdot)$, with parameters $W^k \in \mathbb{R}^{d^k \times |\mathcal{D}^k|}$ where $d_k \in \mathbb{N}$ is a user-specified vector size. A word $\boldsymbol{i}$ is then embedded in a $d = \sum_k d^k$ dimensional space by concatenating all lookup-table outputs:

$$LT_{W^1,\dots,W^K}(\boldsymbol{i})^{\mathrm{T}} = (LT_{W^1}(i^1)^{\mathrm{T}}, \dots, LT_{W^K}(i^K)^{\mathrm{T}})$$

Classifying with Respect to a Predicate In a complex task like SRL, the class label of each word in a sentence depends on a given *predicate*. It is thus necessary to encode in the NN architecture which predicate we are considering in the sentence.

We propose to add a feature for each word that encodes its relative distance to the chosen predicate. For the i^{th} word in the sentence, if the predicate is at position pos_p we use an additional lookup table $LT^{dist_p}(i - pos_p)$.

3.2. Variable Sentence Length

The lookup table layer maps the original sentence into a sequence $\boldsymbol{x}(\cdot)$ of n identically sized vectors:

$$(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n), \quad \forall t \ \boldsymbol{x}_t \in \mathbb{R}^d \,. \tag{1}$$

Obviously the size n of the sequence varies depending on the sentence. Unfortunately normal NNs are not able to handle sequences of variable length.

The simplest solution is to use a *window approach*: consider a window of fixed size ksz around each word we want to label. While this approach works with great success on simple tasks like POS, it fails on more complex tasks like SRL. In the latter case it is common for the role of a word to depend on words far away in the sentence, and hence outside of the considered window.

When modeling long-distance dependencies is important, Time-Delay Neural Networks (TDNNs) (Waibel et al., 1989) are a better choice. Here, *time* refers

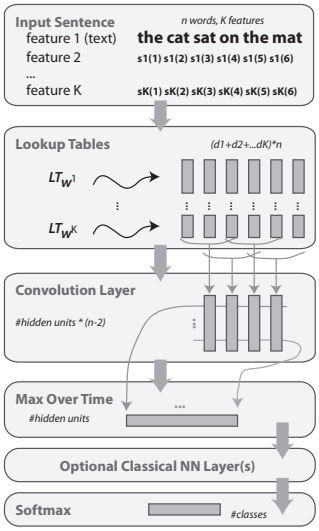


Figure 1. A general deep NN architecture for NLP. Given an input sentence, the NN outputs class probabilities for *one* chosen word. A classical window approach is a special case where the input has a fixed size ksz, and the TDNN kernel size is ksz; in that case the TDNN layer outputs only one vector and the Max layer performs an identity.

to the idea that a sequence has a notion of order. A TDNN "reads" the sequence in an online fashion: at *time* $t \geq 1$, one sees $\boldsymbol{x}_t$, the t^{th} word in the sentence.

A classical TDNN layer performs a *convolution* on a given sequence $\boldsymbol{x}(\cdot)$, outputting another sequence $\boldsymbol{o}(\cdot)$ whose value at time t is:

$$\boldsymbol{o}(t) = \sum_{j=1-t}^{n-t} \boldsymbol{L}_j \cdot \boldsymbol{x}_{t+j} \,, \tag{2}$$

where $\boldsymbol{L}_j \in \mathbb{R}^{n_{hu} \times d}$ $(-n \leq j \leq n)$ are the parameters of the layer (with n_{hu} hidden units) trained by backpropagation. One usually constrains this convolution by defining a *kernel width*, ksz, which enforces

$$\forall \ |j| > (ksz - 1)/2, \quad \boldsymbol{L}_j = \boldsymbol{0} \,. \tag{3}$$

A classical window approach only considers words in a window of size ksz around the word to be labeled. Instead, if we use (2) and (3), a TDNN considers at the same time *all* windows of ksz words in the sentence.

TDNN layers can also be *stacked* so that one can extract local features in lower layers, and more global features in subsequent ones. This is an approach typically used in convolutional networks for vision tasks, such as the LeNet architecture (LeCun et al., 1998).

We then add to our architecture a layer which captures the *most relevant features over the sentence* by feeding

the TDNN layer(s) into a "Max" Layer, which takes the maximum over time (over the sentence) in (2) for each of the n_{hu} output features.

As the layer's output is of fixed dimension (independent of sentence size) subsequent layers can be classical NN layers. Provided we have a way to indicate to our architecture the word *to be labeled*, it is then able to use features extracted from *all* windows of ksz words in the sentence to compute the label of *one* word of interest.

We indicate the word to be labeled to the NN with an additional lookup-table, as suggested in Section 3.1. Considering the word at position pos_w we encode the relative distance between the i^{th} word in the sentence and this word using a lookup-table $LT^{dist_w}(i - pos_w)$.

3.3. Deep Architecture

A TDNN (or window) layer performs a linear operation over the input words. While linear approaches work fairly well for POS or NER, more complex tasks like SRL require nonlinear models. One can add to the NN one or more classical NN layers. The output of the l^{th} layer containing n_{hu_l} hidden units is computed with $\boldsymbol{o^l} = \tanh(\boldsymbol{L^l} \cdot \boldsymbol{o^{l-1}})$, where the matrix of parameters $\boldsymbol{L^l} \in \mathbb{R}^{n_{hu_l} \times n_{hu_{l-1}}}$ is trained by backpropagation.

The size of the last (parametric) layer's output $\boldsymbol{o^{last}}$ is the number of classes considered in the NLP task. This layer is followed by a *softmax* layer (Bridle, 1990) which makes sure the outputs are positive and sum to 1, allowing us to interpret the outputs of the NN as probabilities for each class. The i^{th} output is given by $e^{o_i^{last}} / \sum_j e^{o_j^{last}}$. The whole network is trained with the cross-entropy criterion (Bridle, 1990).

3.4. Related Architectures

In (Collobert & Weston, 2007) we described a NN suited for SRL. This work also used a lookup-table to generate word features (see also (Bengio & Ducharme, 2001)). The issue of labeling with respect to a predicate was handled with a special hidden layer: its output, given input sequence (1), predicate position pos_p, and the word of interest pos_w was defined as:

$$\boldsymbol{o}(t) = \mathcal{C}(t - pos_w,\, t - pos_p) \cdot \boldsymbol{x}_t.$$

The function $\mathcal{C}(\cdot)$ is shared through time t: one could say that this is a variant of a TDNN layer with a kernel width $ksz = 1$ but where the parameters are *conditioned* with other variables (distances with respect to the verb and word of interest).

The fact that $\mathcal{C}(\cdot)$ does not combine several words in the same neighborhood as in our TDNN approach limits the dependencies between words it can model. Also $\mathcal{C}(\cdot)$ is itself a NN inside a NN. Not only does one have to carefully design this additional architecture, but it also makes the approach more complicated to train and implement. Integrating all the desired features in $\boldsymbol{x}()$ (including the predicate position) via lookup-tables makes our approach simpler, more general and easier to tune.

4. Multitasking with Deep NN

Multitask learning (MTL) is the procedure of learning several tasks at the same time with the aim of mutual benefit. This an old idea in machine learning; a good overview, especially focusing on NNs, can be found in (Caruana, 1997).

4.1. Deep Joint Training

If one considers *related* tasks, it makes sense that features useful for one task might be useful for other ones. In NLP for example, POS predictions are often used as features for SRL and NER. Improving generalization on the POS task might therefore improve both SRL and NER.

A NN automatically learns features for the desired tasks in the deep layers of its architecture. In the case of our general architecture for NLP presented in Section 3, the deepest layer (consisting of lookup-tables) *implicitly learns* relevant features for each word in the dictionary. It is thus reasonable to expect that when training NNs on *related* tasks, sharing deep layers in these NNs would improve features produced by these deep layers, and thus improve generalization performance. The last layers of the network can then be task specific.

In this paper we show this procedure performs very well for NLP tasks when sharing the lookup-tables of each considered task, as depicted in Figure 2. Training is achieved in a stochastic manner by looping over the tasks:

1. Select the next task.
2. Select a random training example for this task.
3. Update the NN for this task by taking a gradient step with respect to this example.
4. Go to 1.

It is worth noticing that labeled data for training each task can come from completely different datasets.

4.2. Previous Work in MTL for NLP

The NLP field contains many related tasks. This makes it a natural field for applying MTL, and sev-

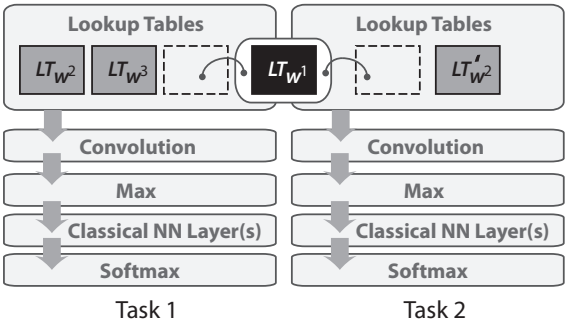


Figure 2. Example of deep multitasking with NN. Task 1 and Task 2 are two tasks trained with the architecture presented in Figure 1. One lookup-table (in black) is shared (the other lookup-tables and layers are task specific). The principle is the same with more than two tasks.

eral techniques have already been explored.

Cascading Features The most obvious way to achieve MTL is to train one task, and then use this task as a feature for another task. This is a very common approach in NLP. For example, in the case of SRL, several methods (e.g., (Pradhan et al., 2004)) train a POS classifier and use the output as features for training a parser, which is then used for building features for SRL itself. Unfortunately, tasks (features) are learnt separately in such a cascade, thus propagating errors from one classifier to the next.

Shallow Joint Training If one possesses a dataset labeled for several tasks, it is then possible to train these tasks jointly in a *shallow* manner: one unique model can predict all task labels at the same time. Using this scheme, the authors of (Sutton et al., 2007) proposed a conditional random field approach where they showed improvements from joint training on POS tagging and noun-phrase chunking tasks. However the requirement of jointly annotated data is a limitation, as this is often not the case. Similarly, in (Miller et al., 2000) NER, parsing and relation extraction were jointly trained in a statistical parsing model achieving improved performance on all tasks. This work has the same joint labeling requirement problem, which the authors avoided by using a predictor to fill in the missing annotations.

In (Sutton & McCallum, 2005a) the authors showed that one could learn the tasks independently, hence using different training sets, by only leveraging predictions jointly in a *test time* decoding step, and still obtain improved results. The problem is, however, that this will not make use of the shared tasks at *training time*. The NN approach used here seems more flexible in these regards.

Finally, the authors of (Musillo & Merlo, 2006) made an attempt at improving the semantic role labeling task by joint inference with syntactic parsing, but their results are not state-of-the-art. The authors of (Sutton & McCallum, 2005b) also describe a negative result at the same joint task.

5. Leveraging Unlabeled Data

Labeling a dataset can be an expensive task, especially in NLP where labeling often requires skilled linguists. On the other hand, unlabeled data is abundant and freely available on the web. Leveraging unlabeled data in NLP tasks seems to be a very attractive, and challenging, goal.

In our MTL framework presented in Figure 2, there is nothing stopping us from jointly training supervised tasks on labeled data *and* unsupervised tasks on unlabeled data. We now present an unsupervised task suitable for NLP.

Language Model We consider a *language model* based on a simple fixed window of text of size ksz using our NN architecture, given in Figure 2. We trained our language model to discriminate a two-class classification task: if the word in the middle of the input window is related to its context or not. We construct a dataset for this task by considering all possible ksz windows of text from the entire of English Wikipedia (`http://en.wikipedia.org`). Positive examples are windows from Wikipedia, negative examples are the same windows but where the middle word has been replaced by a random word.

We train this problem with a ranking-type cost:

$$\sum_{s \in \mathcal{S}} \sum_{w \in \mathcal{D}} \max\left(0,\, 1 - f(s) + f(s^w)\right) , \qquad (4)$$

where $\mathcal{S}$ is the set of sentence windows of text, $\mathcal{D}$ is the dictionary of words, and $f(\cdot)$ represents our NN architecture without the softmax layer and s^w is a sentence window where the middle word has been replaced by the word w. We sample this cost *online* w.r.t. (s, w).

We will see in our experiments that the features (embedding) learnt by the lookup-table layer of this NN clusters semantically similar words. These discovered features will prove very useful for our shared tasks.

Previous Work on Language Models (Bengio & Ducharme, 2001) and (Schwenk & Gauvain, 2002) already presented very similar language models. However, their goal was to give a *probability* of a word given *previous* ones in a sentence. Here, we only want to have a good representation of words: we take advantage of the complete context of a word (before and af-

ter) to predict its relevance. Perhaps this is the reason the authors were never able to obtain a good embedding of their words. Also, using probabilities imposes using a cross-entropy type criterion and can require many tricks to speed-up the training, due to normalization issues. Our criterion (4) is much simpler in that respect.

The authors of (Okanohara & Tsujii, 2007), like us, also take a two-class approach (true/fake sentences). They use a *shallow* (kernel) classifier.

Previous Work in Semi-Supervised Learning For an overview of semi-supervised learning, see (Chapelle et al., 2006). There have been several uses of semi-supervised learning in NLP before, for example in NER (Rosenfeld & Feldman, 2007), machine translation (Ueffing et al., 2007), parsing (McClosky et al., 2006) and text classification (Joachims, 1999). The first work is a highly problem-specific approach whereas the last three all use a self-training type approach (Transductive SVMs in the case of text classification, which is a kind of self-training method). These methods augment the training set with labeled examples from the unlabeled set which are predicted by the model itself. This can give large improvements in a model, but care must be taken as the predictions are of course prone to noise.

The authors of (Ando & Zhang, 2005) propose a setup more similar to ours: they learn from unlabeled data as an auxiliary task in a MTL framework. The main difference is that they use *shallow* classifiers; however they report positive results on POS and NER tasks.

Semantically Related Words Task We found it interesting to compare the embedding obtained with a language model on unlabeled data with an embedding obtained with labeled data. WordNet is a database which contains semantic relations (synonyms, holonyms, hypernyms, ...) between around 150,000 words. We used it to train a NN similar to the language model one. We considered the problem as a two-class classification task: positive examples are pairs with a relation in Wordnet, and negative examples are random pairs.

6. Experiments

We used Sections 02-21 of the PropBank dataset version 1 (about 1 million words) for training and Section 23 for testing as standard in all SRL experiments. POS and chunking tasks use the same data split via the Penn TreeBank. NER labeled data was obtained by running the Stanford Named Entity Recognizer (a CRF based classifier) over the same data.

Table 1. Language model performance for learning an embedding in $wsz = 50$ dimensions (dictionary size: 30,000). For each column the queried word is followed by its index in the dictionary (higher means more rare) and its 10 nearest neighbors (arbitrary using the Euclidean metric).

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED
454	1973	6909	11724	29869
SPAIN	CHRIST	PLAYSTATION	YELLOWISH	SMASHED
ITALY	GOD	DREAMCAST	GREENISH	RIPPED
RUSSIA	RESURRECTION	PSNUMBER	BROWNISH	BRUSHED
POLAND	PRAYER	SNES	BLUISH	HURLED
ENGLAND	YAHWEH	WII	CREAMY	GRABBED
DENMARK	JOSEPHUS	NES	WHITISH	TOSSED
GERMANY	MOSES	NINTENDO	BLACKISH	SQUEEZED
PORTUGAL	SIN	GAMECUBE	SILVERY	BLASTED
SWEDEN	HEAVEN	PSP	GREYISH	TANGLED
AUSTRIA	SALVATION	AMIGA	PALER	SLASHED

Language models were trained on Wikipedia. In all cases, any numeric number was converted as "NUMBER". Accentuated characters were transformed to their non-accentuated versions. All paragraphs containing other non-ASCII characters were discarded. For Wikipedia, we obtain a database of *631M* words. We used WordNet to train the "synonyms" (semantically related words) task.

All tasks use the same dictionary of the 30,000 most common words from Wikipedia, converted to lower case. Other words were considered as *unknown* and mapped to a special word.

Architectures All tasks were trained using the NN shown in Figure 1. POS, NER, and chunking tasks were trained with the window version with $ksz = 5$. We chose linear models for POS and NER. For chunking we chose a hidden layer of 200 units. The language model task had a window size $ksz = 11$, and a hidden layer of 100 units. All these tasks used two lookup-tables: one of dimension wsz for the word in lower case, and one of dimension 2 specifying if the first letter of the word is a capital letter or not.

For SRL, the network had a convolution layer with $ksz = 3$ and 100 hidden units, followed by another hidden layer of 100 hidden units. It had three lookup-tables in the first layer: one for the word (in lower case), and two that encode relative distances (to the word of interest and the verb). The last two lookup-tables embed in 5 dimensional spaces. Verb positions are obtained with our POS classifier.

The language model network had only one lookup-table (the word in lower case) and 100 hidden units. It used a window of size $ksz = 11$.

We show results for different encoding sizes of the word in lower case: $wsz = 15$, 50 and 100.

Table 2. A Deep Architecture for SRL improves by learning auxiliary tasks that share the first layer that represents words as wsz-dimensional vectors. We give word error rates for wsz=15, 50 and 100 and various shared tasks.

	wsz=15	wsz=50	wsz=100
SRL	16.54	17.33	18.40
SRL + POS	15.99	16.57	16.53
SRL + Chunking	16.42	16.39	16.48
SRL + NER	16.67	17.29	17.21
SRL + Synonyms	15.46	15.17	15.17
SRL + Language model	14.42	14.30	14.46
SRL + POS + Chunking	16.46	15.95	16.41
SRL + POS + NER	16.45	16.89	16.29
SRL + POS + Chunking + NER	16.33	16.36	16.27
SRL + POS + Chunking + NER + Synonyms	15.71	14.76	15.48
SRL + POS + Chunking + NER + Language model	14.63	14.44	14.50

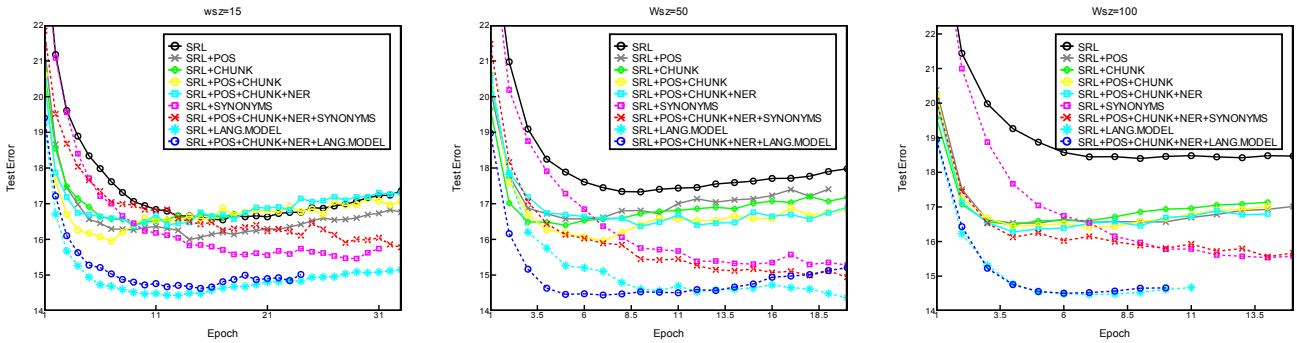


Figure 3. Test error versus number of training epochs over PropBank, for the SRL task alone and SRL jointly trained with various other NLP tasks, using deep NNs.

Results: Language Model Because the language model was trained on a huge database we first trained it *alone*. It takes about a week to train on one computer. The embedding obtained in the word lookup-table was extremely good, even for uncommon words, as shown in Table 1. The embedding obtained by training on labeled data from WordNet "synonyms" is also good (results not shown) however the coverage is not as good as using unlabeled data, e.g. "Dreamcast" is not in the database.

The resulting word lookup-table from the language model was used as an initializer of the lookup-table used in MTL experiments with a language model.

Results: SRL Our main interest was improving SRL performance, the most complex of our tasks. In Table 2, we show results comparing the SRL task alone with the SRL task jointly trained with different combinations of the other tasks. For all our experiments, training was achieved in a few epochs (about a day) over the PropBank dataset as shown in Figure 3. Testing takes 0.015s to label a complete sentence (given one verb).

All MTL experiments performed better than SRL alone. With larger wsz (and thus large capacity) the relative improvement becomes larger from using MTL compared to the task alone, which shows MTL is a good way of regularizing: in fact with MTL results are fairly stable with capacity changes.

The *semi-supervised* training of SRL using the language model performs better than other combinations. Our best model performed as low as 14.30% in per-word error rate, which is to be compared to previously published results of 16.36% with an NN architecture (Collobert & Weston, 2007) and 16.54% for a state-of-the-art method based on parse trees (Pradhan et al., 2004)[1]. Further, our system is the only one not to use POS tags or parse tree features.

Results: POS and Chunking Training takes about 30 min for these tasks alone. Testing time for labeling a complete sentence is about 0.003s. We obtained modest improvements to POS and chunking results us-

[1]Our loss function optimized per-word error rate. We note that many SRL results e.g. the CONLL 2005 evaluation use F1 as a standard measure.

ing MTL. Without MTL (for $wsz = 50$) we obtain 2.95% test error for POS and 4.5% (91.1 F-measure) for chunking. With MTL we obtain 2.91% for POS and 3.8% (92.71 F-measure) for chunking. POS error rates in the 3% range are state-of-the-art. For chunking, although we use a different train/test setup to the CoNLL-2000 shared task (`http://www.cnts.ua.ac.be/conll2000/chunking`) our system seems competitive with existing systems (better than 9 of the 11 submitted systems). However, our system is the only one that does *not use* POS tags as input features.

Note, we did not evaluate NER error rates because we used non-gold standard annotations in our setup. Future work will more thoroughly evaluate these tasks.

7. Conclusion

We proposed a general deep NN architecture for NLP. Our architecture is extremely fast enabling us to take advantage of huge databases (e.g. 631 million words from Wikipedia). We showed our deep NN could be applied to various tasks such as SRL, NER, POS, chunking and language modeling. We demonstrated that learning tasks simultaneously can improve generalization performance. In particular, when training the SRL task jointly with our language model our architecture achieved state-of-the-art performance in SRL without any explicit syntactic features. This is an important result, given that the NLP community considers syntax as a mandatory feature for semantic extraction (Gildea & Palmer, 2001).

References

Ando, R., & Zhang, T. (2005). A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *JMLR*, *6*, 1817–1853.

Bengio, Y., & Ducharme, R. (2001). A neural probabilistic language model. *NIPS 13*.

Bridle, J. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. F. Soulié and J. Hérault (Eds.), *Neurocomputing: Algorithms, architectures and applications*, 227–236. NATO ASI Series.

Caruana, R. (1997). Multitask Learning. *Machine Learning*, *28*, 41–75.

Chapelle, O., Schlkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Adaptive computation and machine learning. Cambridge, Mass., USA: MIT Press.

Collobert, R., & Weston, J. (2007). Fast semantic extraction using a novel neural network architecture. *Proceedings of the 45th Annual Meeting of the ACL* (pp. 560–567).

Gildea, D., & Palmer, M. (2001). The necessity of parsing for predicate argument recognition. *Proceedings of the 40th Annual Meeting of the ACL*, 239–246.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *ICML*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*.

McClosky, D., Charniak, E., & Johnson, M. (2006). Effective self-training for parsing. *Proceedings of HLT-NAACL 2006*.

Miller, S., Fox, H., Ramshaw, L., & Weischedel, R. (2000). A novel use of statistical parsing to extract information from text. *6th Applied Natural Language Processing Conference*.

Musillo, G., & Merlo, P. (2006). Robust Parsing of the Proposition Bank. *ROMAND 2006: Robust Methods in Analysis of Natural language Data*.

Okanohara, D., & Tsujii, J. (2007). A discriminative language model with pseudo-negative samples. *Proceedings of the 45th Annual Meeting of the ACL*, 73–80.

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Comput. Linguist.*, *31*, 71–106.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J., & Jurafsky, D. (2004). Shallow semantic parsing using support vector machines. *Proceedings of HLT/NAACL-2004*.

Rosenfeld, B., & Feldman, R. (2007). Using Corpus Statistics on Entities to Improve Semi-supervised Relation Extraction from the Web. *Proceedings of the 45th Annual Meeting of the ACL*, 600–607.

Schwenk, H., & Gauvain, J. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. *IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 765–768).

Sutton, C., & McCallum, A. (2005a). Composition of conditional random fields for transfer learning. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 748–754.

Sutton, C., & McCallum, A. (2005b). Joint parsing and semantic role labeling. *Proceedings of CoNLL-2005* (pp. 225–228).

Sutton, C., McCallum, A., & Rohanimanesh, K. (2007). Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data. *JMLR*, *8*, 693–723.

Ueffing, N., Haffari, G., & Sarkar, A. (2007). Transductive learning for statistical machine translation. *Proceedings of the 45th Annual Meeting of the ACL*, 25–32.

Waibel, A., abd G. Hinton, T. H., Shikano, K., & Lang, K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *37*, 328–339.

Autonomous Geometric Precision Error Estimation in Low-Level Computer Vision Tasks

Andrés Corrada-Emmanuel CORRADA@CS.UMASS.EDU
Howard Schultz HSCHULTZ@CS.UMASS.EDU
Computer Science Department, University of Massachusetts at Amherst, 140 Governors Drive,Amherst, MA 01003-9264

Abstract

Errors in map-making tasks using computer vision are sparse. We demonstrate this by considering the construction of digital elevation models that employ stereo matching algorithms to triangulate real-world points. This sparsity, coupled with a geometric theory of errors recently developed by the authors, allows for autonomous agents to calculate their own precision independently of ground truth. We connect these developments with recent advances in the mathematics of sparse signal reconstruction or compressed sensing. The theory presented here extends the autonomy of 3-D model reconstructions discovered in the 1990s to their errors.

1. Introduction

Autonomy of robots or intelligent sensors depends on developing algorithms that can assess their own performance independent of ground truth. Consider an Aerial Mapping Appliance (AMA) that must construct a map or 3-D model of the world based on photographs. Researchers in computer vision discovered in the 1990s that a faithful 3-D model of an imaged scene was possible without any knowledge of the positions, or orientations of the camera that took the photographs (Beardsley et al., 1996). This reconstruction is even possible without knowing the internal parameters of the camera (Pollefeys et al., 1999). The geometry of multiple images contains all the necessary information to do this reconstruction. This independence of 3-D model reconstruction from ground truth (in this case, camera positions, etc.) raises the possibility that the errors in the reconstruction can also be recovered autonomously by an intelligent agent such as the AMA.

Autonomous error estimation for 3-D model reconstruction was recently demonstrated to be possible by the authors (Corrada-Emmanuel et al., 2007; Corrada-Emmanuel & Schultz, 2008). The theory depends on making a distinction between accuracy and precision. Knowledge of accuracy is not possible without ground truth. Precision can be estimated autonomously. This paper will demonstrate that autonomous precision estimation is also related to the mathematics of sparse signal reconstruction or compressed sensing (Donoho, 2006a). The precision errors of measurements, not just the measurements, are sparse themselves. This sparsity is the key to their reconstruction.

2. The Distinction between Geometric Accuracy and Precision

The concepts of accuracy and precision are well known to all scientists. The Machine Learning community knows these concepts as bias and variance (Bishop, 2007). Bias refers to how far an estimate is from the true value. Variance captures how noisy that estimate is given the measurements used to compute it. Our meaning of accuracy and precision in 3-D models is analogous to bias and variance but not equivalent. Our definitions are geometrical in nature.

Imagine that one had a set of 3-D models of a scene. Furthermore, along with the models one also has the ground truth or exact locations of points in the scene. The total error of the models can be defined as

$$\sum_{\text{models}} \sum_{\text{points}} [\vec{x}_{\text{model}}(\text{point}) - \vec{x}_{\text{true}}(\text{point})]^2 \quad (1)$$

We can decrease the total error in the models by applying a global transformation to all the models. For

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

example, the models may be systematically off by 1 meter in some direction. The *geometric precision* of the models is defined as the minimum possible total error after application of a global transformation to all the models:

$$\min_{T} \sum_{\text{models}} \sum_{\text{points}} \left[T(\vec{x}_{\text{model}}(\text{point})) - \vec{x}_{\text{true}}(\text{point})\right]^2 . \tag{2}$$

The *geometric accuracy* is defined as difference between the total error and the geometric precision.

We describe some simple examples to clarify these definitions. Imagine a 3-D model reconstruction that is merely a translated example of the real world, i.e. all locations are off by 1 meter to the North. The reconstruction has an accuracy error of 1 meter and zero precision error. A model with zero accuracy error but some precision error can be created by taking a perfect reconstruction and individually randomizing the elevation of the reconstruction with zero mean.

The concept of geometric accuracy can thus be captured by a successive set of transformations that the reader may want to view as encompassing the usual hierarchy of projective, affine and euclidean transformations (Hartley & Zisserman, 2000; Faugeras et al., 2001) or some subset of them. The rest of this paper will use the words accuracy and precision as shorthand for these geometric definitions.

2.1. Autonomous Elevation Difference Equations and Geometric Precision

Our central claim is that precision can be estimated autonomously even as the accuracy of the models is completely unknown. Autonomous geometric precision error estimation is possible by creating quantities that are invariant under global accuracy transformations. In this paper we will consider one such quantity that is useful for characterizing the precision errors in DEMs and has been discussed in our previous papers (Corrada-Emmanuel et al., 2007):

$$\Delta_{P,Q}(x,y) = \frac{1}{P}\sum_{i=1}^{P} Z_i - \frac{1}{Q}\sum_{j=1}^{Q} Z_j \tag{3}$$

$$= \frac{1}{P}\sum_{i=1}^{P} \delta_i - \frac{1}{Q}\sum_{j=1}^{Q} \delta_j, \tag{4}$$

where the integers P and Q are between 1 and the number of models being compared, $1 \leq P \leq M$ and $1 \leq Q \leq M$. A DEM i is a collection of elevation postings at different (x,y) locations, $\{Z_i(x,y)\}$. The precision error in each posting is denoted by $\delta_i(x,y)$, so in general one can write

$$Z_i(x,y) = Z_{\text{true}}(x,y) + \delta_i(x,y). \tag{5}$$

By picking the integers P and Q less than or equal to the number of DEM models, we guarantee that the true value of the elevation cancels out at each posting since

$$\frac{1}{P} * (P * Z_{\text{true}}) - \frac{1}{Q} * (Q * Z_{\text{true}}) = 0 \tag{6}$$

so that equation 4 follows from equation 3.

By considering all possible values P and Q one can find a set of linearly independent equations for the elevation precision errors. We call this independent set the *autonomous difference equations*. Note that these equations are not being used to construct a better estimate of the true elevation by performing some simple averaging over them. Their sole purpose is to probe the errors in the reconstructed elevations. More general expressions that take into account x and y position errors can be constructed but we defer discussion of these to future papers.

Equation 3 can be calculated from the observable elevations. The task of the autonomous agent is to estimate the precision errors $\{\delta_i\}$ in equation 4 and how they are correlated with each other. Once the agent knows these correlations, the precision error of a fused estimated can be decreased while possibly increasing its accuracy error. This may be a suitable action to take since in many computer vision tasks accuracy is cheaper to fix than precision, a point we clarify in our concluding remarks.

3. The Covariance Matrix for Precision Errors

An AMA or robot on a mapping mission will not know beforehand what errors it will make during its activities. Sensors could systematically malfunction. Lighting conditions may be unfavorable at certain viewing angles. These and other factors will inevitably mean that repeated measurements of the same scene will be partly correlated, or their precision may vary widely. How should the 3-D models obtained from different vantage points be fused? How fast is the precision error in the reconstructions decreasing as a function of the collected images? Has the AMA attained a desired precision level and therefore completed its mission? Autonomous mission planning by robots requires answers to these questions.

We argue that a principled approach to answering these questions must rely on an autonomous estima-

tion of the covariance matrix of the 3-D models. Having multiple measurements whose errors are strongly correlated is not much better than a single measurement, for example. Knowing the covariance matrix would allow the agent to discard bad data, understand its rate of error decrease as a function of data collection, and provide a fused estimate that monotonically improves with time.

The covariance matrix for DEM errors is composed of entries of the form $\langle \delta_i \delta_j \rangle - \langle \delta_i \rangle \langle \delta_j \rangle$. For ease of discussion, we will assume that the precision error has been de-meaned so $\langle \delta_i \rangle = 0$ for all i, so the covariance matrix is equivalent to $\langle \delta_i \delta_j \rangle$ in this paper.

It is impossible, generally, to calculate this covariance matrix given a set of measurements. We explain this fully by constructing a linear algebra system for the covariance matrix based on the autonomous difference equations (eqs. 3 and 4) to demonstrate that it defines an under-determined system – one where we have less equations that unknowns.

3.1. An Under-Determined Linear Algebra System for the Covariance Matrix Entries

Squaring the autonomous difference equations and averaging over all the posting locations (x, y) gives a set of linear equations for all the entries in the covariance matrix. We denote this system by

$$S = \Phi \Delta. \tag{7}$$

The vector S is the "signal" of the DEM precision errors. Its components are calculated using equation 3. The matrix Φ consists of the rational fractions that come from expanding the square of equation 4. The vector Δ are the entries $< \delta_i \delta_j >$ of the covariance matrix that we want to estimate.

Equation 7 defines an under-determined linear system because the number of independent entries in the covariance matrix is $M(M+1)/2$ given M models (the matrix is symmetric). The number of independent equations that can be constructed from the autonomous difference equations is equal to $M(M+1)/2 - M$. Therefore, the system is always under-determined by M equations.

3.2. The Correlated-Pair Error Model

This limitation was circumvented in our earlier papers (Corrada-Emmanuel et al., 2007) by *assuming* that the covariance matrix had the simple form

$$\begin{pmatrix} * & * & 0 & 0 & \dots \\ * & * & 0 & 0 & \dots \\ 0 & 0 & * & * & \dots \\ 0 & 0 & * & * & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{8}$$

The block-diagonal shape came from our production of two DEMs from every photographic pair, a practice that differs from the usual photogrammetric convention of producing a single DEM from a photographic pair. We assumed that only the two DEMs from the same photographic pair were correlated with each other. These correlated-pair DEMs gave rise to the block-diagonal form of the covariance matrix. In effect, we assumed that the covariance matrix was sparse since this correlated-pair model only requires $n + n/2$ non-zero terms to be estimated for the covariance matrix.

3.2.1. Asymmetry in Computer Vision Stereo Matching

The reason one can produce two different DEMs from two photographs is that stereo matching algorithms may not be perfectly symmetric in their output (Brown et al., 2003). This means that a DEM produced by matching image A to image B, which we denote by $A \rightarrow B$, will not lead to the same DEM as doing $B \rightarrow A$. Of course, the resulting DEM pair ($A \rightarrow B$, $B \rightarrow A$) is highly correlated. The correlated-pair error model in equation 8 is meant to capture this unknown, but possibly large, cross-correlation between the errors in the pair. One can readily calculate that this block-diagonal error model allows one to calculate the precision error exactly whenever three or more photographs overlap on the same scene.

4. Sparsity of Geometric Precision Error

As useful as the correlated-pair error model may be in certain circumstances it is a *model* and therefore cannot form the foundation of a robust process for error estimation. It is conceivable that DEMs from unrelated photographs could become correlated in their errors due to environmental factors or even instrument malfunction. A robust estimation of the covariance matrix should not depend on any assumptions of how DEMs are correlated.

Recent developments in the mathematics of sparse signal reconstruction or compressed sensing (Donoho, 2006a) offer us a mathematical procedure to deal with

this situation. During a mapping mission photographs taken from different viewing positions and orientations will lead to mapping errors that are uncorrelated but occasionally may have strong correlations between them. We just do not know a priori which DEMs will be correlated with each other, only that these cross-correlations will be sparse.

As we noted in section 3.1, the linear system is at the margin of being completely determined being shy by just M equations. If the covariance matrix was sparse enough in the sense that on the order of M independent entries were zero, the estimation would be robust. We can express this condition by dividing the number of equations we have $(M(M+1)/2 - M)$ by the number we need $(M(M+1)/2)$

$$1 - \frac{2}{M+1}. \tag{9}$$

As the number of models increases, this fraction becomes increasingly near to one – the condition for being well-determined.

We hypothesize that given enough models ($M \to \infty$), any experimental situation can be driven into a sparse regime for the precision error covariance matrix. Note that we are not talking about sparsity of the models themselves, but of the correlations between their precision errors.

The under-determined linear system 7 can be solved by using the ℓ_1-minimization technique advocated in the compressed sensing literature (Donoho, 2006b)

$$\min ||\Delta||_1 \text{ subject to } S = \Phi\Delta. \tag{10}$$

This problem can be solved as a convex optimization problem (Donoho, 2006b) by recasting it as the equivalent linear program:

$$\min \sum_i u_i \text{ subject to} \tag{11}$$

$$u_i + \Delta_i \geq 0 \tag{12}$$

$$u_i - \Delta_i \geq 0 \tag{13}$$

$$\Phi\Delta = S \tag{14}$$

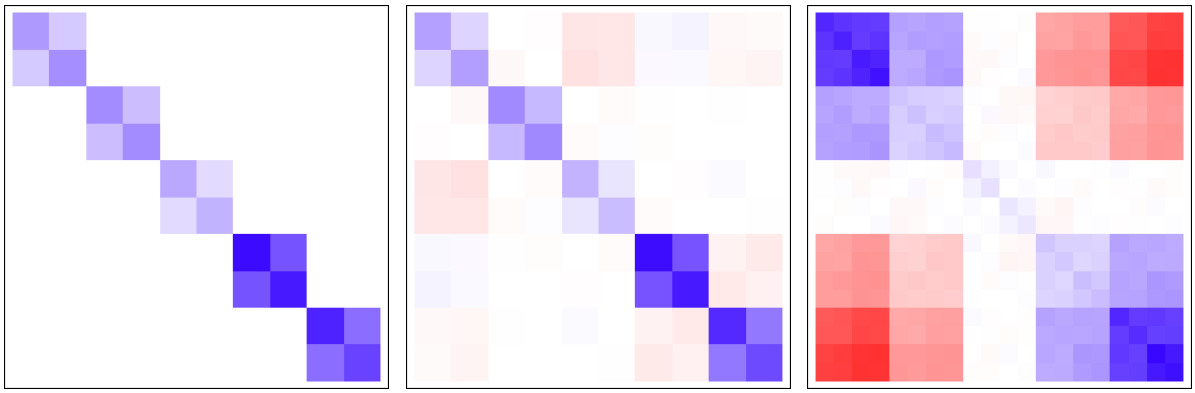
In the experimental section of the paper we will show that this approach reconstructs a covariance matrix for the precision errors that is very close to the correlated-pair model (eq. 8). Some off-diagonal terms hypothesized to be zero are about 5 times smaller than the in-pair cross correlation. We emphasize that this sparsity of errors hypothesis is an experimental assertion. No mathematical proof can be given that this sparsity condition can be met. The applicability of the assertion is based on the experimental realization of the AMA and the features of the terrain. A device built with stable imaging sensors of high quality that is mapping a reasonably static terrain would be a good candidate for a suitable condition that meets our sparsity assumption.

5. More Data Means Higher Resolution Error Maps

The name *compressed sensing* comes from the realization that sparsity implies a low-dimensional or compressible signal. If pictures of a natural scene taken with a $n \times n$ CCD can always be compressed, why take n^2 measurements? The imaging of the scene can be compressed by using less pixels and then reconstructed with an under-determined linear system. This has been dramatically demonstrated by the Rice University one-pixel camera (Wakin et al., 2006). About 1,000 measurements with a single pixel reproduced images captured by a 4,000 pixel CCD. Compressed sensing implies that we are wasting effort by taking too many measurements.

The error theory presented here gives a different perspective on this issue. Yes, reconstructing a 3-D model of the world can be done with less measurements. However, errors are an important aspect of all measurements. How confident can we be of any particular reconstruction? The only way to understand this is to produce not just maps of the territory that is being mapped, i.e. DEMs, but to also produce error maps of the same territory. The procedure for precision error estimation depends on averaging over all postings that a collection of DEMs have in common. Sparsity is only present after this averaging. The error map therefore has a much lower resolution than the DEM itself. Multiple measurements are needed to increase the resolution of this error map. In this view, no measurement is ever wasted – it leads to higher resolution in the error map of the measurements.

This suggests that the resolution of the error map should be studied by decreasing the map area that is used to create the average covariance matrix of the precision errors. As the averaging area is diminished, various cross-correlations between different DEMs will start to turn on. At some point, the number of these off-diagonal terms will be large enough to violate the condition of sparsity and the resolution limit of the error map would be reached. This resolution limit may vary across the mapped area and would naturally depend on the particular dataset. This phenomenon will be demonstrated in the experimental section.



(a) Twenty-Nine Palms correlated-pair model covariance

(b) Twenty-Nine Palms ℓ_1 minimization covariance

(c) Duke Forest covariance

Figure 1. Comparison of the covariance matrix obtained with ℓ_1-minimization versus that obtained by using the correlated-pair error model. All values are in units of m^2. Blue represents positive values, red negative values. The hue scale for figures (a) and (b) is normalized so that a value of 0.12 m^2 results in a completely saturated color pixel. Figure (c) is normalized with 2.5 m^2.

6. Experimental Results

We demonstrate the formalism for sparse precision error estimation by using a set of four aerial images taken of a desert terrain in the Twenty-Nine Palms region in California, USA. The images have been arbitrarily labeled as $\{A, B, C, D\}$. Four photographs allow us to produce 4*3=12 DEMs from all possible *matching chains* of the form $i \rightarrow j$. A blunder removal process, however, automatically identified that two of the DEMs ($B \rightarrow D$, and $D \rightarrow B$) differed in their elevation estimates by more than one meter for all postings. This pair was excluded from our calculations so the results presented here involve the remaining 10 DEMs.

6.1. Random Reconstructions via ℓ_1-Minimization

For 10 DEMs there are on the order of fifty thousand ways to write equation 4. The number of independent equations in this set is 45. An independent set selected from all possible permutations of the difference equations leads to a different reconstruction matrix Φ. To carry out the ℓ_1-minimization estimate of the 10x10 covariance matrix for the DEMs, we randomly selected ten different linearly independent sets and their corresponding Φ matrices. This was done to study the numerical stability of the reconstruction procedure. The statistical average was done over an overlap region of the DEMS that spanned the postings 500 to 1500 where 2000 by 2000 was the original size of the individual DEMs. This was done to exclude edge effects and increase the density of postings on which all DEMs gave an elevation estimate. Only postings for which we had a full 10 measurements were used. The number of postings was equal to 940,010 out of a possible million. Each posting represents an area of $(0.38\ m)^2$.

The reconstruction of the covariance matrix is shown in figure 1(b). The covariance matrix is presented as a 10x10 pixel image. For comparison, the covariance matrix reconstructed with the correlated-pair error model is shown in figure 1(a). No numerically significant variation in the reconstructed covariance matrix was observed with 10 randomly selected Φ matrices so a single figure is sufficient to summarize the results. Note that about 12 entries in the covariance matrix are practically zero – two more than the 10 entries required to define a well-determined linear system.

The ℓ_1-minimization procedure also ascribes most of the cross-correlations to the DEMs that come from asymmetrically matching the same pair of photographs. In addition, the sparse reconstruction has discovered that some of the DEMs are negatively correlated.

Is this error reconstruction correct? At this time we can point to its *self-consistent* character as strong evidence for its correctness. Neither the autonomous elevation difference equations or the ℓ_1-minimization procedure assume that certain DEMs are strongly correlated. Yet the empirically reconstructed matrix clearly shows that the 10 DEMs have a strong 5-pair structure exemplified by the block-diagonal structure. The reconstruction has 'discovered' that we used DEMs from asymmetric matching pairs.

Another self-consistent feature of the reconstruction is

that the more precise a DEM is, the smaller its cross-correlation with its asymmetric pair becomes. This is a behavior that we would expect from a system that is producing increasingly precise estimates.

6.1.1. Duke Forest DEMs

The Twenty-Nine Palms data, just discussed, is extremely high quality. The images were taken with a high-quality photogrammetric instrument. To confirm that precision errors can be recovered in more noisy data, we studied a series of aerial photographs of a forest canopy in the Duke Forest, NC taken with an off-the-shelf digital camera.

We randomly selected a track of images and picked four consecutive images. The images where multi-band and we chose the near-ir and green bands. The combination of bands and asymmetric pair matches resulted in twenty DEMs. The recovered precision error covariance matrix is shown in Figure 1(c). The recovery is similar to that for the Twenty-Nine Palms data, in that the highly-correlated pairs were discovered once again. As befits the noisier data set, the precision error estimated is $\approx 2.0\ m^2$ versus the $0.1\ m^2$ value for the Twenty-Nine Palms images.

6.2. Horizontal Resolution of the Precision Error Covariance Matrix

The self-consistent character of the reconstruction can be exploited further. The linear program that recovers the error has as side constraints that the diagonal terms of the covariance matrix have to be positive, a required property for the $\langle \delta_i^2 \rangle$ terms. To keep the reconstruction as a linear program, we did not require that the cross-correlations terms satisfy the inequality

$$\left| \frac{< \delta_i \delta_j >}{\sqrt{< \delta_i^2 >< \delta_j^2 >}} \right| \leq 1. \qquad (15)$$

The output of the linear program just turns out to satisfy these constraints for this particular dataset. Indeed, we now use the breakdown in these cross-correlation constraints to probe how much resolution can be obtained in the error map of the DEMs.

To study the resolution limit of ℓ_1-minimization procedure, we shrank the size of the area in the Twenty-Nine maps over which the difference equations were averaged. We have no independent way of verifying the validity of the reconstruction except the self-consistency check that the reconstructed vector does indeed represent a covariance matrix – its dimensionless cross-correlations should have an absolute value less than or equal to one.

Surprisingly the resolution of the covariance matrix for this data is on the order of 5x5 postings. We show an example of the covariance matrix for a patch encompassing the postings 500 to 505 in both directions in figure 2(a). The covariance matrix for the patch encompassing the postings 500 to 510 is shown in figure 2(b). The breakdown in reconstruction for the 5x5 patch is most evident in the cross-correlations related to the DEM in position 9. For this particular patch the variance of DEM 9 is calculated as $1.1\ 10^{-5}\ m^2$ and the variance of DEM 1 is calculated as $1.0\ 10^{-1}\ m^2$. The dimensionless cross-correlation between them has a value of 30.0 – the ℓ_1-minimization has not produced a proper covariance matrix for this small patch.

No breakdown is found in the 10x10 patch. Similar results were obtained over a handful of other patches over the mapped scene. Interestingly, an earlier paper by the authors had found that for this dataset the horizontal decorrelation length was in the order of 5 postings, a result that was obtained by a "cheating" experiment that used ray-tracing to establish a pseudo ground truth against which the error at the individual posting level could be calculated.

7. Conclusions and Future Work

We conclude by discussing the utility of estimating precision error even while neglecting or increasing the accuracy error. Geometric accuracy is defined by a global set of transformations. The parameters needed to define it are finite and readily extracted by knowing at most the location for three points in the world. In that sense, accuracy is cheap to obtain. Precision, on the other hand, captures the local variability of the 3-D model reconstructions. The parameters needed to model it, if one wished to do so, are correspondingly large. Therefore, 3-D model precision is expensive for the user to correct since it involves multiple measurements spread over the whole scene. Therefore, the autonomous error estimation algorithm presented here should have application in computer vision tasks where accuracy is not needed. An example of such a task is species identification by shape where the resolution is more important than the absolute size or orientation of the objects. Many more examples can be thought of, where accuracy is not relevant but precision is. One obvious class of problems for which this algorithm would not be helpful is those that require accurate geolocation of an object in the scene. A reasonable guarantee of accuracy can be obtained by using proper external references (GPS, attitude-heading sensors, etc.) but this algorithm is invariant to their accuracy error.

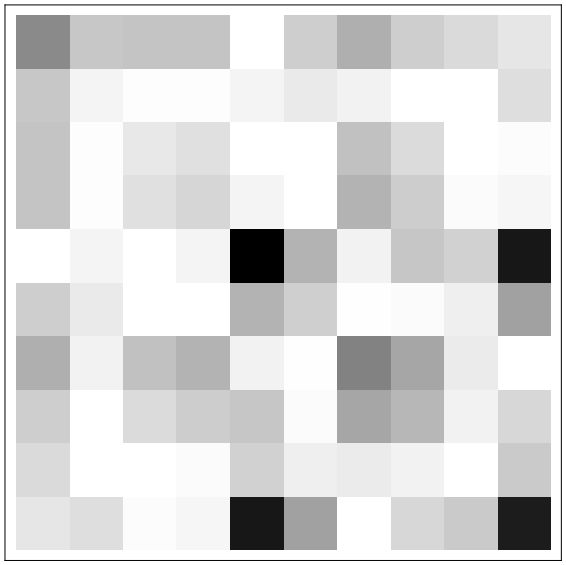
Table 1. Covariance matrix entries along the block diagonal for the ten DEMs in the 29 Palms dataset. Variance is in units of m^2.

DEM	$< \delta_i^2 >$	$< \delta_i \delta_j > / \sqrt{< \delta_i^2 >< \delta_j^2 >}$
AB	0.044	0.45
BA	0.046	
AC	0.056	0.59
CA	0.056	
AD	0.036	0.38
DA	0.031	
BC	0.114	0.73
CB	0.108	
CD	0.100	0.69
DC	0.085	

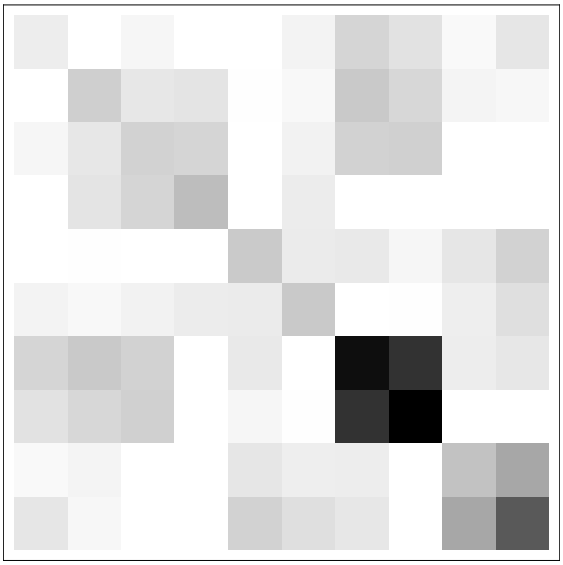
(a) ℓ_1-minimization

DEM	$< \delta_i^2 >$	$< \delta_i \delta_j > / \sqrt{< \delta_i^2 >< \delta_j^2 >}$
AB	0.048	0.50
BA	0.053	
AC	0.054	0.57
CA	0.054	
AD	0.041	0.44
DA	0.036	
BC	0.115	0.73
CB	0.108	
CD	0.104	0.71
DC	0.089	

(b) Correlated-pair error model



(a) Covariance matrix for a patch of size 5x5



(b) Covariance matrix for a patch of size 10x10

The present paper has demonstrated that the covariance matrix of the geometric precision errors can be measured autonomously. The present formalism applies to other areas of Machine Learning, indeed, to any scientific setting where multiple scalar predictions are available for a set of entities. For example, the precision error equations (3) can used for comparing the relevance judgment of various information retrieval algorithms. Instead of elevations, one would use the binary judgment of relevancy to compare the retrieval models.

Future work will continue to explore the utility of the precision error covariance matrix for data fusion (what is the optimal way to combine the DEMs to minimize the total precision error?), and extend the formalism of the precision error equations (3) to multi-dimensional or other non-scalar models such as 3-D locations or parse trees in computational linguistics.

Acknowledgments

This work was supported by grant (IIS-0430742) from the United States National Science Foundation and contract TT0690688 from the Advanced Technology Laboratory, Lockheed-Martin Corporation.

References

Beardsley, P., Torr, P., & Zisserman, A. (1996). 3D model acquisition from extended image sequences. *Proceedings of the 4th European Conference on Computer Vision, ECCV'96* (p. 683). Cambridge, UK: Springer-Verlag. Part 2 (of 2).

Bishop, C. M. (2007). *Pattern recognition and machine learning*. Springer.

Brown, M. Z., Burschka, D., & Hager, G. D. (2003). Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 993–1008. Compilation and indexing terms, Copyright 2007 Elsevier Inc. All rights reserved.

Corrada-Emmanuel, A., Pinette, B., Ostapchenko, A., & Schultz, H. (2007). Improving autonomous estimates of dem uncertainties by exploiting computer matching asymmetries. *8th Conference on Optical 3-D Measurement Techniques*. Zurich, Switzerland.

Corrada-Emmanuel, A., & Schultz, H. (2008). *Autonomous estimates of horizontal decorrelation lengths for digital elevation models* (Technical Report). Computer Science Department, University of Massachusetts at Amherst.

Donoho, D. L. (2006a). Compressed sensing. *IEEE Transactions on Information Theory*, *52*, 1289–1306.

Donoho, D. L. (2006b). For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, *59*, 797–829.

Faugeras, O., Luong, Q.-T., & Papadopoulo, T. (2001). *The geometry of multiple images : the laws that govern the formation of multiple images of a scene and some of their applications*. Cambridge, Mass.: MIT Press.

Hartley, R., & Zisserman, A. (2000). *Multiple view geometry in computer vision*. Cambridge, U.K. ; New York: Cambridge University Press.

Pollefeys, M., Koch, R., & Gool, L. V. (1999). Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, *32*, 7–25.

Wakin, M., Laska, J., Duarte, M., Baro, D., Sarvotham, S., andKevin Kelly, D. T., & Baraniuk, R. (2006). An architecture for compressive imaging. *Proceedings of the International Conference on Image Processing – ICIP 2006*. Atlanta, GA.

Stability of Transductive Regression Algorithms

Corinna Cortes CORINNA@GOOGLE.COM
Google Research, 76 Ninth Avenue, New York, NY 10011.

Mehryar Mohri MOHRI@CIMS.NYU.EDU
Courant Institute of Mathematical Sciences and Google Research, 251 Mercer Street, New York, NY 10012.

Dmitry Pechyony PECHYONY@CS.TECHNION.AC.IL
Technion - Israel Institute of Technology, Haifa 32000, Israel.

Ashish Rastogi RASTOGI@CS.NYU.EDU
Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012.

Abstract

This paper uses the notion of algorithmic stability to derive novel generalization bounds for several families of transductive regression algorithms, both by using convexity and closed-form solutions. Our analysis helps compare the stability of these algorithms. It suggests that several existing algorithms might not be stable but prescribes a technique to make them stable. It also reports the results of experiments with local transductive regression demonstrating the benefit of our stability bounds for model selection, in particular for determining the radius of the local neighborhood used by the algorithm.

1. Introduction

Many learning problems in information extraction, computational biology, natural language processing and other domains can be formulated as *transductive inference* problems (Vapnik, 1982). In the transductive setting, the learning algorithm receives both a labeled training set, as in the standard induction setting, and a set of unlabeled test points. The objective is to predict the labels of the test points. No other test points will ever be considered. This setting arises in a variety of applications. Often, the points to label are known but they have not been assigned a label due to the prohibitive cost of labeling. This motivates the use of transductive algorithms which leverage the unlabeled data during training to improve learning performance.

This paper deals with transductive regression, which arises in problems such as predicting the real-valued labels of the nodes of a known graph in computational biology, or the scores associated with known documents in information extraction or search engine tasks.

Several algorithms have been devised for the specific setting of transductive regression (Belkin et al., 2004b; Chapelle et al., 1999; Schuurmans & Southey, 2002; Cortes & Mohri, 2007). Several other algorithms introduced for transductive classification can be viewed in fact as transductive regression ones as their objective function is based on the squared loss, e.g., (Belkin et al.2004a; 2004b). Cortes and Mohri (2007) also gave explicit VC-dimension generalization bounds for transductive regression that hold for all bounded loss functions and coincide with the tight classification bounds of Vapnik (1998) when applied to classification.

This paper presents novel algorithm-dependent generalization bounds for transductive regression. Since they are algorithm-specific, these bounds can often be tighter than bounds based on general complexity measures such as the VC-dimension. Our analysis is based on the notion of algorithmic stability.

In Sec. 2 we give a formal definition of the transductive regression setting and the notion of stability for transduction. Our bounds generalize the stability bounds given by Bousquet and Elisseeff (2002) for the inductive setting and extend to regression the stability-based transductive classification bounds of (El-Yaniv & Pechyony, 2006). Standard concentration bounds

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

such as McDiarmid's bound (McDiarmid, 1989) cannot be readily applied to the transductive regression setting since the points are not drawn independently but uniformly without replacement from a finite set. Instead, a generalization of McDiarmid's bound that holds for random variables sampled without replacement is used, as in (El-Yaniv & Pechyony, 2006). Sec. 3.1 gives a simpler proof of this bound.

This concentration bound is used to derive a general transductive regression stability bound in Sec. 3.2. In Sec. 4, we present the stability coefficients for a family of local transductive regression algorithms. The analysis in this section is based on convexity. In Sec. 5, we study the stability of other transductive regression algorithms (Belkin et al., 2004a; Wu & Schölkopf, 2007; Zhou et al., 2004; Zhu et al., 2003) based on their closed form solution and propose a modification to the seemingly unstable algorithm that makes them stable and guarantees a non-trivial generalization bound. Finally, Sec. 6 shows the results of experiments with local transductive regression demonstrating the benefit of our stability bounds for model selection, in particular for determining the radius of the local neighborhood used by the algorithm. This provides a partial validation of our bounds and analysis.

2. Definitions

Let us first describe the transductive learning setting. Assume that a full sample X of $m+u$ examples is given. The learning algorithm further receives the labels of a random subset S of X of size m which serves as a training sample. The remaining u unlabeled examples, $x_{m+1}, \ldots, x_{m+u} \in X$, serve as test data. We denote by $X \vdash (S,T)$ a partitioning of X into the training set S and the test set T. The *transductive learning* problem consists of predicting accurately the labels $y_{m+1}, \ldots, y_{m+u}$ of the test examples, no other test examples will ever be considered (Vapnik, 1998).[1] The specific problems where the labels are real-valued numbers, as in the case studied in this paper, is that of *transduction regression*. It differs from the standard (*induction*) regression since the learning algorithm is given the unlabeled test examples beforehand and can thus exploit this information to improve performance.

We denote by $c(h,x)$ the cost of an error of a hypothesis h on a point x labeled with $y(x)$. The cost function commonly used in regression is the squared loss $c(h,x) = (h(x) - y(x))^2$. In the remaining of this paper, we will assume a squared loss but many of our results generalize to other convex cost functions. The training and test errors of h are respectively $\widehat{R}(h) = \frac{1}{m}\sum_{k=1}^{m} c(h,x_k)$ and $R(h) = \frac{1}{u}\sum_{k=1}^{u} c(h,x_{m+k})$. The generalization bounds we derive are based on the notion of transductive algorithmic stability.

Definition 1 (Transduction β-stability)**.** *Let L be a transductive learning algorithm and let h denote the hypothesis returned by L for $X \vdash (S,T)$ and h' the hypothesis returned for $X \vdash (S',T')$. L is said to be* uniformly β-stable *with respect to the cost function c if there exists $\beta \geq 0$ such that for any two partitionings $X \vdash (S,T)$ and $X \vdash (S',T')$ that differ in exactly one training (and thus test) point and for all $x \in X$,*

$$\left|c(h,x) - c(h',x)\right| \leq \beta. \tag{1}$$

3. Transduction Stability Bounds

3.1. Concentration Bound for Sampling without Replacement

Stability-based generalization bounds in the inductive setting are based on McDiarmid's inequality (1989). In the transductive setting, the points are drawn uniformly without replacement and thus are not independent. Therefore, McDiarmid's concentration bound cannot be readily used. Instead, a generalization of McDiarmid's bound for sampling without replacement is needed as in El-Yaniv and Pechyony (2006).

We will denote by $\mathbf{S}_1^m$ a sequence of random variables $S_1, \ldots, S_m$ and write $\mathbf{S}_1^m = \mathbf{x}_1^m$ as a shorthand for the m equalities $S_i = x_i$, $i = 1, \ldots, m$ and $\Pr[\mathbf{x}_{i+1}^m | \mathbf{x}_1^{i-1}, x_i] = \Pr[\mathbf{S}_{i+1}^m = \mathbf{x}_{i+1}^m | \mathbf{S}_1^{i-1} = \mathbf{x}_1^{i-1}, S_i = x_i]$.

Theorem 1 ((McDiarmid, 1989), 6.10)**.** *Let $\mathbf{S}_1^m$ be a sequence of random variables, each S_i taking values in the set X, and assume that a measurable function $\phi : X^m \mapsto \mathbb{R}$ satisfies: $\forall i \in [1,m], \forall x_i, x_i' \in X$,*

$$\left|\mathbb{E}_{\mathbf{S}_{i+1}^m}\left[\phi | \mathbf{S}_1^{i-1}, S_i = x_i\right] - \mathbb{E}_{\mathbf{S}_{i+1}^m}\left[\phi | \mathbf{S}_1^{i-1}, S_i = x_i'\right]\right| \leq c_i.$$

$$\textit{Then, } \forall \epsilon > 0, \quad \Pr\left[|\phi - \mathbb{E}\left[\phi\right]| \geq \epsilon\right] \leq 2\exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2}\right).$$

The following is a concentration bound for sampling without replacement needed to analyze the generalization of transductive algorithms.

Theorem 2. *Let $\mathbf{x}_1^m$ be a sequence of random variables, sampled from an underlying set X of $m+u$ elements without replacement, and let that $\phi : X^m \mapsto \mathbb{R}$*

[1] Another natural setting for transduction is one where the training and test samples are both drawn according to the same distribution and where the test points, but not their labels, are made available to the learning algorithm. However, as pointed out by Vapnik (1998), any generalization bound in the setting we analyze directly yields a bound for this other setting, essentially by taking the expectation.

be a symmetric function such that for all $i \in [1,m]$ and for all $x_1,\ldots,x_m \in X$ and $x'_1,\ldots,x'_m \in X$,

$$\left|\phi(x_1,\ldots,x_m) - \phi(x_1,\ldots,x_{i-1},x'_i,x_{i+1},\ldots,x_m)\right| \leq c.$$

Then, $\forall \epsilon > 0, \quad \Pr\left[\left|\phi - \mathbb{E}\left[\phi\right]\right| \geq \epsilon\right] \leq 2\exp\left(\frac{-2\epsilon^2}{\alpha(m,u)c^2}\right),$

where $\alpha(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2\max\{m,u\})}$.

Proof. For a fixed $i \in [1,m]$, let $g(\mathbf{S}_1^{i-1}) = \mathbb{E}_{\mathbf{S}_{i+1}^m}\left[\phi|\mathbf{S}_1^{i-1}, S_i{=}x_i\right] - \mathbb{E}_{\mathbf{S}_{i+1}^m}\left[\phi|\mathbf{S}_1^{i-1}, S_i{=}x'_i\right]$. Then, $g(\mathbf{x}_1^{i-1}) = \sum_{\mathbf{x}_{i+1}^m} \phi(\mathbf{x}_1^{i-1}, x_i, \mathbf{x}_{i+1}^m) \Pr[\mathbf{x}_{i+1}^m|\mathbf{x}_1^{i-1}, x_i] - \sum_{\mathbf{x'}_{i+1}^m} \phi(\mathbf{x}_1^{i-1}, x'_i, \mathbf{x'}_{i+1}^m) \Pr[\mathbf{x'}_{i+1}^m|\mathbf{x}_1^{i-1}, x'_i]$.

For uniform sampling without replacement, the probability terms can be written as: $\Pr\left[\mathbf{x}_{i+1}^m|\mathbf{x}_1^{i-1}, x_i\right] = \prod_{k=i}^{m-1} \frac{1}{m+u-k} = \frac{u!}{(m+u-i)!}$. Thus, $g(\mathbf{x}_1^{i-1}) = \frac{u!}{(m+u-i)!}[\sum_{\mathbf{x}_{i+1}^m} \phi(\mathbf{x}_1^{i-1}, x_i, \mathbf{x}_{i+1}^m) - \sum_{\mathbf{x'}_{i+1}^m} \phi(\mathbf{x}_1^{i-1}, x'_i, \mathbf{x'}_{i+1}^m)]$. To compute the expression between brackets, we divide the set of permutations $\{\mathbf{x'}_{i+1}^m\}$ into two sets, those that contain x_i and those that do not. If a permutation $\mathbf{x'}_{i+1}^m$ contains x_i we can write it as $\mathbf{x'}_{i+1}^{k-1} x_i \mathbf{x'}_{k+1}^m$, where k is such that $x'_k = x_i$. We then match it up with the permutation $x_i \mathbf{x'}_{i+1}^{k-1} \mathbf{x'}_{k+1}^m$ from the set $\{x_i \mathbf{x}_{i+1}^m\}$. These two permutations contain exactly the same elements, and since the function ϕ is symmetric in its arguments, the difference in the value of the function on the permutations is zero.

In the other case, if a permutation $\mathbf{x'}_{i+1}^m$ does not contain the element x_i, then we simply match it up with the same permutation in $\{\mathbf{x}_{i+1}^m\}$. The matching permutations appearing in the summation are then $x_i \mathbf{x'}_{i+1}^m$ and $x'_i \mathbf{x'}_{i+1}^m$ which clearly only differ with respect to x_i. The difference in the value of the function ϕ in this case can be bounded by c. The number of such permutations is $(m-i)!\binom{m+u-(i+1)}{m-i} = \frac{(m+u-i-1)!}{(u-1)!}$, which leads to the following upper bound: $\sum_{\mathbf{x}_{i+1}^m} \phi(\mathbf{x}_1^{i-1}, x_i, \mathbf{x}_{i+1}^m) - \sum_{\mathbf{x'}_{i+1}^m} \phi(\mathbf{x}_1^{i-1}, x'_i, \mathbf{x'}_{i+1}^m) \leq \frac{(m+u-i-1)!}{(u-1)!} c$, which implies that $|g(\mathbf{x}_1^{i-1})| \leq \frac{u!}{(m+u-i)!} \cdot \frac{(m+u-i-1)!}{(u-1)!} c \leq \frac{u}{m+u-i} c$. Then, combining Theorem 1 with the identity $\sum_{i=1}^m \frac{1}{(m+u-i)^2} \leq \frac{m}{m+u-1/2} \frac{1}{u-1/2}$, yields that $\Pr\left[\left|\phi - \mathbb{E}\left[\phi\right]\right| \geq \epsilon\right] \leq 2\exp\left(\frac{-2\epsilon^2}{\alpha_u(m,u)c^2}\right)$, where $\alpha_u(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2u)}$. The function ϕ is symmetric in m and u in the sense that selecting one of the sets uniquely determines the other set. The statement of the theorem then follows from a similar bound with $\alpha_m(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2m)}$, taking the tighter of the two. □

3.2. Transductive Stability Bound

To obtain a general transductive regression stability bound, we apply the concentration bound of Theorem 2 to the random variable $\phi(S) = R(h) - \widehat{R}(h)$. To do so, we need to bound $\mathbb{E}_S\left[\phi(S)\right]$, where S is a random subset of X of size m, and $|\phi(S) - \phi(S')|$ where S and S' are samples differing by exactly one point.

Lemma 1. *Let H be a bounded hypothesis set ($\forall x \in X, |h(x) - y(x)| \leq B$) and L a β-stable algorithm returning the hypotheses h and h' for two training sets S and S of size m each, respectively, differing in exactly one point. Then,*

$$|\phi(S) - \phi(S')| \leq 2\beta + B^2(m+u)/(mu). \tag{2}$$

Proof. By definition, S and S' differ exactly in one point. Let $x_i \in S$, $x_{m+j} \in S'$ be the points in which the two sets differ. The lemma follows from the observation that for each one of the $m-1$ common labeled points in S and S', and for each one of the $u-1$ common test points in T and T' (recall $T = X \setminus S$, $T' = X \setminus S'$), the difference in cost is bounded by β, while for x_i and x_{m+j}, the difference in cost is bounded by B^2. Then, it follows that $|\phi(S) - \phi(S')| \leq \frac{(u-1)\beta}{u} + \frac{(m-1)\beta}{m} + \frac{B^2}{u} + \frac{B^2}{m} \leq 2\beta + B^2\left(\frac{1}{u} + \frac{1}{m}\right)$. □

Lemma 2. *Let h be the hypothesis returned by a β-stable algorithm L. Then, $|\mathbb{E}_S\left[\phi(S)\right]| \leq \beta$.*

Proof. By definition of $\phi(S)$, its expectation is $\frac{1}{u}\sum_{k=1}^u \mathbb{E}_S\left[c(h, x_{m+k})\right] - \frac{1}{m}\sum_{k=1}^m \mathbb{E}_S\left[c(h, x_k)\right]$. Since $\mathbb{E}_S\left[c(h, x_{m+j})\right]$ is the same for all $j \in [1,u]$, and $\mathbb{E}_S\left[c(h, x_i)\right]$ the same for all $i \in [1,m]$, for any i and j, $\mathbb{E}_S\left[\phi(S)\right] = \mathbb{E}_S\left[c(h, x_{m+j})\right] - \mathbb{E}_S\left[c(h, x_i)\right] = \mathbb{E}_{S'}\left[c(h', x_i)\right] - \mathbb{E}_S\left[c(h, x_i)\right]$. Thus, $\mathbb{E}_S\left[\phi(S)\right] = \mathbb{E}_{S,S' \sim X}\left[c(h', x_i) - c(h, x_i)\right] \leq \beta$. □

Theorem 3. *Let H be a bounded hypothesis set ($\forall x \in X, |h(x) - y(x)| \leq B$) and L a β-stable algorithm. Let h be the hypothesis returned by L when trained on $X \vdash (S,T)$. Then, for any $\delta > 0$, with prob. at least $1-\delta$,*

$$R(h) \leq \widehat{R}(h) + \beta + \left(2\beta + \frac{B^2(m+u)}{mu}\right)\sqrt{\frac{\alpha(m,u)\ln\frac{1}{\delta}}{2}}.$$

Proof. The result follows directly from Theorem 2 and Lemmas 1 and 2. □

This is a general bound that applies to *any* transductive algorithm. To apply it, the stability coefficient β, which depends on m and u, needs to be determined. In the subsequent sections, we derive bounds on β for a number of transductive regression algorithms (Cortes

& Mohri, 2007; Belkin et al., 2004a; Wu & Schölkopf, 2007; Zhou et al., 2004; Zhu et al., 2003).

4. Stability of Local Transductive Regression Algorithms

This section describes and analyzes a general family of local transductive regression algorithms (LTR) generalizing the algorithm of Cortes and Mohri (2007).

LTR algorithms can be viewed as a generalization of the so-called kernel regularization-based learning algorithms to the transductive setting. The objective function that they minimize is of the form:

$$F(h,S) = \|h\|_K^2 + \frac{C}{m}\sum_{k=1}^{m} c(h,x_k) + \frac{C'}{u}\sum_{k=1}^{u} \widetilde{c}(h,x_{m+k}), \quad (3)$$

where $\|\cdot\|_K$ is the norm in the reproducing kernel Hilbert space (RKHS) with associated kernel K, $C \geq 0$ and $C' \geq 0$ are trade-off parameters, and $\widetilde{c}(h,x) = (h(x) - \widetilde{y}(x))^2$ is the error of the hypothesis h on the unlabeled point x with respect to a pseudo-target $\widetilde{y}$.

Pseudo-targets are obtained from neighborhood labels $y(x)$ by a local weighted average. Neighborhoods can be defined as a ball of radius r around each point in the feature space. We will denote by β_{loc} the score-stability coefficient of the local algorithm used, that is the maximal amount by which the two hypotheses differ on an given point, when trained on samples disagreeing on one point. This notion is stronger than that of cost-based stability.

In this section, we use the bounded-labels assumption, that is $\forall x \in S$, $|y(x)| \leq M$. We also assume that for any $x \in X$, $K(x,x) \leq \kappa^2$. We will use the following bound based on the reproducing property and the Cauchy-Schwarz inequality valid for any hypothesis $h \in H : \forall x \in X$,

$$|h(x)| = |\langle h, K(x,\cdot)\rangle| \leq \|h\|_K \sqrt{K(x,x)} \leq \kappa \|h\|_K. \quad (4)$$

Lemma 3. *Let h be the hypothesis minimizing (3). Assume that for any $x \in X$, $K(x,x) \leq \kappa^2$. Then, for any $x \in X$, $|h(x)| \leq \kappa M\sqrt{C+C'}$.*

Proof. The proof is a straightforward adaptation of the technique of (Bousquet & Elisseeff, 2002) to LTR algorithms. By Eqn. 4, $|h(x)| \leq \kappa \|h\|_K$. Let $\mathbf{0} \in \mathbb{R}^{m+u}$ be the hypothesis assigning label zero to all examples. By definition of h,

$$F(h,S) \leq F(\mathbf{0},S) \leq (C+C')M^2.$$

Using $\|h\|_K \leq \sqrt{F(h,S)}$ yields the statement. □

Since $|h(x)| \leq \kappa M\sqrt{C+C'}$, this immediately gives us a bound on $|h(x) - y(x)| \leq M(1 + \kappa\sqrt{C+C'})$. Thus, we are in a position to apply Theorem 3 with $B = AM$, $A = 1 + \kappa\sqrt{C+C'}$.

We now derive a bound on the stability coefficient β. To do so, the key property we will use is the convexity of $h \mapsto c(h,x)$. Note, however, that in the case of $\widetilde{c}$, the pseudo-targets may depend on the training set S. This dependency matters when we wish to apply convexity with two hypotheses h and h' obtained by training on different samples S and S'. For convenience, for any two such fixed hypotheses h and h', we extend the definition of $\widetilde{c}$ as follows. For all $t \in [0,1]$,

$$\widetilde{c}(th+(1-t)h',x) = \big((th+(1-t)h')(x) - (t\widetilde{y}+(1-t)\widetilde{y}')\big)^2.$$

This allows us to use the same convexity property for $\widetilde{c}$ as for c for any two fixed hypotheses h and h', as verified by the following lemma, and does not affect the proofs otherwise.

Lemma 4. *Let h be a hypothesis obtained by training on S and h' by training on S'. Then, for all $t \in [0,1]$,*

$$t\widetilde{c}(h,x) + (1-t)\widetilde{c}(h',x) \geq \widetilde{c}(th+(1-t)h',x). \quad (5)$$

Proof. Let $\widetilde{y} = \widetilde{y}(x)$ be the pseudo-target value at x when the training set is S and $\widetilde{y}' = \widetilde{y}'(x)$ when the training set is S'. For all $t \in [0,1]$,

$$\begin{aligned} &tc(h,x) + (1-t)c(h',x) - c(th+(1-t)h',x) \\ &= t(h(x)-\widetilde{y})^2 + (1-t)(h'(x)-\widetilde{y}')^2 \\ &- \left[t(h(x)-\widetilde{y}) + (1-t)(h'(x)-\widetilde{y}')\right]^2. \end{aligned}$$

The statement of the lemma follows directly by the convexity of $x \mapsto x^2$ over real numbers. □

Let h be a hypothesis obtained by training on S and h' by training on S'. Let $\Delta = h - h'$. Then, for all $x \in X$, $|c(h,x) - c(h',x)| = |\Delta(x)\left((h(x)-y(x)) + (h'(x)-y(x))\right)| \leq 2M(1 + \kappa\sqrt{C+C'})|\Delta(x)|$. As in 4, for all $x \in X$, $|\Delta(x)| \leq \kappa\|\Delta\|_K$, thus for all $x \in X$,

$$|c(h,x) - c(h',x)| \leq 2M(1+\kappa\sqrt{C+C'})\kappa\|\Delta\|_K. \quad (6)$$

Lemma 5. *Assume that for all $x \in X$, $|y(x)| \leq M$. Let S and S' be two samples differing by exactly one point. Let h be the hypothesis returned by the algorithm minimizing the objective function $F(h,S)$, h' be the hypothesis obtained by minimization of $F(h,S')$ and let $\widetilde{y}$ and $\widetilde{y}'$ be the corresponding pseudo-targets. Then,*

$$\begin{aligned} &C\left[c(h',x_i) - c(h,x_i)\right]/m - C'\left[\widetilde{c}(h',x_i) - \widetilde{c}(h,x_i)\right]/u \\ &\leq 2AM\left(\kappa\|\Delta\|_K\left(C/m + C'/u\right) + \beta_{loc}C'/u\right). \end{aligned}$$

where $\Delta = h' - h$ and $A = 1 + \kappa\sqrt{C+C'}$.

Proof. Let $\tilde{c}(h_i, \tilde{y}_i) = \tilde{c}(h, x_i)$ and $\tilde{c}(h'_i, \tilde{y}'_i) = \tilde{c}(h', x_i)$. By Lemma 3 and the bounded-labels assumption,

$$\begin{aligned}
&|\widetilde{c}(h'_i, \widetilde{y}'_i) - \widetilde{c}(h_i, \widetilde{y}_i)| \\
&= |\widetilde{c}(h'_i, \widetilde{y}'_i) - \widetilde{c}(h'_i, \widetilde{y}_i) + \widetilde{c}(h'_i, \widetilde{y}_i) - \widetilde{c}(h_i, \widetilde{y}_i)| \\
&\leq |(\widetilde{y}'_i - \widetilde{y}_i)(\widetilde{y}'_i + \widetilde{y}_i - 2h'_i)| + |(h'_i - h_i)(h'_i + h_i - 2\widetilde{y}_i)|\,.
\end{aligned}$$

By the score-stability of local estimates, $\widetilde{y}'(x_i) - \widetilde{y}(x_i) \leq \beta_{loc}$. Thus,

$$|\widetilde{c}(h'_i, \widetilde{y}'_i) - \widetilde{c}(h_i, \widetilde{y}_i)| \leq 2AM(\beta_{loc} + \kappa\|\Delta\|_K). \quad (7)$$

Using 6 leads after simplification to the statement of the lemma. □

The proof of the following theorem is based on Lemma 4 and Lemma 5 and is reserved to a longer version of this paper.

Theorem 4. *Assume that for all $x \in X$, $|y(x)| \leq M$ and there exists κ such that $\forall x \in X$, $K(x,x) \leq \kappa^2$. Further, assume that the local estimator has uniform stability coefficient β_{loc}. Let $A = 1+\kappa\sqrt{C+C'}$. Then,* LTR *is uniformly β-stable with*

$$\beta \leq 2(AM)^2\kappa^2\left[\frac{C}{m}+\frac{C'}{u}+\sqrt{\left(\frac{C}{m}+\frac{C'}{u}\right)^2+\frac{2C'\beta_{loc}}{AM\kappa^2 u}}\right].$$

Our experiments with LTR will demonstrate the benefit of this bound for model selection (Sec. 6).

5. Stability Based on Closed-Form Solutions

5.1. Unconstrained Regularization Algorithms

In this section, we consider a family of transductive regression algorithms that can be formulated as the following optimization problem:

$$\min_{\mathbf{h}} \mathbf{h}^T\mathbf{Q}\mathbf{h} + (\mathbf{h}-\mathbf{y})^T\mathbf{C}(\mathbf{h}-\mathbf{y}). \quad (8)$$

$\mathbf{Q} \in \mathbb{R}^{(m+u)\times(m+u)}$ is a symmetric regularization matrix, $\mathbf{C} \in \mathbb{R}^{(m+u)\times(m+u)}$ is a symmetric matrix of empirical weights (in practice it is often a diagonal matrix), $\mathbf{y} \in \mathbb{R}^{(m+u)\times 1}$ are the target values of the m labeled points together with the pseudo-target values of the u unlabeled points (in some formulations, the pseudo-target value is 0), and $\mathbf{h} \in \mathbb{R}^{(m+u)\times 1}$ is a column vector whose ith row is the predicted target value for the x_i. The closed-form solution of (8) is given by

$$\mathbf{h} = (\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})^{-1}\mathbf{y}. \quad (9)$$

The formulation (8) is quite general and includes as special cases the algorithms of (Belkin et al., 2004a; Wu & Schölkopf, 2007; Zhou et al., 2004; Zhu et al., 2003). We present a general framework for bounding the stability coefficient of these algorithms and then examine the stability coefficient of each of these algorithms in turn.

For a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ we will denote by $\lambda_M(\mathbf{A})$ its largest eigenvalue and $\lambda_m(\mathbf{A})$ its smallest. Then, for any $\mathbf{v} \in \mathbb{R}^{n\times 1}$, $\lambda_m(\mathbf{A})\|\mathbf{v}\|_2 \leq \|\mathbf{A}\mathbf{v}\|_2 \leq \lambda_M(\mathbf{A})\|\mathbf{v}\|_2$. We will also use in the proof of the following proposition the fact that for symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$, $\lambda_M(\mathbf{AB}) \leq \lambda_M(\mathbf{A})\lambda_M(\mathbf{B})$.

Proposition 1. *Let $\mathbf{h}$ and $\mathbf{h}'$ solve (8), under test and training sets that differ exactly in one point and let $\mathbf{C}, \mathbf{C}', \mathbf{y}, \mathbf{y}'$ be the analogous empirical weight and the target value matrices. Then,*

$$\|\mathbf{h}'-\mathbf{h}\|_2 \leq \frac{\|\mathbf{y}'-\mathbf{y}\|_2}{\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})}+1} + \frac{\lambda_M(\mathbf{Q})\|\mathbf{C}'^{-1}-\mathbf{C}^{-1}\|_2\,\|\mathbf{y}\|_2}{\left(\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C}')}+1\right)\left(\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})}+1\right)}.$$

Proof. Let $\Delta = \mathbf{h}' - \mathbf{h}$ and $\Delta\mathbf{y} = \mathbf{y}' - \mathbf{y}$. Let $\mathfrak{C} = (\mathbf{C}^{-1}\mathbf{Q} + \mathbf{I})$ and $\mathfrak{C}' = (\mathbf{C}'^{-1}\mathbf{Q} + \mathbf{I})$. By definition,

$$\begin{aligned}
\Delta &= \mathfrak{C}'^{-1}\mathbf{y}' - \mathfrak{C}^{-1}\mathbf{y} \\
&= \mathfrak{C}'^{-1}\Delta\mathbf{y} + (\mathfrak{C}'^{-1} - \mathfrak{C}^{-1})\mathbf{y} \\
&= \mathfrak{C}'^{-1}\Delta\mathbf{y} + (\mathfrak{C}^{-1}\left[(\mathbf{C}^{-1} - \mathbf{C}'^{-1})\mathbf{Q}\right]\mathfrak{C}'^{-1})\mathbf{y}.
\end{aligned}$$

$$\text{Thus, } \|\Delta\|_2 \leq \frac{\|\Delta\mathbf{y}\|_2}{\lambda_m(\mathfrak{C})} + \frac{\lambda_M(\mathbf{Q})\|\mathbf{C}'^{-1}-\mathbf{C}^{-1}\|_2 \cdot \|\mathbf{y}\|_2}{\lambda_m(\mathfrak{C}')\lambda_m(\mathfrak{C})}. \quad (10)$$

Furthermore, $\lambda_m(\mathfrak{C}) \geq \frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})}+1$. Plugging this bound back into Eqn. 10 yields:

$$\|\Delta\|_2 \leq \frac{\|\Delta\mathbf{y}\|_2}{\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})}+1} + \frac{\lambda_M(\mathbf{Q})\|\mathbf{C}'^{-1}-\mathbf{C}^{-1}\|_2\|\mathbf{y}\|_2}{\left(\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C}')}+1\right)\left(\frac{\lambda_m(\mathbf{Q})}{\lambda_M(\mathbf{C})}+1\right)}. \quad \square$$

Since $\|\mathbf{h}'-\mathbf{h}\|_\infty$ is bounded by $\|\mathbf{h}'-\mathbf{h}\|_2$, the proposition provides a bound on the score-stability of $\mathbf{h}$ for the transductive regression algorithms of Zhou et al. (2004); Wu and Schölkopf (2007); Zhu et al. (2003). For each of these algorithms, the pseudo-targets used are zero. If we make the bounded labels assumption ($\forall x \in X, |y(x)| \leq M$, for some $M > 0$), it is not difficult to show that $\|\mathbf{y}-\mathbf{y}'\|_2 \leq \sqrt{2}M$ and $\|\mathbf{y}\|_2 \leq \sqrt{m}M$. We now examine each algorithm in turn.

Consistency method (CM) In the CM algorithm (Zhou et al., 2004), the matrix $\mathbf{Q}$ is a normalized Laplacian of a weight matrix $\mathbf{W} \in \mathbb{R}^{(m+u)\times(m+u)}$ that captures affinity between pairs of points in the full sample X. Thus, $\mathbf{Q} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, where $\mathbf{D} \in \mathbb{R}^{(m+u)\times(m+u)}$ is a diagonal matrix, with $[\mathbf{D}]_{i,i} =$

$\sum_j [\mathbf{W}]_{i,j}$. Note that $\lambda_m(\mathbf{Q}) = 0$. Furthermore, matrices $\mathbf{C}$ and $\mathbf{C}'$ are identical in CM, both diagonal matrices with (i,i)th entry equal to a positive constant $\mu > 0$. Thus $\mathbf{C}^{-1} = \mathbf{C}'^{-1}$ and using Prop. 1, we obtain the following bound on the score-stability of the CM algorithm: $\beta_{\texttt{CM}} \leq \sqrt{2}M$.

Local learning regularization ($\texttt{LL}-\texttt{Reg}$) In the $\texttt{LL}-\texttt{Reg}$ algorithm (Wu & Schölkopf, 2007), the regularization matrix $\mathbf{Q}$ is $(\mathbf{I}-\mathbf{A})^T(\mathbf{I}-\mathbf{A})$, where $\mathbf{I} \in \mathbb{R}^{(m+u)\times(m+u)}$ is an identity matrix and $\mathbf{A} \in \mathbb{R}^{(m+u)\times(m+u)}$ is a non-negative weight matrix that captures the local similarity between all pairs of points in X. $\mathbf{A}$ is normalized, i.e. each of its rows sum to 1. Let $C_l, C_u > 0$ be two positive constants. The matrix $\mathbf{C}$ is a diagonal matrix with $[\mathbf{C}]_{i,i} = C_l$ if $x_i \in S$ and C_u otherwise. Let $C_{\max} = \max\{C_l, C_u\}$ and $C_{\min} = \min\{C_l, C_u\}$. Thus, $\|\mathbf{C}'^{-1} - \mathbf{C}^{-1}\|_2 = \sqrt{2}\left(\frac{1}{C_{\min}} - \frac{1}{C_{\max}}\right)$. By the Perron-Frobenius theorem, its eigenvalues lie in the interval $(-1, 1]$ and $\lambda_M(\mathbf{A}) \leq 1$. Thus, $\lambda_m(\mathbf{Q}) \geq 0$ and $\lambda_M(\mathbf{Q}) \leq 4$ and we have the following bound on the score-stability of the $\texttt{LL}-\texttt{Reg}$ algorithm: $\beta_{\texttt{LL}-\texttt{Reg}} \leq \sqrt{2}M + 4\sqrt{m}M\left(\frac{1}{C_{\min}} - \frac{1}{C_{\max}}\right) \leq \sqrt{2}M + \frac{4\sqrt{m}M}{C_{\min}}$.

Gaussian Mean Fields algorithm GMF (Zhu et al., 2003) is very similar to the $\texttt{LL}-\texttt{Reg}$, and admits exactly the same stability coefficient.

Thus, the stability coefficients of the algorithms of $\texttt{CM}, \texttt{LL}-\texttt{Reg}$, and GMF can be large. Without additional constraints on the matrix $\mathbf{Q}$, these algorithms do not seem to be stable enough for the generalization bound of Theorem 3 to converge. A particular example of constraint is the condition $\sum_{i=1}^{m+u} h(x_i) = 0$ used by Belkin et al.'s algorithm (2004a). In the next section, we give a generalization bound for this algorithm and then describe a general method for making the algorithms just examined stable.

5.2. Stability of Constrained Regularization Algorithms

This subsection analyzes constrained regularization algorithms such as the Laplacian-based graph regularization algorithm of Belkin et al. (2004a). Given a weighted graph $G = (X, E)$ in which edge weights represent the extent of similarity between vertices, the task consists of predicting the vertex labels. The hypothesis h returned by the algorithm is solution of the following optimization problem:

$$\begin{aligned} &\min_{h\in H} \ \mathbf{h}^T\mathbf{L}\mathbf{h} + \frac{C}{m}\sum_{i=1}^{m}(h(x_i) - y_i)^2 \\ &\text{subject to: } \sum_{i=1}^{m+u} h(x_i) = 0, \end{aligned} \tag{11}$$

where $\mathbf{L} \in \mathbb{R}^{(m+u)\times(m+u)}$ is a smoothness matrix, e.g., the graph Laplacian, $\{y_i \mid i \in [1, m]\}$ are the target values of the m labeled nodes.

The hypothesis set H in this case can be thought of as a hyperplane in $\mathbb{R}^{m+u}$ that is orthogonal to the vector $\mathbf{1} \in \mathbb{R}^{m+u}$. Maintaining the notation used in (Belkin et al., 2004a), we let P_H denote the operator corresponding to the orthogonal projection on H. For a sample S drawn without replacement from X, define $\mathbf{I}_S \in \mathbb{R}^{(m+u)\times(m+u)}$ to be the diagonal matrix with $[\mathbf{I}_S]_{i,i} = 1$ if $x_i \in S$ and 0 otherwise. Similarly, let $\mathbf{y}_S \in \mathbb{R}^{(m+u)\times 1}$ be the column vector with $[\mathbf{y}_S]_{i,1} = y_i$ if $x_i \in S$ and 0 otherwise. The closed-form solution on a training sample S is given by (Belkin et al., 2004a):

$$\mathbf{h}_S = \left(P_H\left(\frac{m}{C}\mathbf{L} + \mathbf{I}_S\right)\right)^{-1}\mathbf{y}_S. \tag{12}$$

Theorem 5. *Assume that the vertex labels of the graph $G = (X, E)$ and the hypothesis h obtained by optimizing Eqn. 11 are both bounded ($\forall x, |h(x)| \leq M$ and $|y(x)| \leq M$ for some $M > 0$). Let $A = 1 + \kappa\sqrt{C}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$,*

$$R(h) \leq \widehat{R}(h) + \beta + \left(2\beta + \frac{(AM)^2(m+u)}{mu}\right)\sqrt{\frac{\alpha(m,u)\ln\frac{1}{\delta}}{2}},$$

with $\alpha(m,u) = \frac{mu}{m+u-1/2} \cdot \frac{1}{1-1/(2\max\{m,u\})}$ and $\beta \leq (4\sqrt{2}M^2)/(m\lambda_2/C - 1) + (4\sqrt{2m}M^2)/(m\lambda_2/C - 1)^2$, λ_2 is the second smallest eigenvalue of the Laplacian.

Proof. The proof is similar to that of (Belkin et al., 2004a) but uses our general transductive regression bound instead. □

The generalization bound we just presented differs in several respects from that of Belkin et al. (2004a). Our bound explicitly depends on both m and u while theirs shows only a dependency on m. Also, our bound does not depend on the number of times a point is sampled in the training set (parameter t), thanks to our analysis based on sampling without replacement.

Contrasting the stability coefficient of Belkin's algorithm with the stability coefficient of LTR (Theorem 4), we note that it does not depend on C' and β_{loc}. This is because unlabeled points do not enter the objective function, and thus $C' = 0$ and $\widetilde{y}(x) = 0$ for all

$x \in X$. However, the stability does depend on the second smallest eigenvalue λ_2 and the bound diverges as λ_2 approaches $\frac{C}{m}$. In all our regression experiments, we observed that this algorithm does not perform as well in comparison with LTR.

5.3. Making Seemingly Unstable Algorithms Stable

In Sec. 5.2, we saw that imposing additional constraints on the hypothesis, e.g., $\mathbf{h} \cdot \mathbf{1} = 0$, allowed one to derive non-trivial stability bounds. This idea can be generalized and similar non-trivial stability bounds can be derived for "stable" versions of the algorithms presented in Sec. 5.1 $\mathtt{CM}, \mathtt{LL-Reg}$, and GMF. Recall that the stability bound in Prop. 1 is inversely proportional to the smallest eigenvalue $\lambda_m(\mathbf{Q})$. The main difficulty with using the proposition for these algorithms is that $\lambda_m(\mathbf{Q}) = 0$ in each case. Let $\mathbf{v}_m$ denote the eigenvector corresponding to $\lambda_m(\mathbf{Q})$ and let λ_2 be the second smallest eigenvalue of $\mathbf{Q}$. One can modify (8) and constrain the solution to be orthogonal to $\mathbf{v}_m$ by imposing $\mathbf{h} \cdot \mathbf{v}_m = 0$. In the case of (Belkin et al., 2004a), $\mathbf{v}_m = \mathbf{1}$. This modification, motivated by the algorithm of (Belkin et al., 2004a), is equivalent to increasing the smallest eigenvalue to be λ_2.

As an example, by imposing the additional constraint, we can show that the stability coefficient of CM becomes bounded by $O(C/\lambda_2)$, instead of $\Theta(1)$. Thus, if $C = O(1/m)$ and $\lambda_2 = \Omega(1)$, it is bounded by $O(1/m)$ and the generalization bound converges as $O(1/m)$.

6. Experiments

6.1. Model Selection Based on Bound

This section reports the results of experiments using our stability-based generalization bound for model selection for the LTR algorithm. A crucial parameter of this algorithm is the stability coefficient $\beta_{loc}(r)$ of the local algorithm, which computes pseudo-targets $\widetilde{y}_x$ based on a ball of radius r around each point. We derive an expression for $\beta_{loc}(r)$ and show, using extensive experiments with multiple data sets, that the value r^* minimizing the bound is a remarkably good estimate of the best r for the test error. This demonstrates the benefit of our generalization bound for model selection, avoiding the need for a held-out validation set.

The experiments were carried out on several publicly available regression data sets: *Boston Housing*, *Elevators* and *Ailerons*[2]. For each of these data sets, we used $m = u$, inspired by the observation that, all other parameters being fixed, the bound of Theorem 3 is tightest when $m = u$. The value of the input variables were normalized to have mean zero and variance one. For the Boston Housing data set, the total number of examples was 506. For the Elevators and the Ailerons data set, a random subset of 2000 examples was used. For both of these data sets, other random subsets of 2000 samples led to similar results. The Boston Housing experiments were repeated for 50 random partitions, while for the Elevators and the Ailerons data set, the experiments were repeated for 20 random partitions each. Since the target values for the Elevators and the Ailerons data set were extremely small, they were scaled by a factor 1000 and 100 respectively in a pre-processing step.

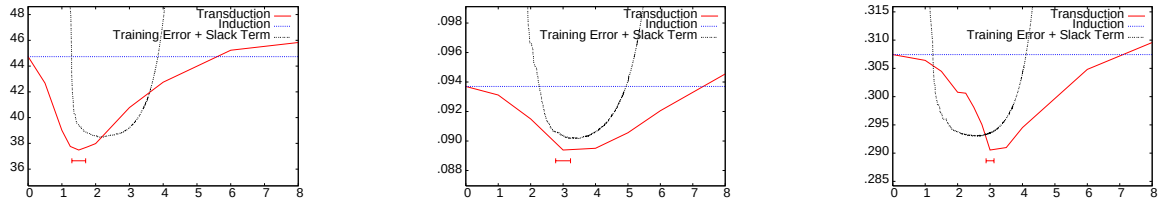
In our experiments, we estimated the pseudo-target of a point $x' \in T$ as a weighted average of the labeled points $x \in N(x')$ in a neighborhood of x'. Thus, $\widetilde{y}_{x'} = \sum_{x \in N(x')} \alpha_x y_x / \sum_{x \in N(x')} \alpha_x$. Weights are defined in terms of a similarity measure $K(x, x')$ captured by a kernel K: $\alpha_x = K(x, x')$. Let $m(r)$ be the number of labeled points in $N(x')$. Then, it is easy to show that $\beta_{loc} \leq 4\alpha_{\max} M/(\alpha_{\min} m(r))$, where $\alpha_{\max} = \max_{x \in N(x')} \alpha_x$ and $\alpha_{\min} = \min_{x \in N(x')} \alpha_x$. Thus, for a Gaussian kernel with parameter σ, $\beta_{loc} \leq 4M/(m(r)e^{-2r^2/\sigma^2})$. To estimate β_{loc}, one needs an estimate of $m(r)$, the number of samples in a ball of radius r from an unlabeled point x'. In our experiments, we estimated $m(r)$ as the number of samples in a ball of radius r from the origin. Since all features are normalized to mean zero and variance one, the origin is also the centroid of the set X.

We implemented a dual solution of LTR and used Gaussian kernels, for which, the parameter σ was selected using cross-validation on the training set. Experiments were repeated across 36 different pairs of values of (C, C'). For each pair, we varied the radius r of the neighborhood used to determine estimates from zero to the radius of the ball containing all points.

Figure 1(a) shows the mean values of the test MSE of our experiments on the Boston Housing data set for typical values of C and C'. Figures 1(b)-(c) show similar results for the Ailerons and Elevators data sets. For the sake of comparison, we also report results for induction. The relative standard deviations on the MSE are not indicated, but were typically of the order of 10%. LTR generally achieves a significant improvement over induction.

The generalization bound we derived in Eqn. 3 consists of the training error and a complexity term that depends on the parameters of the LTR algorithm $(C, C', M, m, u, \kappa, \beta_{loc}, \delta)$. Only two terms depend

[2] www.liaad.up.pt/~ltorgo/Regression/DataSets.html.



(a) (b) (c)

Figure 1. MSE against the radius r of LTR for three data sets: (a) Boston Housing. (b) Ailerons. (c) Elevators. The small horizontal bar indicates the location (mean ± one standard deviation) of the minimum of the empirically determined r.

upon the choice of the radius r: $\widehat{R}(h)$ and β_{loc}. Thus, keeping all other parameters fixed, the theoretically optimal radius r^* is the one that minimizes the training error plus the slack term. The figures also include plots of the training error combined with the complexity term, appropriately scaled. The empirical minimization of the radius r coincides with or is close to r^*. The optimal r based on test MSE is indicated with error bars.

6.2. Stable Versions of Unstable Algorithms

We refer to the stable version of the CM algorithm presented in Sec. 5.1 as CM − STABLE. We compared CM and CM − STABLE empirically on the same datasets, again using $m = u$. For the normalized Laplacian we used k-nearest neighbors graphs based on Euclidean distance. The parameters k and C were chosen by five-fold cross-validation over the training set. The experiment was repeated 20 times with random partitions. The averaged mean-squared errors with standard deviations, are reported in Table 6.2.

Dataset	CM	CM − STABLE
Elevators	0.3228 ± 0.0264	0.3293 ± 0.0286
Ailerons	0.1149 ± 0.0081	0.1184 ± 0.0087
Housing	57.93 ± 6.5	57.92 ± 6.5

We conclude from this experiment that CM and CM − STABLE have the same performance. However, as we showed previously, CM − STABLE has a non-trivial risk bound and thus comes with some guarantee.

7. Conclusion

We presented a comprehensive analysis of the stability of transductive regression algorithms with novel generalization bounds for a number of algorithms. Since they are algorithm-dependent, our bounds are often tighter than those based on complexity measures such as the VC-dimension. Our experiments also show the effectiveness of our bounds for model selection and the good performance of LTR algorithms.

References

Belkin, M., Matveeva, I., & Niyogi, P. (2004a). Regularization and semi-supervised learning on large graphs. *COLT* (pp. 624–638).

Belkin, M., Niyogi, P., & Sindhwani, V. (2004b). *Manifold regularization* (Technical Report TR-2004-06). University of Chicago.

Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *JMLR*, *2*, 499–526.

Chapelle, O., Vapnik, V., & Weston, J. (1999). Transductive Inference for Estimating Values of Functions. *NIPS 12* (pp. 421–427).

Cortes, C., & Mohri, M. (2007). On Transductive Regression. *NIPS 19* (pp. 305–312).

El-Yaniv, R., & Pechyony, D. (2006). Stable transductive learning. *COLT* (pp. 35–49).

McDiarmid, C. (1989). On the method of bounded differences. *Surveys in Combinatorics* (pp. 148–188). Cambridge University Press, Cambridge.

Schuurmans, D., & Southey, F. (2002). Metric-Based Methods for Adaptive Model Selection and Regularization. *Machine Learning*, *48*, 51–84.

Vapnik, V. N. (1982). *Estimation of dependences based on empirical data*. Berlin: Springer.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley-Interscience.

Wu, M., & Schölkopf, B. (2007). Transductive classification via local learning regularization. *AISTATS* (pp. 628–635).

Zhou, D., Bousquet, O., Lal, T., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. *NIPS 16* (pp. 595–602).

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *ICML* (pp. 912–919).

A Rate-Distortion One-Class Model and its Applications to Clustering

Koby Crammer CRAMMER@CIS.UPENN.EDU
Partha Pratim Talukdar PARTHA@CIS.UPENN.EDU
Department of Computer & Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA

Fernando Pereira[1] PEREIRA@GOOGLE.COM
Google, Inc. 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

Abstract

In one-class classification we seek a rule to find a coherent subset of instances similar to a few positive examples in a large pool of instances. The problem can be formulated and analyzed naturally in a rate-distortion framework, leading to an efficient algorithm that compares well with two previous one-class methods. The model can be also be extended to remove background clutter in clustering to improve cluster purity.

1. Introduction

Often we are given a large set of data items among which we would like to find a coherent subset. For instance, in document retrieval we might want to retrieve a small set of relevant documents similar to a few seed documents. In genomics, it is useful to find the set of genes that are strongly co-expressed with a few genes of interest. In both cases, we prefer high-precision answers over high-recall ones.

A popular intuition for this *one-class classification* problem is that of finding a small ball (under some appropriate norm) that contains as many of the seed elements as possible (Tax & Duin, 1999). Most previous approaches to the problem take the point of view of outlier and novelty detection, in which most of the examples are identified as relevant. However, Crammer and Chechik (2004) seek a small subset of relevant examples, rather than keep all but few outliers.

Most approaches to one-class classification use convex cost functions that focus on the large-scale distribution of the data. Those functions grow linearly outside class and and are constant inside it (Schölkopf et al., 1995; Tax & Duin, 1999; Ben-Hur et al., 2001). In a related study, Schölkopf et al. (2001) seek to separate most of the examples from the origin using a single hyperplane. More recently, Crammer and Singer (2003) generalized that approach to the general case of Bregman divergences. In all of those methods, the convexity of the cost function forces the the solution to shrink to the center of mass as the radius of the ball goes to zero, thus ignoring any local substructure.

In contrast to the previous work, Crammer and Chechik (2004) assumed that the distribution of points outside the one class is not relevant, so they chose a cost function that grows linearly inside the class but is constant outside it. This cost function is thus indifferent to the values of the irrelevant instances. A flat cost outside the class is expected to be better than a growing cost when the relevant instances are mostly in a small region, or when there are relatively few relevant instances. Unfortunately, their cost function leads to a non-convex optimization problem that requires an approximate solution.

Using ideas from rate-distortion theory (Cover & Thomas, 1991), we express the one-class problem as a lossy coding of each instance into a few possible instance-dependent codewords. Unlike previous methods that use just two (Crammer & Chechik, 2004) or a small number (Bekkerman & McCallum, 2005) of possible codewords for all instances, the total number of codewords in our method is greater than the number of instances. To preclude trivial codings, we force each instance to associate only with a few possible codewords. Finding the best coding function is an optimization problem for which we provide an efficient algorithm. The optimization has an "inverse temperature" parameter that represents the tradeoff between compression and distortion. As temperature

[1]Work done mainly at the University of Pennsylvania.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

decreases, the solution passes through a series of phase transitions associated with different sizes for the one class. This model outperforms two previous algorithms proposed for the problem, which are effective only in more restricted situations.

Our one-class model is also effective on the task of clustering a set of instances into multiple classes when some of the instances are clutter that should not be included in any cluster. This task can be reduced to an alternation between applications of the one-class algorithm and hard clustering. Initial experiments with synthetic and real world data show that by leaving some instances out of the clusters, the quality of the clustering improves.

2. One-Class as Rate-Distortion Optimization

We are given a set of instances indexed by the integer random variable $1 \leq X \leq m$. Each instance is described by a *point* $\mathbf{v}_x \in \mathbb{R}^d$ (possibly restricted to the simplex), and $p(x) = p(X = x)$ is a prior distribution over instances. Our goal is to find a *small coherent* subset of instances from a large set of possible instances. In particular, the learning task is to find a *centroid* $\mathbf{w}$ in the space such that there are many seed instances $\mathbf{v}_x$ close to it.

We formalize the task as a source coding problem. An instance x is either coded with the one class, with distortion $\mathcal{D}(\mathbf{v}_x \| \mathbf{w})$, and assigned the code 0, or it is coded as itself with zero distortion. The distortion $\mathcal{D}$ can be any Bregman divergence (Censor & Zenios, 1997), which includes as special cases the Euclidean distance and the KL divergence between distributions.

The random variable T represents the code for an instance: if $T = 0$, the instance was coded with the one class, while if $T = x > 0$, the instance is coded as itself. Although T has $m + 1$ distinct values, only one code x is associated with the event $T = x > 0$. The coding process is summarized by the conditional probability $q(t|x)$ of encoding x as t. These constraints mean that $q(t|x) = 0$ if $t \notin \{x, 0\}$, that is, the only nonzero probability outcomes for x are $T = 0$ or $T = x$.

The marginal

$$q(0) = \sum_x p(x)q(0|x) \ , \tag{1}$$

is the probability of assigning any instance to the one class. The other marginals are a product of two terms $q(x) = p(x)q(x|x)$, because of the constraints that $q(t|x) = 0$ for $t \neq 0$ and $t \neq x$. We explicitly allow soft assignments, $0 \leq q(0|x) \leq 1$. As we will see below, instances in the one class have a hard assignment to the class, but instances outside have a soft assignment.

We use the information bottleneck (IB) framework (Tishby et al., 1999) to formalize the assignment process. IB is an information-theoretic approach to regularized unsupervised learning that aims to extract a meaningful representation of some data X based on its association with side information. For generality, we choose here the rate-distortion formulation of the IB, which solves for the assignment by optimizing the tradeoff between two quantities: the amount of compression applied to the source data X, measured by the mutual information $I(T; X)$, and the average distortion between the data and its representation:

$$\min_{\mathbf{w}, \{q(0|x)\}} I(T; X) + \beta D(\mathbf{w}, \{q(0|x)\}) \ . \tag{2}$$

For one-class learning, the distortion term measures how well on average the centroid $\mathbf{w}$ serves as a proxy to each of the instances $\mathbf{v}_x$:

$$D(\mathbf{w}, \{q(0|x)\}) = \sum_x p(x)q(0|x)\mathcal{D}(\mathbf{v}_x \| \mathbf{w}) \ .$$

In contrast with standard rate distortion and IB formulations, the average distortion is computed only for $T = 0$, because the distortion is zero for $T > 0$.

We first rewrite the mutual information term using the constraints $q(t|x) = 0$ if $t \neq x$ and $t \neq 0$:

$$\begin{aligned} I(T; X) &= \textstyle\sum_{x,t} p(x)q(t|x) \log\left(\frac{q(t|x)}{q(t)}\right) \\ &= \textstyle\sum_x p(x) \left[q(0|x) \log\left(\frac{q(0|x)}{q(0)}\right) + q(x|x) \log\left(\frac{q(x|x)}{q(x)}\right)\right] \\ &= \textstyle\sum_x p(x) \left[\begin{array}{l} q(0|x) \log\left(\frac{q(0|x)}{q(0)}\right) + \\ (1 - q(0|x)) \log\left(\frac{q(x|x)}{q(x|x)p(x)}\right) \end{array}\right] . \end{aligned}$$

Then, the minimization (2) can be written as:

$$\begin{aligned} \min_{\{q(t|x), \mathbf{w}\}} \quad & \sum_{x=1}^{m} p(x) \left[q(0|x) \log\left(\frac{q(0|x)}{q(0)}\right)\right. \\ & \left. + (1 - q(0|x)) \log\left(\frac{1}{p(x)}\right)\right] \\ & + \beta \sum_x p(x)q(0|x)\mathcal{D}(\mathbf{v}_x \| \mathbf{w}) \qquad (3) \\ \text{s.t.} \quad & 0 \leq q(0|x) \leq 1, 1 \leq x \leq m \qquad (4) \end{aligned}$$

The corresponding Lagrangian is:

$$\textstyle\sum_{x=1}^{m} p(x) \left[q(0|x) \log\left(\frac{q(0|x)}{q(0)}\right) + (1 - q(0|x)) \log\left(\frac{1}{p(x)}\right)\right] + \beta \sum_x p(x)q(0|x)\mathcal{D}(\mathbf{v}_x \| \mathbf{w}) + \sum_x p(x)\nu_x q(0|x) \ .$$

Setting to zero its derivative with respect to $q(0|x)$, we get:

$$p(x)\left[\log\left(\frac{q(0|x)}{q(0)}\right)+\beta p(x)\mathcal{D}\left(\mathbf{v}_x\|\mathbf{w}\right)+\log p(x)+\nu_x\right]=0\ ,$$

Using the KKT conditions, we solve for $q(0|x)$:

$$q(0|x)=\min\left\{q(0)\frac{e^{-\beta\mathcal{D}(\mathbf{v}_x\|\mathbf{w})}}{p(x)},1\right\}\ . \qquad (5)$$

Setting the derivative of the Lagrangian with respect to $\mathbf{w}$ to zero we get:

$$\mathbf{w}=\frac{\sum_x p(x)q(0|x)\mathbf{v}_x}{\sum_x p(x)q(0|x)}=\sum_x q(x|0)\mathbf{v}_x\ . \qquad (6)$$

That is, the centroid is the average of all the points $\mathbf{v}_x$ weighted by their probability of membership in the single class.Like in the IB, the solution has a set of self-consistent equations: (1), (5), and (6).

Note that this rate-distortion formulation can be expressed as a tradeoff between two information quantities, as in the original IB. When $\mathcal{D}$ is the KL divergence, the optimization in (2) (or (3)) is equivalent to minimizing the tradeoff $I(X;T)-\beta I(T;Y)$ under the above constraints, where the random variable Y gives side information through the vectors $\mathbf{v}_x$.

3. Algorithm

The sequential algorithm of Slonim (2003) finds efficiently a local maximum of the IB objective. The algorithm alternates between selecting an instance and deciding whether moving it to another cluster would improve the objective. The item is reassigned to the cluster which yields the best improvement. A similar algorithm has been proposed for one-class problems (Crammer & Chechik, 2004). At each round, an instance is either removed from the class or added to the class, depending on what would most improve the objective.

We present a different algorithm for our model, inspired by Blahut-Arimoto algorithm and the original IB algorithm. The new algorithm iterates between the self-consistent equations (1), (5), and (6). Analogously to those algorithms, ours alternates between fixing $q(0|x)$ and $q(0)$ and fixing $\mathbf{w}$, and solving for the other parameters. We solve easily for $\mathbf{w}$ by computing the weighted average in (6). To solve for $q(0|x)$ and $q(0)$, let $\mathbf{w}$ be fixed and define $d_x=\mathcal{D}\left(\mathbf{v}_x\|\mathbf{w}\right)$. We now show how to compute $q(0|x)$ and $q(0)$ efficiently.

Eq. (5) cannot be solved directly for $q(0|x)$ because it involves $q(0)$, which in turn depends on $q(0|x)$. However, we can break this cycle by analyzing more carefully the properties of the solution. Let $\mathcal{C}=\{x:q(0|x)=1\}$. From (1) we get:

$$q(0)=\sum_x p(x)q(0|x)=\sum_{x\in\mathcal{C}}p(x)+q(0)\sum_{x\notin\mathcal{C}}e^{-\beta d_x} \qquad (7)$$

Assume that $\mathcal{C}\neq\emptyset$. Solving for $q(0)$, we obtain: $q(0)=\left(\sum_{x\in\mathcal{C}}p(x)\right)/\left(1-\sum_{x\notin\mathcal{C}}e^{-\beta d_x}\right)$. This equation is well defined if $0\leq q(0)\leq 1$, or equivalently:

$$\sum_{x\notin\mathcal{C}}e^{-\beta d_x}\leq 1-\sum_{x\in\mathcal{C}}p(x)\ . \qquad (8)$$

If $\mathcal{C}$ contains all the points, this is trivially satisfied. If $\mathcal{C}=\emptyset$, (7) becomes $q(0)\left(1-\sum_x e^{-\beta d_x}\right)=0$. Therefore, there is a unique β_0 such that for all $\beta\geq\beta_0$ we have $q(0)=0$. If $p(x)>0$ for all x we then have $q(0|x)=0$.

In summary, the solution of the optimization problem is given by the set $\mathcal{C}=\{x:q(0|x)=1\}$. We cannot search for that set naïvely, but fortunately the following lemma gives an efficient way to find the set by sorting its possible members.

Lemma 1 *Let $x_1,\ldots,x_m$ be a permutation of $[1,m]$ such that $0<\beta d_{x_1}+\log p(x_1)\leq\cdots\leq\beta d_{x_m}+\log p(x_m)$. Then $\mathcal{C}=\{x_i:1\leq i\leq k\}$ for some $k\in[0,m]$.*

Proof: Assume that $\mathcal{C}\neq\emptyset$. From (5) we know that $q(0|x)=\min\left\{q(0)e^{-\beta d_x-\log(p(x))},1\right\}$. We now show that if $x_k\in\mathcal{C}$ for some k, then $x_j\in\mathcal{C}$ for all $1\leq j<k$. If $x_k\in\mathcal{C}$, by definition $q(0)e^{-\beta d_{x_k}-\log p(x_k)}\geq 1$. For $j<k$, by hypothesis we have $-\beta d_{x_k}-\log p(x_k)\geq-\beta d_{x_j}-\log p(x_j)$. Thus, $q(0)e^{-\beta d_{x_j}-\log p(x_j)}\geq 1$, and thus $x_j\in\mathcal{C}$. ■

The lemma allows us to solve (3) easily for a fixed $\mathbf{w}$. The inputs for the algorithm are the prior over items $p(x)$, the distortions d_x, and the tradeoff parameter β. First, we order the items x in ascending order of the combined distortion and log-prior $\beta d_x+\log(p(x))$. As in the lemma, we obtain an ordering $x_1,\ldots,x_m$. Among the possible $\mathcal{C}=\{x_i:1\leq i\leq k\}$ that satisfy (8), we choose the one that minimizes the objective. A naïve implementation would require $\mathcal{O}(m\log m)$ time to sort the items, and then additional $\mathcal{O}(m)$ steps for each of the m candidate subsets $\mathcal{C}$, yielding an overall complexity of $\mathcal{O}(m^2)$. However, we can use dynamic programming to compute the objective for $\mathcal{C}\cup\{x_k\}$ from quantities saved from computing the objective for $\mathcal{C}$. Equation (3) can be rearranged as:

Input

- Distortion values d_x for $x \in \{1 \dots m\}$
- Prior $p(x)$ for $x \in \{1 \dots m\}$
- Tradeoff parameter $\beta \geq 0$

Sort the words in accordance to their score

$$\beta d_{x_1} + \log(p(x_1)) \leq \cdots \leq \beta d_{x_m} + \log(p(x_m))$$

Initialize $k = m$, $a_k = 1$, $p_k = 1$, $\mathcal{J}_k = \beta \sum_x p(x) d_x$, $c_{\text{best}} = k$, $\mathcal{J}_{\text{best}} = \mathcal{J}_k$.

Loop: While $k > 0$

1. Compute $a_{k-1} = a_k - e^{-\beta d_{x_{k-1}}}$
2. Compute $p_{k-1} = p_k - p(x_{k-1})$
3. if $p_{k-1} \leq a_{k-1}$
   - Compute $\mathcal{J}_{k-1}$ using (10).
   - If $\mathcal{J}_{k-1} < \mathcal{J}_{\text{best}}$ then set $k_{\text{best}} = k-1$ and $\mathcal{J}_{\text{best}} = \mathcal{J}_{k-1}$.
4. Set $k \leftarrow k - 1$.

Compute: $q(0|x)$ using (5) and (7)

Output: $q(0|x)$

Figure 1. Finding the one class for fixed distortion.

$$H\left[p(x)\right] + \sum_x p(x) \left[q(0|x) \log \left(\frac{p(x) q(0|x)}{q(0)} \right) \right] + \beta \sum_x p(x) q(0|x) d_x$$

The sum can be split according to whether $x \in \mathcal{C}$ and rearranged again:

$$\begin{array}{l} H\left[p(x)\right] + \sum_{x \in \mathcal{C}} p(x) \left[\log\left(p(x)\right) + \beta d_x\right] \\ - \left(\sum_{x \in \mathcal{C}} p(x)\right) \log \left(\sum_{x \in \mathcal{C}} p(x)\right) \\ + \left(\sum_{x \in \mathcal{C}} p(x)\right) \log \left(1 - \sum_{x \notin \mathcal{C}} e^{-\beta d_x}\right) \ . \end{array} \quad (9)$$

Let $\mathcal{C}_k = \{x_i : 1 \leq i \leq k\}$, $\mathcal{J}_k$ the value of the objective on $\mathcal{C}_k$, $p_k = \sum_{j=1}^{k} p(x_j) = \sum_{x \in \mathcal{C}} p(x)$, and by $a_c = 1 - \sum_{j=k+1}^{m} e^{-\beta d_{x_j}} = 1 - \sum_{x \notin \mathcal{C}} e^{-\beta d_x}$. These quantities can be computed recursively as follows. Let $\mathcal{J}_m = \beta \sum_x p(x) d_x$, $p_m = 1$ and $a_m = 1$. Given $\mathcal{J}_k$, p_k and a_k, we can compute the following in unit time: $a_{k-1} = a_k - e^{-\beta d_{x_k}}$ and $p_{k-1} = p_k - p(x_k)$. Finally, by examining (9) we get,

$$\begin{array}{rcl} \mathcal{J}_{k-1} & = & \mathcal{J}_k - p(x_{k-1})[\beta d_{x_{k-1}} + \log p(x_{k-1})] \\ & & + [p_k \log(p_k) - p_k \log(a_k)] \\ & & - [p_{k-1} \log(p_{k-1}) - p_{k-1} \log(a_{k-1})] \end{array} \quad (10)$$

Fig. 1 gives an outline of the algorithm.

The following properties of the solution are worth noting. When the temperature $t = 1/\beta$ is high, all the instances belong to the single cluster with probability 1. As the temperature drops, instances are pulled out of the class, one after the other, as $q(0|x)$ becomes strictly less than 1. Finally, at a critical temperature t_0, all the instances are pulled out of the class, that is, $q(0|x) = 0$ for all x. We show this process in Fig. 2. There are five points, indexed 1 through 5. The distortion of each point is proportional to its index. The y axis is temperature $t = 1/\beta$. For high values of t, all the instances belong to the class with probability 1. At $t \approx 3.1$ the model goes through its first phase transition, as the instance with the highest distortion is pulled out of the class, and its probability of belonging there drops exponentially. There are three more similar phase transition for instances 4, 3 and 2 respectively. Then, at $t \approx 1.5$ the model goes through a discontinuous phase transition and $q(0|x)$ falls to zero for all instances.

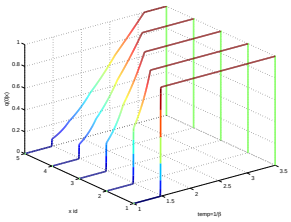


Figure 2. Illustration of the serious of phase transitions of $q(0|x)$ as the temperature $1/\beta$ is modified.

In summary, the algorithm iterates between two steps: compute $\mathbf{w}$ given $q(0|x)$ and $q(0)$ using (6) (expectation), and use the algorithm in Fig. 1 to find $q(0|x)$ and $q(0)$ from $d_x = \mathcal{D}\left(\mathbf{v}_x \| \mathbf{w}\right)$ (maximization). In this aspect the algorithm is similar to EM, and thus we might suspect that it has a maximum-likelihood analog (Slonim & Weiss, 2002).

4. Multiclass Clustering

It seems natural to generalize from one class to multiple classes by replacing the one class centroid with $k > 1$ centroids. There are $k + 1$ outcomes for each instance: either code it using one of the k centroids, or code it with itself. We now formalize this extension.

We might at first think that we could just generalize $q(t|x)$ from the previous model to range over a set of $k+m$ values — m points and k centroids — where for given x the value of $q(t|x)$ is non-zero for at most $k+1$ values of t, the k clusters and self-coding. However, this direct approach leads to a derivation that can not be decomposed nicely as in the one class case, because

$1-q(x|x)$ is not informative about individual clusters, just about their sum.

Instead, we break the coding scheme into two stages. First, given an instance x, we determine whether to code it using one of the centroids or by itself $t = x$. Then, for non-self-coded instances, we decide their cluster. Formally, given x we decide if we want to code it by itself (with probability $q(x|x)$) or by some centroid (with probability $q(0|x)$). Next, if we decide to code the instance with one of the centroids, we denote the centroid's identity by S, and denote the probability of encoding using centroid S given the point identity x and the decision to code it 0 by $r(s|0,x) = \Pr[S = s|0,x]$ We also define the marginals, $q(s,0) = \sum_x p(x)q(0|x)r(s|0,x)$.

As in (2), we write a rate-distortion objective. The rate equals to the mutual information between possible codings and input variables. The distortion-rate optimization is

$$\min_{\{q(\cdot|x)\},r(s|0,x),\{\mathbf{w}_s\}} \beta \sum_{x,s} p(x)q(0|x)r(s|0,x)\mathcal{D}\left(\mathbf{v}_x\|\mathbf{w}_s\right) + \sum_x p(x) \sum_s q(0|x)r(s|0,x) \log\left(\frac{q(0|x)r(s|0,x)}{q(s,0)}\right) + \sum_x p(x)q(x|x) \log\left(\frac{q(x|x)}{q(x)}\right) , \quad (11)$$

subject to normalization $q(0|x) + q(x|x) = 1$ and $\sum_s r(s|0,x) = 1$. The marginals are defined naturally, $q(x) = p(x)q(x|x)$ and $q(0) = \sum_x p(x)q(0|x)$. Before solving the optimization problem we further assume that every point x is associated with exactly one centroid (if any), that is $r(s|0,x) = 1$ or $r(s|0,x) = 0$. From normalization there is only a single cluster s for which $r(s|0,x) = 1$, denoted by $s(x)$. We also denote by $E(s) = \{x : s(x) = s\}$. We do so for two reasons, first, without doing so we could not separate $q(0|x)$ and $q(s,0)$ from each other (for different values of s) and could not get a solution similar to (5). Second, we show below that we can solve this problem by alternating between two algorithms, one of them is the sequential-IB (sIB) (Slonim, 2003) designed for hard clustering. We call this algorithm MCRD (multiclass rate-distortion-based algorithm).

We now solve the optimization analogously to the derivation starting at (3). After writing the Lagrangian we use its derivations to compute self consistent equations. (details omitted for lack of space). First, we have

$$q(0|x) = \min\left\{\frac{q(s(x),0)}{p(x)} e^{-d_{s(x),x}}, 1\right\} , \quad (12)$$

which is the equivalent of (5). Note that the values of $q(0|x)$ are tied for $x \in E(s)$. Thus, there are k sets of equations, each set tying all points in $E(s)$ and the exact value of $q(0|x)$ for $x \in E(s)$ and $q(s,0)$ can be solved separately using the algorithm of Fig. 1.

Next, we can compute the derivative of the Lagrangian with respect to the other variables and obtain self consistent equations

$$\mathbf{w}_s = \frac{\sum_x p(x)q(0|x)r(s|0,x)\mathbf{v}_x}{\sum_x p(x)q(0|x)r(s|0,x)} = \frac{\sum_{x\in E(s)} p(x)q(0|x)\mathbf{v}_x}{\sum_{x\in E(s)} p(x)q(0|x)} ,$$

and $r(s|0,x) \propto q(s,0)e^{-\beta\mathcal{D}(\mathbf{v}_x\|\mathbf{w}_s)}$. The last equation can not be used to solve the problem since we assume that $r(s|0,x)$ is an integer. In practice, we use the following lemma which relates the optimization problem of (11) and the optimization problem of the IB method (Tishby et al., 1999).

Lemma 2 *The following two optimization problems are equivalent up to a linear transformation:*

1. *The optimization problem of* (11) *over* $\mathbf{w}_s$ *and* $r(s|0,x)$*, where we fix* $q(0|x)$ *and* $q(s,0)$*, and* $r(s|0,x) \in \{0,1\}$.
2. *The rate-distortion formulation of the IB method (Slonim, 2003), where the assignment probabilities are either* 0 *or* 1*, and a reweighted prior proportional to* $p(x)q(0|x)$.

(Proof omitted due to lack of space.) Using the lemma and the discussion preceding it, we have an algorithm for MCRD that alternates between two steps: (1) Use the sIB algorithm to set the values of $\mathbf{w}_s$ and $r(s|0,x)$, given $q(0|x)$ and $q(s,0)$, with prior proportional to $p(x)q(0|x)$. (2) Use k calls to the algorithm on Fig. 1 to find $q(0|x)$ and $q(s,0)$ from $d_{s(x),x} = \mathcal{D}\left(\mathbf{v}_x\|\mathbf{w}_s\right)$.

5. Experiments

We compare our algorithm (OCRD-BA) with two previously proposed methods: the IB-related one-class algorithm of Crammer and Chechik (2004) (OC-IB), and a well-known convex optimization method (Tax & Duin, 1999; Schölkopf et al., 2001; Crammer & Singer, 2003) (OC-Convex). We obtained Crammer and Chechik's data and followed their evaluation protocol to achieve comparable results. For lack of space, we just discuss document retrieval experiments, although we obtained qualitatively comparable results on gene expression data as well.

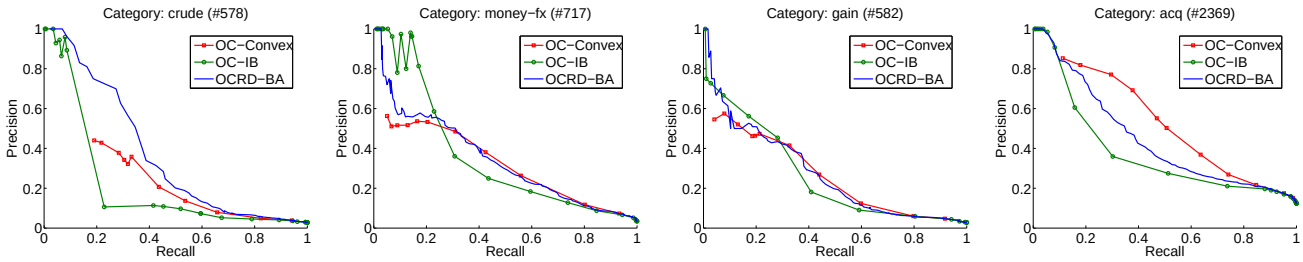


Figure 3. Precision-Recall plots for four (out of five) categories of Reuters-21678 dataset using OC-IB, OC-Convex, and OCRD-BA (this paper).

5.1. Document Retrieval

This is a document retrieval task that was previously described in detail (Crammer & Chechik, 2004, Sec. 6.2). The task uses a subset of the five most frequent categories of Reuters-21578. For each category, half of its documents were used for training, and the remaining half, together with the remaining documents from other categories, were used for evaluation. During training, each of the algorithms searched for a meaningful subset of the training data and generated a centroid. The centroid was used then to label the test data, and to compute recall and precision.

All algorithms used the KL divergence to compare empirical word distributions for different documents. For OC-IB and OC-Convex, we used the parameter values in the previous study (Crammer & Chechik, 2004). For our algorithm, OCRD-BA, we set the prior $p(x)$ to be uniform over the training set, and used a range of values of β that yielded a range of class sizes. We used a single random document to initialize the centroid maintained by OCRD-BA, as was done for OC-IB. We trained five models for each value of β, each using a different random example for initialization, and picked the one which attained the best value of the objective.

After picking a model, we fixed the induced centroid $\mathbf{w}$ and computed the distortion $\mathcal{D}(\mathbf{v}_x \| \mathbf{w})$ for all the test examples $\mathbf{v}_x$. We then ran the first half of our algorithm (Fig. 1) to compute the cluster assignments $q(0|x)$. Finally, a test point $\mathbf{v}_x$ was assigned to the class if $q(0|x) = 1$. We used the actual Reuters labels to plot precision and recall values for different β values.

The results are summarized in Fig. 3, where there is one plot per category (except the *earn* category where all algorithms perform the same). As in the previous study (Crammer & Chechik, 2004), we observe that OC-IB achieves better precision than OC-Convex on low recall values. The previous study argues that OC-Convex converges to the center-of-mass of the data for low values of recall while OC-IB exploits local structure and thus performs better. As recall increases, OC-Convex improves and OC-IB degrades, until OC-Convex performs better than OC-IB for high values of recall.

Our method, OCRD-BA, strikes a balance between the two previous methods: at low values of recall, OCRD-BA is comparable in performance to OC-IB and at higher values of recall OCRD-BA is comparable to OC-Convex. Furthermore, in the *crude* category, our method outperformed both algorithms. This suggests that OCRD-BA is similar to OC-IB for small classes and to OC-Convex for large classes. We discuss this issue later.

5.2. Clustering

We evaluated the MCRD algorithm using a synthetic dataset and a real dataset. The synthetic dataset (Synth4G) has 900 points in the plane. Of those, 400 were generated from 4 Gaussian distributions with $\sigma = 0.1$, 100 points from each Gaussian. The remaining 500 point were generated from a uniform distribution. We ran the algorithm allowing up to five clusters with various values of β. The output of the algorithm for four values of β is plotted in Fig. 4. The title of each plots summarize the value of β used, number of points associated with a cluster, and (in parenthesis) the size of each cluster. For low values of β the algorithm prefers to reduce the rate (over distortion) and effectively group all points into a single cluster. As β increases the algorithm uses more clusters until all 5 possible clusters are used (left panel). As β is increased the algorithm prefers to remove points from the clusters, but still use 5 centroids (second panel), until at some point the algorithm only four clusters are used (third panel). Then, for higher values of β five clusters are used again (right panel). This may be due to the fact, that for large β, the actual length scale is small, and thus, practically, there are more than five

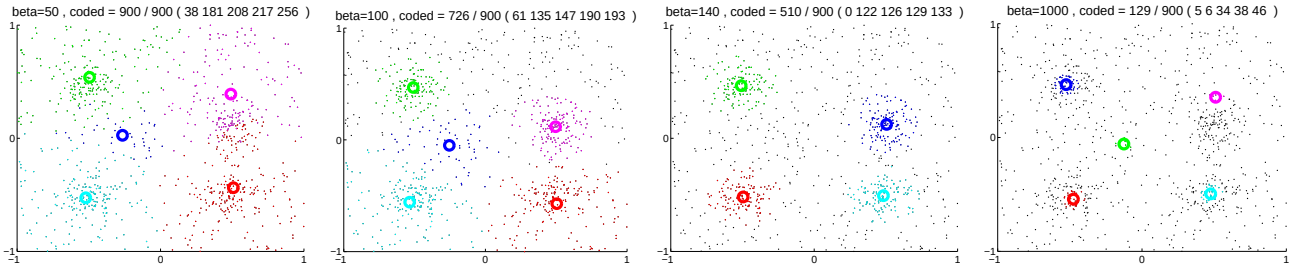


Figure 4. Clusterings produced by MCRD with ($k = 5$) on the synthetic data set for four values of β. Self-coded points are marked by black dots, coded points by colored dots and cluster centroids by bold circles.

clusters (more than five small, dense regions).

We also evaluated on Multi5_1, a real-world high dimensional multiclass dataset which has been used by Slonim et al. (2002) to evaluate the sIB clustering algorithm. The dataset has 500 documents from 5 categories, each represented as a distribution over 2,000 words. We compare the MCRD algorithm ($\beta = 1.6$) with sIB, which by default, uses all the points in clustering thereby achieving 100% recall. We follow Slonim et al., (2002, Sec. 7.4) to get precision at various recall values for sIB, and for other experimental details. The precision at various recall values is summarized in Fig. 5. We observe that MCRD consistently outperforms sIB at all recall levels. Specifically, MCRD achieves very high precision at low recall values, which is one of the objectives of current work. These experimental results further support our hypothesis that better clustering of the data can be obtained if the algorithm is allowed to selectively leave out data points which are unlikely to help the clustering task.

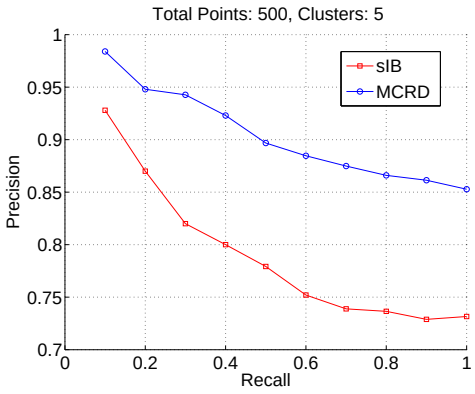


Figure 5. Precision vs. Recall for sIB and MCRD algorithms ($\beta = 1.6$) on the Multi5_1 dataset. Results are obtained by averaging over 5 random permutations of the data. For each permutation, 5 restarts were used and the model with the best objective was selected.

6. Related Work

Crammer and Chechik (2004) proposed to use the information bottleneck for one-class problems. They compressed the points using two possible events: a point can be either belong to the single class or not. In the former case, the distortion is proportional to the distance between the point and the centroid. In the later case, the distortion equals fixed predefined value R, which intuitively sets the diameter of the class. This formulation suffers from some drawbacks. First, it uses two interacting parameters, the R parameter just discussed, and an inverse temperature β to set the hardness of the solution. In practice, they set β to yield only hard solutions. Second, their distortion does not make sense in term of compression, as the compressor effectively can either approximate a point (using the single class) or ignore it.

Instead, we use $m+1$ values for the compression variable T, but regularization forces the compressor to generate sparse solutions. In contrast to a fixed nonzero distortion used for out-of-class points, we use a zero distortion because the out-of-class points are not encoded. As a result, our method uses a single inverse temperature parameter to set the size of the class. A point can either be in the class alone (hard assignment) or both in the class and outside (soft assignment).

The centroid is defined as a weighted average of all of the data. Points which belong to the cluster have the same weight, while other points are weighted proportionally to the exponent of their distance from the centroid. This behavior combines properties of two previous approaches. As in Crammer and Chechik's work, the points belonging to the class give an equal contribution in the location of the centroid. But like in discriminative one-class methods (Crammer & Singer, 2003) points outside the class still affect its centroid. Thus our method uses information from all

the data, unlike discriminative methods that only see outliers (Tax & Duin, 1999).

The fact that the centroid in our model is contributed to by both typical points and outliers may explain the results of the experiments. The OC-IB method (Crammer & Chechik, 2004) works in a low-recall condition, that is, with a small class. In this condition, points outside the cluster will have a negligible effect on the centroid, yielding the OC-IB solution. For large values of β, points outside the class have a stronger effect on the centroid's location, similarly to discriminative methods (Crammer & Singer, 2003). Furthermore, when using the KL divergence, points that are not contributing to the centroids at all (removed from data), would typically have a divergence of infinity to the centroids. Our methods allows to reduce the effect of outliers (by giving them exponential small weight), but still allow them to contribute to the centroids (positive weight).

Gupta and Ghosh (2006) present an extension of the Crammer and Chechik algorith that clusters points while allowing some of them to be ignored. Recently, Lashkari and Golland (2008) proposed an exemplar-based algorithm , in which any point can serve as a centroid (similarly to k-medians). They show that under some choices, some points can be coded by themselves. Our method is different in that it allows centroids that do not coincide with any data point (similar to k-means).

7. Conclusions

Building on the rate-distortion formulation of the information bottleneck method, we cast the problem of identifying a small coherent subset of data as an optimization problem that trades off class size (compression) for accuracy (distortion). We analyzed a rate-distortion view of the model and demonstrated that it goes through a sequence of phase transitions that correspond to different class sizes. We demonstrated that our method combines the best of two previous methods, each of which is good in a narrower range of class sizes. We also showed that our method allows us to move from one-class to standard clustering, but with background noise left out. The proposed approach for one-class learning can be extended to the idea of regularizing by using constraints over a large set of decisions which can be used for other more complex associations among random variables, and in particular for bi-clustering.

Acknowledgments This material is based in part upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185, and by NSF grant IIS-0513778.

References

Bekkerman, R., & McCallum, A. (2005). Disambiguating web appearances of people in a social network. *WWW*. Chiba, Japan.

Ben-Hur, A., Horn, D., Siegelmann, H., & Vapnik, V. (2001). Support vector clustering. *JMLR*, *2*, 125–137.

Censor, Y., & Zenios, S. (1997). *Parallel optimization: Theory, algorithms, and applications*. Oxford Univ. Press, NY, USA.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. Wiley.

Crammer, K., & Chechik, G. (2004). A needle in a haystack: Local one-class optimization. *ICML 23*.

Crammer, K., & Singer, Y. (2003). Learning algorithms for enclosing points in bregmanian spheres. *COLT 16*.

Gupta, G., & Ghosh, J. (2006). Bregman bubble clustering: A robust, scalable framework for locating multiple, dense regions in data. *ICDM*.

Lashkari, D., & Golland, P. (2008). Convex clustering with exemplar-based models. *NIPS*.

Schölkopf, B., Burges, C., & Vapnik, V. (1995). Extracting support data for a given task. *KDD 1*.

Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*, 1443–1472.

Slonim, N. (2003). *The information bottleneck: Theory and applications*. Doctoral dissertation, Hebrew University.

Slonim, N., Friedman, N., & Tishby, N. (2002). Unsupervised document classification using sequential information maximization. *SIGIR*.

Slonim, N., & Weiss, Y. (2002). Maximum likelihood and the information bottleneck. *NIPS*.

Tax, D., & Duin, R. (1999). Data domain description using support vectors. *ESANN* (pp. 251–256).

Tishby, N., Pereira, F., & Bialek, W. (1999). The information bottleneck method. *37th Allerton Conference on Communication, Control, and Computing*. Allerton House, Illinois.

Fast Gaussian Process Methods for Point Process Intensity Estimation

John P. Cunningham JCUNNIN@STANFORD.EDU
Department of Electrical Engineering, Stanford University, Stanford, CA, USA 94305

Krishna V. Shenoy SHENOY@STANFORD.EDU
Department of Electrical Engineering and Neurosciences Program, Stanford University, Stanford, CA, USA 94305

Maneesh Sahani MANEESH@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, UCL London, WC1N 3AR, UK

Abstract

Point processes are difficult to analyze because they provide only a sparse and noisy observation of the intensity function driving the process. Gaussian Processes offer an attractive framework within which to infer underlying intensity functions. The result of this inference is a continuous function defined across time that is typically more amenable to analytical efforts. However, a naive implementation will become computationally infeasible in any problem of reasonable size, both in memory and run time requirements. We demonstrate problem specific methods for a class of renewal processes that eliminate the memory burden and reduce the solve time by orders of magnitude.

1. Introduction

Point processes with temporally or spatially varying intensity functions arise naturally in many fields of study. When the intensity function is itself a random process (often a Gaussian Process), the process is called a doubly-stochastic or Cox point process. Application domains including economics and finance (*e.g.* Basu & Dassios, 2002), neuroscience (*e.g.* Cunningham et al., 2008), ecology (*e.g.* Moller et al., 1998), and others.

Given observed point process data, one can use a Gaussian Process (GP) framework to infer an optimal estimate of the underlying intensity. In this paper we consider GP prior intensity functions coupled with point process observation models. The problem of intensity estimation then becomes a modification of GP regression and inherits the computational complexity inherent in GP methods (*e.g.* Rasmussen & Williams, 2006). The data size n will grow with the length (*e.g.* total time) of the point process. Naive methods will be $\mathcal{O}(n^2)$ in memory requirements (storing Hessian matrices) and $\mathcal{O}(n^3)$ in run time (matrix inversions and determinants). At one thousand data points (such as one second of millisecond-resolution data), a naive solution to this problem is already quite burdensome on a common workstation. At ten thousand or more, this problem is for all practical purposes intractable.

While applications of doubly-stochastic point processes are numerous, there is little work proposing solutions to the serious computational issues inherent in these methods. Thus, the development of efficient methods for intensity estimation would be of broad appeal. In this paper, we do not address the appropriateness of doubly-stochastic point process models for particular applications, but rather we focus on the significant steps required to make such modelling computationally tractable. We build on previous work from both GP regression and large-scale optimization to create a considerably faster and less memory intensive algorithm for doubly-stochastic point-process intensity estimation.

As part of the GP intensity estimation problem we optimize model hyperparameters using a Laplace approximation to the marginal likelihood or evidence. This requires an iterative approach which divides into two major parts. First, at each iteration we must find a modal (MAP) estimate of the intensity function. Second, we must calculate the approximate model evidence and its gradients with respect to GP hyperparameters. Both aspects of this problem present computational and memory problems. We develop methods to reduce the costs of both drastically.

We show that for certain classes of renewal process observation models, MAP estimation may be framed as a tractable convex program. To ensure nonnegativity in the intensity function we use a log barrier Newton method

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

(Boyd & Vandenberghe, 2004), which we solve efficiently by deriving decompositions of matrices with known structure. By exploiting a recursion embedded in the algorithm, we avoid many costly matrix inversions. We combine these advances with large scale optimization techniques, such as conjugate gradients (CG, as used by Gibbs & MacKay, 1997) and fast fourier transform (FFT) matrix multiplication methods.

To evaluate the model evidence, as well as its gradients with respect to hyperparameters, we again exploit the structure imposed by the renewal process framework to find an exact but considerably less burdensome representation. We then show that a further approximation loses little in accuracy, but makes the cost of this computation insignificant.

Combining these advances, we are able to reduce a problem that is effectively computationally infeasible to a problem with minimal memory load and very fast solution time. $\mathcal{O}(n^2)$ memory requirements are eliminated, and $\mathcal{O}(n^3)$ computation is reduced to modestly superlinear.

2. Problem Overview

Define $\mathbf{x} \in \mathbb{R}^n$ to be the intensity function (the high dimensional signal of interest); $\mathbf{x}$ is indexed by input[1] time points $\mathbf{t} \in \mathbb{R}^n$. Let the observed data $\mathbf{y} = \{y_0, \ldots, y_N\} \in \mathbb{R}^{N+1}$ be a set of $N+1$ time indices into the vector $\mathbf{x}$; that is, the ith point event occurs at time y_i, and the intensity at that time is x_{y_i}. Denote all hyperparameters by θ. In general, the prior and observation models are both functions of θ. The GP framework implies a normal prior on the intensity $p(\mathbf{x} \mid \theta) = \mathcal{N}(\mu\mathbf{1}, \Sigma)$, where the nonzero mean is a sensible choice because the intensity function is constrained to be nonnegative. Thus we treat μ as a hyperparameter ($\mu \in \theta$). The positive definite covariance matrix Σ (also a function of θ) is defined by an appropriate kernel such as a squared exponential or Ornstein-Uhlenbeck kernel (see Rasmussen & Williams, 2006, for a discussion of GP kernels). The point-process observation model gives the likelihood $p(\mathbf{y} \mid \mathbf{x}, \theta)$. In this work, we consider renewal processes (*i.e.* one-dimensional point processes with independent event interarrival times), a family of point processes that has both been well-studied theoretically and applied in many domains (Daley & Vere-Jones, 2002).

The GP prior is log concave in $\mathbf{x}$, and the nonnegativity constraint on intensity ($\mathbf{x} \succeq 0$) is convex (constraining $\mathbf{x}$ to be nonnegative is equivalent to solving an unconstrained problem where the prior on the vector $\mathbf{x}$ is a truncated multivariate normal distribution, but this is not the same as truncating the GP prior in the continuous, infinite dimensional function space; see Horrace, 2005). Thus, if the observation model is also log concave in $\mathbf{x}$, the MAP estimate $\mathbf{x}^*$ is unique and can be readily found using a log barrier Newton method (Boyd & Vandenberghe, 2004; Paninski, 2004). Renewal processes are simply defined by their interarrival distribution $f_z(z)$. A common construction for a renewal process with an inhomogeneous underlying intensity is to use the intensity rescaling $m(t_i \mid t_{i-1}) = \int_{t_{i-1}}^{t_i} x(u)du$ (in practice, a discretized sum of $\mathbf{x}$) (Barbieri et al., 2001; Daley & Vere-Jones, 2002). Accordingly, the density for an observation of event times $\mathbf{y}$ can be defined

$$\begin{aligned} p(\mathbf{y}) &= \prod_{i=1}^{N} p(y_i \mid y_{i-1}) \\ &= \prod_{i=1}^{N} |m'(y_i \mid y_{i-1})|\, f_z(m(y_i \mid y_{i-1})) \end{aligned} \quad (1)$$

by a change of variables for the interarrival distribution (Papoulis & Pillai, 2002). Since $m(t)$ is a linear transformation of the intensity function (our variables of interest), the observation model obeys log concavity as long as the distribution primitive $f_z(z)$ is log concave. Examples of suitable renewal processes include the inhomogeneous Poisson, gamma interval, Weibull interval, inverse Gaussian (Wald) interval, Rayleigh interval, and other processes (Papoulis & Pillai, 2002). For this paper, we choose one of these distributions and focus on its details. However, for processes of the form above, the implementation details are identical up to the forms of the actual distributions.

To solve the GP intensity estimation, we first find a MAP estimate $\mathbf{x}^*$ given fixed hyperparameters θ, and then we approximate the model evidence $p(\mathbf{y} \mid \theta)$ (for which we need $\mathbf{x}^*$) and its gradients in θ. Iterating these two steps, we can find the optimal model $\hat{\theta}$ (we do not integrate over hyperparameters). Finally, MAP estimation under these optimal hyperparameters $\hat{\theta}$ gives an optimal estimate of the underlying intensity. This iterative solution for $\hat{\theta}$ can be written:

$$\begin{aligned} \hat{\theta} &= \operatorname*{argmax}_{\theta} p(\theta) p(\mathbf{y} \mid \theta) \\ &\approx \operatorname*{argmax}_{\theta} p(\theta) p(\mathbf{y} \mid \mathbf{x}^*, \theta) p(\mathbf{x}^* \mid \theta) \frac{(2\pi)^{\frac{n}{2}}}{|\Lambda^* + \Sigma^{-1}|^{\frac{1}{2}}}, \end{aligned} \quad (2)$$

where the last term is a Laplace approximation to the intractable form of $p(\mathbf{y} \mid \theta)$, $\mathbf{x}^*$ is the mode of $p(\mathbf{y} \mid \mathbf{x}) p(\mathbf{x})$ (MAP estimate), and $\Lambda^* = -\nabla^2_{\mathbf{x}} \log p(\mathbf{y} \mid \mathbf{x}, \theta) \mid_{\mathbf{x}=\mathbf{x}^*}$. The log concavity of our problem in $\mathbf{x}$ supports the choice of a Laplace approximation. Each of the two major steps in this algorithm (MAP estimation and model selection) involves computational and memory challenges. We address these challenges in Sections 4 and 5.

[1] In this work we restrict ourselves to a single input dimension (which we call time), as it aligns with the family of renewal processes in one-dimension. Some ideas here can be extended to multiple dimensions (*e.g.* if using a spatial Poisson process).

The computational problems inherent in GP methods have been well studied, and much progress has been made in sparsification (*e.g.* Quinonero-Candela & Rasmussen, 2005). Unfortunately, these methods do not apply directly to point process estimation, as there are no distinct training and test sets. The reader might wonder if a coarser grid would be adequate, thereby obviating the detailed methods developed here. We have found in experiments (not shown) that the sacrifice in accuracy required to allow reasonable computational tractability is large, and thus we do not consider the coarse grid a viable option. One could also consider re-expressing the problem in terms of the integrals $m(y_i \mid y_{i-1})$ appearing in Eq. 1. While this is possible in certain cases, it requires additional approximation. Finally, we note that the Laplace approximation is often inferior to Expectation Propagation (EP) (Kuss & Rasmussen, 2005) for GP methods. While many of the same techniques used here could also be used with EP, EP requires additional approximations and computational overhead. We find in experiments (not shown) that EP yields similar accuracy to the Laplace approximation in this domain, but EP incurs increased complexity and computational load.

3. Model Construction

To demonstrate our fast method, we choose the specific observation model of an inhomogeneous gamma interval process (Barbieri et al., 2001) (with hyperparameter $\gamma \in \theta$, $\gamma \geq 1$). If time has been discretized with precision Δ, this can be written

$$p(\mathbf{y} \mid \mathbf{x}, \theta) = \prod_{i=1}^{N} \left[\frac{\gamma x_{y_i}}{\Gamma(\gamma)} \left(\gamma \sum_{k=y_{i-1}}^{y_i - 1} x_k \Delta \right)^{\gamma - 1} \cdot \exp\left\{ -\gamma \sum_{k=y_{i-1}}^{y_i - 1} x_k \Delta \right\} \right], \quad (3)$$

(where we have ignored terms that scale with Δ). Let $f(\mathbf{x}) = -\log p(\mathbf{y} \mid \mathbf{x}, \theta) p(\mathbf{x} \mid \theta)$. Our MAP estimation problem is to minimize $f(\mathbf{x})$ subject to the constraint $\mathbf{x} \succeq 0$ (nonnegativity). In the log barrier method, we consider the above problem as a sequence of convex problems where we seek to minimize, at increasing values of τ, the (unconstrained) objective function

$$f_\tau(\mathbf{x}) = f(\mathbf{x}) - \sum_{k=1}^{n} \left(\frac{1}{\tau} \right) \log(x_k) \quad (4)$$

which has Hessian (positive definite by our log concave construction):

$$H = \nabla^2_{\mathbf{x}} f_\tau(\mathbf{x}) = \Sigma^{-1} + \Lambda, \quad \text{where} \quad \Lambda = B + D, \quad (5)$$

with $D = \mathbf{diag}(x_{y_0}^{-2}, \ldots, 0, \ldots, x_{y_i}^{-2}, \ldots, 0 \ldots, x_{y_N}^{-2}) + (\frac{1}{\tau})\mathbf{diag}(x_1^{-2}, \ldots, x_n^{-2})$ being positive definite and diagonal. B is block diagonal with N blocks $\hat{B}_i$:

$$\hat{B}_i = b_i b_i^T \quad \text{where} \quad b_i = \sqrt{(\gamma - 1)} \left(\sum_{k=y_{i-1}}^{y_i - 1} x_k \right)^{-1} \mathbf{1}. \quad (6)$$

B is thus block rank 1 (with the positive eigenvalue in each block corresponding to the eigenvector b_i). This matrix is key, as we exploit its structure to achieve improvements in computational performance.

4. MAP Estimation Problem

As outlined in Section 2, we first find the MAP estimate $\mathbf{x}^*$ for any model defined by hyperparameters θ. The log barrier method has the intensive requirements of calculating the objective Eq. 4, its gradient $\mathbf{g}$ (in $\mathbf{x}$), and the Newton step $\mathbf{x}_{nt} = -H^{-1}\mathbf{g}$. Each of these calculations is $\mathcal{O}(n^3)$ in run time and $\mathcal{O}(n^2)$ in memory. We show an approach that alleviates these burdens.

4.1. Finding the Newton Step $\mathbf{x}_{nt}$

First we consider the Hessian, $H = \Sigma^{-1} + \Lambda$, which itself contains the costly inverse Σ^{-1}. We would like to avoid this inversion of Σ entirely with the matrix inversion lemma (Sherman-Woodbury-Morrison formula):

$$\begin{aligned} -H^{-1} &= -(\Sigma^{-1} + \Lambda)^{-1} \\ &= -\Sigma + \Sigma R (I + R^T \Sigma R)^{-1} R^T \Sigma \end{aligned} \quad (7)$$

where R is any valid factorization such that $RR^T = \Lambda$. This decomposition preserves symmetry in the remaining matrix inverse (required for CG) and has advantageous numerical properties. With this form, instead of calculating $\mathbf{x}_{nt} = -H^{-1}\mathbf{g}$ directly, we need only multiply the rightmost expression in Eq. 7 with the gradient $\mathbf{g}$. Doing so requires the inversion $(I + R^T \Sigma R)^{-1}\mathbf{v}$ where $\mathbf{v} = R^T \Sigma \mathbf{g}$. CG allows us to avoid directly calculating matrix inverses and instead achieve the desired inversion by iteratively multiplying $(I + R^T \Sigma R)\mathbf{z}$ for different vectors $\mathbf{z}$ (Gibbs & MacKay, 1997).

It is common to precondition the CG method to reduce the number of iterations required for convergence. However, our experience with preconditioning (using both classic preconditioners and some of our own design) was that it actually degraded run-time performance. Preconditioners typically aim to improve the condition number of the Hessian, which indeed they do in this problem. However, the rapidity of CG convergence here is facilitated more by spectral concentration – many eigenvalues being equal or close to 1 – than by overall conditioning. Thus, we found it more effective to use CG inversion directly on $(I + R^T \Sigma R)$.

In general, however, finding the decomposition $\Lambda = RR^T$ is an $\mathcal{O}(n^3)$ operation, which would remove any computational benefit from this approach. For log concave renewal processes, we can derive a valid decomposition in closed form and linear computation time. Since Λ is block diagonal, we consider only one block without loss of generality. Calling this block $\hat{\Lambda}$, we know $\hat{\Lambda} = bb^T + \hat{D}$, where $\hat{D}$ is a diagonal block of the larger diagonal matrix D, and b is defined in Eq. 6. $\hat{D}$ is positive definite, so $T = \hat{D}^{-\frac{1}{2}}$ satisfies $T\hat{D}T = I$ (a similarity transform). Then, calling $\tilde{b} = Tb$, we have $T\hat{\Lambda}T = \tilde{b}\tilde{b}^T + I$. With this form, we see that the general structure of $T\hat{\Lambda}T$ is preserved under the desired matrix decomposition, up to scaling of the components:

$$(\alpha\tilde{b}\tilde{b}^T + I)(\alpha\tilde{b}\tilde{b}^T + I)^T = (\alpha^2\|\tilde{b}\|^2 + 2\alpha)\tilde{b}\tilde{b}^T + I \quad (8)$$

and we want to choose α such that (Eq. 8) equals $\tilde{b}\tilde{b}^T + I$. Using the quadratic formula to find this α, we see then that

$$\tilde{R} = \left(\frac{\sqrt{1 + \|\tilde{b}\|^2} - 1}{\|\tilde{b}\|^2}\right)\tilde{b}\tilde{b}^T + I \quad (9)$$

satisfies $T\hat{\Lambda}T = \tilde{R}\tilde{R}^T$. Since T is diagonal, it easily inverts to $T^{-1} = \hat{D}^{\frac{1}{2}}$. Then:

$$\hat{\Lambda} = T^{-1}\tilde{R}\tilde{R}^T T^{-1} = (T^{-1}\tilde{R})(T^{-1}\tilde{R})^T = \hat{R}\hat{R}^T. \quad (10)$$

To be explicit, we have found that

$$\hat{R} = \left(\frac{\sqrt{1 + \|\hat{D}^{-\frac{1}{2}}b\|^2} - 1}{\|\hat{D}^{-\frac{1}{2}}b\|^2}\right)bb^T\hat{D}^{-\frac{1}{2}} + \hat{D}^{\frac{1}{2}} \quad (11)$$

is a valid decomposition $\hat{R}\hat{R}^T = \hat{\Lambda}$. This decomposition can be seen as a partial rank-one (blockwise) update to a Cholesky factorization (Gill et al., 1974), in that $\hat{D}$ can trivially be factorized to $\hat{D}^{\frac{1}{2}}$. The final form is not, however, a Cholesky factorization, since $\hat{R}$ is not triangular (making a triangular factor would require additional computation and the explicit representation of the Cholesky matrix).

Since all of the products needed to construct $\hat{R}$ can be formed in $\mathcal{O}(m)$ time (where m is the size of the block), and since the larger matrix R can be formed by tiling the blocks $\hat{R}$, we have a total complexity for this decomposition of $\mathcal{O}(n)$. We can then use CG to find the solution to $(I + R^T\Sigma R)^{-1}(R^T\Sigma\mathbf{g})$. With this inversion calculated, we can perform the remaining forward multiplications in Eq. 7; this completes calculation of a Newton step.

In fact, we need not form the matrix R in memory. Instead, we retain each of its component elements (in Eq. 11), and reduce multiplication of a vector by R to a sequence of inner products and multiplications by diagonal matrices, all of which can be stored and calculated in $\mathcal{O}(n)$ time. Thus, we eliminate the need for $\mathcal{O}(n^2)$ storage, and we perform the relevant matrix multiplications in $\mathcal{O}(n)$ time. Since R can be multiplied in linear time, the complexity of multiplying vectors by $(I + R^T\Sigma R)$ depends on multiplying vectors by the covariance matrix Σ.

Since we have evenly spaced resolution of our data $\mathbf{x}$ in time indices t_i, Σ is Toeplitz. This matrix can be embedded in a larger circulant matrix, multiplication by which is simply a convolution operation of the argument vector with a row of this circulant matrix. Thus, the operation can be quickly done in $\mathcal{O}(n\log n)$ using frequency domain multiplications(Silverman, 1982). Further, we need never represent the matrix Σ; we only store the first row of the circulant matrix. Again we have eliminated $\mathcal{O}(n^2)$ memory needs. Other methods for fast kernel matrix multiplications include Fast Gauss Transforms (FGT) (Raykar et al., 2005) and kd-Trees (Shen et al., 2006; Gray & Moore, 2003). We note that the single input dimension (time) enables this Toeplitz structure, and thus an extension to multiple dimensions should use FGT or similar. The regular structure of the data points in any dimension make Σ multiplications very fast with such a method. Further, these methods avoid explicit representation of Σ. Here, the simple FFT approach for this one-dimensional problem significantly outperforms other (more general) methods in both speed and accuracy.

Finally, we note that the matrix $(I + R^T\Sigma R)$ is particularly well suited to CG. Although $R^T\Sigma R$ is full rank by definition, in practice its spectrum has very few large eigenvalues (typically fewer than N, the number of events). Loosely, the matrix looks like identity plus low rank. In practice, the CG method converges with high accuracy almost always in fewer than 50 steps (very often under 30). This is drastically fewer than the worst case of n steps (n of 10^3 to 10^4).

Instead of decomposing $\Lambda = RR^T$, one might have considered using the matrix inversion lemma to write $(\Sigma^{-1} + \Lambda)^{-1} = \Sigma - \Sigma\Lambda(\Lambda + \Lambda\Sigma\Lambda)^{-1}\Lambda\Sigma$. Indeed this valid form enables all of the CG and fast multiplication methods previously discussed. While it may seem that this form's ease of derivation (compared to the matrix decomposition in Eq. 11) warrants its use in general, the matrix to be inverted is poorly conditioned compared to $(I + R^T\Sigma R)$, and thus the inversion requires more CG steps. We have found in testing that the number of CG steps can roughly double. Thus, the decomposition of Eq. 11 is computationally worthwhile.

In this section, we have constructed a fast method for calculating the Newton step that costs $\mathcal{O}(n\log n)$ per CG step and incurs a very small number of CG steps. Also, we have avoided explicit representation of any matrix, so that memory requirements are only linear in the data size n, al-

lowing problem sizes of potentially millions of time steps. These two factors stand in contrast to the cubic run time and quadratic storage needs of a naive method.

4.2. Evaluating the Gradient and Objective

Calculating the objective $f_\tau(\mathbf{x})$ (Eq. 4) and its gradient (both required for the log barrier method) require finding $\Sigma^{-1}(\mathbf{x} - \mu\mathbf{1})$. Note that the kth iterate $\mathbf{x}^{(k)}$ (of the log barrier method) has the form

$$\begin{aligned}(\mathbf{x}^{(k)} - \mu\mathbf{1}) &= \mathbf{x}^{(k-1)} + t^{(k-1)}\mathbf{x}_{nt}^{(k-1)} - \mu\mathbf{1} \\ &= \sum_{j=1}^{k-1} t^{(j)}\mathbf{x}_{nt}^{(j)} + (\mathbf{x}^{(0)} - \mu\mathbf{1}) \quad (12)\end{aligned}$$

where $t^{(j)}$ and $\mathbf{x}_{nt}^{(j)}$ represent the jth iterates of the Newton step size t and the step $\mathbf{x}_{nt}$, and $\mathbf{x}^{(0)}$ is the algorithm initial point. The most logical starting point $\mathbf{x}^{(0)}$ is $\mu\mathbf{1}$, in which case the rightmost term in Eq. 12 drops out. Thus, letting $\mathbf{x}^{(0)} = \mu\mathbf{1}$ and using the form of $\mathbf{x}_{nt} = -H^{-1}\mathbf{g}$ with $-H^{-1}$ defined as in Eq. 7, we write:

$$\Sigma^{-1}(\mathbf{x}^{(k)} - \mu\mathbf{1}) = \sum_{j=1}^{k-1} t^{(j)}\Big(-\mathbf{g}^{(j)} + R^{(j)}(I + R^{(j)T}\Sigma R^{(j)})^{-1}R^{(j)T}\Sigma\mathbf{g}^{(j)}\Big). \quad (13)$$

In the earlier calculation of $\mathbf{x}_{nt}$ (Section 4.1), both of the right hand side arguments in Eq. 13 have already been found. As such, we have a recurrence that obviates the invertion of Σ, with no additional memory demands (Rasmussen & Williams, 2006).

The above steps reduce a naive MAP estimation (of any log concave renewal process) that requires cubic effort and quadratic storage to an algorithm that is modestly superlinear in run time and linear in memory requirements.

5. Model Selection Problem

Having now found $\mathbf{x}^*$ for any hyperparameters θ, the second major part of the problem is to find the negative logarithm of our approximation to the evidence $p(\mathbf{y} \mid \theta)$ in Eq. 2, and its gradients with respect to θ. The approximated log evidence can be written as:

$$\begin{aligned}&-\log\ p(\mathbf{y} \mid \theta) \approx -\log\ p(\mathbf{y} \mid \mathbf{x}^*) \\ &+\ \frac{1}{2}(\mathbf{x}^* - \mu\mathbf{1})^T\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})\ +\ \frac{1}{2}\log|I + \Sigma\Lambda^*| \quad (14)\end{aligned}$$

(ignoring constants). Each of these terms has an explicit and an implicit gradient with respect to θ, where the latter result from the dependence of the MAP estimate $\mathbf{x}^*$ on the hyperparameters (such implicit gradients are typical for the use of Laplace approximation in GP learning; see Rasmussen & Williams, 2006, section 5.5.1). The implicit gradients in this problem are extremely computationally burdensome to calculate (requiring the trace of matrix inversions and matrix-matrix products for each element of $\mathbf{x}$). In empirical tests, we find implicit gradients to be quite small relative to the explicit gradients (often by several orders of magnitude). Ignoring these gradients is undesirable but essential to make this problem computationally feasible. Thus we consider only explicit gradients. This is a common approach for GP methods; see Rasmussen and Williams (2006).

Efficient computation of the first two terms of Eq. 14, as well as their gradients with respect to θ, can be achieved by the fast multiplication method and the recursion derived in Sec. 4. Specifically, the values of the first and second terms of Eq. 14 are calculated during the MAP estimation, so no additional memory or computation is necessary for them. The gradient of the first term is nonzero only with respect to γ and is linear in $\mathbf{x}$ (no matrix multiplications are required). Thus it can be quickly calculated with no additional memory demands. Computation of the gradient of the second term (the prior) can exploit the fact that we calculated $\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})$ in the final step of the MAP estimation. The gradient of this term with respect to μ is a simple inner product $\mathbf{1}^T(\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1}))$ (since we have already calculated the right side of this inner product, this computation is $\mathcal{O}(n)$ in run time and requires no additional memory). The gradient of this term with respect to a kernel hyperparameter θ_i (*e.g.* a lengthscale or variance) is:

$$\frac{d}{d\theta_i}\Big[\frac{1}{2}(\mathbf{x}^* - \mu\mathbf{1})^T\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})\Big] = \frac{1}{2}\big(\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})\big)^T\Big(\frac{d\Sigma}{d\theta_i}\Big)\big(\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})\big). \quad (15)$$

Since we have $\Sigma^{-1}(\mathbf{x}^* - \mu\mathbf{1})$, this gradient only requires one matrix-vector multiplication. $\frac{d\Sigma}{d\theta_i}$ has the same Toeplitz structure as Σ and can thus be quickly multiplied. Thus, calculating the first two terms of Eq. 14 and their gradients adds no complexity to the method developed so far.

Only the term $\frac{1}{2}\log|I + \Sigma\Lambda^*|$ presents difficulty. Determinants in general require $\mathcal{O}(n^2)$ memory and $\mathcal{O}(n^3)$ solve time using a Cholesky or PLU factorization, so we must consider the problem more carefully. We examine the eigenstructure of $(I + \Sigma\Lambda^*)$. Since we are not trying to find a MAP estimate, there is no log barrier term (*i.e.* let $\tau \to \infty$); thus D (from Eq. 5) is rank N only. This means that $\Lambda^* = B + D$ (Eq. 6) is block outer product plus sub rank diagonal, so it is also rank deficient with block rank 2. Thus, it has $2N$ nonzero eigenvalues (two corresponding to each of the N events, one in each block from B and one in each block from D). Using the eigenvalue decomposition

$\Lambda^* = USU^T$, we see

$$\begin{aligned} \log|I + \Sigma\Lambda^*| &= \log|I + \Sigma USU^T| \\ &= \log|U^T||I + \Sigma USU^T||U| \\ &= \log|I + U^T\Sigma US|, \end{aligned} \tag{16}$$

since the orthogonal matrix U has determinant 1 and $U^TU = I$ by definition. Since Λ^* has rank $2N$, we know that S is diagonal with zeros on the last $n - 2N$ entries. By construction, the number of events N is much smaller than the total data size n. Since the determinant of a matrix is the product of its eigenvalues, the unit eigenvalue dimensions of $I + U^T\Sigma US$ can be ignored. We define $\overline{S}$ as the $2N \times 2N$ submatrix of S that is made up of the diagonal block with nonzero diagonal entries. Further define $\overline{U}$ as the corresponding $2N$ eigenvectors. Then, since the other dimensions of $U^T\Sigma US$ contribute nothing to the determinant, we have

$$\begin{aligned} \log|I + \Sigma\Lambda^*| &= \log|I + U^T\Sigma US| \\ &= \log|I + \overline{U}^T\Sigma\overline{U}\,\overline{S}| \\ &= \log|I + \overline{\Sigma}\,\overline{S}|, \end{aligned} \tag{17}$$

where I is now the $2N \times 2N$ identity, and we have further defined $\overline{\Sigma} = \overline{U}^T\Sigma\overline{U}$. Computationally, $\overline{\Sigma}$ is formed by multiplying Σ with the columns of $\overline{U}$. Since Λ^* is block rank 2, both matrices $\overline{S}$ and $\overline{U}$ can be found in closed form (N rank 2 eigendecompositions, one decomposition per block). This calculation of Eq. 17 requires $2N$ matrix multiplications which each have a run time cost of $\mathcal{O}(n\log n)$.

We can make a small approximation that simplifies this problem even further. Typically, N of these $2N$ eigenvalues are substantially larger than the other N. Each block of Λ^* contributes two nonzero eigenvalues. The larger is due to the diagonal entry $x_{y_i}^{-2}$ (from the matrix D) and is nearly axis aligned. The smaller eigenvalue is due to the outer product vector from block $\hat{B}_i$. Examination of the denominators in the definition of $\hat{B}_i$ and D in Eqs. 5 and 6 explains the difference in magnitude, since $x_{y_i}^2$ is much smaller than the square of sums denominator in $\hat{B}_i$. We approximate the eigenvector as the y_i axis and approximate its eigenvalue as the corresponding value in Λ^*. Then $\overline{S}$ is size $N \times N$. This savings is small, but importantly we can form $\overline{\Sigma} = \overline{U}^T\Sigma\overline{U}$ simply by picking out the N rows and columns of Σ corresponding to the event times y_i.

In this formulation, we are left with matrices of size $N \times N$ only, so we have some modest number of $\mathcal{O}(N^3)$ operations; this approach is considerably faster and scales better than the exact method above. We have also reduced $\mathcal{O}(n^2)$ storage to $\mathcal{O}(N^2)$. The following section elucidates the quality of this approximation.

To calculate the gradients with respect to this log determinant term, we also use the approximation of Eq.17. We call our approximate gradient of this term the gradient of the approximation in Eq. 17. This approximation can readily be differentiated with respect to the hyperparameters (again, typical for GP; see Rasmussen & Williams, 2006). Since these approximations are matrices in the event space N (not time space n), these gradients are quickly calculated with a handful of $\mathcal{O}(N^3)$ operations and with storage of $\mathcal{O}(N^2)$.

6. Results and Discussion

The methods developed here maintain computational accuracy while achieving massive speed-up and the elimination of memory burden. First, we have shown a fast method that achieves an accurate approximation of the MAP estimate $\mathbf{x}^*$ in much less time than a naive method. We have made all matrix multiplications implicit, thereby eliminating the memory burden of representing full matrices. We call this piece the "MAP Estimation." Second, we found the approximate model evidence, as well as its gradients, so as to perform model selection on the hyperparameters θ. These calculations, which involved the calculation of a log determinant and its gradients (Eq. 17), were achieved with matrices of significantly reduced dimension, again removing the storage demands of teh naive method. We call this piece the "log determinant approximation." These two pieces must be iterated (as described before Eq. 2) to find both the optimal model $\hat{\theta}$ and the optimal intensity $\mathbf{x}^*$. We call this iterative method (combining the two pieces above) the "full GP intensity estimation." We show here that each piece is fast and accurate, and finally that they combine to make an overall method that is considerably faster than a standard implementation, with minimal sacrifice to accuracy.

To demonstrate results, we pick six representative intensity functions, consisting of sinusoids of various amplitudes (5-100 events/second), means (15-150 events/second), frequencies (1-2 Hz), and lengths (0.5-10 seconds of millisecond resolution data, implying data sizes n of 500 to 10000). This set is by no means exhaustive, but it does indicate how this method outperforms a naive implementation in a range of scenarios. Our testing over many different intensity functions (including those in Cunningham et al., 2008) agrees with the results shown here. We simulate point process data $\mathbf{y}$ from these intensities, and we implement both the naive and the fast method on these process realizations.

All results are given for 2006era Linux (FC4) 64 bit workstations with 2-4GB of RAM running MATLAB (R14sp3, BLAS ATLAS 3.2.1 on AMD processors). The naive method was implemented in MATLAB. The fast method was similarly implemented in MATLAB with some use of the C-MEX interface for linear operations such as multiplication of a vector by the (implicitly represented) matrix R.

Table 1. Performance for fast and naive methods. Results averaged over 10 independent trials.

	Data Set					
	1	2	3	4	5	6
Data Size(n)	500	1000	1000	2000	4000	10000
Num. Events (N)[1]	20-30	30-40	140-160	55-70	55-70	140-160
MAP Estimation						
Fast Solve Time(s)	0.12	0.17	0.46	0.32	7.6	37.9
Naive Solve Time(s)	7.04	40.5	39.5	333	3704	1day[3]
Speed Up	**58**×	**232**×	**86**×	**1043**×	**493**×	**2000**×[3]
MS Error (Fast vs. Naive)[2]	4.3e-4	4.2e-4	2.1e-4	5.2e-6	6.1e-6	-
Avg. CG Iters.	6.4	5.5	16.2	8.1	29.9	49.7
Log Determinant Approximation						
Fast Solve Time(s)	6.5e-4	1.8e-3	1.9e-2	2.8e-3	2.8e-3	2.5e-2
Naive Solve Time(s)	0.24	1.02	0.97	5.7	34.7	540[3]
Speed Up	**375**×	**566**×	**52**×	**2058**×	**1.3e4**×	**2.2e4**×[3]
Avg. Acc. of Fast Approx.	99.1%	98.8%	99.8%	98.9%	99.7%	-
Avg. Model Selection Iters.	54.3	54.6	89.1	68.1	39.4	40.7
Full GP Intensity Estimation (Iterative Model Selection and MAP Estimation)						
Fast Solve Time(s)	4.4	7.1	30.3	18.7	128	423
Naive Solve Time(s)	443	3094	4548	2.4e4	1.5e5	1month[3]
Speed Up	**105**×	**451**×	**150**×	**1512**×	**1166**×	**1e4**×[3]
MS Error (Fast vs. Naive)[2]	0.10	0.03	10.8	0.01	0.01	-

[1] Entries show a range of data used.
[2] Squared norm of $x(t)$ is roughly 10^3 to 10^5, so these errors are insignificant.
[3] Unable to complete naive method; numbers estimated from cubic scaling.

First we demonstrate the utility of our fast MAP estimation method on problems of several different sizes and with different $\mathbf{x}$. We compare the fast MAP estimation to a naive implementation, demonstrating the average mean squared (MS) error (between the fast and naive estimates) and the average solve time. These results are found in the first part of Table 1. The squared norm of $\mathbf{x}$ is roughly 10^3 to 10^5, so the errors shown (the difference between the naive and fast methods) are vanishingly small. Thus, the fast MAP estimation gives an extremely accurate approximation of the naive MAP estimate. For all practical purposes, the fast MAP estimation method is exact.

The naive method scales in run time as the cube of data size n, as expected. The fast method and the speed-up factor do not appear to scale linearly in the data size. Indeed, run time depends heavily on the number of CG iterations required to solve the MAP estimation. This number of CG steps depends on problem size n, number of events N, and hyperparameters such as the lengthscale of the covariance matrix. Even so, major gains are achieved.

Second, we demonstrate our model selection accuracy and speed-up (the log determinant approximation). We run the full iterative fast method with both MAP estimation and evidence model selection. At each iterate of θ, we calculate evidence and its gradients using both the fast and naive methods. In the second section of Table 1, we show average solution times for calculating the log determinant in both naive and fast methods, and we compare their accuracy. For the sake of brevity, we demonstrate only the calculation of $\log |I + \Sigma\Lambda^*|$, not its gradients with respect to the hyperparameters. Those calculations show very similar speed-ups and are as well approximated. Thus, the log determinant is calculated to 99-100% accuracy with the naive method, and we have a highly accurate approximation.

Finally, the full intensity estimation problem requires iterative evidence calculations and MAP estimations, so we must also demonstrate the accuracy of the full fast method versus the full naive method. The last part of Table 1 shows this result (Full GP Intensity Estimation). We see that all data sets converge to quite similar results in both the fast and the naive methods, and the fast method enjoys significant speed-up. The MS errors shown compare the result

of the fast method to the result of the naive method and are very small compared to the squared norm of $\mathbf{x}$ (10^3 to 10^5).

We have demonstrated a method for inferring optimal intensity estimates from an observation of renewal process data, and we have exploited problem structure to make this method computationally attractive. As an extension, we also developed this fast GP technique for multiple observations $\mathbf{y}^{(i)}$ of the same underlying $\mathbf{x}$. It uses the same approach with comparable performance improvements. As such, we do not report it here.

Since we avoid all explicit representations of $n \times n$ matrices, our memory requirements are very minor for a problem of this size. The major run time improvements in Table 1 require effectively no loss of accuracy from an exact naive approach, and thus the additional technical complexity of this approach is well justified. Having fast, scalable methods for point process intensity estimation problems may mean the difference between theoretically interesting approaches and methods that become well used in practice.

Acknowledgments

This work was supported by NIH-NINDS-CRCNS-R01, the Michael Flynn SGF, NSF, Gatsby, CDRF, BWF, ONR, Sloan, and Whitaker. We thank Stephen Boyd for helpful discussions, Drew Haven for technical support, and Sandy Eisensee for administrative assistance.

References

Barbieri, R., Quirk, M., Frank, L., Wilson, M., & Brown, E. (2001). Construction and analysis of non-poisson stimulus-response models of neural spiking activity. *J Neurosci Methods*, *105*, 25–37.

Basu, S., & Dassios, A. (2002). A Cox process with log-normal intensity. *Insurance: Mathematics and Economics*, *31*, 297–302.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Cunningham, J. P., Yu, B. M., Shenoy, K. V., & Sahani, M. (2008). Inferring neural firing rates from spike trains using Gaussian processes. In *Advances in NIPS, 20*.

Daley, D., & Vere-Jones, D. (2002). *An introduction to the theory of point processes*. New York: Springer.

Gibbs, M., & MacKay, D. (1997). Efficient Implementation of Gaussian Processes. *Preprint*.

Gill, P., Golub, G., Murray, W., & Saunders, M. (1974). Methods for Modifying Matrix Factorizations. *Mathematics of Computation*, *28*, 505–535.

Gray, A., & Moore, A. (2003). Nonparametric density estimation: Toward computationsl tractability. *SIAM Int'l Conference on Data Mining.*.

Horrace, W. (2005). Some results on the multivariate truncated normal distribution. *J Multivariate Analysis*, *94*, 209–221.

Kuss, M., & Rasmussen, C. (2005). Assessing approximate inference for binary gaussian process classification. *Journal of Machine Learning Res.*, *6*, 1679–1704.

Moller, J., Syversveen, A., & Waagepetersen, R. (1998). Log Gaussian Cox processes. *Scandanavian J. of Stats.*, *25*, 451–482.

Paninski, L. (2004). Log-concavity results on Gaussian process methods for supervised and unsupervised learning. *Advances in NIPS*, *16*.

Papoulis, A., & Pillai, S. (2002). *Probability, random variables, and stochastic processes*. Boston: McGraw Hill.

Quinonero-Candela, J., & Rasmussen, C. (2005). A Unifying View of sparse approximate Gaussian process regression. *J. Machine Learning*, *6*, 1939–1959.

Rasmussen, C., & Williams, C. (2006). *Gaussian processes for machine learning*. Cambridge: MIT Press.

Raykar, V., Yang, C., Duraiswami, R., & Gumerov, N. (2005). Fast computation of sums of Gaussians in high dimensions. *University of Maryland Tech. Report CS-TR-4767/UMIACS-TR-2005-69*.

Shen, Y., Ng, A., & Seeger, M. (2006). Fast Gaussian Process Regression using KD-trees. *Advances in NIPS*, *18*.

Silverman, B. (1982). Kernel density estimation using the fast fourier transform. *Journal of Royal Stat. Soc. Series C: Applied Stat.*, *33*.

Self-taught Clustering

Wenyuan Dai[†] DWYAK@APEX.SJTU.EDU.CN
Qiang Yang[‡] QYANG@CSE.UST.HK
Gui-Rong Xue[†] GRXUE@APEX.SJTU.EDU.CN
Yong Yu[†] YYU@APEX.SJTU.EDU.CN
(†) Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China
(‡) Hong Kong University of Science and Technology, Clearwater Bay, Kowloon, Hong Kong

Abstract

This paper focuses on a new clustering task, called *self-taught clustering*. Self-taught clustering is an instance of *unsupervised transfer learning*, which aims at clustering a small collection of target unlabeled data with the help of a large amount of *auxiliary* unlabeled data. The target and auxiliary data can be different in topic distribution. We show that even when the target data are not sufficient to allow effective learning of a high quality feature representation, it is possible to learn the useful features with the help of the auxiliary data on which the target data can be clustered effectively. We propose a co-clustering based self-taught clustering algorithm to tackle this problem, by clustering the target and auxiliary data simultaneously to allow the feature representation from the auxiliary data to influence the target data through a common set of features. Under the new data representation, clustering on the target data can be improved. Our experiments on image clustering show that our algorithm can greatly outperform several state-of-the-art clustering methods when utilizing irrelevant unlabeled auxiliary data.

1. Introduction

Clustering (Jain & Dubes, 1988) aims at partitioning objects into groups, so that the objects in the same groups are relatively similar, while the objects in different groups are relatively dissimilar. Clustering has a long history in machine learning (MacQueen, 1967), and recent works on clustering research have focused on improving the clustering performance using the prior knowledge in semi-supervised clustering (Wagstaff et al., 2001) and supervised clustering (Finley & Joachims, 2005).

In the past, semi-supervised clustering incorporates pairwise supervision, such as *must-link* or *cannot-link* constraints (Wagstaff et al., 2001), to bias clustering results. Supervised clustering methods learn distance functions from a small sample of auxiliary *labeled* data (Finley & Joachims, 2005). Different from these clustering problems, in this paper, we address a new clustering task where we use a large amount of *auxiliary unlabeled* data to enhance the clustering performance of a small amount of *target* unlabeled data. In our problem, we do not have any labeled data or pairwise supervisory constraint knowledge. All we have are the auxiliary data which are totally unlabeled and may be irrelevant to the target data. Our target data consist of a collection of unlabeled data from which it may be insufficient to learn a good feature representation. Thus, applying clustering directly on these target data may give very poor performance. However, with the help of auxiliary data, we are able to uncover a good feature set to enable high quality clustering on the target data.

Our problem can be considered as an instance of transfer learning, which makes use of knowledge gained from one learning task to improve the performance of another, even when these learning tasks or domains follow different distributions (Caruana, 1997). However, since all the data are unlabeled, we can consider it as an instance of *unsupervised transfer learning* (Teh et al., 2006). This unsupervised transfer learning problem could also be viewed as a clustering version of the self-taught learning (Raina et al., 2007), which uses irrelevant unlabeled data to help supervised learning. Thus, we refer to our problem as *self-taught clustering*

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

(a) diamond (b) platinum

(c) ring (d) titanium

Figure 1. Example for common features among different types of objects, using images as the instance.

(or STC for abbreviation).

To tackle the problem, we observe that the performance of clustering highly relies on data representation when the objective function and the distance measure are fixed. Therefore, to improve the clustering performance, one alternative way is to seek a better data representation. We observe that different objects may share some common or relevant features. For example, in Figure 1, `diamond` and `ring` share quite a lot of features about "diamond"; `ring` and `platinum` share quite a lot of features about "platinum"; moreover, `platinum` and `titanium` share quite a lot of features about "metal". In this situation, the auxiliary data can be used to help uncover a better data representation to benefit the target data set. Our approach to tackling this problem is by using co-clustering (Dhillon et al., 2003), so that the commonality can be found in the feature spaces that corresponds to similar semantic meanings.

In our solution to the self-taught clustering problem, two clustering operations, on the target data and the auxiliary data are respectively performed together. This is done through co-clustering. We extend the information theoretic co-clustering algorithm (Dhillon et al., 2003) which minimizes *loss in mutual information* before and after co-clustering. An iterative algorithm is proposed to monotonically reduce the objective function. The experimental results show that our algorithm can greatly improve the clustering performance by effectively using auxiliary unlabeled data, as compared to several other state-of-the-art clustering algorithms.

2. Problem Formulation

For clarity, we first define the self-taught clustering task. Let X and Y be two discrete random variables, taking values from two value sets $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_m\}$, respectively. X and Y correspond to the target and auxiliary data. Let Z be a discrete random variable, taking values from the value set $\{z_1, \ldots, z_k\}$, that corresponds to the *common* feature space of both target and auxiliary data.

Let $p(X, Z)$ be the joint probability distribution with respect to X and Z, and $q(Y, Z)$ be the joint probability distribution with respect to Y and Z. In general, $p(X, Z)$ and $q(Y, Z)$ can be considered as two $n \times k$ and $m \times k$ matrices respectively, which can be estimated from data observations. For example, consider the case that $x_1 = \{z_1, z_3\}$, $x_2 = \{z_2\}$, and $x_3 = \{z_2, z_3\}$. Then, the joint probability distribution $p(X, Z)$ can be estimated as

$$p(X, Z) = \begin{bmatrix} 0.2 & 0.0 & 0.2 \\ 0.0 & 0.2 & 0.0 \\ 0.0 & 0.2 & 0.2 \end{bmatrix}. \quad (1)$$

We wish to cluster X into N partitions $\tilde{X} = \{\tilde{x}_1, \ldots, \tilde{x}_N\}$ and Y into M clusters $\tilde{Y} = \{\tilde{y}_1, \ldots, \tilde{y}_M\}$. Furthermore, Z can be clustered into K feature clusters $\tilde{Z} = \{\tilde{z}_1, \ldots, \tilde{z}_K\}$. We use $C_X : X \mapsto \tilde{X}$, $C_Y : Y \mapsto \tilde{Y}$ and $C_Z : Z \mapsto \tilde{Z}$ to denote three clustering functions, which map variables in the three value sets to their corresponding clusters. For brevity, in the following, we will use $\tilde{X}$, $\tilde{Y}$ and $\tilde{Z}$ to denote $C_X(X)$, $C_Y(Y)$ and $C_Z(Z)$, respectively.

Our objective is to find a good clustering function C_X for the target data, with the help of the clusters C_Y on the auxiliary data and C_Z on the common feature space.

3. The Self-taught Clustering Algorithm

In this section, we present our co-clustering based *self-taught clustering* (STC) algorithm, and then discuss its theoretical properties based on information theory.

3.1. Objective Function for Self-taught Clustering

We extend the information theoretic co-clustering (Dhillon et al., 2003) to model our self-taught clustering algorithm. In the information theoretic co-clustering, the objective function of co-clustering is defined as minimizing *loss in mutual information* between instances and features, before and after co-clustering. Formally, using the target data X and their feature space Z for illustration, the objective function can be expressed as

$$I(X, Z) - I(\tilde{X}, \tilde{Z}), \quad (2)$$

where $I(\cdot\,;\,\cdot)$ denotes the mutual information between two random variables (Cover & Thomas, 1991) that $I(X; Z) = \sum_{x \in X} \sum_{z \in Z} p(x, z) \log \frac{p(x,z)}{p(x)p(z)}$. Moreover, $I(\tilde{X}, \tilde{Z})$ corresponds to the joint probability distribu-

tion $p(\tilde{X}, \tilde{Z})$ which is defined as

$$p(\tilde{x}, \tilde{z}) = \sum_{x \in \tilde{x}} \sum_{z \in \tilde{z}} p(x, z). \quad (3)$$

For example, for the joint probability $p(X, Z)$ in Equation (1), suppose that the clustering on X is $\tilde{X} = \{\tilde{x}_1 = \{x_1, x_2\}, \tilde{x}_2 = \{x_3\}\}$, and the clustering on Z is $\tilde{Z} = \{\tilde{z}_1 = \{z_1, z_2\}, \tilde{z}_2 = \{z_3\}\}$. Then,

$$p(\tilde{X}, \tilde{Z}) = \begin{bmatrix} 0.4 & 0.2 \\ 0.2 & 0.2 \end{bmatrix}. \quad (4)$$

In this work, we model our self-taught clustering algorithm (STC) as performing co-clustering operations on the target data X and auxiliary data Y, simultaneously, while the two co-clusters share the same features clustering $\tilde{Z}$ on the feature set Z. Thus, the objective function can be formulated as

$$\mathcal{J} = I(X, Z) - I(\tilde{X}, \tilde{Z}) + \lambda \Big[I(Y, Z) - I(\tilde{Y}, \tilde{Z}) \Big]. \quad (5)$$

In Equation (2), $I(X, Z) - I(\tilde{X}, \tilde{Z})$ is computed on the co-clusters on the target data X, while $I(Y, Z) - I(\tilde{Y}, \tilde{Z})$ on the auxiliary data Y. λ is a trade-off parameter to balance the influence between the target data and the auxiliary data which we will test in our experiments. From Equation (5), we can see that, although the two co-clustering objective functions $I(X, Z) - I(\tilde{X}, \tilde{Z})$ and $I(Y, Z) - I(\tilde{Y}, \tilde{Z})$ are performed separately, they share the same feature clustering $\tilde{Z}$. This is the "bridge" to transfer the knowledge between the target and auxiliary data.

Our remaining task is to minimize the value of the objective function in Equation (5)[1]. However, minimizing Equation (5) is not an easy task, since it is non-convex and there are no good solutions currently to directly optimize this objective function. In the following, we will rewrite the objective function in Equation (5) into the form of Kullback-Leibler divergence (Cover & Thomas, 1991) (KL divergence), and minimize the reformulated objective function.

3.2. Optimization for Co-clustering

We first define two new probability distributions $\tilde{p}(X, Z)$ and $\tilde{q}(Y, Z)$ as follows.

Definition 1 *Let $\tilde{p}(X, Z)$ denote the joint probability distribution of X and Z with respect to the co-clusters (C_X, C_Z); formally,*

$$\tilde{p}(x, z) = p(\tilde{x}, \tilde{z}) \frac{p(x)}{p(\tilde{x})} \frac{p(z)}{p(\tilde{z})}, \quad (6)$$

where $x \in \tilde{x}$ and $z \in \tilde{z}$. Therefore, with regard to Equations (1) and (4), $\tilde{p}(X, Z)$ is given by

$$\tilde{p}(X, Z) = \begin{bmatrix} 0.089 & 0.178 & 0.133 \\ 0.044 & 0.089 & 0.067 \\ 0.067 & 0.133 & 0.200 \end{bmatrix}. \quad (7)$$

Likewise, let $\tilde{q}(Y, Z)$ denote the joint probability distribution of Y and Z with respect to the co-clusters (C_Y, C_Z). We have

$$\tilde{q}(y, z) = q(\tilde{y}, \tilde{z}) \frac{q(y)}{q(\tilde{y})} \frac{q(z)}{q(\tilde{z})}, \quad (8)$$

where $y \in \tilde{y}$ and $z \in \tilde{z}$.

Using the probability distributions $\tilde{p}(X, Z)$ and $\tilde{q}(Y, Z)$ defined above, we can reformulate the objective function in Equation (5) into a form based on KL divergence (Cover & Thomas, 1991).

Lemma 1 *When the clusters C_X, C_Y and C_Z are fixed, the objective function in Equation (5) can be reformulated as*

$$\begin{aligned} & I(X; Z) - I(\tilde{X}; \tilde{Z}) + \lambda \Big[I(Y; Z) - I(\tilde{Y}; \tilde{Z}) \Big] \quad (9) \\ & = D(p(X, Z) || \tilde{p}(X, Z)) + \lambda \, D(q(Y, Z) || \tilde{q}(Y, Z)), \end{aligned}$$

where $D(\cdot||\cdot)$ denotes the KL divergence between two probability distributions (Cover & Thomas, 1991), where $D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$.

Proof Based on the Lemma 2.1 in (Dhillon et al., 2003), $I(X, Z) - I(\tilde{X}, \tilde{Z}) = D(p(X, Z)||\tilde{p}(X, Z))$. Similarly, $I(Y, Z) - I(\tilde{Y}, \tilde{Z}) = D(q(Y, Z)||\tilde{q}(Y, Z))$. Therefore, Lemma 1 can be proved straightforwardly. □

Lemma 1 converts the loss in mutual information to the KL divergence between the distributions p and $\tilde{p}$, and between q and $\tilde{q}$, respectively. However, the probability distributions in Lemma 1 are joint distributions, and are therefore difficult to optimize. Hence, in Lemma 2, we rewrite the objective function in Lemma 1 as a conditional probability form. We then show how to optimize the objective function in the new form.

Lemma 2 *The KL divergence with respect to joint probability distributions can be reformulated as*

$$\begin{aligned} & D(p(X, Z)||\tilde{p}(X, Z)) \\ & = \sum_{\tilde{x} \in \tilde{X}} \sum_{x \in \tilde{x}} p(x) D(p(Z|x)||\tilde{p}(Z|\tilde{x})) \quad (10) \\ & = \sum_{\tilde{z} \in \tilde{Z}} \sum_{z \in \tilde{z}} p(z) D(p(X|z)||\tilde{p}(X|\tilde{z})). \quad (11) \end{aligned}$$

[1] To be mentioned, in this paper, our minimization is for a fixed numbers of clusters N, M and K.

Similarly,

$$D(q(Y,Z)||\tilde{q}(Y,Z)) = \sum_{\tilde{y}\in\tilde{Y}}\sum_{y\in\tilde{y}} q(y)D(q(Z|y)||\tilde{q}(Z|\tilde{y})) \quad (12)$$

$$= \sum_{\tilde{z}\in\tilde{Z}}\sum_{z\in\tilde{z}} q(z)D(q(Y|z)||\tilde{q}(Y|\tilde{z})). \quad (13)$$

Proof We only give the proof to Equation (10). Using an identical argument, Equations (11), (12) and (13) can be easily derived.

$$D(p(X,Z)||\tilde{p}(X,Z)) = \sum_{\tilde{x}\in\tilde{X}}\sum_{\tilde{z}\in\tilde{Z}}\sum_{x\in\tilde{x}}\sum_{z\in\tilde{z}} p(x,z)\log\frac{p(x,z)}{\tilde{p}(x,z)}.$$

Since $\tilde{p}(x,z) = p(x)\frac{p(\tilde{x},\tilde{z})}{p(\tilde{x})}\frac{p(z)}{p(\tilde{z})} = p(x)\tilde{p}(z|\tilde{x})$, we have

$$\begin{aligned} &D(p(X,Z)||\tilde{p}(X,Z)) \\ &= \sum_{\tilde{x}\in\tilde{X}}\sum_{\tilde{z}\in\tilde{Z}}\sum_{x\in\tilde{x}}\sum_{z\in\tilde{z}} p(x)p(z|x)\log\frac{p(x)p(z|x)}{p(x)\tilde{p}(z|\tilde{x})} \\ &= \sum_{\tilde{x}\in\tilde{X}}\sum_{x\in\tilde{x}} p(x)\sum_{\tilde{z}\in\tilde{Z}}\sum_{z\in\tilde{z}} p(z|x)\log\frac{p(z|x)}{\tilde{p}(z|\tilde{x})} \\ &= \sum_{\tilde{x}\in\tilde{X}}\sum_{x\in\tilde{x}} p(x)D(p(Z|x)||\tilde{p}(Z|\tilde{x})). \end{aligned}$$

□

From Lemma 2 and Equation (10), we can see that minimizing $D(p(Z|x)||\tilde{p}(Z|\tilde{x}))$ for a single x can reduce the value of $D(p(X,Z)||\tilde{p}(X,Z))$ and thus can then decrease global optimization function in Equation (9). Therefore, if we iteratively choose the best cluster $\tilde{x}$ for each x to minimize $D(p(Z|x)||\tilde{p}(Z|\tilde{x}))$, the objective function will be minimized monotonically. Formally,

$$C_X(x) = \arg\min_{\tilde{x}\in\tilde{X}} D(p(Z|x)||\tilde{p}(Z|\tilde{x})). \quad (14)$$

Using a similar argument on Y and Z, we have

$$C_Y(y) = \arg\min_{\tilde{y}\in\tilde{Y}} D(q(Z|y)||\tilde{q}(Z|\tilde{y})), \quad (15)$$

and

$$C_Z(z) = \arg\min_{\tilde{z}\in\tilde{Z}} p(z)D(p(X|z)||\tilde{p}(X|\tilde{z})) + \lambda\, q(z)D(q(Y|z)||\tilde{q}(Y|\tilde{z})). \quad (16)$$

Based on Equation (14), (15) and (16), an alternative way to minimize the objective function in Equation (9) is derived, as shown in Algorithm 1.

In Algorithm 1, in each iteration, our self-taught clustering algorithm (STC) minimizes the objective function by choosing the best $\tilde{x}$, $\tilde{y}$ and $\tilde{z}$ for each x, y and z based on Equations (14), (15) and (16). As we discussed above, this can reduce the value of the global objective function in Equation (9). In the following theorem, we show the monotonically decreasing property of the objective function of the STC algorithm.

Algorithm 1 The Self-taught Clustering Algorithm: STC

Input: A target unlabeled data set X; an auxiliary unlabeled data set Y; the feature space Z shared by both X and Y; the initial clustering functions $C_X^{(0)}$, $C_Y^{(0)}$ and $C_Z^{(0)}$; the number of iterations T.

Output: The final clustering function $C_X^{(T)}$ on the target data X.

Procedure STC

1: Initialize $p(X,Z)$ and $q(Y,Z)$ based on the data observations on X, Y, and Z.
2: Initialize $\tilde{p}^{(0)}(X,Z)$ based on $p(X,Z)$, $C_X^{(0)}$, $C_Z^{(0)}$, and Equation (6).
3: Initialize $\tilde{q}^{(0)}(Y,Z)$ based on $q(Y,Z)$, $C_Y^{(0)}$, $C_Z^{(0)}$, and Equation (8).
4: **for** $t \leftarrow 1,\ldots,T$ **do**
5: Update $C_X^{(t)}(X)$ based on p, $\tilde{p}^{(t-1)}$, and Equation (14).
6: Update $C_Y^{(t)}(Y)$ based on q, $\tilde{q}^{(t-1)}$, and Equation (15).
7: Update $C_Z^{(t)}(Z)$ based on p, q, $\tilde{p}^{(t-1)}$, $\tilde{q}^{(t-1)}$, and Equation (16).
8: Update $\tilde{p}^{(t)}$ based on based on $p(X,Z)$, $C_X^{(t)}$, $C_Z^{(t)}$, and Equations (6).
9: Update $\tilde{q}^{(t)}$ based on based on $q(Y,Z)$, $C_Y^{(t)}$, $C_Z^{(t)}$, and Equations (8).
10: **end for**
11: Return $C_X^{(T)}$ as the final clustering function on the target data X.

Theorem 1 *In Algorithm 1, let the value of objective function $\mathcal{J}$ in the t-th iteration be*

$$\mathcal{J}(C_X^{(t)},C_Y^{(t)},C_Z^{(t)}) = D(p(X,Z)||\tilde{p}^{(t)}(X,Z)) + \lambda\, D(q(Y,Z)||\tilde{q}^{(t)}(Y,Z)). \quad (17)$$

Then,

$$\mathcal{J}(C_X^{(t)},C_Y^{(t)},C_Z^{(t)}) \geq \mathcal{J}(C_X^{(t+1)},C_Y^{(t+1)},C_Z^{(t+1)}). \quad (18)$$

Proof (Sketch) Since in each iteration, the clustering functions are updated based on Equations (14), (15) and (16), which locally minimize the values of $D(p(X,Z)||\tilde{p}(X,Z))$ and $D(q(Y,Z)||\tilde{q}(Y,Z))$, the objective function is monotonically non-increasing as a result. Theorem 1 follows as a consequence. □

Note that, although STC is able to minimize the objective function value in Equation (9), it is only able to find a locally optimal one. Finding the global optimal solution is NP-hard. The next corollary emphasizes the convergence property of our algorithm STC.

Corollary 1 *Algorithm 1 converges in a finite number of iterations.*

Proof (Sketch) The convergence of our algorithm STC can be proved straightforwardly based on the monotonical decreasing property in Theorem 1, and the finiteness of the solution space. □

3.3. Complexity Analysis

We now analyze the computational cost of our algorithm STC. Suppose that the total number of (x, z) co-occurrences in the target data set X is L_1, and the total number of (y, z) co-occurrences in the auxiliary data set Y is L_2. In each iteration, updating the target instance clustering C_X takes $O(N \cdot L_1)$. Updating the auxiliary instance clustering C_Y takes $O(M \cdot L_2)$. Moreover, updating the feature clustering C_Z takes $O(K \cdot (L_1 + L_2))$. Since the number of iterations is T, the time complexity of our algorithm is $O(T \cdot ((K + N) \cdot L_1 + (K + M) \cdot L_2)))$. In the following experiments, it is shown that $T = 10$ is enough for convergence. Usually, the number of clusters N, M and K can be considered as constants, so that the time complexity of STC is $O(L_1 + L_2)$.

Considering space complexity, our algorithm needs to store all the (x, z) and (y, z) co-occurrences and their corresponding probabilities. Thus, the space complexity is $O(L_1 + L_2)$. This indicates that the time complexity and the space complexity of our algorithm are all linear on the input. We conclude that the algorithm scales well.

4. Experiments

In this section, we evaluate our self-taught clustering algorithm STC on the image clustering tasks, and show effectiveness of STC.

4.1. Data Sets

We conduct our experiments on eight clustering tasks generated based on the Caltech-256 image corpus (Griffin et al., 2007). There are a total of 256 categories in the Caltech-256 data set, where we randomly chose 20 categories from this corpus. For each category, 70 images are randomly selected to form our clustering tasks. Six binary clustering tasks, one 3-way clustering task, and one 5-way clustering task were generated using these 20 categories, as shown in Table 1. The first column in Table 1 presents the categories with respect to the target unlabeled data. For each clustering task, we used the data from the corresponding categories as target unlabeled data, while the data from the remaining categories as the auxiliary unlabeled data.

For data preprocessing, we used the "bag-of-words" method (Li & Perona, 2005) to represent images in our experiments. Interesting points in images are found and described by SIFT descriptor (Lowe, 2004). Then, we clustered all the interesting points to get the codebook, and set the number of clusters to 800. Using this codebook, each image can be represented as a vector in the subsequent learning processes.

4.2. Evaluation Criteria

In these experiments, we used *entropy* to measure the quality of clustering results, which reveals the purity of clusters. Specifically, the entropy for a cluster $\tilde{x}$ is defined as $H(\tilde{x}) = -\sum_{c \in C} p(c|\tilde{x}) \log_2 p(c|\tilde{x})$, where c represents a category label in the evaluation corpus, and $p(c|\tilde{x})$ is defined as $p(c|\tilde{x}) = \frac{|\{x|\ell(x)=c \wedge x \in \tilde{x}\}|}{|\tilde{x}|}$, where $\ell(x)$ denotes the *true* label of x in the evaluation corpus. The *total entropy* for the whole clustering is defined as the weighted sum of the entropy with respect to all the clusters; formally, $H(\tilde{X}) = \sum_{\tilde{x} \in \tilde{X}} \frac{|\tilde{x}|}{n} H(\tilde{x})$. The quality of clustering $\tilde{X}$ is evaluated using the entropy $H(\tilde{X})$.

4.3. Empirical Analysis

We compared our algorithm STC to several state-of-the-art clustering methods as baseline methods. For each baseline method considered below, we have two different options: one is to apply the baseline method on the target data only, which we refer to as `separate`, and the other is to apply on the combined data consisting of target data and the auxiliary data, which we refer as `combined`. The first baseline method is a traditional 1D-clustering solution `CLTUO` (Zhao & Karypis, 2002) using its default parameter. The second baseline method is clustering on the target data under a new feature representation that is first constructed through feature clustering (on the target or the combined data set); this baseline is designed to evaluate the effectiveness of co-clustering based method as opposed to naively constructing new data representation for clustering. We refer to this class of baseline methods as `Feature Clustering`. The third baseline method is an information theoretic co-clustering method applied to the target (or the combined) data set (Dhillon et al., 2003), which we refer to as `Co-clustering`. This base-

Table 1. Performance in terms of entropy for each data set and evaluation method.

Data Set	CLUTO		Feature Clustering		Co-clustering		STC
	separate	combined	separate	combined	separate	combined	
eyeglass vs sheet-music	0.527	0.966	0.669	0.669	0.630	0.986	0.187
airplane vs ostrich	0.352	0.696	0.512	0.479	0.426	0.753	0.252
fern vs starfish	0.865	0.988	0.588	0.953	0.741	0.968	0.575
guitar vs laptop	0.923	0.965	0.999	0.970	0.925	1.000	0.569
hibiscus vs ketch	0.371	0.446	0.659	0.649	0.399	0.793	0.252
cake vs harp	0.882	0.879	0.998	0.911	0.860	0.996	0.772
car-side, tire, frog	1.337	1.385	1.362	1.413	1.316	1.275	1.000
cd, comet, vcr, diamond-ring, skyscaper	1.663	1.827	1.755	1.751	1.715	1.772	1.274
Average	0.865	1.019	0.943	0.974	0.877	1.068	0.610

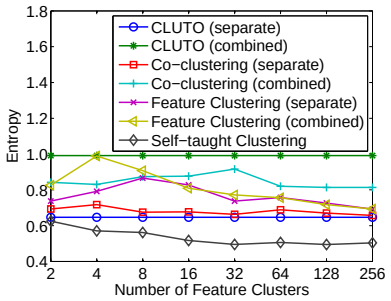


Figure 2. The entropy curves as a function of different number of feature clusters.

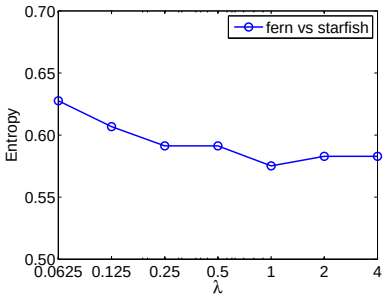


Figure 3. The entropy curves as a function of different trade-off parameter λ.

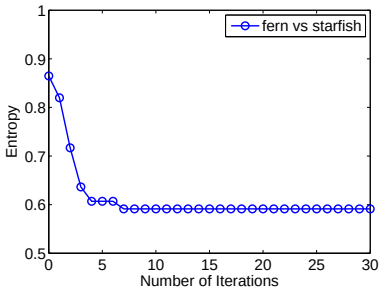


Figure 4. The entropy curves as a function of different number of iterations.

line is designed to test the effectiveness of our special co-clustering model for self-taught clustering.

Table 1 presents the clustering performance in entropy according to each data set and each evaluation method. From this table, we can see that `Feature Clustering` and `Co-clustering` perform somewhat worse than `CLUTO`. This is a little different from the results shown in the previous literatures such as (Dhillon et al., 2003). In our opinion, it is because our self-taught clustering problem focuses on a different situation from the previous ones; that is, the target data are insufficient for traditional clustering algorithms. In our experiments, there are only 70 instances in each category, which is too few to build a good feature clustering partition. Therefore, the performance of `Feature Clustering` and `Co-clustering` declines. Moreover, the performance with respect to `combined` is worse than that with respect to `separate` in general. We believe that it is because the target data and the auxiliary data are more or less independent of each other, and thus the topics in the combined data set may be biased towards the auxiliary data and thus harm the clustering performance on the target data. In general, our algorithm STC greatly outperforms the three baseline methods. We observe that the reason for the outstanding performance of STC is that the co-clustering part of STC makes feature clustering result consistent with the clustering result on both the target data and the auxiliary data. Therefore, using this feature clustering as the new data representation, the clustering performance of the target data is improved.

In our STC algorithm, it is assumed that we have already known the number of feature clusters K. However, in reality, this number should be carefully tuned. In these experiments, we tuned this parameter empirically. Figure 2 presents the entropy curves with respect to different number of feature clusters given by `CLUTO`, `Feature Clustering`, `Co-clustering` and STC respectively. The entropy in Figure 2 is the average over 6 binary image clustering tasks. Note that the curve given by `CLUTO` never changes, since `CLUTO` does not incorporate feature clustering. From this figure, we can see `Feature Clustering` and `Co-clustering` perform somewhat unstably as a function of the increasing number of feature clustering. We believe the reason is that there are only too few instances in each clustering task, which makes the traditional clustering results unreliable. Our algorithm STC incorporates a large amount of auxiliary unlabeled data, so that its variance is much smaller than that of traditional clustering algorithms. STC performs increasingly better in general, along with the increasing number of feature clustering, until the number of feature clusters reaches 32. When the number of feature clusters is greater than 32, the performance of STC becomes insensitive to the number of feature clusters. We believe a number of feature clustering which is no less than 32 will be sufficient to make STC perform well. In these experiments, we set the number of feature clustering

to 32.

We next tested the choice for the trade-off parameter λ in our algorithm STC (refer to Equation (5)). Generally, it is difficult to theoretically determine the value of the trade-off parameter λ. Instead, in this work, we tuned this parameter empirically on the data set `fern vs starfish`. Figure 3 presents the entropy curve given by STC along with changing trade-off parameter λ. From this figure, it can be seen that, when λ decreases, which implies that the weights of the auxiliary unlabeled data lower, the performance of STC declines rapidly. On the other hand, when λ is sufficiently large, i.e. $\lambda > 1$, the performance of STC is relatively insensitive to the parameter λ. This indicates the auxiliary data can help the clustering on the target data in our clustering tasks. In these experiments, we set the trade-off parameter λ to one, which is the best point in Figure 3.

Since our algorithm STC is iterative, the convergence property is also important to evaluate. Theorem 1 and Corollary 1 have already proven the convergence of STC theoretically. Here, we analyze the convergence of STC empirically. Figure 4 shows the entropy curve given by STC corresponding to different number of iterations on the data set `fern vs starfish`. From this figure, we can see that STC converges very well after 7 iterations, while the performance of STC reaches the lowest point when STC converges. This indicates that our algorithm STC converges very fast and very well. In these experiments, we set the number of iterations T to 10. We believe 10 iterations are enough for STC to converge.

5. Related Work

In this section, we review several past research works that are related to our work, including semi-supervised clustering, supervised clustering and transfer learning.

Semi-supervised clustering improves clustering performance by incorporating additional constraints provided by a few labeled data, in the form of *must-links* (two examples must in the same cluster) and *cannot-links* (two examples cannot in the same cluster) (Wagstaff et al., 2001). It finds a balance between satisfying the pairwise constraints and optimizing the original clustering criteria function. In addition to (Wagstaff et al., 2001), Basu et al. (2002) used a small amount of labeled data to generate initial seed clusters in K-means and constrained K-means algorithm by labeled data. Basu et al. (2004) generalized the previous semi-supervised clustering algorithms and proposed a probabilistic framework based on *hidden Markov random fields* that combines the constraints and clustering distortion measures in a general framework. Recent semi-supervised clustering works include (Nelson & Cohen, 2007; Davidson & Ravi, 2007).

Supervised clustering is another branch of work designed to improve clustering performance with the help of a collection of auxiliary labeled data. To address the supervised clustering problem, Finley and Joachims (2005) proposed an SVM-based supervised clustering algorithm by optimizing a variety of different clustering functions. Daumé III and Marcu (2005) developed a Bayesian framework for supervised clustering based on Dirichlet process prior.

Transfer learning emphasizes the transferring of knowledge across different domains or tasks. For example, multi-task learning (Caruana, 1997) or clustering (Teh et al., 2006) learns the common knowledge among different related tasks. Wu and Dietterich (2004) investigated methods for improving SVM classifiers with *auxiliary* training data sources. Raina et al. (2006) proposed to learn logistic regression classifiers by incorporating labeled data from irrelevant categories through constructing informative prior from the irrelevant labeled data. Raina et al. (2007) proposed a new learning strategy known as *self-taught learning*, which utilizes irrelevant unlabeled data to enhance the classification performance.

In this paper, we propose a new clustering framework called *self-taught clustering* which is an instance of *unsupervised transfer learning*. The basic idea is to use irrelevant unlabeled data to help the clustering of a small amount of target data. To our best knowledge, our self-taught clustering problem is novel in capturing a large class of machine learning problems.

6. Conclusions and Future Work

In this paper, we investigated an unsupervised transfer learning problem called *self-taught clustering*, and developed a solution by using an unlabeled auxiliary data to help improve the target clustering results. We proposed a co-clustering based self-taught clustering algorithm (STC) to solve this problem. In our algorithm, two co-clusterings are performed simultaneously on the target data and the auxiliary data to uncover the shared feature clusters. Our empirical results show that the auxiliary data can help the target data to construct a better feature clustering as data representation. Under the new data representation, the clustering performance on the target data is indeed enhanced, and our algorithm can greatly outperform several state-of-the-art clustering methods in the ex-

periments.

In this work, we tackled the self-taught clustering by finding a better feature representation using co-clustering. In the future, we will explore several other ways in finding common feature representations.

Acknowledgements

Qiang Yang thanks Hong Kong CERG grants 621307 and CAG grant HKBU1/05C. We thank the anonymous reviewers for their greatly helpful comments.

References

Bach, F. R., Lanckriet, G. R. G., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. *Proceedings of the Twenty-first International Conference on Machine Learning* (pp. 6–13).

Basu, S., Banerjee, A., & Mooney, R. J. (2002). Semi-supervised clustering by seeding. *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 27–34).

Basu, S., Bilenko, M., & Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 59–68).

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. Wiley-Interscience.

Daumé III, H., & Marcu, D. (2005). A bayesian model for supervised clustering with the dirichlet process prior. *Journal of Machine Learning Research*, *6*, 1551–1577.

Davidson, I., & Ravi, S. S. (2007). Intractability and clustering with constraints. *Proceedings of the Twenty-fourth International Conference on Machine Learning* (pp. 201–208).

Dhillon, I. S., Mallela, S., & Modha, D. S. (2003). Information-theoretic co-clustering. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 89–98).

Finley, T., & Joachims, T. (2005). Supervised clustering with support vector machines. *Proceedings of the Twenty-second International Conference on Machine Learning* (pp. 217–224).

Griffin, G., Holub, A., & Perona, P. (2007). *Caltech-256 object category dataset* (Technical Report 7694). California Institute of Technology.

Jain, A. J., & Dubes, R. C. (1988). *Algorithms for clustering data*. Englewood, NJ: Prentice-Hall.

Li, F.-F., & Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2* (pp. 524–531).

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 1:281–297).

Nelson, B., & Cohen, I. (2007). Revisiting probabilistic models for clustering with pair-wise constraints. *Proceedings of the Twenty-fourth International Conference on Machine Learning* (pp. 673–680).

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. *Proceedings of the Twenty-fourth International Conference on Machine Learning* (pp. 759–766).

Raina, R., Ng, A. Y., & Koller, D. (2006). Constructing informative priors using transfer learning. *Proceedings of the Twenty-third International Conference on Machine Learning* (pp. 713–720).

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, *101*, 1566–1581.

Wagstaff, K., Cardie, C., Rogers, S., & Schrödl, S. (2001). Constrained k-means clustering with background knowledge. *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 577–584).

Wu, P., & Dietterich, T. G. (2004). Improving svm accuracy by training on auxiliary data sources. *Proceedings of the Twenty-first International Conference on Machine Learning* (pp. 110–117).

Zhao, Y., & Karypis, G. (2002). Evaluation of hierarchical clustering algorithms for document datasets. *Proceedings of the Eleventh International Conference on Information and Knowledge Management* (pp. 515–524).

Hierarchical Sampling for Active Learning

Sanjoy Dasgupta DASGUPTA@CS.UCSD.EDU
Daniel Hsu DJHSU@CS.UCSD.EDU
Department of Computer Science and Engineering, University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093-0404

Abstract

We present an active learning scheme that exploits cluster structure in data.

1. Introduction

The *active learning* model is motivated by scenarios in which it is easy to amass vast quantities of unlabeled data (images and videos off the web, speech signals from microphone recordings, and so on) but costly to obtain their labels. It shares elements with both supervised and unsupervised learning. Like supervised learning, the goal is ultimately to learn a classifier. But like unsupervised learning, the data come unlabeled. More precisely, the labels are hidden, and each of them can be revealed only at a cost. The idea is to query the labels of just a few points that are especially informative about the decision boundary, and thereby to obtain an accurate classifier at significantly lower cost than regular supervised learning. Indeed, there are canonical examples in which active learning provably yields exponentially lower label complexity than supervised learning (Cohn et al., 1994; Freund et al., 1997; Dasgupta, 2005; Balcan et al., 2006; Balcan et al., 2007; Castro & Nowak, 2007; Hanneke, 2007; Dasgupta et al., 2007). However, these examples are highly specific, and the wider efficacy of active learning remains to be characterized.

Sampling bias. A typical active learning heuristic might start by querying a few randomly-chosen points, to get a very rough idea of the decision boundary. It might then query points that are increasingly closer to its current estimate of the boundary, with the hope of rapidly honing in. Such heuristics immediately bring to the forefront the unique difficulty of active learning, the fundamental characteristic that separates it from other learning models: *sampling bias*. As training proceeds, and points are queried based on increasingly confident assessments of their informativeness, the training set quickly diverges from the underlying data distribution. It consists of an unusual subset of points, hardly a representative subsample; why should a classifier trained on these strange points do well on the overall distribution? In section 2, we make this intuition concrete, and show how ill-managed sampling bias causes many active learning heuristics to not be consistent: even with infinitely many labels, they fail to converge to a good hypothesis.

The two faces of active learning. The recent literature offers two distinct narratives for explaining when active learning is helpful. The first has to do with *efficient search through the hypothesis space*. Each time a new label is seen, the set of plausible classifiers (those roughly consistent with the labels seen so far) shrinks somewhat. Using active learning, one can explicitly select points whose labels will shrink this set as fast as possible. Most theoretical work in active learning attempts to formalize this intuition.

The second argument for active learning has to do with *exploiting cluster structure in data*. Suppose, for instance, that the unlabeled points form five nice clusters; with luck, these clusters will be "pure" and only five labels will be necessary! Of course, this is hopelessly optimistic. In general, there may be no nice clusters, or there may be viable clusterings at many different resolutions. The clusters themselves may only be mostly-pure, or they may not be aligned with labels at all. In this paper, we present a scheme for cluster-based active learning that is statistically consistent and never has worse label complexity than supervised learning. In cases where there exists cluster structure (at whatever resolution) that is loosely aligned with class labels, the scheme detects and exploits it.

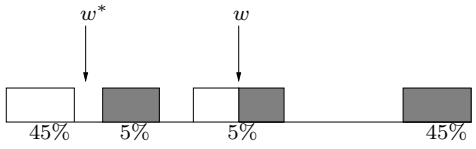
Our model. We start with a hierarchical clustering of the unlabeled points. This should be constructed so that some pruning of it is weakly informative of the

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

class labels. We describe an active learning strategy with good statistical properties, that will discover and exploit any informative pruning of the cluster tree. For instance, suppose it is possible to prune the cluster tree to m leaves (m unknown) that are fairly pure in the labels of their constituent points. Then, after querying just $O(m)$ labels, our learner will have a fairly accurate estimate of the labels of the *entire* data set. These can then be used as is, or as input to a supervised learner. Thus, our scheme can be used in conjunction with any hypothesis class, no matter how complex.

2. Active Learning and Sampling Bias

Many active learning heuristics start by choosing a few unlabeled points at random and querying their labels. They then repeatedly do something like this: fit a classifier $h \in H$ to the labels seen so far; and query the label of the unlabeled point closest to the decision boundary of h (or the one on which h is most uncertain, or something similar). Such schemes make intuitive sense, but do not correctly manage the bias introduced by adaptive sampling. Consider this 1-d example:



Here the data lie in four groups on the line, and are (say) distributed uniformly within each group. Filled blocks have a + label, while clear blocks have a − label. Most of the data lies in the two extremal groups, so an initial random sample has a good chance of coming entirely from these. Suppose the hypothesis class consists of thresholds on the line: $H = \{h_w : w \in \mathbb{R}\}$ where $h_w(x) = \mathbf{1}(x \geq w)$. Then the initial boundary will lie somewhere in the center group, and the first query point will lie in this group. So will every subsequent query point, forever. As active learning proceeds, the algorithm will gradually converge to the classifier shown as w. But this has 5% error, whereas classifier w^* has only 2.5% error. Thus the learner is not consistent: even with infinitely many labels, it returns a suboptimal classifier.

The problem is that the second group from the left gets overlooked. It is not part of the initial random sample, and later on, the learner is mistakenly confident that the entire group has a − label. And this is just in one dimension; in high dimension, the problem can be expected to be worse, since there are more places for this troublesome group to be hiding out. For a discussion of this problem in text classification, see the recent paper of Schutze et al. (2006).

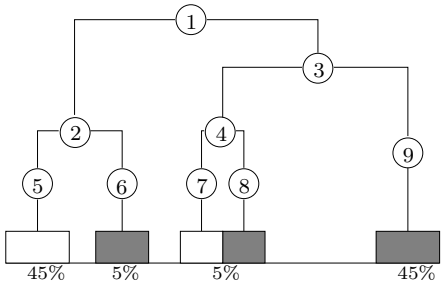


Figure 1. The top few nodes of a hierarchical clustering.

Sampling bias is the most fundamental challenge posed by active learning. This paper presents a broad framework for managing this bias that is provably sound.

3. A Clustering-Based Framework for Guiding Sampling

Our active learner starts with a hierarchical clustering of the data. Figure 1 shows how this might look for the example of the previous section.

Here only the top few nodes of the hierarchy are shown; their numbering is immaterial. At any given time, the learner works with a particular partition of the data set, given by a pruning of the tree. Initially, this is just $\{1\}$, a single cluster containing everything. Random points are drawn from this cluster and their labels are queried. Suppose one of these points, x, lies in the rightmost group. Then it is a random sample from node 1, but also from nodes 3 and 9. Based on these random samples, each node of the tree maintains statistics about the relative numbers of positive and negative instances seen. A few samples reveal that the top node 1 is very mixed while nodes 2 and 3 are substantially more pure. Once this transpires, the partition $\{1\}$ will be replaced by $\{2, 3\}$. Subsequent random samples will be chosen from either 2 or 3, according to a sampling strategy favoring the less-pure node. A few more queries down the line, the pruning will likely be refined to $\{2, 4, 9\}$. This is when the benefits of the partitioning scheme become most obvious; based on the samples seen, it can be concluded that cluster 9 is (almost) pure, and thus (almost) no more queries will be made from it until the rest of the space has been partitioned into regions that are similarly pure.

The querying can be stopped at any stage; then, each cluster in the current partition gets assigned the majority label of the points queried from it. In this way,

the *entire* data set gets labeled, and the number of erroneous labels induced is kept to a minimum. If desired, these labels can be used for a subsequent round of supervised learning, with any learning algorithm and any hypothesis class.

3.1. Preliminary Definitions

The cost of a pruning. Say there are n unlabeled points, and we have a hierarchical clustering represented by a binary tree T with n leaves. For any node v of the tree, denote by T_v both the subtree rooted at v and also the data points contained in this subtree (at its leaves). A *pruning* of the tree is a subset of nodes $\{v_1, \ldots, v_m\}$ such that the T_{v_i} are disjoint and together cover all the data. At any given stage, the active learner will work with a partition of the data set given by a pruning of T. In the analysis, we will also deal with a *partial pruning*: a subset of a pruning.

A weight of a node $v \in T$ is the proportion of the data set in T_v: $w_v =$ (number of leaves of T_v)$/n$. Likewise, the weight of a partial pruning is the fraction of the data set that it covers, $w(P) = \sum_{v \in P} w_v$. A full pruning has weight 1.

Suppose there are k possible labels, and that their proportions in T_v are $p_{v,l}$ for $l = 1, \ldots, k$. Then the error introduced by assigning all points in T_v their majority label is $\epsilon_v = 1 - \max_l p_{v,l}$. Consequently, the error induced by a particular pruning (or partial pruning) P—that is, the fraction of incorrect labels when each cluster of P is assigned its majority label—is

$$\epsilon(P) = \frac{1}{w(P)} \sum_{v \in P} w_v \epsilon_v$$

In pruning the tree, it always helps to go as far down as possible, *provided* we can accurately estimate the majority labels in those nodes.

Empirical estimates for individual nodes. Due of limited sampling, we will only have labels from some of the nodes, and even for those, we may not be able to correctly determine the majority label. If we assign label l to all the points in T_v, the induced error is $\epsilon_{v,l} = 1 - p_{v,l}$. Likewise, when each cluster v of pruning (or partial pruning) P is assigned label $L(v) \in \{1, 2, \ldots, k\}$, the error induced is

$$\epsilon(P, L) = \frac{1}{w(P)} \sum_{v \in P} w_v \epsilon_{v,L(v)}.$$

We will at any given time have only very imperfect estimates of the $p_{v,l}$'s and thus of these various error probabilities. Fix any node v, and suppose that at time t, we have queried $n_v(t)$ random points contained in that node. This gives us estimates of its class probabilities, $p_{v,l}(t)$. Correspondingly, our estimate of $\epsilon_{v,l}$ will be $\epsilon_{v,l}(t) = 1 - p_{v,l}(t)$.

Table 1. Key quantities in the algorithm and analysis. The indexing (t) specifies the empirical quantity at time t.

d_v	depth of node v in tree
d_P	maximum depth of nodes in P
w_v	weight of node v
$p_{v,l}$	fraction of label l in node v
$L^*(v)$	majority label of node v (that is, $\arg\max_l p_{v,l}$)
$n_v(t)$	number of points sampled from node v
$p_{v,l}(t)$	fraction of label l in points sampled from T_v
$\mathcal{A}(t)$	admissible (node,label) pairs
$\epsilon_{v,l}(t)$	$1 - p_{v,l}(t)$
$\widetilde{\epsilon}_{v,l}(t)$	$\epsilon_{v,l}(t)$ if $(v,l) \in \mathcal{A}(t)$; otherwise 1

The quality of these estimates can be assessed using generalization bounds. At any given time t, we can associate with each node v and label l a confidence interval $[p_{v,l}^{\mathrm{LB}}, p_{v,l}^{\mathrm{UB}}]$ within which we expect the true probability $p_{v,l}$ to lie. One possibility is to use $[\max(p_{v,l}(t) - \Delta_{v,l}(t), 0), \min(p_{v,l}(t) + \Delta_{v,l}, 1)]$, for $\Delta_{v,l}(t) \approx \frac{1}{n_v(t)} + \sqrt{\frac{p_{v,l}(t)(1-p_{v,l}(t))}{n_v(t)}}$. In Lemma 1, we will give a precise value for $\Delta_{v,l}(t)$ for which we are able to assert that (with high probability) *every* $p_{v,l}$ is *always* within this interval. However, there are other ways of constructing confidence intervals as well. The most accurate is simply to use the binomial (or hypergeometric) distribution directly.

When are we confident about the majority label of a subtree? As mentioned above, it is advantageous to descend as far as possible in the tree, provided we are confident about our estimate of the majority label. To this end, define

$$A_{v,l}(t) = \text{true} \quad \Leftrightarrow \quad (1 - p_{v,l}^{\mathrm{LB}}(t)) < \beta \cdot \min_{l' \neq l} (1 - p_{v,l'}^{\mathrm{UB}}(t)). \tag{1}$$

$A_{v,l}$ asserts that l is an *admissible* label for node v, in the weak sense that it incurs at most β times as much error as any other label. To see this, notice that label l gets at most $1 - p_{v,l}^{\mathrm{LB}}(t)$ fraction of the points wrong, whereas l' gets at least $1 - p_{v,l'}^{\mathrm{UB}}(t)$ fraction of the points wrong. In our experiments, we use $\beta = 2$, in which case

$$A_{v,l}(t) = \text{true} \quad \Leftrightarrow \quad p_{v,l}^{\mathrm{LB}}(t) > 2 p_{v,l'}^{\mathrm{UB}}(t) - 1 \ \forall l' \neq l.$$

For any given v, t, several different labels l might satisfy this criterion, for instance if $p_{v,l}^{\mathrm{LB}}(t) = p_{v,l}^{\mathrm{UB}}(t) = 1/k$ for all labels l. When there are only two possible labels, the criterion further simplifies to $p_{v,l}^{\mathrm{LB}}(t) > 1/3$.

We will maintain a set of (v, l) pairs for which the condition $A_{v,l}(t)$ is either true or was true sometime in the past:

$$\mathcal{A}(t) = \{(v, l) : A_{v,l}(t') \text{ for some } t' \leq t\}.$$

$\mathcal{A}(t)$ is the set of admissible (v, l) pairs at time t. We use it to stop ourselves from descending too far down tree T when only a few samples have been drawn. Specifically, we say pruning P and labeling L are *admissible* in tree T at time t if:

- $L(v)$ is defined for P and ancestors of P in T.
- $(v, L(v)) \in \mathcal{A}(t)$ for any node v that is a strict ancestor of P in T.
- For any node $v \in P$, there are two options:
  - either $(v, L(v)) \in \mathcal{A}(t)$;
  - or there is no l for which $(v, l) \in \mathcal{A}(t)$. In this case, if v has parent u, then $(u, L(v)) \in \mathcal{A}(t)$.

This final condition implies that if a node in P is not admissible (with any label), then it is forced to take on an admissible label of its parent.

Empirical estimate of the error of a pruning. For any node v, the empirical estimate of the error induced when all of subtree T_v is labeled l is $\epsilon_{v,l}(t) = 1 - p_{v,l}(t)$. This extends to a pruning (or partial pruning) P and a labeling L:

$$\epsilon(P, L, t) = \frac{1}{w(P)} \sum_{v \in P} w_v \epsilon_{v,L(v)}(t).$$

This can be a bad estimate when some of the nodes in P have been inadequately sampled. Thus we use a more conservative adjusted estimate:

$$\widetilde{\epsilon}_{v,l}(t) = \begin{cases} 1 - p_{v,l}(t) & \text{if } (v, l) \in \mathcal{A}(t) \\ 1 & \text{if } (v, l) \notin \mathcal{A}(t) \end{cases}$$

with $\widetilde{\epsilon}(P, L, t) = (1/w(P)) \sum_{v \in P} w_v \widetilde{\epsilon}_{v,L(v)}(t)$. The various definitions are summarized in Table 1.

Picking a good pruning. It will be convenient to talk about prunings not just of the entire tree T but also of subtrees T_v. To this end, define the *score of v at time t*—denoted $s(v, t)$—to be the adjusted empirical error of the best admissible pruning and labeling (P, L) of T_v. More precisely, $s(v, t)$ is

$$\min\{\widetilde{\epsilon}(P, L, t) : (P, L) \text{ admissible in } T_v \text{ at time } t\}.$$

Written recursively, $s(v, t)$ is the minimum of

- $\widetilde{\epsilon}_{v,l}(t)$, for all l;
- $\frac{w_a}{w_v} s(a, t) + \frac{w_b}{w_v} s(b, t)$, whenever v has children a, b and $(v, l) \in \mathcal{A}(t)$ for some l.

Algorithm 1 Cluster-adaptive active learning

Input: Hierarchical clustering of n unlabeled points; batch size B
$P \leftarrow \{\text{root}\}$ (current pruning of tree)
$L(\text{root}) \leftarrow 1$ (arbitrary starting label for root)
for time $t = 1, 2, \ldots$ until the budget runs out **do**
  for $i = 1$ to B **do**
    $v \leftarrow \text{select}(P)$
    Pick a random point z from subtree T_v
    Query z's label l
    Update empirical counts and probabilities $(n_u(t), p_{u,l}(t))$ for all nodes u on path from z to v
  end for
  In a bottom-up pass of T, update $\mathcal{A}$ and compute scores $s(u, t)$ for all nodes $u \in T$ (see text)
  for each (selected) $v \in P$ **do**
    Let (P', L') be the pruning and labeling of T_v achieving scores $s(v, t)$
    $P \leftarrow (P \setminus \{v\}) \cup P'$
    $L(v) \leftarrow L'(u)$ for all $u \in P'$
  end for
end for
for each cluster $v \in P$ **do**
  Assign each point in T_v the label $L(v)$
end for

Starting from the empirical estimates $p_{v,l}(t), p^{\text{LB}}_{v,l}, p^{\text{UB}}_{v,l}$, it is possible to update the set $\mathcal{A}(t)$ and to compute all the $\widetilde{\epsilon}_{v,l}(t)$ and $s(v, t)$ values in a single linear-time, bottom-up pass through the tree.

3.2. The Algorithm

Algorithm 1 contains the active learning strategy. It remains to specify the the manner in which the hierarchical clustering is built and the procedure **select**. *Regardless of how these decisions are made*, the algorithm is statistically sound in that the confidence intervals $p_{v,l} \pm \Delta_{v,l}(t)$ are valid, and these in turn validate the guarantees for admissible prunings/labelings. This leaves a lot of flexibility to explore different clustering and sampling strategies.

The select procedure. This controls the selective sampling. Some options:

(1) Choose $v \in P$ with probability $\propto w_v$. This is similar to random sampling.

(2) Choose v with probability $\propto w_v(1 - p^{\text{UB}}_{v,L(v)}(t))$. This is an active learning rule that reduces sampling

in regions of the space that have already been observed to be fairly pure in their labels.

(3) For each subtree $(T_z, z \in P)$, find the observed majority label, and assign this label to all points in the subtree; fit a classifier h to this data; and choose $v \in P$ with probability $\propto \min\{|\{x \in T_v : h(x) = +1\}|, |\{x \in T_v : h(x) = -1\}|\}$. This biases sampling towards regions close to the current decision boundary.

Building a hierarchical clustering. The scheme works best when there is a pruning P of the tree such that $|P|$ is small and a significant fraction of its constituent clusters are almost-pure. One option is to run a standard hierarchical clustering algorithm, like average linkage, perhaps with a domain-specific distance function (or one generated from a neighborhood graph). Another option is to use a bit of labeled data to guide the construction of the hierarchy.

3.3. Naive Sampling

First consider the naive sampling strategy in which a node $v \in P$ is selected in proportion to its weight w_v. We'll show that if there is an almost-pure pruning with m nodes, then only $O(m)$ labels are needed before the entire data is labeled almost-perfectly. Proofs are deferred to the full version of the paper.

Theorem 1 *Pick any $\delta, \eta > 0$ and any pruning Q with $\epsilon(Q) \leq \eta$. With probability at least $1 - \delta$, the learner induces a labeling (of the data set) with error $\leq (\beta + 1)\epsilon(Q) + \eta$ when the number of labels seen is*

$$Bt = O\left(\frac{\beta+1}{\beta-1} \cdot \frac{|Q|}{\eta} \log \frac{2^{d_Q} kB|Q|}{\eta\delta}\right).$$

Recall that β is used in the definition of an admissible label (equation (1)); we use $\beta = 2$ in our experiments.

The number of prunings with m nodes is about 4^m; and these correspond to roughly $(4k)^m$ possible classifications (each of the m clusters can take on one of k labels). Thus this result is what one would expect if one of these classifiers were chosen by supervised learning. In our scheme, we do not evaluate such classifiers directly, but instead evaluate the subregions of which they are composed. We start our analysis with confidence intervals for $p_{v,l}$ and n_v.

Lemma 1 *Pick any $\delta > 0$. With probability at least $1 - \delta$, the following holds for all nodes $v \in T$, all labels l, and all times t.*

(a) $|p_{v,l} - p_{v,l}(t)| \leq \Delta_{v,l} \leq \Delta_{v,l}(t)$, where

$$\Delta_{v,l} = \frac{2}{3n_v(t)} \log \frac{1}{\delta'} + \sqrt{\frac{2p_{v,l}(1-p_{v,l})}{n_v(t)} \log \frac{1}{\delta'}}.$$

$$\Delta_{v,l}(t) = \frac{5}{n_v(t)} \log \frac{1}{\delta'} + \sqrt{\frac{9p_{v,l}(t)(1-p_{v,l}(t))}{2n_v(t)} \log \frac{1}{\delta'}}.$$

for $\delta' = \delta/(kBt^2 d_v^2)$.

(b) $n_v(t) \geq Btw_v/2$ if $Btw_v \geq 8 \log(t^2 2^{2d_v}/\delta)$.

Our empirical assessment of the quality of a pruning P is a blend of sampling estimates $p_{v,l}(t)$ and perfectly known values w_v. Next, we examine the rate of convergence of $\epsilon(P, L, t)$ to the true value $\epsilon(P, L)$.

Lemma 2 *Assume the bounds of Lemma 1 hold. There is a constant c such that for all prunings (or partial prunings) $P \subset T$, all labelings L, and all t,*

$$w(P) \cdot |\epsilon(P, L, t) - \epsilon(P, L)| \leq c \cdot \left(\frac{|P|}{Bt} \log \frac{kBt^2 2^{d_P}}{\delta} + \sqrt{w(P)\epsilon(P, L)\frac{|P|}{Bt} \log \frac{kBt^2 2^{d_P}}{\delta}}\right).$$

Lemma 2 gives useful bounds on $\epsilon(P, L, t)$. Our algorithm uses the more conservative estimate $\widetilde{\epsilon}(P, L, t)$, which is identical to $\epsilon(P, L, t)$ except that it automatically assigns an error of 1 to any $(v, L(v)) \notin \mathcal{A}(t)$, that is to say, any (node, label) pair for which insufficiently many samples have been seen. We need to argue that for nodes v of reasonable weight, and their majority labels $L^*(v)$, we will have $(v, L^*(v)) \in \mathcal{A}(t)$.

Lemma 3 *There is a constant c' such that $(v, l) \in \mathcal{A}(t)$ for any node v with majority label l and*

$$w_v \geq \max\left(\frac{8}{Bt} \log \frac{t^2 2^{2d_v}}{\delta}, \frac{\beta+1}{\beta-1} \cdot \frac{c'}{Bt} \log \frac{kBt^2 d_v^2}{\delta}\right).$$

The purpose of the set $\mathcal{A}(t)$ is to stop the algorithm from descending too far in the tree. We now quantify this. Suppose there is a good pruning that contains a node q whose majority label is $L^*(q)$. However, our algorithm descends far below q, to some pruning P (and associated labeling L) of T_q. By the definition of admissible pruning, this can only happen if $(q, L(q))$ lies in $\mathcal{A}(t)$. Under such circumstances, it can be proved that (P, L) is not too much worse than $(q, L^*(q))$.

Lemma 4 *For any node q, let (P, L) be the admissible pruning and labeling of T_q found by our algorithm at time t. If $(q, L(q)) \in \mathcal{A}(t)$, then $\epsilon(P, L) \leq (\beta + 1)\epsilon_q$.*

Proof sketch of Theorem 1. Let Q, t be as in the theorem statement, and let L^* denote the optimal labeling (by majority label) of each node. Define V to be the set of all nodes v with weight exceeding the bound in Lemma 3. As a result, $(v, L^*(v)) \in \mathcal{A}(t)$ for all $v \in V$.

Suppose that at time t, the learning scheme is using some pruning P with labeling L. We will decompose P and Q into three groups of nodes each: (i) $P_a \subset P$ are strict ancestors of $Q_a \subset Q$; (ii) $P_d \subset P$ are strict descendants of $Q_d \subset Q$; and (iii) the remaining nodes are common to P and Q.

Since nodes of P_a were never expanded to Q_a, we can show $w(P_a)\epsilon(P_a, L) \leq w(P_a)\epsilon(Q_a, L^*) + 2\eta/3 + w(Q_a \setminus V)$. Meanwhile, from Lemma 4 we have $w(P_d)\epsilon(P_d, L) \leq (\beta+1)w(Q_d)\epsilon(Q_d, L^*) + w(Q_d \setminus V)$. Putting it all together, we get $\epsilon(P, L) - \epsilon(Q, L^*) \leq \eta + (\beta+1)\epsilon(Q)$, under the conditions on t. ∎

3.4. Active Sampling

Suppose our current pruning and labeling are (P, L). So far we have only discussed the naive strategy of choosing query nodes $u \in P$ with probability proportional to w_u. For active learning, a more intelligent and adaptive strategy is needed. A natural choice is to pick u with probability proportional to $w_u \epsilon^{\mathrm{UB}}_{u,L(u)}(t)$, where $\epsilon^{\mathrm{UB}}_{u,l} = 1 - p^{\mathrm{LB}}_{u,l}(t)$ is an upper bound on the error associated with node u. This takes advantage of large, pure clusters: as soon as their purity becomes evident, querying is directed elsewhere.

Fallback analysis. Can the adaptive strategy perform worse than naive random sampling? There is one problematic case. Suppose there are only two labels, and that the current pruning P consists of two nodes (clusters), each with 50% probability mass; however, cluster A has impurity (minority label probability) 5% while B has impurity 50%. Under our adaptive strategy, we will query 10 times more from B than from A. But suppose B cannot be improved: any attempts to further refine it lead to subclusters which are also 50% impure. Meanwhile, it might be possible to get the error in A down to zero by splitting it further. In this case, random sampling, which weighs A equally to B, does better than the active learning scheme.

Such cases can only occur if the best pruning has high impurity, and thus active learning still yields a pruning that is not much worse than optimal. To see this, pick any good pruning Q (with optimal labeling L^*), and let's see how adaptive sampling fares with respect to Q. Suppose our scheme is currently working with a pruning P and labeling L. Divide P into two regions: $P_0 = \{p \in P : p \in T_v \text{ for some } v \in Q\}$ and $P_1 = P \setminus P_0$. The danger is that we will sample too much from P_0, where no further improvement is needed (relative to Q), and not enough from P_1. But it can be shown that *either* the active strategy samples from P_1 at least half as often as the random strategy would, *or* the current pruning is already pretty good, in that

$$\epsilon(P, L) \leq 2\epsilon(Q, L^*) + \text{terms involving sampling error.}$$

Benefits of active learning. Active sampling is sure to help when the hierarchical clustering has some large, fairly-pure clusters near the top of the tree. These will be quickly identified, and very few queries will subsequently be made in those regions. Consider an idealized example in which there are only two possible labels and each node in the tree is either pure or $(1/3, 2/3)$-impure. Specifically: (i) each node has two children, with equal probability mass; and (ii) each impure node has a pure child and an impure child. In this case, active sampling can be seen to yield a convergence rate $1/n^2$ in contrast to the $1/n$ rate of random sampling.

The example is set up so that the selected pruning P (with labeling L) always consists of pure nodes $\{a_1, a_2, \ldots, a_d\}$ (at depths $1, 2, \ldots, d$) and a single impure node b (at depth d). These nodes have weights $w_{a_i} = 2^{-i}$, $i = 1, \ldots, d$, and $w_b = 2^{-d}$; the impure node causes the error of the best pruning to be $\varepsilon = 2^{-d}/3$. The goal, then, is to sample enough from node b to cut this error in half (say, because the target error is $\varepsilon/2$). This can be achieved with a constant number of queries from node b, since this is enough to render the majority label of its pure child admissible and thus offer a superior pruning.

If we were to completely ignore the pure nodes, then the next several queries could all be made in node b; we thus halve the error with only a constant number of queries. Continuing this way leads to an exponential improvement in convergence rate. Such a policy of neglect is fine in our present example, but this would be imprudent in general: after all, the nodes we ignore may actually turn out impure, and only further sampling would reveal them as such. We instead select a node u with probability proportional to $w_u \epsilon^{\mathrm{UB}}_{u,L(u)}(t)$, and thus still select a pure node a_i with probability roughly proportional to $w_{a_i}/n_{a_i}(t)$. This allows for some cautionary exploration while still affording an improved convergence rate.

The chance of selecting the impure node b is

$$\frac{w_b \epsilon^{\mathrm{UB}}_{b,L(b)}}{w_b \epsilon^{\mathrm{UB}}_{b,L(b)} + \sum_{i=1}^{d} w_{a_i} \epsilon^{\mathrm{UB}}_{a_i,L(a_i)}} \geq \Omega\left(\frac{\varepsilon}{\varepsilon + \frac{d}{\sum_{i=1}^{d} n_{a_i}(t)}}\right).$$

The inequality follows (with high probability) because the error bound for b is always at least the true error ε (up to constants), while another argument shows that

$$\sum_{i=1}^{d} w_{a_i} \epsilon^{\mathrm{UB}}_{a_i, L(a_i)} = O\left(\sum_{i=1}^{d} \frac{w_{a_i}}{n_{a_i}(t)}\right) = O\left(\frac{d}{\sum_{i=1}^{d} n_{a_i}(t)}\right).$$

We need to argue that the pure nodes do not get queried too much. Well, if they have been queried at least $\sqrt{d/\varepsilon} = O(\sqrt{(1/\varepsilon)\log 1/\varepsilon})$ times, the chance of selecting b is $\Omega(\sqrt{\varepsilon/d})$; another $O(\sqrt{d/\varepsilon})$ queries with active sampling suffice to land a constant number in node b—just enough to cut the error in half. Overall, the number of queries needed is then $O(\sqrt{(1/\varepsilon)\log(1/\varepsilon)})$, considerably less than the $O(1/\varepsilon)$ required of random sampling.

4. Experiments

How many label queries can we save by exploiting cluster structure with active learning? Our analysis suggests that the savings is tied to *how well the cluster structure aligns with the actual labels*. To evaluate how accommodating real world data is in this sense, we studied the performance of our active learner on several natural classification tasks.

4.1. Classification Tasks

When used for classification, our active learning framework decomposes into three parts: (1) unsupervised hierarchical clustering of the unlabeled data, (2) cluster-adaptive sampling (Algorithm 1, with the second variant of **select**), and (3) supervised learning on the resulting fully labeled data. We used standard statistical procedures, Ward's average linkage clustering and logistic regression, for the unsupervised and supervised components, respectively, in order to assess just the role of the cluster-adaptive sampling method.

We compared the performance of our active learner to two baseline active learning methods, random sampling and margin-based sampling, that only train a classifier on the subset of queried labeled data. Random sampling chooses points to label at random, and margin-based sampling chooses to label the points closest to the decision boundary of the current classifier (as described in Section 2). Again, we used logistic regression with both of these methods.

A few details: We ran each active learning method 10 times for each classification task, allowing the budget of labels to grow in small increments. For each budget size, we evaluated the resulting classifier on a test set, computed its misclassification error, and averaged this error over the repeated trials. Finally, we used

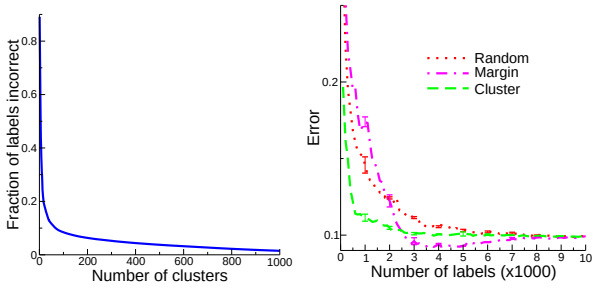


Figure 2. Results on OCR digits. Left: Errors of the best prunings in the OCR digits tree. Right: Test error curves on classification task.

ℓ_2-regularization with logistic regression, choosing the trade-off parameter with 10-fold cross validation.

OCR digit images. We first considered multi-class classification of the MNIST handwritten digit images.[1] We used 10000 training images and 2000 test images.

The tree produced by Ward's hierarchical clustering method was especially accommodating for cluster-adaptive sampling. Figure 2 (left) depicts this quantitatively; it shows the error of the best k-pruning of the tree for several values of k. For example, the tree had a pruning of 50 nodes with about 12% error. Our active learner found such a pruning using just 400 labels.

Figure 2 (right) plots the test errors of the three active learning methods on the multi-class classification task. Margin-based sampling and cluster-adaptive sampling both outperformed random sampling, with margin-based sampling taking over a little after 2000 label queries. The initial advantage of cluster-adaptive sampling reflects its ability to discover and subsequently ignore relatively pure clusters at the onset of sampling. Later on, it is left sampling from clusters of easily confused digits (e.g. 3's, 5's, and 8's).

The test error of the margin-based method appeared to actually dip below the test error of classifier trained using all of the training data (with the correct labels). This appears to be a case of fortunate sampling bias. In contrast, cluster-adaptive sampling avoids this issue by concentrating on converging to the same result as if it had all of the correct training labels.

Newsgroup text. We also considered four pairwise binary classification tasks with the 20 Newsgroups data set. Following Schohn and Cohn (2000), we chose four pairs of newsgroups that varied in difficulty. We used a version of the data set that removes duplicates and some newsgroup-identifying headers, but other-

[1] http://yann.lecun.com/exdb/mnist/

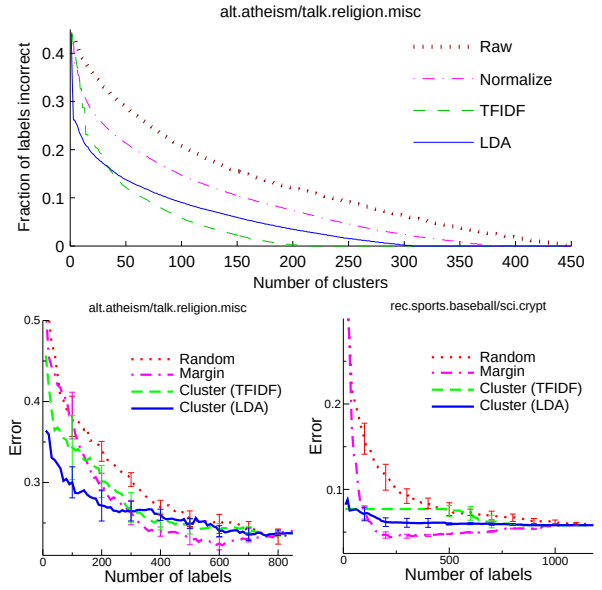


Figure 3. Results on newsgroup text. Top: Errors of the best prunings in various trees for atheism/religion pair. Bottom: Test error curves on newsgroup tasks.

wise represents each document as a simple word count vector.[2] Each newsgroup had about 1000 documents, and the data for each pair were partitioned into training and test sets at a 2:1 ratio. We length-normalized the count vectors before training the logistic regression models in order to speed up the training and improve classification performance.

The initial word count representation of the newsgroup documents yielded poor quality clusterings, so we tried various techniques for preprocessing text data before clustering with Ward's method: (1) normalize each document vector to unit length; (2) apply TF/IDF and length normalization to each document vector; and (3) infer a posterior topic mixture for each document using a Latent Dirichlet Allocation model trained on the same data (Blei et al., 2003). For the last technique, we used Kullback-Leibler divergence as the notion of distance between the topic mixture representations. Figure 3 (top) plots the errors of the best prunings. Indeed, the various changes-of-representation and specialized notions of distance help build clusterings of greater utility for cluster-adaptive active learning.

In all four pairwise tasks, both margin-based sampling and cluster-adaptive sampling outperformed random sampling. Figure 3 (bottom) shows the test errors on two of these newsgroup pairs. We observed the same effects regarding cluster-adaptive sampling and margin-based sampling as in the OCR digits data.

[2] http://people.csail.mit.edu/jrennie/20Newsgroups/

4.2. Rare Category Detection

To demonstrate its versatility, we applied our cluster-adaptive sampling method to a rare category detection task. We used the Statlog Shuttle data, a set of 43500 examples from seven different classes; the smallest class comprises a mere 0.014% of the whole. To discover at least one example from each class, random sampling needed over 8000 queries (averaged over several trials). In contrast, cluster-adaptive sampling needed just 880 queries; it sensibly avoided sampling much from clusters confidently identified as pure, and instead focused on clusters with more potential.

Acknowledgements. Support provided by the Engineering Institute at Los Alamos National Laboratory and the NSF under grants IIS-0347646 and IIS-0713540.

References

Balcan, M.-F., Beygelzimer, A., & Langford, J. (2006). Agnostic active learning. *ICML*.

Balcan, M.-F., Broder, A., & Zhang, T. (2007). Margin based active learning. *COLT*.

Blei, D., Ng, A., & Jordan, M. (2003). Latent dirichlet allocation. *JMLR*, *3*, 993–1022.

Castro, R., & Nowak, R. (2007). Minimax bounds for active learning. *COLT*.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*, 201–221.

Dasgupta, S. (2005). Coarse sample complexity bounds for active learning. *NIPS*.

Dasgupta, S., Hsu, D., & Monteleoni, C. (2007). A general agnostic active learning algorithm. *Neural Information Processing Systems*.

Freund, Y., Seung, H., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*, 133–168.

Hanneke, S. (2007). A bound on the label complexity of agnostic active learning. *ICML*.

Schohn, G., & Cohn, D. (2000). Less is more: active learning with support vector machines. *ICML*.

Schutze, H., Velipasaoglu, E., & Pedersen, J. (2006). Performance thresholding in practical text classification. *ACM International Conference on Information and Knowledge Management*.

Learning to Classify with Missing and Corrupted Features

Ofer Dekel OFERD@MICROSOFT.COM
Microsoft Research, 1 Microsoft Way, Redmond, WA 98052 USA

Ohad Shamir OHADSH@CS.HUJI.AC.IL
The Hebrew University, Jerusalem 91904, Israel

Abstract

After a classifier is trained using a machine learning algorithm and put to use in a real world system, it often faces noise which did not appear in the training data. Particularly, some subset of features may be missing or may become corrupted. We present two novel machine learning techniques that are robust to this type of classification-time noise. First, we solve an approximation to the learning problem using linear programming. We analyze the tightness of our approximation and prove statistical risk bounds for this approach. Second, we define the online-learning variant of our problem, address this variant using a modified Perceptron, and obtain a statistical learning algorithm using an online-to-batch technique. We conclude with a set of experiments that demonstrate the effectiveness of our algorithms.

1. Introduction

Supervised machine learning techniques often play a central role in solving complex real-world classification problems. First, we collect a training set of labeled examples and present this set to a machine learning algorithm. Then, the learning algorithm constructs a classifier, which can be put to use as a component in a working system. The process of collecting the training set and constructing the classifier is called the *training phase*, whereas everything that occurs after the hypothesis has been determined is called the *classification phase*. In many cases, the training phase can be performed under sterile and controlled conditions, and care can be taken to collect a high quality training set. In contrast, the classification phase often takes place in the noisy and uncertain conditions of the real world, and some of the features that were available during the training phase may be missing or corrupted. In this paper, we explore the possibility of anticipating and preparing for this type of classification-time noise.

The problem of corrupted and missing features occurs in a variety of different classification settings. For example, say that our goal is to learn an automatic medical diagnosis system. Each instance represents a patient, each feature contains the result of a medical test performed on that patient, and the purpose of the system is to detect a certain disease. When constructing the training set, we go to the trouble of carefully performing every possible test on each patient. However, when the learned classifier is eventually deployed as part of a diagnosis system, and applied to new patients, it is highly unlikely that all of the test results will be available. Technical difficulties may prevent certain tests from being performed. Different patients may have different insurance policies, each covering a different set of tests. A patient's blood sample may become contaminated, replacing the features that correspond to blood tests with random noise, while having no effect on other features. We would still like our diagnosis system to make accurate predictions. Alternatively, our goal may be to train a fingerprint recognition system that controls the lock on a door. After a few days of flawless operation, a user with greasy fingers comes along and leaves an oily smudge on the fingerprint scanner panel. From then on, all of the features measured from the area under the smudge are either distorted or cannot be extracted altogether. Ideally, the fingerprint recognition system should continue operating.

We take a worst-case approach to our problem, and assume that the set of affected features is chosen by an adversary individually per instance. More specifically, we assume that each feature is assigned an a-priori importance value and the adversary may remove or corrupt any feature subset whose total value is upper-bounded by a predefined parameter. In many natural settings, missing and damaged features are not actually chosen adversarially, but we find it beneficial to have our algorithm as robust as possible.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

We present two different learning algorithms for our problem, each with pros and cons. The first approach formulates the learning problem as a linear program (LP), in a way that closely resembles the quadratic programming formulation of the Support Vector Machine (Vapnik, 1998). However, the number of constraints in this LP grows exponentially with the number of features. Using tricks from convex analysis, we derive a related polynomial-size LP, and give conditions under which it is an exact reformulation of the original exponential-size LP. When these conditions do not hold, the polynomial-size LP still approximates the exponential-size LP, and we prove an upper bound on the approximation difference. Despite the fact that the distribution of training examples is different from the distribution of examples observed during the classification phase, we prove a statistical generalization bound for this approach.

Letting m denote the size of our training set and n the number of features, our polynomial LP formulation uses $O(mn)$ variables and $O(mn)$ sparse constraints. Depending on the dataset, this can still be rather large for off-the-shelf LP solvers. We see this as a shortcoming of our first approach, which brings us to our second algorithmic approach. We define an online learning problem, which is closely related to the original statistical learning problem. We devise a modified version of the Perceptron algorithm (Rosenblatt, 1958) for this online problem, and convert this Perceptron into a statistical learning algorithm using an online-to-batch conversion technique (Cesa-Bianchi et al., 2004). This approach benefits from the computational efficiency of the online Perceptron, and from the generalization properties and theoretical guarantees provided by the online-to-batch technique. Experimentally, we observe that the efficiency of our second approach seems to come at the price of accuracy.

Choosing an adequate regularization scheme is one of the keys to solving this problem successfully. Many existing machine learning algorithms, such as the Support Vector Machine, use L_2 regularization to promote statistical generalization. When L_2 regularization is used, the learning algorithm may put a large weight on one feature and compensate by putting a small weight on another feature. This promotes classifiers that focus their weight on the features that contribute the most. For example, in the degenerate case where one of the features actually equals the label, an L_2 regularized learning algorithm is likely to put most of its weight on that one feature. Some algorithms use L_1 regularization to further promote sparse solutions. In the context of our work, sparsity actually makes a classifier more susceptible to adversarial feature-corrupting noise. Here we prefer dense classifiers, which hedge their bets as much as possible. Both of the algorithms presented in this paper achieve this density by using a L_∞ regularization scheme. It is interesting to note that the choice of the L_∞ norm emerges as a natural one in the theoretical analysis of our first, LP-based learning approach.

1.1. Related Work

Previous papers on "noise-robust learning" mainly deal with the problem of learning with a noisy training set, a research topic which is entirely orthogonal to ours. The learning algorithms presented in (Dietterich & Bakiri, 1995) and (Gamble et al., 2007) try to be robust to general additive noise that appears at classification time, but not necessarily to feature deletion or corruption. (**?**) presents adversarial learning as a one-shot two-player game between the classifier and an adversary, and designs a robust learning algorithm from a Bayesian-learning perspective. Our approach shares the motivation of (**?**) but is otherwise significantly different. In the related field of online learning, where the training and classification phases are interlaced and cannot be distinguished, (Littlestone, 1991) proves that the Winnow algorithm can tolerate various types of noise, both adversarial and random.

Our work is most closely related to the work in (Globerson & Roweis, 2006), and its more recent enhancement in (Teo et al., 2008). Our motivation is the same as theirs, and the approaches share some similarities. Our experiments, presented in Sec. 4, suggest that our algorithms achieve considerably better performance, but we would also like to emphasize more fundamental differences between the two approaches: We allow features to have different a-priori importance levels, and we take this information into account in our algorithm and analysis. Our approach uses L_∞ regularization to promote a dense solution, where (Globerson & Roweis, 2006) uses L_2 regularization. Our second approach, which uses online-to-batch conversion techniques, is entirely novel. Finally, we prove statistical generalization bounds for our algorithms despite the change in distribution at classification time.

2. A Linear Programming Formulation

In this section, and throughout the paper, we use lowercase bold-face letters to denote vectors, and their plain-face counterparts to denote each vector's components. We also use the notation $[n]$ as shorthand for $\{1, \ldots, n\}$.

2.1. Feature Deleting Noise

We first examine the case where features are missing at classification time. Let $\mathcal{X} \subseteq \mathbb{R}^n$ be an instance space and let $\mathcal{D}$ be a probability distribution on the product space $\mathcal{X} \times \{\pm 1\}$. We receive a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ sampled i.i.d. from $\mathcal{D}$, which we use to learn our classifier. We assign each feature $j \in [n]$ a value $v_j \geq 0$. Informally, we think of v_j as the a-priori *informativeness* of fea-

ture j, or as the importance of feature j to the classification task. It can also represent the cost of obtaining the feature (such as the price of a medical test). Next, we define the value of a subset J of features as the sum of values of the features in that subset, and we denote $V(J) = \sum_{j\in J} v_j$. For instance, we frequently use $V([n])$ when referring to $\sum_{j=1}^{n} v_j$ and $V([n] \setminus J)$ when referring to $\sum_{j\notin J} v_j$. Next, we fix a noise-tolerance parameter N in $[0, V([n])]$ and define $P = V([n]) - N$. During the classification phase, instances are generated in the following way: First, a pair $(\mathbf{x}, y)$ is sampled from $\mathcal{D}$. Then, an adversary selects a subset of features $J \subset [n]$ such that $V([n] \setminus J) \leq N$, and replaces x_j with 0 for all $j \notin J$. The adversary selects J for each instance individually, and with full knowledge of the inner workings of our classifier. The noise-tolerance parameter N essentially acts as an upper bound on the amount of damage the adversary is allowed to inflict. We would like to use the training set S (which does not have missing features) to learn a binary classifier that is robust to this specific type of classification-time noise.

We focus on learning linear margin-based classifiers. A linear classifier is defined by a weight vector $\mathbf{w} \in \mathbb{R}^n$ and a bias term $b \in \mathbb{R}$. Given an instance $\mathbf{x}$, which is sampled from $\mathcal{D}$, and a set of coordinates J left intact by the adversary, the linear classifier outputs $b + \sum_{j\in J} w_j x_j$. The sign of $b + \sum_{j\in J} w_j x_j$ constitutes the actual binary prediction, while $|b + \sum_{j\in J} w_j x_j|$ is understood as the degree of confidence in that prediction. A classification mistake occurs if and only if $y(b + \sum_{j\in J} w_j x_j) \leq 0$, so we define the *risk* of the linear classifier $(\mathbf{w}, b)$ as

$$\mathcal{R}(\mathbf{w}, b) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}} \Big(\begin{array}{l} \exists J \text{ with } V([n] \setminus J) < N \\ \text{s.t. } y\big(b + \sum_{j\in J} w_j x_j\big) \leq 0 \end{array} \Big) \ . \quad (1)$$

Since $\mathcal{D}$ is unknown, we cannot explicitly minimize Eq. (1). Thus, we turn to the empirical estimate of Eq. (1), the *empirical risk*, defined as

$$\frac{1}{m}\sum_{i=1}^{m} \Big[\!\Big[\min_{J\,:\,V([n]\setminus J)\leq N} y_i\big(b + \textstyle\sum_{j\in J} w_j x_{i,j}\big) \leq 0 \Big]\!\Big], \quad (2)$$

where $[\![\pi]\!]$ denotes the indicator function of the predicate π. Minimizing the empirical risk directly constitutes a difficult combinatorial optimization problem. Instead, we formulate a linear program that closely resembles the formulation of the Support Vector Machine (Vapnik, 1998). We choose a margin parameter $\gamma > 0$ and a regularization parameter $C > 0$, and solve the problem

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \ & \tfrac{1}{m\gamma}\textstyle\sum_{i=1}^{m} \xi_i \quad (3) \\ \text{s.t. } & \forall\, i \in [m] \quad \forall J \,:\, V([n] \setminus J) \leq N \\ & \qquad y_i\big(b + \textstyle\sum_{j\in J} w_j x_{i,j}\big) \geq \tfrac{\gamma V(J)}{P} - \xi_i \ , \\ & \forall\, i \in [m] \ \ \xi_i \geq 0 \ , \quad \|\mathbf{w}\|_\infty \leq C \ . \end{aligned}$$

The objective function of Eq. (3) is called the *empirical hinge-loss* obtained on the sample S. Since ξ_i is constrained to be non-negative, each training example contributes a non-negative amount to the total loss. Moreover, the objective function of Eq. (3) upper bounds the empirical risk of $(\mathbf{w}, b)$. More specifically, for any feasible point $(\mathbf{w}, b, \boldsymbol{\xi})$ of Eq. (3), ξ_i upper bounds γ times the indicator function of the event

$$\min_{J\,:\,V([n]\setminus J)\leq N} y_i\big(b + \textstyle\sum_{j\in J} w_j x_{i,j}\big) \leq 0 \ .$$

To see this, note that for a given example $(\mathbf{x}_i, y_i)$, if there exists a feature subset J such that $V([n] \setminus J) \leq N$ and $y_i(b + \sum_{j\in J} w_j x_j) \leq 0$ then the first constraint in Eq. (3) enforces $\xi_i \geq \gamma V(J)/P$. The assumption $V([n] \setminus J) \leq N$ now implies that $V(J) \geq P$, and therefore $\xi_i \geq \gamma$. If such a set J does not exist, then the second constraint in Eq. (3) enforces $\xi_i \geq 0$.

The optimization problem above actually does more than minimize an upper bound on the empirical risk. It also requires the margin attained by the feature subset J to grow with proportion to $V(J)$. While a true adversary would always inflict the maximal possible damage, our optimization problem also prepares for the case where less damage is inflicted, requiring the confidence of our classifier to increase as less noise is introduced. We also restrict $\mathbf{w}$ to a hyper-box of radius C, which controls the complexity of the learned classifier and promotes dense solutions. Moreover, this constraint is easy to compute and makes our algorithms more efficient. Although Eq. (3) is a linear program, it is immediately noticeable that the size of its constraint set may grow exponentially with the number of features n. For example, if $v_j = 1$ for all $j \in [n]$ and if N is a positive integer, then the linear program contains over $\binom{n}{N}$ constrains per example. We deal with this problem below.

2.2. A Polynomial Approximation

Taking inspiration from (Carr & Lancia, 2000), we find an efficient approximate formulation of Eq. (3), which turns out to be an exact reformulation of Eq. (3) when $v_j \in \{0, 1\}$ for all $j \in [n]$. Specifically, we replace Eq. (3) with

$$\begin{aligned} \min \ & \tfrac{1}{m\gamma}\textstyle\sum_{i=1}^{m} \xi_i \quad (4) \\ \text{s.t. } & \forall\, i \in [m] \ \ P\lambda_i - \textstyle\sum_{j=1}^{n} \alpha_{i,j} + y_i b \geq -\xi_i \\ \forall\, i \in [m] \, \forall\, j \in [n] \ & \ y_i w_j x_{i,j} - \tfrac{\gamma v_j}{P} \geq \lambda_i v_j - \alpha_{i,j} \ , \\ \forall\, i \in [m] \, \forall\, j \in [n] \ & \ \alpha_{i,j} \geq 0 \ , \\ \forall\, i \in [m] \ & \ \lambda_i \geq 0 \text{ and } \xi_i \geq 0 \ , \\ & \ \|\mathbf{w}\|_\infty \leq C \ , \end{aligned}$$

where the minimization is over $\mathbf{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$, $\boldsymbol{\xi} \in \mathbb{R}^m$, $\boldsymbol{\lambda} \in \mathbb{R}^m$, and $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m$, each in $\mathbb{R}^n$. The number of

variables and the number of constraints in this problem are both $O(mn)$. The following theorem explicitly relates the optimization problem in Eq. (4) with the one in Eq. (3).

Theorem 1. *If $(\mathbf{w}^\star, b^\star, \boldsymbol{\xi}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\alpha}_1^\star, \ldots, \boldsymbol{\alpha}_m^\star)$ is an optimal solution to Eq. (4), then $(\mathbf{w}^\star, b^\star, \boldsymbol{\xi}^\star)$ is a feasible point of Eq. (3), and therefore the value of Eq. (4) upper-bounds the value of Eq. (3). Moreover, if $v_j \in \{0,1\}$ for all $j \in [n]$, then $(\mathbf{w}^\star, b^\star, \boldsymbol{\xi}^\star)$ is an optimal solution to Eq. (3). Finally, if it does not hold that $v_j \in \{0,1\}$ for all $j \in [n]$, and assuming $\|\mathbf{x}_i\| \leq 1$ for all i, then the difference between the value of Eq. (4) and the value of Eq. (3) is at most C/γ.*

As a first step towards proving Thm. 1, we momentarily forget about the optimization problem at hand and focus on another question: given a specific triplet $(\mathbf{w}, b, \boldsymbol{\xi})$, is it a feasible point of Eq. (3) or not? More concretely, for each training example $(\mathbf{x}_i, y_i)$, we would like to determine if for all J with $V([n] \setminus J) \leq N$ it holds that

$$y_i\big(b + \textstyle\sum_{j\in J} w_j x_{i,j}\big) \;\geq\; \frac{\gamma V(J)}{P} - \xi_i \;\;. \tag{5}$$

We can answer this question by comparing $-\xi_i$ with the value of the following integer program:

$$\begin{array}{ll} \min\limits_{\boldsymbol{\tau}\in\{0,1\}^n} & y_i b + \sum_{j=1}^n \tau_j \left(y_i w_j x_{i,j} - \frac{\gamma v_j}{P}\right) \\ \text{s.t.} & P \;\leq\; \sum_{j=1}^n \tau_j v_j \;\;. \end{array} \tag{6}$$

For example, if the value of this integer program is less than $-\xi_i$, then let $\boldsymbol{\tau}'$ be an optimal solution and we have that $y_i(b + \sum_{j=1}^n \tau'_j w_j x_{i,j}) < (\gamma \sum_{j=1}^n \tau'_j v_j)/P - \xi_i$. Namely, the set $J = \{j \in [n] : \tau'_j = 1\}$ violates Eq. (5). On the other hand, if there exists some J with $V([n] \setminus J) \leq N$ that violates Eq. (5) then its indicator vector is a feasible point of Eq. (6) whose objective value is less than $-\xi_i$.

Directly solving the integer program in Eq. (6) may be difficult, so instead we examine the properties of the following linear relaxation:

$$\begin{array}{ll} \min\limits_{\boldsymbol{\tau}} & y_i b + \sum_{j=1}^n \tau_j \left(y_i w_j x_{i,j} - \frac{\gamma v_j}{P}\right) \\ \text{s.t.} & \forall j \in [n] \;\; 0 \leq \tau_j \leq 1 \;\; \text{and} \;\; P \leq \sum_{j=1}^n \tau_j v_j \;\;. \end{array} \tag{7}$$

To analyze this relaxation we require the following lemma.

Lemma 1. *Fix an example $(\mathbf{x}_i, y_i)$, a linear classifier $(\mathbf{w}, b)$, and a scalar $\xi_i > 0$, and let θ be the value of Eq. (7) with respect to these choices. (a) If $\theta \geq -\xi_i$ then Eq. (5) holds. (b) In the special case where $v_j \in \{0,1\}$ for all $j \in [n]$ and where N is an integer, $\theta \geq -\xi_i$ if and only if Eq. (5) holds. (c) There exists a minimizer of Eq. (7) with at most one coordinate in $(0,1)$.*

The proof of the lemma is straightforward but technical, and is omitted due to lack of space. Lemma 1 tells us that comparing the value of the linear program in Eq. (7) with $-\xi_i$ provides a sufficient condition for Eq. (5) to hold for the example $(\mathbf{x}_i, y_i)$. Moreover, this condition becomes both sufficient and necessary in the special case where $v_j \in \{0,1\}$ for all $j \in [n]$. We now proceed with proving the first part of Thm. 1 using claim (a) in Lemma 1. The remaining parts of the theorem follow similarly from claims (b) and (c) in the lemma.

Proof of Theorem 1. Let $(\mathbf{w}^\star, b^\star, \boldsymbol{\xi}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\alpha}_1^\star, \ldots, \boldsymbol{\alpha}_m^\star)$ be an optimal solution to the linear program in Eq. (4). Specifically, it holds for all $i \in [m]$ that $\boldsymbol{\alpha}_i^\star$ and $\lambda_i^\star$ are non-negative, that $P\lambda_i^\star - \sum_{j=1}^n \alpha_{i,j}^\star + y_i b^\star \geq -\xi_i^\star$, and that

$$\forall\, j \in [n] \;\; y_i w_j^\star x_{i,j} - \frac{\gamma v_j}{P} \;\geq\; \lambda_i^\star v_j - \alpha_{i,j}^\star \;\;.$$

Therefore, it also holds that the value of the following optimization problem

$$\begin{array}{ll} \max\limits_{\boldsymbol{\alpha}_i, \lambda_i} & P\lambda_i \;-\; \sum_{j=1}^n \alpha_{i,j} + y_i b^\star \\ \text{s.t.} & \forall\, j \in [n] \;\; y_i w_j^\star x_{i,j} - \frac{\gamma v_j}{P} \geq \lambda_i v_j - \alpha_{i,j} \;, \\ & \forall\, j \in [n] \;\; \alpha_{i,j} \geq 0 \;\text{ and }\; \lambda_i \geq 0 \;\;, \end{array} \tag{8}$$

is at least $-\xi_i^\star$. The strong duality principle of linear programming (Boyd & Vandenberghe, 2004) states that the value of Eq. (8) equals the value of its dual optimization problem, which is:

$$\begin{array}{ll} \min\limits_{\boldsymbol{\tau}} & y_i b^\star + \sum_{j=1}^n \tau_j \left(y_i w_j^\star x_{i,j} - \frac{\gamma v_j}{P}\right) \\ \text{s.t.} & \forall\, j \in [n] \;\; 0 \leq \tau_j \leq 1 \quad \text{and} \quad P \;\leq\; \sum_{j=1}^n \tau_j v_j \;\;. \end{array} \tag{9}$$

In other words, the value of Eq. (9) is also at least $-\xi_i^\star$. Using claim (a) of Lemma 1, we have that

$$y_i\big(b^\star + \textstyle\sum_{j\in J} w_j^\star x_{i,j}\big) \;\geq\; \frac{\gamma V(J)}{P} - \xi_i^\star \;\;,$$

holds for all J with $V([n] \setminus J) \leq N$. The optimization problem in Eq. (4) also constrains $\|\mathbf{w}\|_\infty \leq C$ and $\xi_i \geq 0$ for all $i \in [m]$, thus, $(\mathbf{w}^\star, b^\star, \boldsymbol{\xi}^\star)$ satisfies the constraints in Eq. (3). Since Eq. (3) and Eq. (4) have the same objective function, the value of Eq. (3) is upper bounded by the value of Eq. (4). □

2.3. Generalization Bounds

We now prove a generalization bound on the risk of the classifier learned in our framework, using PAC-Bayesian techniques (McAllester, 2003). Throughout, we assume that $\|\mathbf{x}\|_\infty \leq 1$ with probability 1 over $\mathcal{D}$. For simplicity, we assume that the bias term b is 0, and that $v_j > 0$ for all j. These assumptions can be relaxed at the cost of a somewhat more complicated analysis. Given a classifier $\mathbf{w}$, let $\ell_\gamma(\mathbf{w}, \mathbf{x}, y)$ denote the γ-loss attained on the example $(\mathbf{x}, y)$, defined as

$$\Big[\!\Big[\min_{J\,:\,V([n]\setminus J)\leq N} y \sum_{j\in J} w_j x_j \;<\; \frac{\gamma V(J)}{P} \Big]\!\Big] \;\;, \tag{10}$$

where $[\![\cdot]\!]$ again denotes the indicator function. Note that $\mathbb{E}[\ell_0(\mathbf{w}, \mathbf{x}, y)] = \mathcal{R}(\mathbf{w}, 0)$, where $\mathcal{R}$ is defined in Eq. (1).

Theorem 2. *Let S be a sample of size m drawn i.i.d from $\mathcal{D}$. For any $\delta > 0$, with probability at least $1 - \delta$, it holds for all $\mathbf{w} \in \mathbb{R}^n$ with $\|\mathbf{w}\|_\infty \leq C$ that the risk associated with $\mathbf{w}$ is at most*

$$\sup \left\{ \epsilon \ : \ \mathrm{KL}\left(\frac{1}{m} \sum_{i=1}^m \ell_\gamma(\mathbf{w}, \mathbf{x}_i, y_i) \ \middle\| \ \epsilon \right) \leq \frac{\beta(m,\delta,\gamma)}{m-1} \right\},$$

where $\beta(m, \delta, \gamma) = \ln(m/\delta) + \sum_{j=1}^n \ln(4PC/(\gamma v_j))$ and KL *is the Kullback-Leibler divergence. The above is upper-bounded by the empirical γ-loss (which equals $\frac{1}{m} \sum_{i=1}^m \ell_\gamma(\mathbf{w}, \mathbf{x}_i, y_i)$), plus the additional term*

$$\sqrt{\frac{2}{m} \sum_{i=1}^m \ell_\gamma(\mathbf{w}, \mathbf{x}_i, y_i) \frac{\beta(m,\delta,\gamma)}{m-1}} + \frac{2\beta(m, \delta, \gamma)}{m-1} \ .$$

Proof sketch. The proof follows along similar lines to the PAC-Bayesian bound for linear classifiers in (McAllester, 2003). First, define the axis-aligned box $B = \prod_{j=1}^n [w_j - \frac{\gamma v_j}{2P}, w_j + \frac{\gamma v_j}{2P}] \cap [-C, C]$. We use the margin concept to upper bound $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell_0(\mathbf{w}, \mathbf{x}, y)]$ by the expected $\ell_{\gamma/2}$ loss over $\mathcal{D}$ of a classifier sampled uniformly from $B \cap [-C, C]^n$. We can upper bound this expected loss using the PAC-Bayesian theorem (McAllester, 2003), where the uniform distribution over $B \cap [-C, C]^n$ is the posterior classifier distribution, and the uniform distribution over $[-C, C]^n$ is the prior. The bound we get is defined in terms of the average empirical $\ell_{\gamma/2}$ loss of a random classifier from B, plus a complexity term dependent on the volume ratio between B and $[-C, C]^n$. Finally, this average loss can be upper bounded by the empirical ℓ_γ loss of $\mathbf{w}$ by repeating the technique of the first stage. The weaker bound stated in the theorem follows from a lower bound on the KL divergence, presented in (McAllester, 2003). □

It is interesting to note that L_∞ regularization emerges as the most natural one in this setting, since it induces the most convenient type of margin for relating the $\ell_0, \ell_{\gamma/2}, \ell_\gamma$ loss functions as described above. This lends theoretical support to our choice of the L_∞ norm in our algorithms.

2.4. Feature Corrupting Noise

We now shift our attention to the case where a subset of the features is corrupted with random noise, and show that the the same LP approach used to handle missing features can also deal with corrupted features if the margin parameter γ in Eq. (4) is sufficiently large. For simplicity, we shall assume that all features are supported on $[-1, 1]$ with zero mean. Unlike the feature deleting noise, we now assume that the each feature selected by the adversary is replaced with noise sampled from some distribution, also supported on $[-1, 1]$ and having zero mean. The following theorem relates the risk of a classifier in the above setting, to its expected γ-loss in the feature deletion setting, where the latter can be bounded with Thm. 2.

Theorem 3. *Let ϵ, C, and N be arbitrary positives, and let γ be at least $C\sqrt{N \ln(1/\epsilon)/2}$. Assume that we solve Eq. (4) with parameters γ, C, N and with $v_j = 1$ for all $j \in [n]$. Let $\mathbf{w}$ be the resulting linear classifier, and assume for simplicity that the bias term b is zero. Let f be a random vector-valued function on $\mathcal{X}$, such that for every $\mathbf{x} \in \mathcal{X}$, $f(\mathbf{x})$ is the instance $\mathbf{x}$ after the feature corruption scheme described above. Then, using ℓ_γ as defined in Eq. (10), for $(\mathbf{x}, y)$ drawn randomly from $\mathcal{D}$, we have:*

$$\Pr\left(y\langle \mathbf{w}, f(\mathbf{x})\rangle \leq 0\right) \ \leq \ \mathbb{E}\left[\ell_\gamma(\mathbf{w}, \mathbf{x}, y)\right] + \epsilon \ .$$

Proof. Let $(\mathbf{x}, y)$ be an example and let J denote the feature subset which remains uncorrupted by the adversary. Using Hoeffding's bound and our assumption on γ, we have that $\Pr\left(y \sum_{j\notin J} w_j f_j(\mathbf{x}) \leq -\gamma\right)$ is upper bounded by ϵ. Therefore, with probability at least $1 - \epsilon$ over the randomness of f, $y\langle \mathbf{w}, f(\mathbf{x})\rangle$ is equal to:

$$y \sum_{j\in J} w_j x_j + y \sum_{j\notin J} w_j f_j(\mathbf{x}) \ > \ y \sum_{j\in J} w_j x_j - \gamma \ . \quad (11)$$

Thus, with probability at least $1 - \epsilon$, $\Pr(y\langle \mathbf{w}, f(\mathbf{x})\rangle < 0)$ is upper bounded by $\mathbb{E}[\ell_\gamma(\mathbf{w}, \mathbf{x}, y)]$. Otherwise, with probability at most ϵ, $\Pr(y\langle \mathbf{w}, f(\mathbf{x})\rangle < 0) \leq 1$. □

We conclude with an interesting observation. In the feature corruption setting, making a correct prediction boils down to achieving a sufficiently large margin on the uncorrupted features. Let $r \in (0, 1)$ be a fixed ratio between N and n, and let n grow to infinity. Assuming a reasonable degree of feature redundancy, the term $y \sum_{j\in J} w_j x_j$ grows as $\Theta(n)$. On the other hand, Hoeffding's bound tells us that $y \sum_{j\notin J} w_j x_j$ grows only as $O(\sqrt{N})$. Therefore, for r arbitrarily close to 1 and a large enough n, the first sum in Eq. (11) dominates the second. Namely, by setting $\gamma = \Omega(\sqrt{N})$ in Eq. (4), our ability to withstand feature corruption matches our ability to withstand feature deletion.

3. Solving the Problem with the Perceptron

We now turn to our second learning algorithm, taking a different angle on the problem. We momentarily forget about the original statistical learning problem and instead define a related online prediction problem. In online learning there is no distinction between the training phase and the classification phase, so we cannot perfectly replicate the classification-time noise scenario discussed above. Instead, we assume that an adversary removes features from every instance that is presented to the algorithm. We address this online problem with a modified version of the

Perceptron algorithm (Rosenblatt, 1958) and use an online-to-batch conversion technique to convert the online algorithm back into a statistical learning algorithm. The detour through online learning gives us efficiency while the online-to-batch technique provides us with the statistical generalization properties we are interested in.

3.1. Perceptron with Projections onto the Cube

We start with a modified version of the well-known Perceptron algorithm (Rosenblatt, 1958), which observes a sequence of examples $\big((\mathbf{x}_i, y_i)\big)_{i=1}^m$, one example at a time, and incrementally builds a sequence $\big((\mathbf{w}_i, b_i)\big)_{i=1}^m$ of linear margin-based classifiers, while constraining them to a hyper-cube. Before processing example i, the algorithm has the vector $\mathbf{w}_i$ and the bias term b_i stored in its memory. An adversary takes the instance $\mathbf{x}_i$ and reveals only a subset J_i of its features to the algorithm, attempting to cause the online algorithm to make a prediction mistake. In choosing J_i, the adversary is restricted by the constraint $V([n] \setminus J) \leq N$. Next, the algorithm predicts the label associated with $\mathbf{x}_i$ to be

$$\text{sign}\left(b_i + \textstyle\sum_{j \in J_i} w_{i,j} x_{i,j}\right) \ .$$

After the prediction is made, the correct label y_i is revealed and the algorithms suffers a hinge-loss $\xi(\mathbf{w}, b, \mathbf{x}, y)$, defined as

$$\left[\max_{J\,:\,V([n]\setminus J)\leq N} \frac{\gamma V(J)}{P} \ - \ y\big(b + \textstyle\sum_{j\in J} w_j x_j\big)\right]_+ \ , \quad (12)$$

where $P = V([n]) - N$ and $[\alpha]_+$ denotes the hinge function, $\max\{\alpha, 0\}$. Note that $\xi(\mathbf{w}_i, b_i, \mathbf{x}_i, y_i)$ upper-bounds γ times the indicator of a prediction mistake on the current example, for any choice of J_i made by the adversary. We choose to denote the loss by ξ to emphasize the close relation between $\xi(\mathbf{w}_i, b_i, \mathbf{x}_i, y_i)$ and ξ_i in Eq. (3). Due to our choice of loss function, we can assume that the adversary chooses the subset J_i that inflicts the greatest loss.

The algorithm now uses the correct label y_i to construct the pair $(\mathbf{w}_{i+1}, b_{i+1})$, which is used to make the next prediction. If $\xi(\mathbf{w}, b, \mathbf{x}, y) = 0$, the algorithm defines $\mathbf{w}_{i+1} = \mathbf{w}_i$ and $b_{i+1} = b_i$. Otherwise, the algorithm defines $\mathbf{w}_{i+1}$ using the following coordinate-wise update

$$j \in [n] \quad w_{i+1,j} \ = \ \begin{cases} [w_{i,j} + y_i \tau x_{i,j}]_{\pm C} & \text{if } j \in J_i \\ w_{i,j} & \text{otherwise} \end{cases} \ ,$$

and $b_{i+1} = [b_i + y_i \tau]_{\pm C}$, where $\tau = \frac{\sqrt{n+1}C}{\sqrt{2m}}$ and $[\alpha]_{\pm C}$ abbreviates the function $\max\big\{\min\{\alpha, C\}, -C\big\}$. This update is nothing more than the standard Perceptron update with constant learning rate τ, with an added projection step onto the hyper-cube of radius C. The specific value of τ used above is the value that optimizes the cumulative loss bound below. As in the previous section, restricting the online classifier to the hyper-cube helps us control its complexity, while promoting dense classifiers. It also comes in handy in the next stage, when we convert the online algorithm into a statistical learning algorithm.

Using a rather straightforward adaptation of standard Perceptron loss bounds, to the case where the hypothesis is confined to the hyper-cube, leads us to the following theorem, which compares the cumulative loss suffered by the algorithm with the cumulative loss suffered by any fixed hypothesis in the hyper-cube of radius C.

Theorem 4. *Choose any $C > 0$ and let $\mathbf{w}^\star \in \mathbb{R}^n$ and $b^\star \in \mathbb{R}$ be such that $\|\mathbf{w}^\star\|_\infty \leq C$ and $|b^\star| \leq C$. Let $\big((\mathbf{x}_i, y_i)\big)_{i=1}^m$ be an arbitrary sequence of examples, with $\|\mathbf{x}_i\|_1 \leq 1$ for all i. Assume that this sequence is presented to our modified Perceptron, and let $\xi(\mathbf{w}_i, b_i, \mathbf{x}_i, y_i)$ be as defined in Eq. (12). Then it holds that $\frac{1}{\gamma m}\sum_{i=1}^m \xi(\mathbf{w}_i, b_i, \mathbf{x}_i, y_i)$ is upper-bounded by*

$$\frac{1}{\gamma m}\sum_{i=1}^m \xi(\mathbf{w}^\star, b^\star, \mathbf{x}_i, y_i) \ + \ \frac{C}{\gamma}\sqrt{\frac{2(n+1)}{m}} \ .$$

The next step is to convert our online algorithm into a statistical learning algorithm.

3.2. Converting Online to Batch

To obtain a statistical learning algorithm, with risk guarantees, we assume that the sequence of examples presented to the modified Perceptron algorithm is a training set sampled i.i.d. from the underlying distribution $\mathcal{D}$. We turn to the simple averaging technique presented in (Cesa-Bianchi et al., 2004) and define $\bar{\mathbf{w}} = \frac{1}{m}\sum_{i=1}^m \mathbf{w}_{i-1}$ and $\bar{b} = \frac{1}{m}\sum_{i=1}^m b_{i-1}$. $(\bar{\mathbf{w}}, \bar{b})$ is called the *average hypothesis*, and defines our robust classifier. We use the derivation in (Cesa-Bianchi et al., 2004) to prove that the average classifier provides an adequate solution to our original problem.

Note that the loss function we use, defined in Eq. (12), is bounded and convex in its first two arguments. This allows us to apply (Cesa-Bianchi et al., 2004, Corollary 2) to relate the risk of $(\bar{\mathbf{w}}, \bar{b})$ with the cumulative online loss suffered by the Perceptron. It also allows us to apply Hoeffding's bound to relate the expected loss of any fixed classifier $(\mathbf{w}^\star, b^\star)$ with its empirical loss on the training set. Combining both bounds results in the following corollary.

Corollary 1. *For any $\delta > 0$, with probability at least $1 - \delta$ over the random sampling of S, our algorithm constructs $(\bar{\mathbf{w}}, \bar{b})$ such that $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\xi(\bar{\mathbf{w}}, \bar{b}, \mathbf{x}, y)\right]$ is at most*

$$\min_{(\mathbf{w},b)\in\mathcal{H}} \mathbb{E}\left[\xi(\mathbf{w}, b, \mathbf{x}, y)\right] \ + (3C + \phi)\sqrt{\frac{2(n + 1 + \ln(\frac{2}{\delta}))}{m}} \ ,$$

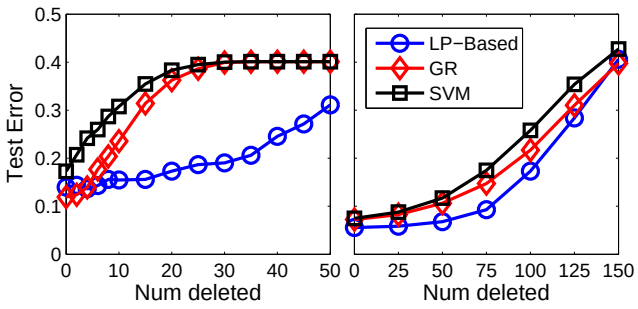


Figure 1. A comparison of our LP-based approach with the algorithm of (Globerson & Roweis, 2006) (GR) and with SVM on SPAM (left) and MNIST (right), with random noise.

where $\phi = \gamma \max_{J:V([n]\setminus J)\leq N} \big(V(J)/P\big)$*, and* $\mathcal{H}$ *is the set of all pairs* $(\mathbf{w}, b)$ *such that* $\|\mathbf{w}\|_\infty \leq C$ *and* $|b| \leq C$*.*

Using the fact that the hinge loss upper-bounds γ times the indicator function of a prediction mistake, regardless of the adversary's choice of the feature set, we have that the expected hinge loss upper-bounds $\gamma\mathcal{R}(\bar{\mathbf{w}}, \bar{b})$.

4. Experiments and Conclusions

We compare the performance of our two algorithms (LP-based and online-to-batch) with that of a linear L_2 SVM (Joachims, 1998) and with the results reported in (Globerson & Roweis, 2006). We used the GLPK package (`http://www.gnu.org/software/glpk`) to solve the LP formulation of our LP-based algorithm.

We begin with a highly illustrative sanity check. We generated a synthetic dataset of 1000 linearly separable instances in $\mathbb{R}^{20}$ and added label noise by flipping each label with probability 0.2. Then, we added two copies of the actual label as additional features to each instance, for a total of 22 features. We randomly split the data into equally sized training and test sets, and trained an SVM classifier on the training set. We set $v_j = 1$ for $j \in [20]$ and $v_{21} = v_{22} = 10$, expressing our prior knowledge that the last two features are more valuable. Using these feature values, we applied our technique with different values of the parameter N. We removed one or both of the high-value features from the test set and evaluated the classifiers. With only one feature removed both SVM and our approach attained a test error of zero. With two features removed, the test error of the SVM classifier jumped to 0.477 ± 0.004 (over 100 random repetitions of the experiment), indicating that it essentially put all of its weight on the two perfect features. With the noise parameter set to $N = 20$, our approach attained a test error of only 0.22 ± 0.002. This is only marginally above the best possible error rate for this setting.

Following the lead of (Globerson & Roweis, 2006), we conducted experiments using the SPAM and MNIST datasets. The SPAM dataset, taken from the UCI repository, is a collection of spam and non-spam e-mails. Spam can be detected by different word combinations, so we expect considerable feature redundancy in this dataset. The MNIST dataset is a collection of pixel-maps of handwritten digits. Again, following (Globerson & Roweis, 2006), we focused on the binary problem of distinguishing the digit 4 from the digit 7. Adjacent pixels often contain redundant information, making MNIST well-suited for our needs.

On each dataset, we performed 2 types of experiments. The first type follows exactly the protocol used in (Globerson & Roweis, 2006). Namely, the algorithm is trained with a small training set of 50 instances, and its performance is tested in the face of *random* feature-deleting noise, which uniformly deletes N non-zero features from each test instance, for various choices of N. Notice that this setting deviates from the adversarial setting considered so far, and the reason for conducting this experiment is to compare our results to those reported in (Globerson & Roweis, 2006). A validation set is used for parameter tuning. We did not test our online-to-batch algorithm within this setting, since it has little advantage with such a small training set. The results are presented in Fig. 1, and show test error as a function of the number of deleted features. Compared to its competitors, our algorithm has a clear and substantial advantage.

The second type of experiment simulates more closely the adversarial setting discussed throughout the paper. Using 10-fold cross-validation, we corrupted each test instance using a greedy adversary, which deletes the most valuable features of each instance until either the limit N is reached or all useful features are deleted. $1/9$ of the training set was used for parameter tuning. Due to computational considerations when running our LP-based algorithm, we performed a variant of bagging by randomly splitting the training set into chunks, training on each chunk individually, and finally averaging the resulting weight vectors. In contrast, our online-to-batch algorithm trained on the entire training set at once, and so did the SVM algorithm. We repeated this process for different values of N. For the SPAM dataset, we repeated this entire experiment twice, once with features values v_j set uniformly to 1, and once with v_j set using a mutual information heuristic. Formally, we set

$$v_j = \tfrac{1}{Z} \max_{c\in\mathbb{R}} I\big([\![x_j > c]\!]; y\big) \ ,$$

where Z is such that $\sum v_j = n$, and where $I([\![x_j > c]\!]; y)$ is the mutual information between the predicate $[\![x_j > c]\!]$ and the label y, over all examples in the training set. Intuitively, we are calculating the amount of information contained in each individual feature on the label, provided that

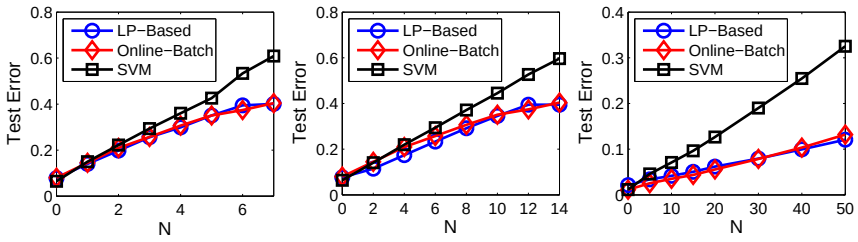


Figure 2. Experiments on SPAM with $\forall j \in J, v_j = 1$ (left) and with v_j set with a mutual information heuristic (center). Experiments on MNIST with v_j set with a mutual information heuristic (right).

we are looking only at linear threshold functions. When experimenting with the MNIST dataset, we only used the values of v_j set by our heuristic. This is a natural choice since the features of MNIST are of markedly different importance levels. For example, the corner pixels, which are always zero, are completely uninformative, while other pixels may be very informative. The results are presented in Fig. 2, and show test error as a function of N. Clearly, our algorithms have the advantage. SVM repeatedly puts all of its eggs in a small number of baskets, and is severely punished for this, while our technique anticipates the actions of the adversary and hedges its bets accordingly.

Moreover, the results in Fig. 2 demonstrate the tradeoffs between our LP-based and online-to-batch algorithms. Although we have handicapped the LP-based algorithm by chunking the training set, its performance is comparable and sometimes superior to that of the online-to-batch algorithm. With less or without chunking, we expect its performance to be even better.

We conclude that our proposed algorithms successfully withstand feature corruption at classification time, and considerably improve upon the current state of the art. On a more general note, this work has interesting connections to a recent trend in machine learning research, which is to develop sparse classifiers supported on a small subset of the features. In our setting, we are interested in the exact opposite, and the efficacy of using the L_∞ norm is clearly demonstrated here. The trade-off between robustness and sparsity provides fertile ground for future research.

References

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Carr, R. D., & Lancia, G. (2000). Compact vs. exponential-size LP relaxations SANDIA Report 2000-2170.

Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2004). On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, *50*, 2050–2057.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*, 263–286.

Gamble, E., Macskassy, S., & Minton, S. (2007). Classification with pedigree and its applicability to record linkage. *Workshop on Text-Mining & Link-Analysis*.

Globerson, A., & Roweis, S. (2006). Nightmare at test time: robust learning by feature deletion. *Proceedings of ICML 23* (pp. 353–360).

Joachims, T. (1998). Making large-scale support vector machine learning practical. In *Advances in kernel methods - support vector learning*. MIT Press.

Littlestone, N. (1991). Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. *Proceedings of the COLT 4* (pp. 147–156).

McAllester, D. A. (2003). Simplified PAC-bayesian margin bounds. *Proceedings of COLT 16* (pp. 203–215).

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–407.

Teo, C.-H., Globerson, A., Roweis, S., & Smola, A. (2008). Convex learning with invariances. *Advances in NIPS 21*.

Vapnik, V. N. (1998). *Statistical learning theory*. Wiley.

Maximum Likelihood Rule Ensembles

Krzysztof Dembczyński KDEMBCZYNSKI@CS.PUT.POZNAN.PL
Institute of Computing Science, Poznań University of Technology, Poznań, 60-965, Poland

Wojciech Kotłowski WKOTLOWSKI@CS.PUT.POZNAN.PL
Institute of Computing Science, Poznań University of Technology, Poznań, 60-965, Poland

Roman Słowiński RSLOWINSKI@CS.PUT.POZNAN.PL
Institute of Computing Science, Poznań University of Technology, Poznań, 60-965, Poland
Systems Research Institute, Polish Academy of Sciences, Warsaw, 01-447, Poland

Abstract

We propose a new rule induction algorithm for solving classification problems via probability estimation. The main advantage of decision rules is their simplicity and good interpretability. While the early approaches to rule induction were based on sequential covering, we follow an approach in which a single decision rule is treated as a base classifier in an ensemble. The ensemble is built by greedily minimizing the negative loglikelihood which results in estimating the class conditional probability distribution. The introduced approach is compared with other decision rule induction algorithms such as SLIPPER, LRI and RuleFit.

1. Introduction

Decision rule is a logical statement of the form: "if *condition* then *response*". It can be treated as a simple classifier that gives a constant response for the objects satisfying the condition part, and abstains from the response for all the other objects. Induction of decision rules has been widely considered in the early machine learning approaches (Michalski, 1983; Cohen, 1995; Fürnkranz, 1996), and rough set approaches to knowledge discovery (Stefanowski, 1998). The most popular algorithms were based on a sequential covering procedure (also known as separate-and-conquer approach). In this technique, a rule is learned which covers a part of the training examples, then examples are removed from the training set and the process is repeated until no examples remain.

Although it seems that decision (classification) trees are much more popular in data mining and machine learning applications, recently we are able to observe again a growing interest in decision rule models. As an example, let us mention such algorithms as RuleFit (Friedman & Popescu, 2005), SLIPPER (Cohen & Singer, 1999), Lightweight Rule Induction (LRI) (Weiss & Indurkhya, 2000). All these algorithms follow a specific iterative approach to decision rule generation by treating each decision rule as a subsidiary base classifier in the ensemble. This approach can be seen as a generalization of the sequential covering, because it approximates the solution of the prediction task by sequentially adding new rules to the ensemble without adjusting those that have already been added (RuleFit is an exception since it generates the trees first and then transforms them to rules). Each rule is fitted by concentrating on objects which were hardest to classify correctly by rules already present in the ensemble. All these algorithms can be explained within the framework of boosting (Freund & Schapire, 1997; Mason et al., 1999; Friedman et al., 2000) or forward stagewise additive modeling (FSAM) (Hastie et al., 2003), a greedy procedure for minimizing a loss function on the dataset.

The algorithm proposed in this paper, Maximum Likelihood Rule Ensembles (MLRules), benefits from the achievements in boosting machines (Freund & Schapire, 1997; Mason et al., 1999; Friedman et al., 2000; Friedman, 2001). Its main idea consists in rule induction by greedily minimizing the negative loglikelihood (also known as logit loss in binary classification case) to estimate the conditional class probability distribution. Minimization of such loss function with a

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

tree being a base classifier has already been used in LogitBoost (Friedman et al., 2000) and MART (Friedman, 2001), however, here we show a modified procedure, adapted to the case when the decision rule is a base classifier in the ensemble. In contrary to RuleFit (where trees are generated first), rules are generated directly; in contrary to SLIPPER and LRI, negative loglikelihood loss is used. Moreover, our approach is distinguished from other approaches to rule induction by the fact of estimating the class probability distribution instead of single classification and by using the same single measure (value of the negative loglikelihood) at all stages of the learning procedure: setting the best cuts (conditions), stopping the rule's growth and determining the response (weight) of the rule. We derive the algorithm for two optimization techniques, depending on whether we expand the loss function to the first order (fitting to the gradient) or to the second order (Newton steps). We report experiments showing the performance of MLRules and comparing them with the competitive rule ensemble methods.

The paper is organized as follows. In Section 2, the problems of classification is described. Section 3 presents a framework for learning rule ensembles. Section 4 is devoted to the problem of a single rule generation. In Section 5 we discuss the issue of convergence of the method, and we propose a modification to the main algorithm. Section 6 contains experimental results. The last section concludes the paper and outlines further research directions.

2. Problem Statement

In the classification problem, the aim is to predict the unknown class label $y \in \{1, \ldots, K\}$ of an object using known values of the attributes $\mathbf{x} = (x_1, x_2, \ldots, x_m)$. This is done by constructing a classification function $f(\mathbf{x})$ that predicts accurately the value of y. The accuracy of a single prediction is measured in terms of the loss function $L(y, f(\mathbf{x}))$, while the overall accuracy of the function $f(\mathbf{x})$ is measured by the expected loss (risk) over the data distribution $P(\mathbf{x}, y)$:

$$R(f) = \mathbb{E}[L(y, f(\mathbf{x}))].$$

Since $P(\mathbf{x}, y)$ is unknown, the risk-minimizing function (Bayes classifier), $f^* = \arg\min_f \mathbb{E}[L(y, f(\mathbf{x}))]$, is also unknown. The learning procedure uses only a set of training examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ to construct f to be a good approximation of f^*. Usually, it is performed by minimization of the empirical risk:

$$R_{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)), \tag{1}$$

where function f is chosen from a restricted family of functions. The most commonly used loss function is 0-1 loss, $L_{0\text{-}1}(y, f(\mathbf{x})) = 1 - \delta_{y,f(\mathbf{x})}$, where $\delta_{ij} = 1$ if $i = j$, otherwise $\delta_{ij} = 0$. If the correct class is predicted, classification function is not penalized, otherwise the unit penalty is imposed. Bayes classifier has the following form:

$$f^*(\mathbf{x}) = \arg \min_{k \in \{1,\ldots,K\}} \Pr(y = k|\mathbf{x}). \tag{2}$$

The 0-1 loss has several drawbacks. Firstly, if we introduce unequal costs of misclassification, $f^*(\mathbf{x})$ does not longer have the form (2). Moreover, 0-1 loss is insensitive to the "confidence" of prediction: minimization of 0-1 loss results only in finding the most probable class, without estimating its probability. On the contrary, probability estimation provides us with the conditional class distribution $P(y|\mathbf{x})$, by which we can measure the prediction confidence. Moreover, all we need to obtain the Bayes classifier for *any* loss function is the conditional probability distribution. Here we consider the estimation of probabilities using the well-known maximum likelihood estimation (MLE) method. MLE can be stated as the empirical risk minimization by taking the negative logarithm of the conditional likelihood (negative log-likelihood) as the loss function:

$$\ell = \sum_{i=1}^{n} -\log P(y_i|\mathbf{x}_i). \tag{3}$$

We model probabilities $P(1|\mathbf{x}), \ldots, P(K|\mathbf{x})$ with a vector $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_K(\mathbf{x}))$ using the multinomial logistic transform:

$$P(y|\mathbf{x}) = \frac{e^{f_y(\mathbf{x})}}{\sum_{k=1}^{K} e^{f_k(\mathbf{x})}}. \tag{4}$$

Then (3) has the form:

$$\ell(\mathbf{f}) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} e^{f_k(\mathbf{x}_i)} \right) - f_{y_i}(\mathbf{x}_i). \tag{5}$$

This expression (with the exception that vector function $\mathbf{f}$ is used instead of scalar f) has the form of (1) if we identify $L(y_i, \mathbf{f}(\mathbf{x}_i)) = \log \left(\sum_{k=1}^{K} e^{f_k(\mathbf{x}_i)} \right) - f_{y_i}(\mathbf{x}_i)$. It is worth mentioning that the Bayes function $\mathbf{f}^*(\mathbf{x})$ is obtained by the inverse of (4).

3. Rules Ensembles

In this section, we describe the scheme of learning rule ensembles. Let X_j be the set of all possible values of attribute j. Condition part of the rule consists of

a conjunction of elementary expressions of the form $x_j \in S_j$, where x_j is the value of object $\mathbf{x}$ on attribute j and S_j is a subset of X_j, $j \in \{1, \ldots, m\}$. We assume that in the case of ordered value sets, S_j has the form of the interval $[s_j, \infty)$ or $(-\infty, s_j]$ for some $s_j \in X_j$, so that the elementary expressions take the form $x_j \geq s_j$ or $x_j \leq s_j$. For nominal attributes, we consider elementary expressions of the form $x_j = s_j$ or $x_j \neq s_j$. Let Φ be the set of elementary expressions constituting the condition part of the rule and let $\Phi(\mathbf{x})$ be an indicator function equal to 1 if $\mathbf{x}$ satisfies the condition part of the rule (all elementary expressions in the condition part), otherwise $\Phi(\mathbf{x}) = 0$. We say that a rule *covers* an object $\mathbf{x}$, if $\Phi(\mathbf{x}) = 1$. The response of the rule is a vector $\boldsymbol{\alpha} \in \mathbb{R}^K$ assigned to the region defined by Φ. Therefore, we define a decision rule as:

$$\mathbf{r}(\mathbf{x}) = \boldsymbol{\alpha}\Phi(\mathbf{x}). \tag{6}$$

Notice that the decision rule takes only two values, $\mathbf{r}(\mathbf{x}) \in \{\boldsymbol{\alpha}, \mathbf{0}\}$, depending whether $\mathbf{x}$ satisfies the condition part or not. In this paper, we assume the classification function is a linear combination of M rules:

$$\mathbf{f}(\mathbf{x}) = \sum_{m=1}^{M} \mathbf{r}_m(\mathbf{x}). \tag{7}$$

Using (4), we can obtain conditional probabilities from (7). Moreover, from (4) it follows that $P(y|\mathbf{x})$ is a monotone function of $f_y(\mathbf{x})$. Therefore, from (2) we have that object $\mathbf{x}$ is classified to the class with the highest $f_k(\mathbf{x})$. Thus, combination (7) has very simple interpretation as a voting procedure: rules vote for each class k, and object $\mathbf{x}$ is classified to the class with the highest vote.

The construction of an optimal rules ensemble minimizing the negative loglikelihood (empirical risk) is a hard optimization problem. That is why we follow here a forward stagewise strategy (Hastie et al., 2003), i.e. the rules are added one by one, greedily minimizing the loss function:

$$\mathbf{r}_m = \arg\min_{\mathbf{r}} \ell(\mathbf{f}_{m-1} + \mathbf{r}) = \arg\min_{\Phi, \boldsymbol{\alpha}} \ell(\mathbf{f}_{m-1} + \Phi\boldsymbol{\alpha}), \tag{8}$$

where $\mathbf{r}_m$ is a rule obtained in the m-th iteration and $\mathbf{f}_{m-1}$ is the rule ensemble after $m-1$ iterations. It has been shown (Hastie et al., 2003) that "shrinking" the base classifier while adding it to the ensemble improves the prediction accuracy. That is why we set:

$$\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \nu \cdot \mathbf{r}_m(\mathbf{x}),$$

where $\nu \in (0, 1]$ is the shrinkage parameter, which constitutes a trade-off between accuracy and interpretability. Higher values ($\nu \sim 1$) produce smaller ensembles, while low values ($\nu \sim 0.1$) produce larger but more accurate ones.

4. Generation of a Single Rule

In this section, we describe how the algorithm generates single rules. In order to obtain a rule, one has to solve (8). The optimization procedure is still computationally hard. Therefore, we restrict analysis to the rules voting for only one class, so that the response of the rule has the form $\boldsymbol{\alpha} = \alpha\mathbf{v}$, where $\mathbf{v}$ is a vector with only one non-zero coordinate $v_k = 1$, for some $k = 1, \ldots, K$, and α is a positive real value.

We propose two heuristic procedures for solving (8). The first, called gradient method (Mason et al., 1999), approximates $\ell(\mathbf{f} + \alpha\mathbf{v}\Phi)$ up to the first order with respect to α:

$$\ell(\mathbf{f} + \alpha\mathbf{v}\Phi) \simeq \ell(\mathbf{f}) + \alpha\ell'(\mathbf{f}, \mathbf{v}\Phi), \tag{9}$$

where

$$\ell'(\mathbf{f}, \mathbf{v}\Phi) = \left.\frac{\partial \ell(\mathbf{f} + \alpha\mathbf{v}\Phi)}{\partial \alpha}\right|_{\alpha=0}. \tag{10}$$

Since the first term in (9) is constant, minimization of the loss for *any* positive α is equivalent to minimization of the second term. Thus, if we define:

$$\mathcal{L}_m(\Phi) = \min_{\mathbf{v}} -\ell'(\mathbf{f}, \mathbf{v}\Phi) \tag{11}$$

(we remind, that there are only K possible vectors $\mathbf{v}$, so the arg min operation can be done by simply checking all K possibilities), then Φ_m can be obtained by minimizing $\mathcal{L}_m(\Phi)$.

The second heuristic, Newton method, approximates $\ell(\mathbf{f} + \alpha\mathbf{v}\Phi)$ up to the second order:

$$\ell(\mathbf{f} + \alpha\mathbf{v}\Phi) \simeq \ell(\mathbf{f}) + \alpha\ell'(\mathbf{f}, \mathbf{v}\Phi) + \frac{\alpha^2}{2}\ell''(\mathbf{f}, \mathbf{v}\Phi), \tag{12}$$

where $\ell'(\mathbf{f}, \mathbf{v}\Phi)$ is defined as before, and:

$$\ell''(\mathbf{f}, \mathbf{v}\Phi) = \left.\frac{\partial^2 \ell(\mathbf{f} + \alpha\mathbf{v}\Phi)}{\partial \alpha^2}\right|_{\alpha=0}. \tag{13}$$

Due to convexity of the loglikelihood, expression (12) is minimized by the Newton step:

$$\alpha = -\frac{\ell'(\mathbf{f}, \mathbf{v}\Phi)}{\ell''(\mathbf{f}, \mathbf{v}\Phi)}. \tag{14}$$

By substituting (14) into (12), and taking the square root, we get:

$$\mathcal{L}_m(\Phi) = \min_{\mathbf{v}} -\frac{\ell'(\mathbf{f}, \mathbf{v}\Phi)}{\sqrt{\ell''(\mathbf{f}, \mathbf{v}\Phi)}}, \tag{15}$$

and we can obtain Φ_m by minimizing $\mathcal{L}_m(\Phi)$.

Algorithm 1 MLRules

input: set of n training examples $\{(y_i, \mathbf{x}_i)\}_1^n$,
M – number of decision rules.
output: rule ensemble $\{r_m(\mathbf{x})\}_1^M$.

$\mathbf{f}_0 := \boldsymbol{\alpha}_0$.
for $m = 1$ **to** M **do**
$\Phi_m(\mathbf{x}) = \arg\min_\Phi \mathcal{L}_m(\Phi)$
$\boldsymbol{\alpha}_m = -\ell'(\mathbf{f}, \mathbf{v}\Phi)/\ell''(\mathbf{f}, \mathbf{v}\Phi)$
$\mathbf{r}_m(\mathbf{x}) = \boldsymbol{\alpha}_m \Phi_m(\mathbf{x})$
$\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \nu \mathbf{r}_m(\mathbf{x})$
end for

Expressions $\ell'(\mathbf{f}, \mathbf{v}\Phi)$ and $\ell''(\mathbf{f}, \mathbf{v})$ have a very simple form. Let $\mathbf{v}$ be such that $v_k = 1$. Then:

$$\ell'(\mathbf{f}, \mathbf{v}\Phi) = \sum_{\Phi(\mathbf{x}_i)=1} p_{ik} - \delta_{k,y_i}, \tag{16}$$

$$\ell''(\mathbf{f}, \mathbf{v}\Phi) = \sum_{\Phi(\mathbf{x}_i)=1} p_{ik}(1 - p_{ik}), \tag{17}$$

where $p_{ik} = P(k|\mathbf{x}_i)$ and $\delta_{i,j} = 1$ iff $i = j$. To calculate $\mathcal{L}_m(\Phi)$, these expressions must be obtained for each k.

What we still need for finding Φ_m using both gradient and Newton techniques, is a fast procedure for minimizing $\mathcal{L}_m(\Phi)$, regardless whether it is defined by (11) or (15). We propose the following simple iterative procedure: at the beginning, Φ_m is empty (no elementary expressions are specified) and we set $\mathcal{L}_m(\Phi) = 0$. In each step, an elementary expression $x_j \in S_j$ is added to Φ_m that minimizes $\mathcal{L}_m(\Phi)$ (if it exists). Such expression is searched by sequentially testing the elementary expressions, attribute by attribute. For ordered attributes, each expression of the form $x_j \geq s_j$ or $x_j \leq s_j$ is tested, for every $s_j \in X_j$; for nominal attributes, we test each expression of the form $x_j = s_j$ or $x_j \neq s_j$, for every $s_j \in X_j$. Adding new expressions is repeated until $\mathcal{L}_m(\Phi)$ cannot be decreased. We also simultaneously obtain $\mathbf{v}_m$, i.e. the value of $\mathbf{v}$ for which the minimum is reached in (11) or (15). Notice that since $\mathcal{L}_m(\Phi) = 0$ at the beginning, $\mathcal{L}_m(\Phi)$ must be strictly negative at the end, otherwise no rule will be generated. The procedure for finding optimal Φ is very fast and proved to be efficient in computational experiments. The ordered attributes can be sorted once before generating any rule. This procedure resembles the way the decision trees are generated. Here, we look, however, for only one path from the root to the leaf. Moreover, let us notice that a minimal value of $\mathcal{L}_m(\Phi)$ is a natural stop criterion in building a single rule and we do not use any other measures (e.g. impurity measures) for choosing the optimal cuts.

Having found Φ_m, we can obtain α_m by solving the following convex line-search problem:

$$\alpha_m = \arg\min_\alpha \ell(\mathbf{f} + \alpha \mathbf{v}_m \Phi_m). \tag{18}$$

To speed up the computations, we follow, however, simpler procedure and obtain α_m by the Newton step (14). The whole procedure for constructing the rule ensemble is presented as Algorithm 1. We call this procedure MLRules. Note that we start with $\mathbf{f}(\mathbf{x})$ equal to $\boldsymbol{\alpha_0}$, which is a "default rule" with fixed $\Phi(\mathbf{x}) \equiv 1$, while $\mathbf{v}_0$ and α_0 are obtained as usual. Since the response always indicates the majority class, such a rule serves as a default classification when no other rule covers a given object.

In our implementation of the algorithm, we employed the resampling technique (Friedman & Popescu, 2003), which is known to improve both accuracy and computational complexity. To obtain less correlated rules, we search for Φ_m, using (11) or (15), on a random subsample (drawn without replacement) of the training set of size $\eta < n$. Then, however, the response α_m is obtained using *all* of the training objects (including those objects, which have not been used to obtain Φ_m). This usually decreases $|\alpha_m|$, so it plays the role of a regularization method, and avoids overfitting the rule to the training set.

5. Extensions

In this section, we shortly discuss the problem of convergence and propose two simple extensions of the main algorithm.

5.1. Convergence

The procedure of obtaining α_m with the Newton step does not always decrease the empirical risk and does not guarantee the convergence of the algorithm. However, using a simple backtracking line-search is sufficient for convergence: we start with α_m obtained by the Newton step. If $\ell(\mathbf{f} + \alpha \mathbf{v}\Phi) < \ell(\mathbf{f})$, the procedure stops; otherwise repeat $\alpha_m := \alpha_m/2$ until the above condition is satisfied. This procedure ends after a finite number of steps, since from the definition of $\mathcal{L}_m$, either (11) or (15), it follows that $\mathcal{L}_m = 0$ if and only if $\ell'(\mathbf{f}, \mathbf{v}\Phi) = 0$, so if a rule is generated, $\mathcal{L}_m < 0$ and $\mathbf{v}\Phi$ is a descent direction. Therefore, ℓ is decreased in each iteration. Since ℓ is bounded from below, the procedure converges, i.e.: $\lim_{m\to\infty} \ell(\mathbf{f}_m) = \ell^\infty$. In the implementation of the algorithm we do not use such a procedure since the algorithm is stopped after M rounds anyway.

This raises the question, whether ℓ^∞ is the solution with the minimum achievable value of negative log-

likelihood in the class of rule ensembles $\mathcal{F}$, i.e. if ℓ^∞ is equal to $\ell^* = \inf_{\mathbf{f}\in\mathcal{F}} \ell(\mathbf{f})$? The answer is negative because a greedy procedure is used to find the condition part of rule Φ. Then, even if a "descent direction" rule exists, the procedure may fail to find it (although the resampling strategy improves the procedure by randomly perturbing the training set in each iteration, which helps to avoid "local minima"). Nevertheless, this questions seems not to be of practical importance here, since we fix the maximal number of rules M. This is due to the empirical evidences showing that ensemble methods sometimes overfit on real-life data when the size of the ensemble is too large. In the next subsection, we describe another stopping condition, independent of the parameter M.

5.2. Avoiding overfitting

A decision rule has the form of m-dimensional rectangle. It can be shown, that the class of m-dimensional rectangles has Vapnik-Chervonenkis (VC) dimension equal to $2m$ and the VC dimension does not depend on the number of cuts. This is contrary to the tree classifier, for which the VC dimension grows to infinity with increasing number of cuts (nodes). Therefore, in case of tree ensembles, one usually specifies some constraints on tree complexity, e.g. maximal number of nodes, while in case of a rule ensemble no such constraints are necessary.

The theoretical results (Schapire et al., 1998) suggest that an ensemble with a simple base classifier (with low VC dimension) and high prediction confidence (margin) on the dataset generalizes well, regardless of the size of the ensemble. Nevertheless, we conducted the computational experiments which show that the performance of rule ensemble can deteriorate as the number of rules grows, especially for the problems with high noise level. Similar phenomenon has been observed for other boosting algorithms, in particular for AdaBoost (Mason et al., 1999; Friedman et al., 2000; Dietterich, 2000). Therefore, we propose a procedure for stopping the ensemble growth, based on the simple "holdout set" analysis.

Each rule is induced from the subsample of size $\eta < n$ without replacement. Thus, there are $n - \eta$ objects which do not take part in the induction procedure and can be used as a holdout set to estimate the quality of the induced rule. Since each rule votes for a single class, we calculate a simple 0-1 error (accuracy) of such a rule on the covered objects from the holdout set. A rule is *acceptable* if the holdout error is better (lower) than random guessing. Then, the stopping condition has the following form: in any p subsequent iterations at least q rules are not acceptable. Such "averaging" over the iterations removes variations and allows us to observe the longer-term behavior of rule acceptability. We set $p = 10$ and $q = 8$, and those values were obtained by noticing, that when the null hypothesis states that rules are not worse than random guessing, at least 8 unacceptable rules must be obtained in 10 trials to reject the null hypothesis in the binomial test with confidence level 0.05. Another possibility for stopping the ensemble growth is running the internal cross validation, but such procedure has not been used in the experiment due to computational complexity.

5.3. Ordinal classification

It is often the case that a meaningful order relation between class labels exists. For example, in recommender systems, users are often asked to evaluate items on five value ("stars") scale. Such problems are often referred to as ordinal classification problems. Here we show how the order relation can be taken into account in MLRules. Without loss of generality, we assume that the order between classes is concordant with the order between class labels coded as natural numbers $Y = \{1, \ldots, K\}$.

To capture the ordinal properties of Y, we only take into account rules voting for "at least" and "at most" class unions, where by "at least" class union we mean set $\{k, \ldots, K\}$ for some k, while by "at most" class union we mean $\{1, \ldots, k\}$. Such rules can be incorporated by considering the vectors $\mathbf{v}$ in the response of the rule to be of the form: $\mathbf{v} = \{-1, \ldots, -1, 1, \ldots, 1\}$ (vote for "at least" union) or $\mathbf{v} = \{1, \ldots, 1, -1, \ldots, -1\}$ (vote for "at most" union), so that the rule increases the probability of a class union, and not of a single class.

The whole algorithm remains the same, apart from the formulas (16) and (17), which now takes the form:

$$\ell'(\mathbf{f}, \mathbf{v}\Phi) = \sum_{\Phi(\mathbf{x}_i)=1} \sum_{k=1}^{K} v_k (p_{ik} - \delta_{k,y_i}), \tag{19}$$

$$\ell''(\mathbf{f}, \mathbf{v}\Phi) = \sum_{\Phi(\mathbf{x}_i)=1} \sum_{1=t}^{K} v_k p_{ik} \left(1 - \sum_{k=1}^{K} v_k p_{ik}\right). \tag{20}$$

The experimental verification of the usefulness of such rule representation is postponed for future research due to the lack of space.

6. Experimental Results

In this section, we show the results of the computational experiments on real datasets. First, we examine

the behavior of the ensemble as the number of rules increases. Then, we compare our algorithm with existing approaches to rule induction.

6.1. Error curves

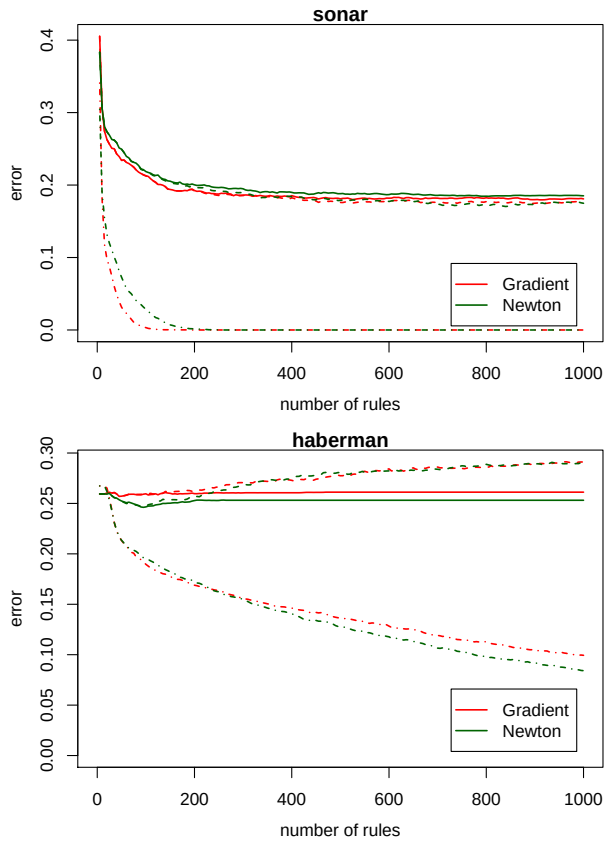


Figure 1. Train and test errors as functions of the ensemble size, obtained by splitting the data into train (66%) and test (33%) sets and averaging over 50 random splits. The lower lines (dashed-dotted) correspond to the train error, upper solid lines – to the test error with stopping condition described in Section 5.2, upper dashed line – test error when the stopping condition was not applied. Parameters of the MLRules are: $\nu = 0.1, \eta = 0.5n$

In Figure 6.1, we present the train and test error as a function of the ensemble size M for two real datasets, taken from the UCI Repository (Asuncion & Newman, 2007). On the *sonar* dataset, both ensembles (gradient and Newton) do not overfit and the test error decreases even if the number of rules reaches 1000; this is a rather typical situation. An atypical one can be found on the second dataset (*haberman*), where from some point, test error starts to increase. However, then a stopping condition described in Section 5.2 is satisfied which prevents rule ensemble from overfitting.

6.2. Comparison with other rule ensemble algorithms

To check the performance of MLRules on the real datasets, we compare them with three existing rule induction algorithms. SLIPPER (Cohen & Singer, 1999) was proposed within the AdaBoost reweighting scheme and uses an induction procedure which involves pruning. LRI (Weiss & Indurkhya, 2000) generates rules in the form of a DNF-formula and uses a specific reweighting scheme based on the cumulative errors. RuleFit (Friedman & Popescu, 2005) is based on FSAM framework (Hastie et al., 2003), but it uses the regression trees as base classifiers and then transforms them to rules. All three approaches are thus based on some boosting/reweighting strategy. According to our knowledge, RuleFit has not been compared with SLIPPER and LRI yet.

We used 35 files taken from UCI Repository (Asuncion & Newman, 2007), among which 20 files are binary classification tasks and 15 are multi-class tasks. We omit characteristics of the datasets due to lack of space. We tested four classifiers on each dataset (MLRules with gradient and Newton steps, LRI and SLIPPER) and RuleFit on binary datasets only (RuleFit does not handle multi-class case). We selected the following parameters for each method:

- SLIPPER: we set maximum number of iteration to 500, rest of parameters were set to default (we kept the internal cross validation, used to choose the optimal number of rules).
- LRI: According to (Weiss & Indurkhya, 2000), we set the rule length to 5 and froze feature after 50 rounds; we also chose 200 rules per class and 2 disjuncts since some previous tests showed that those values work well in practice.
- RuleFit: According to the experiment in (Friedman & Popescu, 2005), we chose mixed rule-linear mode, set average tree size to 4, increased the number of trees to 500, and chose subsample size η as $\min\{0.5n, 100 + 6\sqrt{n}\}$.
- MLRules: We set $\eta = 0.5n, \nu = 0.1, M = 500$, but for the tree biggest datasets (*letter*, *optdigits*, *pendigits*) we increased M to 2000 (to compare, LRI had 26×200 rules for *letter*). Those parameters have not been optimized on the UCI data. We used artificial data to this end and due to the space limit we omit the characteristic of the data generating model.

Each test was performed using 10-fold cross validation

Table 1. Test errors and ranks (in parenthesis). MLRules.G and MLRules.N are MLRules with gradient and Newton method, respectively. Average ranks in the last row correspond to comparing LRI and MLRules on all 35 files.

DATASET	SLIPPER	LRI	RULEFIT	MLRULES.G	MLRULES.N
			BINARY-CLASS DATASETS		
HABERMAN	0.268 (3.0)	0.275(5.0)	0.272(4.0)	0.262 (2.0)	0.249 (1.0)
BREAST-C	0.279 (3.0)	0.293(4.0)	0.297(5.0)	0.259 (1.0)	0.273 (2.0)
DIABETES	0.254 (4.0)	0.254(3.0)	0.262(5.0)	0.247 (1.0)	0.253 (2.0)
CREDIT-G	0.277 (5.0)	0.239(1.0)	0.259(3.0)	0.241 (2.0)	0.260 (4.0)
CREDIT-A	0.170 (5.0)	0.122(1.0)	0.132(3.0)	0.133 (4.0)	0.130 (2.0)
IONOSPHERE	0.065 (3.0)	0.068(4.0)	0.085(5.0)	0.060 (1.0)	0.063 (2.0)
COLIC	0.150 (4.0)	0.161(5.0)	0.147(3.0)	0.139 (2.0)	0.133 (1.0)
HEPATITIS	0.167 (2.0)	0.180(3.0)	0.194(4.0)	0.162 (1.0)	0.201 (5.0)
SONAR	0.264 (5.0)	0.149(2.0)	0.197(4.0)	0.120 (1.0)	0.154 (3.0)
HEART-STATLOG	0.233 (5.0)	0.196(4.0)	0.185(3.0)	0.167 (1.0)	0.174 (2.0)
LIVER-DISORDERS	0.307 (5.0)	0.266(1.0)	0.307(4.0)	0.275 (2.0)	0.278 (3.0)
VOTE	0.050 (5.0)	0.039(3.0)	0.050(5.0)	0.034 (1.0)	0.037 (2.0)
HEART-C	0.195 (5.0)	0.185(3.0)	0.189(4.0)	0.165 (2.0)	0.155 (1.0)
HEART-H	0.200 (5.0)	0.183(3.0)	0.183(4.0)	0.180 (2.0)	0.170 (1.0)
BREAST-W	0.043 (5.0)	0.033(2.0)	0.041(4.0)	0.031 (1.0)	0.034 (3.0)
SICK	0.016 (2.0)	0.018(4.0)	0.019(5.0)	0.016 (3.0)	0.012 (1.0)
TIC-TAC-TOE	0.024 (2.0)	0.122(5.0)	0.053(3.0)	0.113 (4.0)	0.003 (1.0)
SPAMBASE	0.059 (5.0)	0.049(3.0)	0.059(4.0)	0.047 (2.0)	0.046 (1.0)
CYLINDER-BANDS	0.217 (4.0)	0.165(2.0)	0.381(5.0)	0.144 (1.0)	0.193 (3.0)
KR-VS-KP	0.006 (2.0)	0.031(5.0)	0.029(4.0)	0.010 (3.0)	0.005 (1.0)
AVG. RANK	3.9	3.15	4.05	1.85	2.05
			MULTI-CLASS DATASETS		
ANNEAL	0.018	0.007(3.0)	—	0.006 (1.5)	0.006 (1.5)
BALANCE-SCALE	0.17	0.088(2.0)	—	0.078 (1.0)	0.091 (3.0)
ECOLI	0.211	0.140(2.0)	—	0.149 (3.0)	0.140 (1.0)
GLASS	0.340	0.285(3.0)	—	0.244 (1.0)	0.248 (2.0)
IRIS	0.080	0.053(2.0)	—	0.053 (2.0)	0.053 (2.0)
LETTER	0.821	0.069(1.0)	—	0.137 (3.0)	0.088 (2.0)
SEGMENT	0.215	0.021(2.0)	—	0.029 (3.0)	0.020 (1.0)
SOYBEAN	0.505	0.413(3.0)	—	0.073 (2.0)	0.067 (1.0)
VEHICLE	0.301	0.210(1.0)	—	0.236 (3.0)	0.216 (2.0)
VOWEL	0.448	0.059(1.0)	—	0.148 (3.0)	0.104 (2.0)
CAR	0.045	0.054(2.0)	—	0.057 (3.0)	0.028 (1.0)
CMC	0.477	0.435(1.0)	—	0.437 (2.0)	0.439 (3.0)
DERMATOLOGY	0.161	0.057(3.0)	—	0.019 (1.0)	0.024 (2.0)
OPTDIGITS	0.560	0.019(1.0)	—	0.026 (3.0)	0.021 (2.0)
PENDIGITS	0.460	0.010(2.0)	—	0.014 (3.0)	0.010 (1.0)
AVG. RANK	—	2.26	—	1.9	1.84

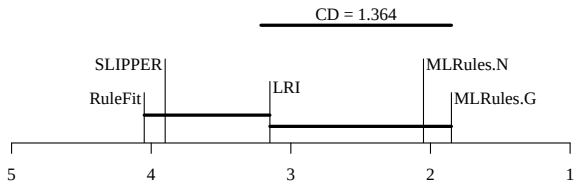


Figure 2. Critical difference diagram

(with exactly the same train/test splits for each classifier) and average 0-1 loss on the test set was calculated. The results are shown in Table 6.2.

We first restrict the analysis to binary-class problems only. To compare multiple classifiers on the multiple datasets, we follow Demšar (2006), and make the Friedman test, which uses ranks of each algorithm to check whether all the algorithms perform equally well (null hypothesis). Friedman statistics gives 33.28 which exceeds the critical value 9.488 (for confidence level 0.05), so we can reject the null hypothesis and state that classifiers are not equally good. Next, we proceed to a post-hoc analysis and calculate the *critical difference* (CD) according to the Nemeneyi statistics. We obtain $CD = 1.364$ which means that algorithms with difference in average ranks more than 1.364 are significantly different. In Figure 6.2 average ranks were marked on a line, and groups of classifiers that are not significantly different were connected. This shows that both MLRules algorithms are not sig-

nificantly different to LRI, however they outperform both SLIPPER and RuleFit. On the other hand, none of three well-known rule ensemble algorithms (LRI, SLIPPER, RuleFit) is significantly better to any other.

The situation remains roughly the same if we compare the algorithms using all 35 datasets. We exclude RuleFit (it does not work with multi-class problems) and SLIPPER (its results are very poor, the worst almost every time[1]). Thus, we end up with 3 algorithms. Friedman statistics gives 3.53 which does not exceed the critical value 5.991, so that the null hypothesis cannot be rejected. Note that the difference in ranks decreased, mainly because LRI performs excellent on the largest datasets (letters and digits recognition).

It is interesting to check how much of the improvement in accuracy of MLRules comes from shrinkage, resampling and regularizing the rule response, because those techniques can also be simply incorporated to SLIPPER and LRI. We plan to investigate this issue in our future research.

7. Conclusions and Future Research

We proposed a new rule induction algorithm for solving classification problems, called MLRules, based on the maximum likelihood estimation method and using boosting strategy in rule induction. In contrary to previously considered algorithms, it estimates the conditional class probability distribution and therefore can work with any cost matrix for classification. We considered two optimization techniques, based on gradient and Newton steps, and introduced a stopping condition to avoid overfitting. The performance of MLRules was verified on a large collection of datasets, both binary- and multi-class. Our algorithm is competitive or outperforms the best existing approaches to rule induction.

We also suggested the way in which MLRules can capture the order between classes and therefore can solve the ordinal classification problems. We plan to verify this issue experimentally in the future.

References

Asuncion, A., & Newman, D. J. (2007). UCI machine learning repository.

Cohen, W. W. (1995). Fast effective rule induction. *ICML* (pp. 115–123).

Cohen, W. W., & Singer, Y. (1999). A simple, fast, and effective rule learner. *National Conference on Artificial Intelligence* (pp. 335–342).

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, *40*, 139–158.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.

Friedman, J. H., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 337–407.

Friedman, J. H., & Popescu, B. E. (2003). *Importance sampled learning ensembles* (Technical Report). Dept. of Statistics, Stanford University.

Friedman, J. H., & Popescu, B. E. (2005). *Predictive learning via rule ensembles* (Technical Report). Dept. of Statistics, Stanford University.

Fürnkranz, J. (1996). Separate-and-conquer rule learning. *Artificial Intelligence Review*, *13*, 3–54.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2003). *Elements of statistical learning: Data mining, inference, and prediction*. Springer.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). *Functional gradient techniques for combining hypotheses*, 33–58. MIT Press.

Michalski, R. S. (1983). *A theory and methodology of inductive learning*, 83–129. Tioga Publishing.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, *26*, 1651–1686.

Stefanowski, J. (1998). On rough set based approach to induction of decision rules. *Rough Set in Knowledge Discovering* (pp. 500–529). Physica Verlag.

Weiss, S. M., & Indurkhya, N. (2000). Lightweight rule induction. *ICML* (pp. 1135–1142).

[1] The SLIPPER software documentation says: "it is an *experiment* multi-class version" and "there is known bug which occasionally causes incorrect output", so we decided not to consider multi-class results.

Learning from Incomplete Data with Infinite Imputations

Uwe Dick DICK@MPI-SB.MPG.DE
Peter Haider HAIDER@MPI-SB.MPG.DE
Tobias Scheffer SCHEFFER@MPI-SB.MPG.DE
Max Planck Institute for Computer Science, Saarbrücken, Germany

Abstract

We address the problem of learning decision functions from training data in which some attribute values are unobserved. This problem can arise, for instance, when training data is aggregated from multiple sources, and some sources record only a subset of attributes. We derive a generic joint optimization problem in which the distribution governing the missing values is a free parameter. We show that the optimal solution concentrates the density mass on finitely many imputations, and provide a corresponding algorithm for learning from incomplete data. We report on empirical results on benchmark data, and on the email spam application that motivates our work.

1. Introduction

In many applications, one has to deal with training data with incompletely observed attributes. For instance, training data may be aggregated from different sources. If not all sources are capable of providing the same set of input attributes, the combined training sample contains incompletely observed data. This situation occurs in email spam detection, where it is helpful to augment the content of an email with real-time information about the sending server, such as its blacklist status. This information is available for all training emails that arrive at a mail server under one's own control, and it is also available at application time. But if one wants to utilize training emails from public archives, this information is missing.

We adress a learning setting in which values are *missing at random:* here, the presence or absence of values does not convey information about the class labels. If this condition is not met, it is informative to consider the presence or absence of values as additional input to the decision function. Techniques for learning from incomplete data typically involve a distributional model that imputes missing values, and the desired final predictive model. Prior work on learning from incomplete data is manifold in the literature, and may be grouped by the way the distributional model is used.

The first group models the distribution of missing values in a first step, and learns the decision function based on the distributional model in a second step. Shivaswamy et al. (2006) formulate a loss function that takes a fixed proportion of the probability mass of each instance into account, with respect to the estimated distribution of missing values. They derive second order cone programs which renders the method applicable only to very small problems. Other examples include Williams and Carin (2005), Williams et al. (2005), and Smola et al. (2005).

The second group estimates the parameters of a distributional model and the final predictive model jointly. As an example, recently Liao et al. (2007) propose an EM-algorithm for jointly estimating the imputation model and a logistic regression classifier with linear kernel, assuming the data arises from a mixture of multivariate Gaussians.

The third group makes no model assumption about the missing values, but learns the decision function based on the visible input alone. For example, Chechik et al. (2007) derive a geometrically motivated approach. For each example, the margin is re-scaled according to the visible attributes. This procedure specifically aims at learning from data with values that are *structurally missing*—as opposed to *missing at random*. Chechik et al. (2007) find empirically that the procedure is not adequate when values are missing at random.

Jointly learning a distributional model and a kernel predictive model relates to the problem of learning a

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

kernel function from a prescribed set of parameterized kernels. This problen drew a lot of attention recently; see, for example, Argyriou et al. (2005) and Micchelli and Pontil (2007).

Estimating the distributional model first and training the predictive model in a second step leaves the user free to choose any learning algorithm for this second step. However, a harder problem has to be solved than would be necessary. If one is only interested in a decision function that minimizes the desired loss, knowing the values or distribution of the missing attributes in the training set is not actually required. Furthermore, errors made in the imputation step and errors made in estimating the parameters of the predictive model can add up in a sequential procedure.

Consequently, we investigate learning the decision function and the distribution of imputations dependently. Unlike prior work on this topic, we develop a solution for a very general class of optimization criteria. Our solution covers a wide range of loss functions for classification and regression problems. It comes with all the usual benefits of kernel methods. We derive an optimization problem in which the distribution governing the missing values is a free parameter. The optimization problem searches for a decision function and a distribution governing the missing values which together minimize a regularized empirical risk.

No fixed parametric form of the distributional model is assumed. A regularizer that can be motivated by a distributional assumption may *bias* the distributional model *towards* a prior belief. However, the regularizer may be overruled by the data, and the resulting distributional model may be different from any parametric form. We are able to prove that there exists an optimal solution based on a distribution that is supported by finitely many imputations. This justifies a greedy algorithm for finding a solution. We derive manifestations of the general learning method and study them empirically.

The paper is structured as follows. After introducing the problem setting in Section 2, we derive an optimization problem in Section 3. Section 4 proves that there is an optimal solution that concentrates the density mass on finitely many imputations and presents an algorithm. Example instantiations of the general solution are presented in Section 5. We empirically evaluate the method in Section 6. Section 7 concludes.

2. Problem Setting

We address the problem of learning a decision function f from a training sample in which some attribute values are unobserved.

Let $\mathbf{X}$ be a matrix of n training instances $\mathbf{x}_i$ and let $\mathbf{y}$ be the vector of corresponding target values y_i. Instances and target values are drawn *iid* from an unknown distribution $p(\mathbf{x}, y)$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ denotes the set of possible target values. Matrix $\mathbf{Z}$ indicates which features are observed. A value of $z_{il} = 1$ indicates that x_{il}, the l-th feature of the i-th example, is observed. Values are *missing at random:* y_i is conditionally independent of $\mathbf{z}_i$ given $\mathbf{x}_i$.

The goal is to learn a function $f : \mathbf{x} \mapsto y$ that predicts target values for *completely observed* examples. The decision function should incur only a minimal true risk $R(f) = \int L(y, f(\mathbf{x}))p(\mathbf{x}, y)d\mathbf{x}dy$, where L is a loss function for the task at hand.

As a means to minimizing the true risk, we seek a function f in the *reproducing kernel Hilbert space* $\mathcal{H}_k$ induced by a kernel k that minimizes a regularized empirical risk functional $R(f) = \sum_{i=1}^{n} l(y_i, f(\mathbf{x}_i)) + \eta\|f\|_k^2$. We demand k to be a Mercer kernel. Loss function l approximates the true loss L. The *representer theorem* allows us to write the minimizer as a sum over functions in $\mathcal{H}_k$ centered at training instances: $f(\mathbf{x}) = \sum_{j=1}^{n} c_j k(\mathbf{x}_j, \mathbf{x})$.

The learning problem from completely observed data would amount to solving Optimization Problem 1.

Optimization Problem 1 (Primal learning problem, observed data). *Over* $\mathbf{c}$*, minimize*

$$R(\mathbf{c}, k) = \sum_{i=1}^{n} l\Big(y_i, \sum_{j=1}^{n} c_j k(\mathbf{x}_j, \mathbf{x}_i)\Big) + \eta \sum_{i,j=1}^{n} c_i c_j k(\mathbf{x}_j, \mathbf{x}_i)$$

We require that the loss function be defined in such a way that Optimization Problem 1 can be written in the dual form of Optimization Problem 2. A wide range of loss functions satisfies this demand; we will later see that this includes hinge loss and squared loss.

Optimization Problem 2 (Dual of learning problem). *Given* $a < 0$*, over* $\mathbf{c}$*, maximize*

$$a \langle \mathbf{c}, \mathbf{K}\mathbf{c} \rangle - R^*(\mathbf{c})$$

subject to the constraints

$$\forall_{i=1}^{m_1^*} g_i^*(\mathbf{c}) \leq 0, \qquad \forall_{j=1}^{m_2^*} h_j^*(\mathbf{c}) = 0. \tag{1}$$

$R^*(\mathbf{c})$ denotes a differentiable convex function of the dual variables $\mathbf{c}$ which we demand to be independent of the kernel matrix $\mathbf{K}$. The inequality constraints g_i^* are differentiable convex and the equality constraints h_j^* differentiable affine. We like to note that the requirement of independence between R^* and $\mathbf{K}$ is not

very restrictive in practice, as we will see in chapter 5. Furthermore, we demand strong duality to hold between Optimization problems 1 and 2.

3. Learning from Incomplete Data in One Step

If any instance $\mathbf{x}_i$ has unobserved features, then $k(\mathbf{x}_i, \mathbf{x})$ and, consequently, the decision function f are not properly defined. In order to learn from incomplete data, we will marginalize the decision function and risk functional by the observable attributes and integrate over all unobserved quantities. To this end, we define $\boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}} \subset \mathbb{R}^{n \times d}$ as a matrix of imputations constrained by $\omega_{il} = x_{il}$ if $z_{il} = 1$. We demand $\Omega_{\mathbf{X}}^{\mathbf{Z}}$ to be compact for the rest of this paper. Let $\boldsymbol{\omega}_i$ denote the i-th row of $\boldsymbol{\omega}$. Then we can define a family of kernels $K(\boldsymbol{\omega})(\mathbf{x}_j, \mathbf{x}_i) = k(\boldsymbol{\omega}_j, \boldsymbol{\omega}_i)$. Any probability measure $p(\boldsymbol{\omega})$ on imputations induces a marginalization of the kernel by the observable variables. Equation 2 integrates over all imputations of unobserved values; it can be evaluated based on the observed values.

$$K(p)(\mathbf{x}_j, \mathbf{x}_i) = \int_{\boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}}} k(\boldsymbol{\omega}_j, \boldsymbol{\omega}_i) dp(\boldsymbol{\omega}) \tag{2}$$

Any probability measure $p(\boldsymbol{\omega})$ constitutes an optimization criterion $R(\mathbf{c}, K(p))$. In the absence of knowledge about the true distribution of missing values, $p(\boldsymbol{\omega})$ becomes a free parameter. Note that $p(\boldsymbol{\omega})$ is a continuous probability measure that is not constrained to any particular parametric form; the space of parameters is therefore of infinite dimensionality.

It is natural to add a regularizer $Q(p)$ that reflects prior belief on the distribution of imputations $p(\boldsymbol{\omega})$ to the optimization criterion, in addition to the empirical risk and regularizer on the predictive model. The regularizer is assumed to be continuous in p. The regularizer does not *constrain* $p(\boldsymbol{\omega})$ to any specific class of distribution, but it reflects that some distributions are believed to be more likely. Without a regularizer, the criterion can often be minimized by imputations which move instances with missing values far away from the separator, thereby removing their influence on the outcome of the learning process. This leads to Optimization Problem 3.

Optimization Problem 3 (Learning problem with infinite imputations). *Given n training examples with incomplete feature values, $\gamma > 0$, kernel function k, over all $\mathbf{c}$ and p, minimize*

$$\tilde{R}_{k,\gamma}(\mathbf{c}, p) \quad = \quad R(\mathbf{c}, K(p)) + \gamma Q(p) \tag{3}$$

subject to the constraints

$$\forall \boldsymbol{\omega} : \; p(\boldsymbol{\omega}) \geq 0, \qquad \int_{\boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}}} p(\boldsymbol{\omega}) d\boldsymbol{\omega} = 1.$$

Each solution to Optimization Problem 3 integrates over *infinitely* many different imputations. The search space contains all continuous probability measures on imputations, the search is guided by the regularizer Q. The regularization parameter γ determines the influence of the regularization on the resulting distribution. For $\gamma \to \infty$ the solution of the optimization reduces to the solution obtained by first estimating the distribution of missing attribute values that minimizes the regularizer. For $\gamma \to 0$ the solution is constituted by the distribution minimizing the risk functional R.

4. Solving the Optimization Problem

In this section, we devise a method for efficiently finding a solution to Optimization Problem 3. Firstly, we show that there exists an optimal solution $\hat{\mathbf{c}}, \hat{p}$ with $\hat{p}$ supported on at most $n+2$ imputations $\boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}}$. Secondly, we present an algorithm that iteratively finds the optimal imputations and parameters minimizing the regularized empirical risk.

4.1. Optimal Solution with Finite Combination

In addition to the parameters $\mathbf{c}$ of the predictive models, continuous probability measure $p(\boldsymbol{\omega})$ contributes an infinite set of parameters to Optimization Problem 3. The implementation of imputations as parameters of a kernel family allows us to show that there exists an optimal probability measure $\hat{p}$ for Equation 3 such that $\hat{p}$ consists of finitely many different imputations.

Theorem 1. *Optimization Problem 3 has an optimal solution $\hat{\mathbf{c}}, \hat{p}$ in which $\hat{p}$ is supported by at most $n+2$ imputations $\boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}}$.*

Proof. The compactness of $\Omega_{\mathbf{X}}^{\mathbf{Z}}$ and the continuity of $\mathbf{K}$ immediately imply that *there exists some* solution to Optimization Problem 3. It remains to be shown that at least one of the solutions is supported by at most $n+2$ imputations. Let $\bar{\mathbf{c}}, \bar{p}$ be *any* solution and let all requirements of the previous section hold. The idea of this proof is to construct a correspondence between distributions over imputations and vectors in $\mathbb{R}^{n+1}$, where a finite support set is known to exist. Define $\mathbf{S}(\boldsymbol{\omega}) = \mathbf{K}(\boldsymbol{\omega})\bar{c} \in \mathbb{R}^n$ and $D = \{(\mathbf{S}(\boldsymbol{\omega})^\top, Q(\boldsymbol{\omega}))^\top : \boldsymbol{\omega} \in \Omega_{\mathbf{X}}^{\mathbf{Z}}\} \subset \mathbb{R}^{n+1}$. Since $\Omega_{\mathbf{X}}^{\mathbf{Z}}$ is compact and $K(\cdot)$ and $Q(\cdot)$ are continuous by definition, D is compact as well. We define a measure over D as $\mu(A \times B) = \bar{p}(\{\boldsymbol{\omega} : \mathbf{S}(\boldsymbol{\omega}) \in A \wedge Q(\boldsymbol{\omega}) \in B\})$.

Then, by Carathéodory's convex hull theorem, there exists a set of k vectors $\{(\mathbf{s}_1^\top, q_1)^\top, \ldots, (\mathbf{s}_k^\top, q_k)^\top\} \subseteq D$ with $k \leq n+2$ and nonnegative constants ν_i with

$\sum_{i=1}^{k} \nu_i = 1$, such that

$$\int_D (\mathbf{s}^\top, q)^\top d\mu((\mathbf{s}^\top, q)^\top) = \sum_{i=1}^{k} (\mathbf{s}_i^\top, q_i)^\top \nu_i.$$

For each i, select any $\boldsymbol{\omega}_i$ such that $(\mathbf{S}(\boldsymbol{\omega}_i)^\top, Q(\boldsymbol{\omega}_i)) = (\mathbf{s}_i^\top, q_i)$. We construct $\hat{p}$ by setting $\hat{p}(\boldsymbol{\omega}) = \sum_{i=1}^{k} \nu_i \delta_{\boldsymbol{\omega}_i}$, where $\delta_{\boldsymbol{\omega}_i}$ denotes the Dirac measure at $\boldsymbol{\omega}_i$. The optimal $\hat{\mathbf{c}}$ results as $\arg\min_{\mathbf{c}} R(\mathbf{c}, K(\hat{p}))$. We have

$$\begin{aligned} \int_D \mathbf{s} d\mu((\mathbf{s}^\top, q)^\top) &= \sum_{i=1}^{k} \mathbf{s}_i \nu_i, \quad \text{and} \\ \int_D q d\mu((\mathbf{s}^\top, q)^\top) &= \sum_{i=1}^{k} q_i \nu_i. \end{aligned}$$

Then

$$\begin{aligned} \mathbf{K}(\bar{p})\bar{\mathbf{c}} &= \left(\int_{\Omega_{\mathbf{X}}^{\mathbf{Z}}} \mathbf{K}(\boldsymbol{\omega}) d\bar{p}(\boldsymbol{\omega}) \right) \bar{\mathbf{c}} = \int_{\Omega_{\mathbf{X}}^{\mathbf{Z}}} \mathbf{S}(\boldsymbol{\omega}) d\bar{p}(\boldsymbol{\omega}) \\ &= \int_D \mathbf{S}(\boldsymbol{\omega}) d\mu \left((\mathbf{S}(\boldsymbol{\omega})^\top, Q(\boldsymbol{\omega}))^\top \right) \\ &= \sum_{i=1}^{k} \mathbf{s}_i \nu_i = \int_{\Omega_{\mathbf{X}}^{\mathbf{Z}}} \mathbf{S}(\boldsymbol{\omega}) d\hat{p}(\boldsymbol{\omega}) \\ &= \int_{\Omega_{\mathbf{X}}^{\mathbf{Z}}} \mathbf{K}(\boldsymbol{\omega}) d\hat{p}(\boldsymbol{\omega}) \bar{\mathbf{c}} = \mathbf{K}(\hat{p})\bar{\mathbf{c}}. \end{aligned}$$

Likewise,

$$\begin{aligned} Q(\bar{p}) &= \int_D Q(\boldsymbol{\omega}) \, d\mu \left((\mathbf{S}(\boldsymbol{\omega})^\top, Q(\boldsymbol{\omega}))^\top \right) \\ &= \sum_{i=1}^{k} q_i \nu_i = Q(\hat{p}). \end{aligned}$$

Since $Q(p)$ does not depend on $\mathbf{c}$, $\bar{\mathbf{c}} = \arg\min_{\mathbf{c}} R(\mathbf{c}, \mathbf{K}(\bar{p}))$, and by strong duality, $\bar{\mathbf{c}} = \arg\max_{\mathbf{c}} a \langle \mathbf{c}, \mathbf{K}(\bar{p})\mathbf{c} \rangle - R^*(\mathbf{c})$. This implies that the Karush-Kuhn-Tucker conditions hold for $\bar{\mathbf{c}}$, namely there exist constants $\kappa_i \geq 0$ and λ_j such that

$$a\mathbf{K}(\bar{p})\bar{\mathbf{c}} - \nabla R^*(\bar{\mathbf{c}}) + \sum_i \kappa_i \nabla g_i^*(\bar{\mathbf{c}}) + \sum_j \lambda_j \nabla h_j^*(\bar{\mathbf{c}}) = 0$$

$$\forall_i \;\; g_i^*(\bar{\mathbf{c}}) \leq 0, \quad \forall_j \;\; h_i^*(\bar{\mathbf{c}}) = 0, \quad \forall_i \;\; \kappa_i g_i^*(\bar{\mathbf{c}}) = 0$$

It is easy to see that therefore $\bar{\mathbf{c}}$ is also a maximizer of $a \langle \mathbf{c}, \mathbf{K}(\hat{p})\mathbf{c} \rangle - R^*(\mathbf{c})$, because $\mathbf{K}(\bar{p})\bar{\mathbf{c}} = \mathbf{K}(\hat{p})\bar{\mathbf{c}}$ and the Karush-Kuhn-Tucker conditions still hold. Their sufficiency follows from the fact that $\mathbf{K}(p)$ is positive semi-definite for any p, and the convexity and affinity premises. Thus,

$$\begin{aligned} & R(\bar{\mathbf{c}}, K(\bar{p})) + \gamma Q(\bar{p}) \\ &= \left[\min_{\mathbf{c}} R(\mathbf{c}, K(\bar{p})) \right] + \gamma Q(\bar{p}) \\ &= \left[\max_{\mathbf{c}} a \langle \mathbf{c}, \mathbf{K}(\bar{p})\mathbf{c} \rangle - R^*(\mathbf{c}) \right] + \gamma Q(\bar{p}) \\ &= [a \langle \bar{\mathbf{c}}, \mathbf{K}(\bar{p})\bar{\mathbf{c}} \rangle - R^*(\bar{\mathbf{c}})] + \gamma Q(\bar{p}) \\ &= [a \langle \bar{\mathbf{c}}, \mathbf{K}(\hat{p})\bar{\mathbf{c}} \rangle - R^*(\bar{\mathbf{c}})] + \gamma Q(\hat{p}) \\ &= \left[\max_{\mathbf{c}} a \langle \mathbf{c}, \mathbf{K}(\hat{p})\mathbf{c} \rangle - R^*(\mathbf{c}) \right] + \gamma Q(\hat{p}) \\ &= \left[\min_{\mathbf{c}} R(\mathbf{c}, K(\hat{p})) \right] + \gamma Q(\hat{p}) \\ &= R(\hat{\mathbf{c}}, K(\hat{p})) + \gamma Q(\hat{p}). \end{aligned}$$

We have now established that there exists a solution with at most $n + 2$ imputations. □

4.2. Iterative Optimization Algorithm

This result justifies the following greedy algorithm to find an optimal solution to Optimization Problem 3. The algorithm works by iteratively optimizing Problem 1 (or, equivalently, 2), and updating the distribution over the missing attribute values. Let $p_{\bar{\boldsymbol{\omega}}}$ denote the distribution $p(\boldsymbol{\omega}) = \delta_{\bar{\boldsymbol{\omega}}}$. Algorithm 1 shows the steps.

Algorithm 1 Compute optimal distribution of imputations on $\Omega_{\mathbf{X}}^{\mathbf{Z}}$

Initialization: Choose $p^{(1)} = p_{\boldsymbol{\omega}^{(1)}}$; e.g., $\boldsymbol{\omega}_{il}^{(1)} = 0$ for all $z_{il} \neq 1$
for $t = 1 \ldots$ **do**
1. $\hat{\mathbf{c}} \leftarrow \arg\min_{\mathbf{c}} R(\mathbf{c}, K(p^{(t)}))$
2. Find $\boldsymbol{\omega}^{(t+1)} \in \Omega_{\mathbf{X}}^{\mathbf{Z}}$: $\tilde{R}_{k,\gamma}(\hat{\mathbf{c}}, p_{\boldsymbol{\omega}^{(t+1)}}) < \tilde{R}_{k,\gamma}(\hat{\mathbf{c}}, p^{(t)})$. If no such $\boldsymbol{\omega}^{(t+1)}$ exists, terminate.
3. $\beta_t \leftarrow \arg\min_{\beta \in (0,1]} \left[\min_{\mathbf{c}} \tilde{R}_{k,\gamma}(\mathbf{c}, \beta p_{\boldsymbol{\omega}^{(t+1)}} + (1-\beta)p^{(t)}) \right]$
4. $p^{(t+1)} \leftarrow \beta_t p_{\boldsymbol{\omega}^{(t+1)}} + (1-\beta_t)p^{(t)}$
5. $\forall j < t : \; \beta_j \leftarrow \beta_j(1-\beta_t)$
end for

Step 1 consists of minimizing the regularized empirical risk functional R, given the current distribution. In step 2 a new imputation is constructed which improves on the current objective value. Since in general $\tilde{R}_{k,\gamma}(\mathbf{c}, p_{\boldsymbol{\omega}})$ is not convex in $\boldsymbol{\omega}$, one cannot find the *optimal* $\boldsymbol{\omega}$ efficiently. But the algorithm only requires to find *any* better $\boldsymbol{\omega}$. Thus it is reasonable to perform gradient ascent on $\boldsymbol{\omega}$, with random restarts in case the found local optimum does not satisfy the inequality of step 2. In step 3 and 4 the optimal distribution consisting of the weighted sum of currently used Dirac impulses $\sum_{i=1}^{t} \beta_i \delta_{\boldsymbol{\omega}_i}$ and the new imputation $\delta_{\boldsymbol{\omega}^{(t+1)}}$ is computed. This step is convex in β if

$\tilde{R}_{k,\gamma}(\mathbf{c}, \beta p_{\boldsymbol{\omega}^{(t+1)}} + (1-\beta)p^{(t)})$ is linear in β. By looking at Optimization Problem 2, we see that this is the case for R. Thus the convexity depends on the choice for Q (see Sect. 5.2). Step 5 updates the weights of the previous imputations.

The algorithm finds t imputations $\boldsymbol{\omega}^{(j)}$ and their weights β_j, as well as the optimal example coefficients $\mathbf{c}$. We can construct the classification function f as

$$f(\mathbf{x}) \quad = \quad \sum_{j=1}^{t} \sum_{i=1}^{n} \beta_j \mathbf{c}_i k(\boldsymbol{\omega}_i^{(j)}, \mathbf{x}). \tag{4}$$

Note that the value $n+2$ is an upper bound for the number of basic kernels which constitute the optimal solution. The algorithm is not guaranteed to terminate after $n+2$ iterations, because the calculated imputations are not necessarily optimal. In practice, however, the number of iterations is usually much lower. In our experiments, the objective value of the optimization problem converges in less than 50 iterations.

5. Example Learners

In this chapter we present manifestations of the generic method, which we call *weighted infinite imputations*, for learning from incomplete data that we use in the experimental evaluation.

Recall from Section 3 the goal to learn a decision function f from incomplete data that minimizes the expected risk $R(f) = \int L(y, f(\mathbf{x}))p(\mathbf{x}, y)d\mathbf{x}dy$. In classification problems the natural loss function L becomes the *zero-one loss*, whereas in regression problems the loss depends on the specific application; common choices are the *squared error* or the ϵ-*insensitive* loss. The considerations in the previous chapters show that, in order to learn regression or classification functions from training instances with missing attribute values, we only have to specify the dual formulation of the preferred learning algorithm on complete data and a regularizer on the distribution of imputations p.

5.1. Two Standard Learning Algorithms

For binary classification problems, we choose to approximate the zero-one by the *hinge* loss and perform *support vector machine* learning. The dual formulation of the SVM is given by $R^{SVM}(\mathbf{c}, k) = \sum_{i=1}^{n} \frac{c_i}{y_i} - \frac{1}{2}\sum_{i,j=1}^{n} c_i c_j k(\mathbf{x}_j, \mathbf{x}_i)$ subject to the constraints $0 \leq \frac{c_i}{y_i} \leq \frac{1}{\eta}$ and $\sum_{i=1}^{n} c_i = 0$. We see that the demands of Optimization Problem 2 are met and a finite solution can be found. Taking the SVM formulation as the dual Optimization Problem 2 gives us the means – in conjunction with an appropriate regularizer Q – to learn a classification function f from incomplete data.

For regression problems, the loss depends on the task at hand, as noted above. We focus on penalizing the *squared error*, though we like to mention that the approach works for other losses likewise. One widely used learning algorithm for solving the problem is *kernel ridge regression*. Again, we can learn the regression function f from incomplete data by using the same principles as described above. Kernel ridge regression minimizes the regularized empirical risk $\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 + \eta\|f\|^2$. The dual formulation $R^{KRR}(\mathbf{c}, k) = \sum_{i=1}^{n} c_i y_i - \frac{1}{4}\sum_{i=1}^{n} c_i^2 + \frac{1}{4\eta}\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j)$ again meets the demands of the dual optimization problem 2. Substituting its primal formulation for R in step 1 of Algorithm 1 and in Eqn. 3 solves the problem of learning the regression function from incomplete data after specifying a regularizer Q.

5.2. Regularizing towards Prior Belief in Feature Space

A regularizer on the distribution of missing values can guide the search towards distributions $\hat{\boldsymbol{\omega}}$ that we believe to be likely. We introduce a regularization term which penalizes imputations that are different from our prior belief $\hat{\boldsymbol{\omega}}$. We choose to penalize the sum of squared distances between instances $\mathbf{x}_i$ and $\hat{\boldsymbol{\omega}}_i$ in *feature space* $\mathcal{H}_k$ induced by kernel k. We define the squared distance regularization term Q^{sq} as

$$\begin{aligned} Q^{sq}(k, \hat{\boldsymbol{\omega}}) \quad &= \quad \sum_{i=1}^{n} \|\phi_k(\mathbf{x}_i) - \phi_k(\hat{\boldsymbol{\omega}}_i)\|_2^2 \\ &= \quad \sum_{i=1}^{n} k(\mathbf{x}_i, \mathbf{x}_i) - 2k(\mathbf{x}_i, \hat{\boldsymbol{\omega}}_i) + k(\hat{\boldsymbol{\omega}}_i, \hat{\boldsymbol{\omega}}_i). \end{aligned}$$

Note that when using Q^{sq}, step 3 of Algorithm 1 becomes a convex minimization procedure.

5.3. Imputing the Mean in Feature Space

In principle any imputation we believe is useful for learning a good classifier can be used as $\hat{\boldsymbol{\omega}}$. Several models of the data can be assumed to compute corresponding optimal imputations. We like to mention one interesting model, namely the class-based mean imputation in the *feature space* $\mathcal{H}_k$ induced by kernel k. This model imputes missing values such that the sum of squared distances between completed instances to the class-dependent mean in feature space is minimal over all possible imputations. $\hat{\boldsymbol{\omega}} = \arg\min_{\boldsymbol{\omega}} \sum_{i=1}^{n} \|\phi_k(\boldsymbol{\omega}_i) - \frac{1}{n_{y_i}}\sum_{j:y_j=y_i} \phi_k(\boldsymbol{\omega}_j)\|_2^2$, where n_y denotes the number of instances with label y. Simple algebraic manipulations show that this is equivalent to

minimizing the sum of squared distances between all instances $\sum_{v\in\{-1,1\}} \frac{1}{n_v} \sum_{i,j:y_i=y_j=v} \|\phi_k(\boldsymbol{\omega}_i) - \phi_k(\boldsymbol{\omega}_j)\|_2^2 = \sum_{v\in\{-1,1\}} \frac{1}{n_v} \sum_{i,j:y_i=y_j=v} \big[k(\boldsymbol{\omega}_i,\boldsymbol{\omega}_i) - 2k(\boldsymbol{\omega}_i,\boldsymbol{\omega}_j) + k(\boldsymbol{\omega}_j,\boldsymbol{\omega}_j)\big]$

Definition 1 (Mean in Feature Space). *The class-based* mean in feature space *imputation method imputes missing values $\hat{\boldsymbol{\omega}}$ which optimize*

$$\hat{\boldsymbol{\omega}} = \arg\min_{\boldsymbol{\omega}} \sum_{v\in\{-1,+1\}} \frac{1}{n_v} \sum_{i,j:y_i=y_j=v} \big[k(\boldsymbol{\omega}_i,\boldsymbol{\omega}_i) - 2k(\boldsymbol{\omega}_i,\boldsymbol{\omega}_j) + k(\boldsymbol{\omega}_j,\boldsymbol{\omega}_j)\big]$$

Note that this model reduces to the standard mean in input space when using the linear kernel.

6. Empirical Evaluation

We evaluate the performance of our generic approach *weighted infinite imputations* for two example realizations. We test for classification performance on the email spam data set which motivates our investigation. Furthermore, we test on seven additional binary classification problems and three regression problems.

6.1. Classification

We choose to learn the decision function for the binary classification task by substituting the risk functional of the *support vector machine*, $-R^{SVM}$, as presented in section 5.1 for R and the squared distance regularizer Q^{sq} (Section 5.2) for Q in Optimization Problem 3.

For the motivating problem setting, we assemble a data set of 2509 spam and non-spam emails, which are preprocessed by a linear text classifier which is currently in use at a large webspace hosting company. This classifier discriminates reasonably well between spam and non-spam, but there is still a small fraction of misclassified emails. The classifier has been trained on about 1 million emails from a variety of sources, including spam-traps as well as emails from the hosting company itself, recognizing more than 10 million distinct text features. On this scale, training a support vector machine with Gaussian kernel is impractical, therefore we employ a two-step procedure. We discard the contents of the emails and retain only their spam score from the text classifier and their size in bytes as content features in the second-step classifier. At the time of collection of the emails, we record auxiliary real-time information about the sending servers. This includes the number of valid and invalid receiver addresses of all emails seen from the server so far, and the mean and standard deviation of the sizes and spam scores of all emails from the server. Such information is not available for emails from external sources, but will be available when classifying unseen emails. We randomly draw 1259 emails, both spam and non-spam, with server information, whereas half of those were drawn from a set of misclassified spam-emails. We augment this set with 1250 emails drawn randomly from a source without server information for which only 2 of the 8 attributes are observed.

To evaluate the common odd versus even digits discrimination, random subsets of 1000 training examples from the USPS handwritten digit recognition set are used. We test on the remaining 6291 examples. Additionally, we test on KDD Cup 2004 Physics (1000 train, 5179 test, 78 attributes) data set and on the 4-view land mine detection data (500, 213, 41) as used by Williams and Carin (2005). In the latter, instances consist of 4 views on the data, each from a separate sensor. Consequently, we randomly select complete views as missing. From the UCI machine learning repository we take the Breast (277 instances, 9 features), Diabetes (768, 8), German (1000, 20), and Waveform (5000, 21) data sets. Selection criteria for this subset of the repository were minimum requirements on sample size and number of attributes.

On each data set we test the performance of *weighted infinite imputation* using four different regularization imputations $\hat{\boldsymbol{\omega}}$ for the regularizer $Q^{sq}(K(p),\hat{\boldsymbol{\omega}})$. These imputations are computed by *mean imputation in input space* (**MeanInput**) and *mean imputation in feature space* (**MeanFeat**) as by Definition 1. Additionally we use the *EM* algorithm to compute the attributes imputed by the maximum likelihood parameters of an assumed multivariate Gaussian distribution with no restrictions on the covariate matrix (**Gauss**), and a Gaussian Mixture Model with 10 Gauss centers and spherical covariances (**GMM**).

Four learning procedures based on single imputations serve as reference methods: the **MeanInput**, **MeanFeat**, **Gauss**, and **GMM** reference methods first determine a single imputation, and then invoke the learning algorithm.

All experiments use a spheric Gaussian kernel. Its variance parameter σ as well as the SVM-parameter η are adjusted using the regular SVM with a training and test split on fully observed data. All experiments on the same data set use this resulting parameter setting. Results are averaged over 100 runs were in each run training and test split as well as missing attributes are chosen randomly. If not stated otherwise, 85% of attributes are marked missing on all data sets. In order to evaluate our method on the email data set, we perform 20-fold cross-validation. Since the emails with

Table 1. Classification accuracies and standard errors for all data sets. Higher accuracy values are written in bold face, "∗" denotes significant classification improvement.

		MeanInput	Gauss	GMM	MeanFeat
Email	Single imp	0.9571 ± 0.0022	0.9412 ± 0.0037	0.9505 ± 0.0030	0.9570 ± 0.0022
	WII	0.9571 ± 0.0022	$\mathbf{0.9536 \pm 0.0022}$ ∗	$\mathbf{0.9527 \pm 0.0024}$	$\mathbf{0.9600 \pm 0.0019}$ ∗
USPS	Single imp	0.8581 ± 0.0027	0.8688 ± 0.0022	0.9063 ± 0.0012	0.8581 ± 0.0027
	WII	$\mathbf{0.8641 \pm 0.0027}$ ∗	$\mathbf{0.8824 \pm 0.0024}$ ∗	$\mathbf{0.9105 \pm 0.0015}$ ∗	$\mathbf{0.8687 \pm 0.0027}$ ∗
Physics	Single imp	0.6957 ± 0.0035	0.5575 ± 0.0038	0.6137 ± 0.0050	0.6935 ± 0.0028
	WII	$\mathbf{0.7084 \pm 0.0039}$ ∗	$\mathbf{0.6543 \pm 0.0055}$ ∗	$\mathbf{0.6881 \pm 0.0049}$ ∗	$\mathbf{0.7036 \pm 0.0032}$ ∗
Mine	Single imp	0.8650 ± 0.0025	0.8887 ± 0.0023	0.8916 ± 0.0023	0.8660 ± 0.0026
	WII	$\mathbf{0.8833 \pm 0.0026}$ ∗	$\mathbf{0.8921 \pm 0.0021}$	$\mathbf{0.8946 \pm 0.0022}$ ∗	$\mathbf{0.8844 \pm 0.0026}$ ∗
Breast	Single imp	0.7170 ± 0.0055	0.7200 ± 0.0048	0.7164 ± 0.0048	0.7085 ± 0.0057
	WII	$\mathbf{0.7184 \pm 0.0056}$	$\mathbf{0.7243 \pm 0.0048}$ ∗	$\mathbf{0.7212 \pm 0.0050}$ ∗	$\mathbf{0.7152 \pm 0.0057}$ ∗
Diabetes	Single imp	0.7448 ± 0.0025	0.7053 ± 0.0036	0.7154 ± 0.0043	0.7438 ± 0.0026
	WII	$\mathbf{0.7455 \pm 0.0025}$	$\mathbf{0.7234 \pm 0.0036}$ ∗	$\mathbf{0.7389 \pm 0.0031}$ ∗	$\mathbf{0.7439 \pm 0.0024}$
German	Single imp	0.7331 ± 0.0029	0.7058 ± 0.0029	0.7056 ± 0.0028	$\mathbf{0.7364 \pm 0.0029}$
	WII	$\mathbf{0.7368 \pm 0.0025}$ ∗	$\mathbf{0.7118 \pm 0.0030}$ ∗	$\mathbf{0.7120 \pm 0.0028}$ ∗	0.7357 ± 0.0027
Waveform	Single imp	0.8700 ± 0.0019	0.8241 ± 0.0031	0.7827 ± 0.0049	0.8679 ± 0.0020
	WII	0.8700 ± 0.0019	$\mathbf{0.8612 \pm 0.0019}$ ∗	$\mathbf{0.8583 \pm 0.0020}$ ∗	$\mathbf{0.8686 \pm 0.0020}$ ∗

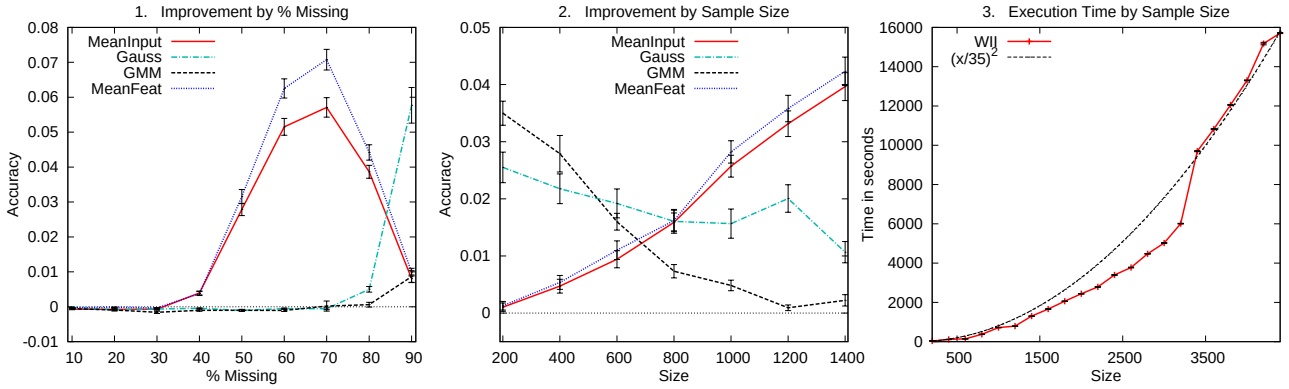


Figure 1. Detailed results on USPS classification task.

missing attributes cannot be used as test examples, the test sets are only taken from the fully observed part of the data set.

Table 6.1 shows accuracies and standard errors for the *weighted infinite imputations* (WII) method with squared distance regularization compared to all single imputations $\hat{\boldsymbol{\omega}}$ on each data set. Regularization parameter γ is automatically chosen for each run based on the performance on a separate tuning set. Baselines are obtained by first imputing $\hat{\boldsymbol{\omega}}$ and learning the classifier in a second step. The *weighted infinite imputations* method outperforms the single imputation in virtually all settings. We test for significant improvements with a paired t-test on the 5% significance level. Significant improvements are marked with a "∗" in the table.

We explore the dependence of classification performance on training sample size and the percentage of missing attribute values in more detail. The first graph in Figure 1 shows improvements in classification accuracy of our method over the single imputations depending on the percentage of missing values. Graph 2 shows classification accuracy improvements depending on the size of the labeled training set. Both experiments are performed on USPS data set and we again adjust γ separately for each run based on the performance on the tuning set. We note that similar results are obtained for the other classification problems. The *weighted infinite imputation* method can improve classification accuracy even when only 30% of the attribute values are missing. It shows, though, that it works best if at least 60% are missing, depending on $\hat{\boldsymbol{\omega}}$. On the other hand, we see that it works for all training set sizes, again depending on $\hat{\boldsymbol{\omega}}$. Similar results are obtained for the other data sets.

Table 2. Mean squared error results and standard errors for regression data sets. Smaller mean squared errors are written in bold face, "∗" denotes significant improvement.

		MeanInput	Gauss	GMM	MeanFeat
Housing	Single imp	193.0908 ± 19.9408	288.6192 ± 41.5954	160.4940 ± 16.2004	1134.5635 ± 101.9452
	WII	$\mathbf{66.5144 \pm 0.8958}$ ∗	$\mathbf{62.3073 \pm 0.8479}$ ∗	$\mathbf{66.7959 \pm 0.9173}$ ∗	$\mathbf{64.7926 \pm 0.9619}$ ∗
Ailerons	Single imp	81.7671 ± 4.5862	172.5037 ± 8.6705	79.8924 ± 4.0297	193.5790 ± 10.4899
	WII	$\mathbf{11.8034 \pm 0.1494}$ ∗	$\mathbf{8.7505 \pm 0.0932}$ ∗	$\mathbf{11.7595 \pm 0.1530}$ ∗	$\mathbf{11.8220 \pm 0.1387}$ ∗
Cpu_act	Single imp	10454.176 ± 962.598	15000.380 ± 973.100	10123.172 ± 933.143	15710.812 ± 1099.603
	WII	$\mathbf{306.257 \pm 12.500}$ ∗	$\mathbf{204.180 \pm 5.058}$ ∗	$\mathbf{305.651 \pm 13.627}$ ∗	$\mathbf{247.988 \pm 8.010}$ ∗

To evaluate the convergence of our method, we measure classification accuracy after each iteration of the learning algorithm. It shows that classification accuracy does not change significantly after about 5 iterations for a typical γ, in this case $\gamma = 10^5$ for the USPS data set. On average the algorithm terminates after about 30-40 iterations. The computational demands of the *weighted infinite imputation* method are approximately quadratic in the training set size for the classification task, as can be seen in Graph 3 of Figure 1. This result depends on the specific risk functional R and its optimization implementation. Nevertheless, it shows that risk functionals which are solvable in quadratic time do not change their computational complexity class when learned with incomplete data.

6.2. Regression

We evaluate the *weighted infinite imputations* method on regression problems using the squared error as loss function. Consequently, risk functional R^{KRR} (Sect. 5.1) is used as R and again the squared distance regularizer Q^{sq} for Q in Optimization Problem 3. From UCI we take the Housing data (506, 14), and from the Weka homepage cpu_act (1500, 21) and ailerons (2000, 40). Ridge parameter η and RBF-kernel parameter σ were again chosen such that they lead to best results on the completely observed data. Regularization parameter γ was chosen based on the performance on a tuning set consisting of 150 examples. Results are shown in Table 2. We can see that our method outperforms the results obtained with the single imputations significantly for all settings.

7. Conclusion

We devised an optimization problem for learning decision functions from incomplete data, where the distribution p of the missing attribute values is a free parameter. The investigated method makes only minor assumptions on the distribution by the means of a regularizer on p that can be chosen freely. By simultaneously optimizing the function and the distribution of imputations, their dependency is taken into account properly. We presented a proof that the optimal solution for the joint learning problem concentrates the density mass of the distribution on finitely many imputations. This justifies the presented iterative algorithm that finds a solution. We showed that instantiations of the general learning method consistently outperform single imputations.

Acknowledgments

We gratefully acknowledge support from STRATO Rechenzentrum AG.

References

Argyriou, A., Micchelli, C., & Pontil, M. (2005). Learning convex combinations of continuously parameterized basic kernels. *Proceedings of the 18th Conference on Learning Theory.*

Chechik, G., Heitz, G., Elidan, G., Abbeel, P., & Koller, D. (2007). Max-margin classification of incomplete data. *Advances in Neural Information Processing Systems 19.*

Liao, X., Li, H., & Carin, L. (2007). Quadratically gated mixture of experts for incomplete data classification. *Proceedings of the 24th International Conference on Machine learning.*

Micchelli, C., & Pontil, M. (2007). Feature space perspectives for learning the kernel. *Machine Learning, 66.*

Shivaswamy, P. K., Bhattacharyya, C., & Smola, A. J. (2006). Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research, 7.*

Smola, A., Vishwanathan, S., & Hofmann, T. (2005). Kernel methods for missing variables. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics.*

Williams, D., & Carin, L. (2005). Analytical kernel matrix completion with incomplete multi-view data. *Proceedings of the ICML 2005 Workshop on Learning With Multiple Views.*

Williams, D., Liao, X., Xue, Y., & Carin, L. (2005). Incomplete-data classification using logistic regression. *Proceedings of the 22nd International Conference on Machine learning.*

An Object-Oriented Representation for Efficient Reinforcement Learning

Carlos Diuk CDIUK@CS.RUTGERS.EDU
Andre Cohen ACOHEN@CS.RUTGERS.EDU
Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
RL^3 Laboratory, Department of Computer Science, Rutgers University, Piscataway, NJ USA

Abstract

Rich representations in reinforcement learning have been studied for the purpose of enabling generalization and making learning feasible in large state spaces. We introduce Object-Oriented MDPs (OO-MDPs), a representation based on objects and their interactions, which is a natural way of modeling environments and offers important generalization opportunities. We introduce a learning algorithm for deterministic OO-MDPs and prove a polynomial bound on its sample complexity. We illustrate the performance gains of our representation and algorithm in the well-known Taxi domain, plus a real-life videogame.

1. Introduction

In the standard Markov Decision Process (MDP) formalization of the *reinforcement-learning* (RL) problem (Sutton & Barto, 1998), a decision maker interacts with an environment consisting of finite state and action spaces. Algorithms for RL in MDP environments suffer from what is known as the *curse of dimensionality*: an exponential explosion in the total number of states as a function of the number of *state variables*. Learning in environments with extremely large state spaces is challenging if not infeasible without some form of generalization. Exploiting the underlying structure of a problem can enable generalization and has long been recognized as important in representing sequential decision tasks (Boutilier et al., 1999).

In this paper, we propose an extension to the standard MDP formalism, which we call *Object-Oriented MDPs (OO-MDPs)*, and present an efficient learning algorithm for deterministic OO-MDPs. We claim that this object-based approach is a natural way of viewing and describing many real-life domains that enables multiple opportunities for generalization. There are many ways of incorporating objects into models for learning and decision making—this paper explores one particular approach as a first attempt to understand the issues that arise.

Our representation has multiple connections with other formalisms proposed in the Relational Reinforcement Learning literature (van Otterlo, 2005), but emphasizes simplicity and tractability over expressive power. Our representation starts from attributes that can be directly perceived by the agent, rather than predicates or propositions introduced by the designer (although we allow the encoding of prior knowledge in propositional form). A similar formalism, *relational MDPs (RMDPs)*, was introduced by Guestrin et al. (2003) in the context of planning, and is based on the same insight. While our formalism has similarities to *RMDPs*, we introduce a number of changes, mainly in the way transition dynamics are described, to enable efficient learning and generalization.

To present and test our approach, we first provide benchmark experiments in the well-known Taxi domain (Dietterich, 2000). We further demonstrate its applicability by designing an agent that can solve an interesting problem in the real-life videogame *Pitfall*[1].

2. Notation

We use a standard Markov Decision Process (MDP) notation throughout this paper (Puterman, 1994). A finite MDP M is a five tuple $\langle \mathcal{S}, \mathsf{A}, T, \mathcal{R}, \gamma \rangle$. We use $T(s'|s,a)$ to denote the transition probability of state s' given state–action pair (s,a) and $R(s,a)$ to denote the expected reward value. A *deterministic MDP* is one in which there is a single next state s' for every given state s and action a; that is, $\forall s \in \mathcal{S}, a \in \mathsf{A}, \exists s' \in \mathcal{S} : T(s'|s,a) = 1$.

3. Object-oriented Representation

We will use the Taxi domain, defined by Dietterich (2000), as an example to introduce our formalism. Taxi is a grid-

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] ©1982 Activision, Inc.

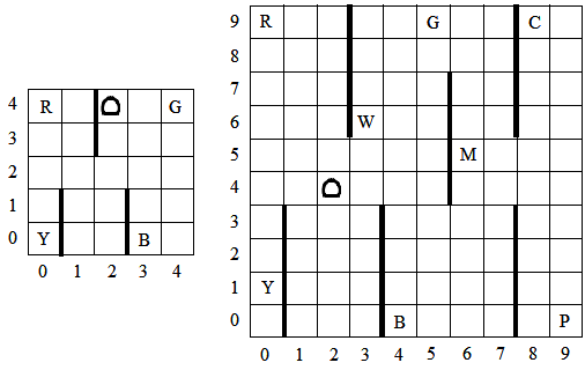


Figure 1. The taxi domain. (a) Original 5×5 Taxi problem. (b) Extended 10×10 version, with a different wall distribution and 8 possible passenger locations and destinations.

world domain (see Figure 1.a), where a *taxi* has the task of picking up a *passenger* in one of a pre-designated set of locations (identified in the figure by the letters *Y, G, R, B*) and dropping it off at a goal *destination*, also one of the pre-designed locations. The set of actions for the taxi are *North*, *South*, *East*, *West*, *PICKUP* and *DROPOFF*. Walls in the grid limit the taxi's movements.

A common factored-state representation for the Taxi problem uses Dynamic Bayesian Networks (DBNs) to indicate how state variables influence each other. For example, the location of the *taxi* after a *North* action only depends on its current location and is independent of the *passenger* or *destination* variables.

We depart from this representation and introduce one based on objects and their interactions. Many elements in our representation are similar to those of *relational MDPs* (Guestrin et al., 2003) with significant differences in the way we represent transition dynamics. Similar to *RMDPs*, we define a set of *classes* $\mathcal{C} = \{C_1, \ldots, C_c\}$. Each class includes a set of *attributes* $Att(C) = \{C.a_1, \ldots, C.a_a\}$, and each attribute has a *domain* $Dom(C.a)$. A particular environment will consist of a set of *objects* $O = \{o_1, \ldots, o_o\}$, where each object is an instance of one class: $o \in C_i$. The state of an object o.*state* is a value assignment to all its attributes. The state of the underlying MDP is the union of the states of all its objects: $s = \bigcup_{i=1}^{o} o_i.state$.

An OO-MDP representation of Taxi has four object classes: *Taxi*, *Passenger*, *Destination* and *Wall*. *Taxi*, *Passenger* and *Destination* have attributes x and y, which define their location in the grid. *Passenger* also has a Boolean attribute *in-taxi*, which specifies whether the passenger is inside the taxi. *Walls* have an attribute that indicates their position in the grid. The Taxi domain, in its 5×5 version shown in Figure 1.a, has one object of each class *Taxi*, *Passenger*, and *Destination*, and multiple (26) objects of class *Wall*. This list of objects points out a significant feature of the OO-MDP representation. Whereas in the classical MDP model, the effect of encountering walls is felt as a property of specific locations in the grid, the OO-MDP view is that wall interactions are the same regardless of their location. As such, agents' experience can transfer gracefully throughout the state space.

When two objects interact in some way, they define a *relation* between them. A combination of the relation established, plus the internal states of the two objects, determines an *effect*—a change in value of one or multiple attributes in either or both interacting objects. This behavior is defined at the class level, meaning that different objects that are instances of the same class behave in the same way when interacting with other objects. Formally, a relation $r : C_i \times C_j \rightarrow$ *Boolean* is a function, defined at the class level, over the combined attributes of objects of classes C_i and C_j. Its value gets defined when instantiated by two objects $o_1 \in C_i$ and $o_2 \in C_j$. For our Taxi representation, we will define 5 relations: $touch_N(o_1, o_2)$, $touch_S(o_1, o_2)$, $touch_E(o_1, o_2)$, $touch_W(o_1, o_2)$ and $on(o_1, o_2)$, which define whether an object $o_2 \in C_j$ is exactly one cell North, South, East or West of an object $o_1 \in C_i$, or if both objects are overlapping (same x, y coordinates). Different domains require different relations.

When the object $taxi_i \in$ *Taxi* is on the northern edge of the grid and tries to perform a *North* action, it hits some object $wall_j \in$ *Wall* and the observed behavior is that it doesn't move. We say that a $touch_N(taxi_i, wall_j)$ relation has been established and the effect of an action *North* under that condition is *no-change*. On the other hand, if $\neg touch_N(taxi_i, wall_j)$ is true and the taxi performs the action *North*, the effect will be $taxi_i.y \leftarrow taxi_i.y + 1$. As stated before, these behaviors are defined at the class level, so we can refer in general to the relation $touch_N(Taxi, Wall)$ as producing the same kind of effects on any instance of $taxi_i \in$ *Taxi* and $wall_j \in$ *Wall*.

We define some properties of these transition dynamics more formally in the next section.

3.1. Transition Dynamics

Every state s induces a certain value assignment to all attributes of all objects—and therefore all relations—in the domain. Transitions are determined by interactions between objects. Every pair of objects $o_1 \in C_i$ and $o_2 \in C_j$, their internal states $o_1.state$ and $o_2.state$, an action a, and the set of relations $r(o_1, o_2)$ that are true—or false—at the current state, determine an effect—a change of value in some of the objects' attributes.

Definition 1 *An* effect *is a single operation over a single attribute att in the OO-MDP. We will group effects into* types, *based on the kind of operation they perform. Ex-*

amples of types are arithmetic *(increment att by* 1*, subtract* 2 *from att), and* constant assignment *(set att to* 0*).*

Definition 2 *A* term t *is any Boolean function. In our OO-MDP representation, we will consider terms representing either a relation between two objects, a certain possible value of an attribute of any of the objects or, more generally, any Boolean function defined over the state space that encodes prior knowledge. All transition dynamics in an OO-MDP are determined by the different possible settings of a special set of terms called* T*.*

Definition 3 *A* condition *is a set* T_c *of terms and negations of terms from* T *that* must *be true in order to produce a particular effect* e *under a given action* a*.*

We can summarize an OO-MDP transition cycle as follows:

1: **while** agent is acting **do**
2: Agent observes current state s and returns action a.
3: From state s, the environment extracts all relations that currently hold between objects and observes the value of all attributes of all objects, assigning a True/False value to all terms in T.
4: For each (if any) fulfilled condition in T_c, there's an effect that will occur, determining a set of effects to be applied to s.
5: If no conditions were fulfilled, no change takes place to s.
6: Otherwise, the environment uses the set of effects to compute s'. New state $s \leftarrow s'$.
7: The environment chooses a reward r from $R(s, a)$.
8: Agent is told r.
9: **end while**

4. *DOORMAX*: Learning and Solving Deterministic OO-MDPs

We introduce Deterministic Object-Oriented Rmax (*DOORMAX*), an algorithm for learning and solving deterministic OO-MDPs. *DOORMAX* is correct and, as we will show, provably efficient under the following assumptions.

Assumption 1 *For each action and each attribute, only effects of one type can occur.*

Assumption 2 *For every action* a*, attribute att and effect type* t*, there is a set* $\mathcal{CE}$ *of* condition–effect *pairs that determine changes to att given* a*. No effect can appear twice on this list, and there are at most* k *different pairs—*$|\mathcal{CE}| \leq k$*. Plus, no conditions* T_i *and* T_j *in the set* $\mathcal{CE}$ *contain each other:* $\neg(T_i \subset T_j \vee T_j \subset T_i)$*. The number of terms or negations of terms in any condition is bounded by a known constant* M*.*

Assumption 3 *Effects are* invertible*, that is, given states* s *and* s'*, for each attribute att and each effect type we can determine a unique effect that would transform att from its value in* s *to its value in* s'*.*

4.1. Definitions and Data Structures

We introduce some definitions, notation, and data structures that will be used to describe *DOORMAX*:

- T is the union of all terms t that will be involved in the conditions that determine the transition dynamics of the environment described by the OO-MDP, plus their negations $\neg t$, with $|T| = 2n$.
- For every state $s \in S$, the function *cond*(s) returns the subset of terms in T that are true in s.
- A condition $T_c \subseteq T$ is represented by a string c_S of length n, where $c_S^i = 1$ if $t_i \in T_c$, $c_S^i = 0$ if $\neg t_i \in T_c$ and $c_S^i =$ * if $t_i \notin T_c \wedge \neg t_i \notin T_c$.
- Given two conditions represented as strings c_1 and c_2, we define the commutative operator $\oplus : c \times c \rightarrow c$ as follows:

c_1	c_2	$c_1 \oplus c_2$
0	0	0
1	1	1
0	1	*
0|1	*	*

- A condition c_1 *matches* another condition c_2, noted $c_1 \models c_2$, if $\forall 1 \leq i \leq n : c_1^i = * \vee c_1^i = c_2^i$.
- For any states s and s' and attribute *att*, the function *eff*$_{att}(s, s')$ returns one effect of each type that would transform attribute *att* in s into its value in s'.
- A *prediction* p is a pair $(p.model, p.\text{effect})$, where $p.model$ is a condition that represents the set of terms that need to be true for $p.$effect to occur.
- For each action a, each attribute *att* and each effect type *type*, a set of predictions *pred*$(a, att, type)$ is maintained. We refer to the set of *models* in a set of predictions as *pred*$(a, att, type).models$.
- If an action a produces no effect from a given state s ($s' = s$), we call the induced condition *cond*(s) a *failure condition*. We define F_a to be a set of failure conditions for action a.
- Two effects are *incompatible* if, for any initial value of an attribute, applying these two effects would yield two different final values.

4.2. OO-MDP Representation of Taxi

To facilitate understanding of the notation and data structures, we present a full example of our representation in the Taxi domain.

The set of terms T, which determines the transition dynamics of the OO-MDP, includes the four $touch_{N/S/E/W}$ relations between the taxi and the walls; the relevant relations between the taxi and the passenger and destination; the attribute value *passenger.in-taxi* $= T$; and all their negations:

$$\{\ touch_{N/S/E/W}(taxi, wall),\quad on(taxi, passenger),\quad \neg touch_{N/S/E/W}(taxi, wall),\quad \neg on(taxi, passenger),\quad on(taxi, destination),\quad \neg on(taxi, destination),\quad passenger.in\text{-}taxi = T,\quad passenger.in\text{-}taxi = F\ \}$$

Consider the state s where the taxi is in position $(2, 4)$ (as in Figure 1.a), the passenger is inside the taxi, and the destination is G. For this state, the function *cond*(s) returns:

$$\{\ touch_N(taxi, wall),\quad \neg touch_S(taxi, wall),\quad \neg touch_E(taxi, wall),\quad touch_W(taxi, wall),\quad \neg on(taxi, passenger),\quad \neg on(taxi, destination),\quad passenger.in\text{-}taxi = T\ \}.$$

The corresponding 7-character string representation for this condition is 1001001, following the prior order for the terms.

Let's now assume that the agent tries to perform the action *East*, which takes it to state s' where the taxi is in location $(3, 4)$. The corresponding *cond*(s') is similar, except that now the taxi is not touching a wall to its West ($\neg touch_W(taxi, wall)$). The corresponding string representation of the new condition is: 1000001. The observed effect is that the taxi moved to location $(3, 4)$. In our representation, two effect types are allowed: arithmetic and constant assignment. Therefore, the function $eff_{taxi.x}(s, s')$ will return two values: increment(1) and set-to(3).

Now, the agent takes another *East* action, and gets to state s'', where location is $(4, 4)$, it's touching a wall to the East and standing on the destination. *cond*(s'') can now be represented as 1010011. The two observed effects to $taxi.x$ are increment(1) and set-to(4). Note that the transition model for an OO-MDP need not predict the changes to the conditions, only to the attributes. The condition values are then derived separately using the knowledge of the relevant relations and their definitions.

Finally, we'll consider separately the actions that produce no effect. Let's assume the agent also attempted an action *North* from each of the previous states, which resulted in it hitting a wall and staying in the same state. We treat these cases differently: The corresponding conditions 1001001, 1000001 and 1010011 will be identified as failure conditions for action *North* and incorporated into the set F_{North}. Whenever we observe a new condition c_i such that any existing condition in F_{North} matches it, we predict that performing a *North* action will have no effect.

4.3. Learning Algorithm

The *DOORMAX* algorithm (Algorithm 1) follows the general structure of most RL algorithms in the Rmax family, which work as follows. Using examples of transitions (s, a, s'), a learning algorithm constructs the transition model T. The learning algorithm must satisfy the KWIK (knows what it knows) conditions (Li et al., 2008), which say: (1) all predictions must be accurate (assuming a valid hypothesis class), and (2) however, the learning algorithm may also return $\perp$, which indicates that it cannot yet predict the output for this input. The sample complexity or KWIK bound of a learning algorithm is the maximum number of times it returns $\perp$. In the Rmax setting, any transition that cannot yet be predicted is assumed to lead to a fictious $s_{\max}$ state from which maximum reward can be obtained.

Algorithm 1 *DOORMAX*: `main()` method

```
// Set up data structures:
for all actions a ∈ A do
    F_a ← ∅
    for all attributes att ∈ ⋃_{c∈C} Att(c) do
        for all effect types type do
            pred(a, att, type) ← ∅
            Add pred(a, att, type) to set of active predictions P
        end for
    end for
end for
while ¬(Termination criterion) do
    Observe current state s.
    Choose action a according to exploration policy, based on prediction for T(s'|s, a) returned by predictTransition(s, a).
    Observe new state s'.
    Update learned model using method addExperience(s, a, s', k).
end while
```

The two main routines of the algorithm are `predictTransition` (Algorithm 2), which predicts the next state given a current state and action based on the current model, and `addExperience` (Algorithm 3), which learns a model of the OO-MDP. If `predictTransition` is not able to predict a next state with accuracy, it returns $s_{\max}$.

To help understand these routines, we present a couple of intuitions, based on the Taxi examples presented in the previous section. Notice that if we applied the $\oplus$ operator to

cond(s) and *cond*(s'), the two conditions from which an *East* action produced an increment(1) effect, we would obtain: $1001001 \oplus 1000001 = 100$*$001$. The resulting condition indicates that the term $\textit{touch}_W(\textit{wall}, \textit{taxi})$ is irrelevant with respect to action *East* and effect increment(1). If we also compare the two pairs of effects obtained, we observe that we consistently observed increment(1), whereas set-to(3) and set-to(4) are incompatible effects. These observations constitute the central ideas for the learning algorithm.

Algorithm 2 `predictTransition(s,a)` method

Inputs: state s and action a.
Output: a predicted state $s' \in S \cup \{s_{\max}\}$.
if $\exists c \in F_a$ s.t. $c \models$ *cond*(s) **then**
  // The current condition is a known failure condition.
  Return s
else
  for all attributes *att* $\in \bigcup_{c \in C}$ *Att*(c) **do**
    $E \leftarrow \emptyset$
    for all effect types *type* **do**
      if $\exists p \in$ *pred*$(a, \textit{att}, \textit{type})$ s.t. $p.\textit{model} \models \textit{cond}(s)_S$ **then**
        Add p.effect to E
      end if
    end for
    if $E = \emptyset \vee \exists e_i, e_j \in E$ s.t. e_i and e_j are incompatible **then**
      Return $s_{\max}$
    else
      // Set E contains all the individual operations that need to be applied to attributes in s in order to convert it to s'.
      $s' \leftarrow$ apply E to s
      Return s'
    end if
  end for
end if

5. Analysis

Under the current assumptions, the effects of a given action on a given attribute assuming effects of a given type can be learned with a worst-case bound of $O(n^M)$, where $n = |T|$ is the number of terms and M is the maximum number of terms involved in any of the conditions. This worst-case bound can be guaranteed by a variant of *SLF-Rmax*, an algorithm introduced by Strehl et al. (2007).

The uniqness assumption, Assumption 2, is not needed for *SLF-Rmax* to achieve this worst-case bound. However, *DOORMAX*, by taking advantage of this assumption, is able to learn faster in many domains. Some empirical evidence to support this claim appears in Section 6.

If we assume M is a constant, SLF-Rmax can be used to provide guaranteed efficient results. However, for many domains DOORMAX will result much more efficient in practice. We conjecture that the two approaches can be run in parallel, to achieve the best of both.

Intuitively, the good empirical results of DOORMAX lie in the way condition-effects are learned each time they are observed. The worst-case occurs when the agent observes an exponential amount of failures before observing instances of the set of effects it needs to learn.

We now show that the problem of learning the transition dynamics of an OO-MDP has polynomial sample complexity in the KWIK setting, when by sample we only refer to the cases where an effect is observed (as opposed to failure samples where $s' = s$).

We split the proof in two parts. First, we show that learning the right (condition, effect) pairs for a single action and attribute is KWIK-learnable, and then we show that learning the right effect type for each action–attribute, given all the possible effect types, is also KWIK learnable.

Theorem 1 *The transition model for a given action a, attribute att and effect type type in a deterministic OO-MDP is KWIK-learnable with a bound of $O(nk+k+1)$, where n is the number of terms in a condition and k is the maximum number of effects per action–attribute.*

Proof:

Given state s and action a, the predictor for effect type *type* will return $\perp$ if *cond*(s) is not a known failure condition and there is no condition in *pred*$(a, \textit{att}, \textit{type})$ that matches *cond*(s). In that case, it gets to observe s' and updates its model with *cond*(s) and the observed effect e. We show that the number of times the model can be updated until it always has a correct prediction is $O(nk + k + 1)$:

- if the effect e has never been observed before for this particular action, attribute and effect type, it gets added to *pred*$(a, \textit{att}, \textit{type})$. This outcome happens at most k times, which is the maximum number of different effects allowed per action-attribute-type combination.
- if the effect e has never been observed, but $|\textit{pred}(a, \textit{att}, \textit{type})| = k$, the algorithm concludes that the current effect type is not the correct one for this action–attribute, and it removes all predictions of this type from its set $\mathcal{P}$. This event can only happen once.
- if the effect e is such that there already exists a prediction for it, $\perp$ is only returned if the existing condition

in the model does not match *cond*(s). This case can only happen if a term in the model is a 0 or 1 and the observation is the opposite. Once it happens, that term becomes a *, so there will never be another mismatch for that term, as * matches either 0 or 1. In the worst case, with every $\perp$ returned, one term at a time gets converted into *. These updates can only happen n times for each effect in *pred*$(a, att, type)$, for a total of nk times.

Therefore, there can be at most $nk + k + 1$ updates to the model for a particular action a, attribute *att* and effect type *type* before *pred*$(a, att, type)$ either has a correct prediction or gets eliminated. □

Corollary 1 *The transition model for a given action and attribute in a deterministic OO-MDPs is KWIK-learnable with a bound of $O(h(nk + k + 1) + (h - 1))$, where n is the number of terms in a condition, k is the max number of effects per action–attribute, and h is the number of effect types.*

Proof: Whenever *DOORMAX* needs to predict s' given state s and action a, it will consult its current predictions for each attribute and effect type. It will return $\perp$ if:

- for any of the h effect types $type_i$, *pred*$(a, att, type_i)$ returns $\perp$. As shown in Theorem 1, *pred*$(a, att, type_i)$ can only return $\perp$ up to $nk + k + 1$ times. Therefore, this case can only happen $h(nk + k + 1)$ times.
- for some attribute *att*, there are two effect types $type_1$ and $type_2$ such that *pred*$(a, att, type_1) \neq$ *pred*$(a, att, type_2)$. When this happens, we get to observe the actual effect e, which will necessarily mismatch one of the predictions. The model will therefore be updated by removing either *pred*$(a, att, type_1)$ or *pred*$(a, att, type_2)$ from its set of predictions. This case can only occur $h - 1$ times for a given action and attribute.

We have shown that, in total, *DOORMAX* will only predict $\perp$ $O(h(nk + k + 1) + (h - 1))$ times before having an accurate model of the transition dynamics for an action and attribute in the OO-MDP. □

6. Experiments

First, we use the Taxi domain to demonstrate how *DOORMAX* makes use of the OO-MDP representation to outperform *Factored-Rmax*, an algorithm based on a factored-state MDP representation. Second, we show how *DOORMAX* and *Factored-Rmax* scale when the size of the state space increases, by comparing them on the 10×10 version of Taxi. Finally, we demonstrate how *DOORMAX* can be

Algorithm 3 `addExperience(s,a,s',k)` method

Inputs: an observation $< s, a, s' >$; k, the maximum number of different effects possible for any action, attribute and effect type.
if $s = s'$ **then**
  // Found a failure condition for action a, update F_a
  Remove all $c \in F_a$ s.t. *cond*$(s) \models c$.
  $F_a \leftarrow F_a \cup \{$*cond*$(s)\}$
else
  for all attributes *att* $\in \bigcup_{c \in C}$ *Att*(c) **do**
    for all $e \in$ *eff*$_{att}(s, s')$ **do**
      Find a prediction $p \in$ *pred*$(a, att, e.type)$ such that p.effect $= e$
      if $\exists p$ **then**
        // We already have a (condition, effect) prediction for current a, *att*, and *type*. Update condition and verify that there are no overlaps.
        $p.model \leftarrow p.model \oplus cond(s)_S$.
        if $\exists c \in$ (*pred*$(a, att, e.type) \setminus p$).*models* s.t. $p.model \models c$ **then**
          // Conditions overlap, violating an assumption, meaning it is not the right *type* of effect for this action and attribute.
          Remove *pred*$(a, att, e.type)$ from $\mathcal{P}$
        end if
      else
        // We observed an effect for which we had no prediction. If its condition does not overlap an existing condition, then add this new prediction.
        if $\exists c \in$ *pred*$(a, att, e.type)$.*models* s.t. *cond*$(s) \models c \vee c \models$ *cond*(s) **then**
          Remove *pred*$(a, att, e.type)$ from $\mathcal{P}$
        else
          Add (*cond*$(s), e)$ to *pred*$(a, att, e.type)$.
          // Verify that there aren't more than k predictions for this action, attribute and type.
          if $|$*pred*$(a, att, e.type)| > k$ **then**
            Remove *pred*$(a, att, e.type)$ from $\mathcal{P}$
          end if
        end if
      end if
    end for
  end for
end if

applied to succesfuly model and solve a real-life problem, the *Pitfall* videogame.

6.1. Taxi

The first experiments we present are based on the Taxi domain previously introduced. We run experiments on two versions: the original 5×5-grid version presented by Dietterich (2000), which consists of 500 states, and an extended 10×10-grid version with 8 passenger locations and destinations, with 7200 states (see Figure 1). The purpose of the extended version is to demonstrate how *DOORMAX* scales by properly generalizing its knowledge about conditions and effects when more objects of the same known classes are introduced.

We compare *DOORMAX* against *Factored-Rmax*, an algorithm from the Rmax family that uses a factored-state MDP and models transitions using a DBN provided as input. Both algorithms are model based and use Rmax-style exploration, so we hope to be able to truly compare the underlying representations.

The representation used for *DOORMAX* was described in the previous sections. In the case of *Factored-Rmax*, we provide a DBN with some derived features that make learning faster. The state variables used are the *Taxi* x and y locations, plus two Boolean features: *in-taxi*, representing whether the *passenger* is in the *taxi*, and *at-destination*, representing whether the *taxi* is standing at the *passenger*'s *destination*.

The experiments for both algorithms and both versions of the Taxi problem were repeated 100 times, and the results averaged. For each experiment, we run a series of episodes, each starting from a random start state. We evaluate the agent's learned policy after each episode on a set of six "probe" combinations of $\langle$taxi (x,y) location, passenger location, passenger destination$\rangle$. The probe states used were: $\{(2,2), Y, R\}$, $\{(2,2), Y, G\}$, $\{(2,2), Y, B\}$, $\{(2,2), R, B\}$, $\{(0,4), Y, R\}$, $\{(0,3), B, G\}$. We report the number of steps taken before learning an optimal policy for these six start states.

The results are shown in the following table, with the last column showing the ratio between the results for the 10×10 version vs the 5×5 one:

	Taxi 5×5	Taxi 10×10	Ratio
Number of states	500	7200	14.40
Factored Rmax			
# steps	1676	19866	11.85
Time per step	43.59ms	306.71ms	7.03
OO-Rmax			
# steps	529	821	**1.55**
Time per step	13.88ms	293.72ms	21.16

We can see how *DOORMAX* not only learns with significantly less sample complexity, but also how well it scales to the larger problem. After increasing the number of states by more than 14 times, *DOORMAX* only requires 1.55 times the experience.

The main difference between *DOORMAX* and *Factored-Rmax* is their internal representation, and the kind of generalization it enables. After just a few examples in which $\neg$*touch*$_N$(*taxi*, *wall*) is true, *DOORMAX* learns that the action *North* has the effect of incrementing *taxi*.y by 1, whereas under *touch*$_N$(*taxi*, *wall*) it fails. This knowledge, as well as its equivalent for *touch*$_{S/E/W}$, is generalized to all 25 (or 100) different locations. *Factored-Rmax* only knows that variable *taxi*.y' in state s' depends on its value in state s, but still needs to learn the transition dynamics for each possible value of *taxi*.y (5 or 10 different values). In the case of actions *East* and *West*, it's even worse, as walls make *taxi*.x' depend on both *taxi*.x and *taxi*.y, which are 25 (or 100) different values.

As *DOORMAX* is based on interactions between objects, it learns that the relation between taxi and wall is independent of the wall location. Each new wall is therefore the same as any known wall, rather than a new exception in the movement rules, the kind *Factored-Rmax* needs to learn.

6.2. Pitfall

Pitfall is a video game released in 1982 by Activision for the Atari game console. The goal is to have the main character (*Man*) traverse a series of screens while collecting as many points as possible while avoiding obstacles (such as holes, pits, logs, crocodiles and walls) and under the time constraint of 20 minutes. All transitions in Pitfall are deterministic. Our goal in this experiment was to have the *Man* cross the first screen from the left to the right with as few actions as possible. Figure 2 ilustrates this first screen.

Our experiments were run using a modified Atari emulator that ran the actual game and detected objects from the displayed image. We used a simple heuristic that identified objects by color clusters and sent joystick commands to the emulator to influence the play. For each frame of the game, a list of object locations was sent to an external learning module that analyzed the state of the game and returned an action to be executed before the emulator continued on to the next frame. If we consider that we start from screen pixels, the flat state representation for Pitfall is enormous: $16^{640x420}$. By breaking it down into basic objects, through an object recognition mechanism, the state space is in the order of the number of objects to the number of possible locations of each object: $6^{640x420}$. OO-MDPs allow for a very succint representation of the problem, that can be learned with only a few experience samples.

Figure 2. Initial screen of Pitfall.

The first screen contains six object types: *Man*, *Hole*, *Ladder*, *Log*, *Wall* and *Tree*. Objects have the attributes x, y, $width$ and $height$, which define their location on the screen and dimension. The *Man* also has a Boolean attribute of $direction$ that specifies which way he is facing. We extended the $touch_X$ relation from Taxi to describe diagonal relations between objects, including: $touch_{NE}(o_i, o_j)$, $touch_{NW}(o_i, o_j)$, $touch_{SW}(o_i, o_j)$ and $touch_{SE}(o_i, o_j)$. These relations were needed to properly capture the effects of moving on and off of ladders.

In our implementation of *DOORMAX*, we defined seven actions: *WalkRight*, *WalkLeft*, *JumpLeft*, *JumpRight*, *Up*, *Down* and *JumpUp*. For each of these actions, however, the emulator has to actually execute a set sequence of smaller frame-specific actions. For example, *WalkLeft* requires four frames: one to tell Pitfall to move the *Man* to the left, and three frames where no action is taken to allow for the animation of the *Man* to complete. Effects are represented as arithmetic increments or decrements to the attributes x, y, $width$, $height$, plus a constant assignment of either R or L to the attribute $direction$.

The starting state of Pitfall is fixed, and given that all transitions are deterministic, only one run of *DOORMAX* was necessary to learn the dynamics of the environment. *DOORMAX* learns an optimal policy after 494 actions, or 4810 game frames, exploring the area beneath the ground as well as the objects en route to the goal. Once the transition dynamics are learned, restarting the game results in the *Man* exiting the first screen through the right, after jumping the hole and the log, in 94 actions (905 real game frames), which is what the optimal policy requires.

A few examples of the (condition, effect) pairs learned by *DOORMAX* are shown below:

Action	Condition	Effects
WalkRight	direction $= L$	{direction $= R$, $\Delta x = +8$}
WalkRight	$touch_E(Man, Wall)$	no-effect
JumpRight	direction $= R$	$\Delta x = +214$
Up	$on(Man, Ladder)$	$\Delta y = +8$

7. Conclusions and Future Work

We introduced OO-MDPs, an object-oriented representation for reinforcement-learning problems that provides a natural way of modeling a broad set of domains, while enabling generalization. We presented *DOORMAX*, a learning algorithm for deterministic OO-MDPs that not only outperforms state-of-the-art algorithms for factored-state representations, but also scales very nicely with respect to the size of the state space, as long as transition dynamics between objects do not change. We presented bounds for learning transition dynamics of determinstic OO-MDPs in the KWIK framework.

One limitation of our work is that we do not yet have a provably efficient algorithm for stochastic domains, which is part of our future work. While OO-MDPs can model stochastic transitions, a more complex learning algorithm would be needed to learn transitions effectively in the face of noise.

The second component of our future research is the extension of the object-oriented model to be able to handle inheritance. We hope to be able to exploit knowledge about objects being part of a common super-class to learn their behaviors faster. Ideally, algorithms could also learn the object definitions and classes automatically, as well.

References

Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1–94.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303.

Guestrin, C., Koller, D., Gearhart, C., & Kanodia, N. (2003). Generalizing plans to new environments in relational mdps. *IJCAI* (pp. 1003–1010).

Li, L., Littman, M. L., & Walsh, T. J. (2008). Knows what it knows: A framework for self-aware learning. *Twenty-Fifth International Conference on Machine Learning*.

Puterman, M. L. (1994). *Markov decision processes—discrete stochastic dynamic programming*. New York, NY: John Wiley & Sons, Inc.

Strehl, A. L., Diuk, C., & Littman, M. L. (2007). Efficient structure learning in factored-state mdps. *AAAI* (pp. 645–650). AAAI Press.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. The MIT Press.

van Otterlo, M. (2005). *A survey of reinforcement learning in relational domains* (Technical Report TR-CTIT-05-31). CTIT Technical Report Series, ISSN 1381-3625.

Optimizing Estimated Loss Reduction for Active Sampling in Rank Learning

Pinar Donmez PINARD@CS.CMU.EDU
Jaime G. Carbonell JGC@CS.CMU.EDU
Language Technologies Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 USA

Abstract

Learning to rank is becoming an increasingly popular research area in machine learning. The ranking problem aims to induce an ordering or preference relations among a set of instances in the input space. However, collecting labeled data is growing into a burden in many rank applications since labeling requires eliciting the relative ordering over the set of alternatives. In this paper, we propose a novel active learning framework for SVM-based and boosting-based rank learning. Our approach suggests sampling based on maximizing the estimated loss differential over unlabeled data. Experimental results on two benchmark corpora show that the proposed model substantially reduces the labeling effort, and achieves superior performance rapidly with as much as 30% relative improvement over the margin-based sampling baseline.

1. Introduction

Learning to rank has recently drawn broad attention among machine learning researchers (Joachims, 2002; Freund et al., 2003; Cao et al., 2006). The objective of rank learning is to induce a mapping (ranking function) from a predefined set of instances to a set of partial (or total) orders. For instance, in recommendation systems each customer is represented with a set of features ranging from the income level to age and her preference order over a set of products (e.g. movies in Netflix). The ranking task is to learn a mapping from the feature space to a set of permutations of the products. The applications include document retrieval, collaborative filtering, product rating, and so on. In this paper, we are interested in IR applications, and focus on document retrieval. A number of queries are provided such that each query is associated with an ordering of documents indicating the relevance of each document to the given query. Like many other ranking applications, this requires a human expert to carefully examine the documents in order to assign relevance-based permutations. It is often unrealistic to spend extensive human effort and money for labeling in ranking. Thus, it is crucial to design methods that will considerably reduce the labeling effort without significantly sacrificing ranking accuracy.

The active learning paradigm addresses this type of problem. The central idea is to start with only a small amount of labeled examples and sequentially select new examples to be labeled by an oracle. The selected examples are then added to the training set. It is clear that labeling data in ranking requires a complete (or partial) ordering of data whereas in classification labeling considers only absolute class assignments. The target domain of a set of permutations is more complex than that of absolute classes. Hence, it is even more crucial to select the most informative examples to be labeled in order to learn a ranking model using fewer labeled examples.

In this paper, we propose a novel active sampling framework for SVM rank learning (Joachims, 2002), or RankSVM in short, and RankBoost (Freund et al., 2003). The proposed method considers the capacity of an unlabeled example to update the current model if rank-labeled and added to the training set. We show that this capacity can be defined as a function that estimates the error of a ranker introduced by the addition of a new example. The capacity function takes different forms in RankSVM and RankBoost due to different formulations of the ranking function. For example in the case of RankSVM, the ranking function is defined via a normal vector which is a weighted sum of the support vectors whereas the ranking func-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

tion is a weighted sum of weak learners in RankBoost. However, in both cases, the proposed strategy selects the samples which are estimated to produce a faster convergence from the current predicted ranking to the true ranking. Our empirical evaluations on two benchmark corpora from topic distillation tasks in TREC competitions show a significant advantage favoring our method against the margin-based sampling heuristic of (Brinker, 2004; Yu, 2005) and a random sampling baseline.

The rest of the paper is organized as follows: Section 2 provides a brief literature review to the related work. Section 3 motivates the choice of the proposed active learning framework and introduces two novel methods for active learning in the RankSVM and RankBoost settings. In Section 4, we report the experimental results and demonstrate the effectiveness of our methods on benchmark datasets. Finally, we offer our conclusions and next steps in Section 5.

2. Related Work

A number of strategies have been proposed for active learning in the classification framework. Some of those center around the version space (Mitchell, 1982) reduction principle (Tong & Koller, 2000): selecting unlabeled instances that limit the volume of the version space the most, or equivalently selecting the ones with the smallest margin. Some of the others adopt the idea of reducing the generalization error (Roy & McCallum, 2001; Xu et al., 2003; Nguyen & Smeulders, 2004; Donmez et al., 2007): the selection of the unlabeled data that has the highest affect on the test error, i.e. points in the maximally uncertain and highly dense regions of the underlying data distribution (Xu et al., 2003; Nguyen & Smeulders, 2004; Donmez et al., 2007).

Unfortunately, it is not straightforward to extend these theoretical principles to ranking problems. The generalization power of ranking functions is measured by different evaluation metrics than the ones used for classification. Moreover, the classical performance metrics for ranking, such as MAP (Mean Average Precision), precision at the k^{th} rank cut-off, NDCG (Normalized Discounted Cumulative Gain), etc., are harder to directly optimize than the classical loss functions for classification, i.e. log loss, 0/1 loss, squared loss, etc.

Recently, there have been attempts to address the challenges in active sampling for rank learning. Brinker (2004) uses a notion of the margin as an approximation to reducing the volume of the version space. The margin in the ranking scenario is defined as the minimum difference of scores between two instances assuming the ranking solution is a real-valued scoring function. Yu (2005) adopted the same notion of margin for SVM rank learning and proposed a batch mode for instance selection that minimizes the sum of the rank score differences of all data pairs within a set of samples. Yu (2005) proposed an efficient implementation which considers only the rank-adjacent pairs and showed that this strategy is optimal in terms of selecting the most ambiguous set of samples with respect to the ranking function. The major drawback of this margin-based sampling method of (Brinker, 2004; Yu, 2005) is that a scoring function for ranking may assign very similar scores to instances with the same rank label since the ranking function does not distinguish between the relative order of two relevant or two non-relevant examples. However, such instances do not carry any additional information for the rank learner to distinguish between the relevant and the non-relevant data.

Another recent development in active rank learning is the divergence-based sampling method of (Amini et al., 2006). The proposed method selects the samples at which two different ranking functions maximally disagree. One of the two functions is the current ranking function trained on the labeled data, and the other is a randomized function obtained by cross validation. The divergence-based strategy is effective only when provided with a sufficiently large initial labeled set, which is impractical for many real-world ranking applications, such as document retrieval.

3. Active Sampling in Rank Learning

3.1. Motivation

This section presents a novel method for active learning using RankSVM and RankBoost. Roy and McCallum (2001) argue that an optimal active learner is the one that asks for the labels of the examples that, once incorporated into training, would result in the lowest expected error on the test set. The expected error on the test set can be estimated using the posterior distribution $\hat{P}_D(y \mid x)$ of class labels estimated from the training set using some loss function L

$$E_{\hat{P}_D} = \int_x L(P(y \mid x), \hat{P}_D(y \mid x))P(x) \qquad (1)$$

Their aim is then to select the point x^* such that when added to the training set with a chosen label y^*, the classifier trained on the new set $\{D + (x^*, y^*)\}$ would have less error than any other candidate x.

$$\forall(x, y) E_{\hat{P}_{D+(x^*,y^*)}} \leq E_{\hat{P}_{D+(x,y)}} \qquad (2)$$

Since the true label y^* is unknown, the expectation calculation is carried out by calculating the estimated error for each possible label $y \in Y$, and then taking the average weighted by the current learner's posterior $\hat{P}_D(y \mid x)$. The naive implementation of this method would be quite inefficient and almost intractable on large datasets. Roy and McCallum (2001) address this problem using fast updates for a Naive Bayes classifier. Although efficient re-training procedures are available for some learners such as SVMs (Cauwenberghs & Poggio, 2000), it would still be infeasible for ranking tasks, especially considering the interactive nature of ranking systems. In this paper, we propose a method to estimate how likely the addition of a new example will result in the lowest expected error on the test set *without any re-training on the enlarged training set*. Our method is based on the likelihood of an example to change the current hypothesis significantly. There are a number of reasons why we believe this is a reasonable indicator for estimating that error:

- Adding a new data point to the labeled set can only change the error on the test set if it changes the current learner.
- The more significant that change, the greater chance to learn the true hypothesis faster.
- We note that a big change in the current hypothesis might not always lead to better generalization. However, as more data is sampled and the hypothesis gets closer to the truth, it is less likely that a single outlier could hurt the performance noticeably.

In the following sections, we briefly review the RankSVM and the RankBoost algorithms and propose a novel active learning method for each.

3.2. Preliminaries

Assume the data is represented as a set of feature vectors $\vec{x} \in \mathbb{R}^d$ and corresponding labels (ranks) $y \in Y = \{r_1, r_2, ..., r_n\}$ where n denotes the number of ranks. We assume binary relevance in this paper, though our framework can be generalized to multi-level ranking scenarios as long as the rank learner works on pairwise preference relationships, which is the case for the majority of rank learning algorithms. Features are numerical values for attributes in the data. Assume further that there exists a preference relationship between data vectors such that $y_i \succ y_j$ denotes $\vec{x}_i$ is ranked higher than $\vec{x}_j$. A perfect ranking function $f \in F$ preserves the order relationships between instances:

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow f(\vec{x}_i) > f(\vec{x}_j)$$

Suppose we are given a set of instances $D = \{(\vec{x}_i, y_i) : (\vec{x}_i, y_i) \in X \times Y\}_{i=1}^m$. The objective for rank learning is to learn a mapping $f : X \times Y \mapsto \mathbb{R}$ that minimizes a given loss function on the training data.

3.3. SVM Rank Learning

Assume $f \in F$ is a linear function, i.e. $f(\vec{x}) = \langle \vec{w}, \vec{x} \rangle$, that satisfies

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow \langle \vec{w}, \vec{x}_i \rangle > \langle \vec{w}, \vec{x}_j \rangle$$

The SVM model targeting this problem can be formulated as a Quadratic Optimization problem:

$$\min_{\vec{w}} \frac{1}{2}\|\vec{w}\|^2 + C \sum \xi_{ij} \tag{3}$$
$$\text{subject to } \langle \vec{w}, \vec{x}_i \rangle \geq \langle \vec{w}, \vec{x}_j \rangle + 1 - \xi_{ij} \ , \ \xi_{ij} \geq 0 \ \forall i, j$$

The above optimization can be equivalently written by re-arranging the constraints and substituting the trade-off parameter C for $\lambda = \frac{1}{2C}$ as follows:

$$\min_{\vec{w}} \sum_{k=1}^{K} \left[1 - z_k \left\langle \vec{w}, \vec{x}_k^1 - \vec{x}_k^2 \right\rangle\right]_+ + \lambda \|\vec{w}\|^2 \tag{4}$$

where $[\ldots]_+$ indicates the standard hinge loss. $\vec{x}^1 - \vec{x}^2$ is a pairwise difference vector whose label z is positive, i.e., $z = +1$ if $\vec{x}^1 \succ \vec{x}^2$ and $z = -1$ otherwise. K is the total number of such pairs in the training set. Finally, a ranked list is obtained by sorting the instances according to the output of the ranking function in descending order.

3.4. Active Sampling for RankSVM

Let us consider a candidate example $\vec{x} \in U$, where U is the set of unlabeled examples. Assume $\vec{x}$ is incorporated into the labeled set with a rank label $y \in Y$. We denote the total loss on the instance pairs that include $\vec{x}$ by a function of $\vec{x}$ and $\vec{w}$, i.e. $D(\vec{x}, \vec{w}) = \sum_{j=1}^{J_y} \left[1 - z_j \langle \vec{w}, \vec{x}_j - \vec{x} \rangle\right]_+$ where J_y is the number of examples in the training set with a different label than the label y of $\vec{x}$. For instance, J_y is the number of negative(non-relevant) examples in the training set if y is assumed to be positive(relevant), and vice versa. The objective function to be minimized by RankSVM then becomes:

$$\min_{\vec{w}} \left\{ \lambda \|\vec{w}\|^2 + \sum_{k=1}^{K} \left[1 - z_k \left\langle \vec{w}, \vec{x}_k^1 - \vec{x}_k^2 \right\rangle\right]_+ + D(\vec{x}, \vec{w}) \right\} \tag{5}$$

Assume $\vec{w}^*$ is the solution to the optimization in Equation 4, and it is unique. Burges and Crisp (2000) show

the necessary and sufficient conditions for the uniqueness of the SVM solution. There are only rare cases where uniqueness does not hold, thus it is a rather safe assumption to make. Since we do not actually rerun the optimization problem on the enlarged data, we restrict ourselves to the current solution(hypothesis) $\vec{w}^*$. Instead of re-optimizing, we estimate the effect of adding each candidate instance on the training loss using the current solution to tell how much incorporating x into the labeled set is likely to change the current hypothesis. First, let us consider two cases.

1. Assume $\vec{w}^* = \text{argmin}_{\vec{w}}\, D(\vec{w}, \vec{x})$
   Then, $\vec{w}^*$ is also the solution to the optimization problem in Equation 5, combining the assumption with $\vec{w}^*$ being the solution to Equation 4. That means, adding $\vec{x}$ to the training set would not change the current hypothesis. From an active learning point of view, this example is useless since the learning algorithm is indifferent to its inclusion.

2. Assume $\vec{w}^* \neq \text{argmin}_{\vec{w}}\, D(\vec{w}, \vec{x})$
   This is the situation where the current solution could be different if that example $\vec{x}$ were incorporated into training. The magnitude of the difference depends on the magnitude of the deviation of $D(\vec{w}^*, \vec{x})$ from its optimal value, $min_{\vec{w}} D(\vec{w}, \vec{x})$.

We now study the second case in more detail. Let $\hat{\vec{w}}$ be the weight vector that minimizes $D(\vec{w}, \vec{x})$, i.e. $\hat{\vec{w}} = \text{argmin}_{\vec{w}}\, D(\vec{w}, \vec{x})$. Then, as the difference $\|\vec{w}^* - \hat{\vec{w}}\|$ increases it becomes less likely that $\vec{w}^*$ is optimal for Equation 5. In other words, the current solution $\vec{w}^*$ is in most need of updating in order to compensate for the loss on the new pairs. Let us write $\hat{\vec{w}}$ in terms of $\vec{w}^*$ as follows:

$$\hat{\vec{w}} = \vec{w}^* - \Delta w$$

Minimizing $D(\vec{w}, \vec{x})$ requires working with the hinge loss, the direct optimization of which is difficult due to the discontinuity of the derivative. However, it can still be solved using a gradient-descent-type algorithm[1]. Recall the objective function to be minimized:

$$\min_{\vec{w}} D(\vec{w}, \vec{x}) = \min_{\vec{w}} \sum_{j=1}^{J_y} \left[1 - z_j \left\langle \vec{w}, \vec{x}_j - \vec{x} \right\rangle\right]_+ \qquad (6)$$

The derivative of the above equation with respect to $\vec{w}$ at a single point $\vec{x}_j$, $\Delta\vec{w}_j$, is:

$$\Delta\vec{w}_j = \begin{cases} 0 & \text{if } z_j \left\langle \vec{w}, \vec{x}_j - \vec{x} \right\rangle \geq 1 \\ -z_j(\vec{x}_j - \vec{x}) & \text{if } z_j \left\langle \vec{w}, \vec{x}_j - \vec{x} \right\rangle < 1 \end{cases} \qquad (7)$$

[1]For a detailed discussion on solving SVM rank learning using gradient descent, see (Cao et al., 2006).

Algorithm 1 RankBoost

Input: initial data distribution D_1 over $X \times X$
for $t = 1$ **to** T **do**
  Train a weak learner on D_t
  Obtain the weak ranking $h_t : X \mapsto \mathbb{R}$
  Choose a weight $\alpha_t \in \mathbb{R}$ for h_t
  $D_{t+1}(\vec{x}^1, \vec{x}^2) = \frac{D_t(\vec{x}^1, \vec{x}^2) exp(-\alpha_t(h_t(\vec{x}^1) - h_t(\vec{x}^2)))}{Z_t}$
end for

We substitute $\vec{w}$ in Equation 7 for the current weight vector $\vec{w}^*$ to estimate how the solution of Equation 6 deviates from it, i.e. $\|\vec{w}^* - \hat{\vec{w}}\| = \|\Delta\vec{w}\|$. We can now write the magnitude of the total derivative as a function of $\vec{x}$ and the rank label y as follows:

$$\begin{aligned} g(\vec{x}, y) &= \|\Delta\vec{w}\| = \sum_j \|\Delta\vec{w}_j\| \qquad (8) \\ &= \sum_{j=1}^{J_y} \begin{cases} 0 & \text{if } z_j \left\langle \vec{w}^*, \vec{x}_j - \vec{x} \right\rangle \geq 1 \\ \| - z_j(\vec{x}_j - \vec{x})\| & \text{if } z_j \left\langle \vec{w}^*, \vec{x}_j - \vec{x} \right\rangle < 1 \end{cases} \end{aligned}$$

$g(\vec{x}, y)$ estimates how likely the current hypothesis is to be updated to minimize the loss introduced as a result of the addition of the example $\vec{x}$ with the rank label y. Thus, we use this function to estimate the ability of each unlabeled candidate example to change the current learner if incorporated into training. Since the true labels of the candidate examples are unknown, we use the current learner to estimate the true label probabilities. Then, we can take the expectation of $g(\vec{x}, y)$ by taking the weighted sum over the current posterior $\hat{P}(y \mid \vec{x})$ for all $y \in Y$. Among all the unlabeled examples, we choose the one with the highest value for that expectation:

$$\begin{aligned} \vec{x}^* &= \underset{\vec{x} \in U}{\text{argmax}} \sum_{y \in Y} \hat{P}(y \mid \vec{x}) g(\vec{x}, y) \\ &= \underset{\vec{x} \in U}{\text{argmax}} \Big\{ \hat{P}(y = 1 \mid \vec{x}) g(\vec{x}, y = 1) + \qquad (9) \\ &\qquad \hat{P}(y = -1 \mid \vec{x}) g(\vec{x}, y = -1) \Big\} \end{aligned}$$

3.5. RankBoost Learning

RankBoost is a boosting algorithm designed for ranking problems. Like all algorithms in boosting family, RankBoost learns a weak learner on each round, and maintains a distribution D_t over the ranked pairs, $X \times X$, to emphasize the pairs whose relative order is the hardest to learn. An outline of the algorithm is given as Algorithm 1. Z_t is a normalization constant, and the final ranking is a weighted sum of the weak rankings $H(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$. For more details and theoretical discussion see (Freund et al., 2003).

3.6. Active Sampling for RankBoost

This section introduces a similar method for active sampling for the RankBoost algorithm (Freund et al., 2003). Consider a candidate point $\vec{x} \in U$ and assume it is merged into the training set with rank label $y \in Y$. Unlike RankSVM, RankBoost algorithm does not directly operate with an optimization function. But the ranking loss with respect to the distribution at time t can be written as:

$$\sum_{\vec{x}^1,\vec{x}^2} D_t(\vec{x}^1,\vec{x}^2) I(H(\vec{x}^2) \geq H(\vec{x}^1)) \tag{10}$$

where I is defined to be 1 if the predicate holds and 0 otherwise. Hence, this is a sum over misranked pairs, assuming $\vec{x}^1 \succ \vec{x}^2$. The distribution at time $T+1$ can be written as:

$$D_{T+1}(\vec{x}^1,\vec{x}^2) = D_1(\vec{x}^1,\vec{x}^2) \frac{exp(H(\vec{x}^2) - H(\vec{x}^1))}{\prod_t Z_t} \tag{11}$$

The initial distribution term D_1 can be dropped without loss of generality, assuming it is uniform (which is reasonable given the fact that we do not have prior information about the data). Similarly to RankSVM, we would like to estimate how much the current ranking function would change if the point $\vec{x}$ were in the training set. We estimate this deviation by the difference in the ranking loss after enlarging the current labeled set with each example $\vec{x} \in U$. The ranking loss on the enlarged set with respect to the distribution D_{T+1} is:

$$\sum_{\vec{x}^1,\vec{x}^2} \frac{exp(H(\vec{x}^2) - H(\vec{x}^1))}{\prod_t Z_t} I(H(\vec{x}^2) \geq H(\vec{x}^1)) + \sum_{\vec{x}^j,\vec{x}} \frac{exp(H(\vec{x}^j) - H(\vec{x}))}{\prod_t Z_t} I(H(\vec{x}^j) \geq H(\vec{x})) \tag{12}$$

Note that the rank label y of $\vec{x}$ is assumed to be positive (relevant) with $\vec{x} \succ \vec{x}_j$ in this case. We have a similar calculation for the case where y is assumed to be negative (non-relevant). We adopt the distribution D_{T+1} because 1) it can easily be written in terms of the final ranking function, 2) it contains information about which pairs remain the hardest to determine after the iterative weight updates. Then, the difference in the ranking loss between the current and the augmented set simply becomes:

$$\Delta L(\vec{x}, y=1) = \sum_{\vec{x}^j,\vec{x}} \frac{exp(H(\vec{x}^j) - H(\vec{x}))}{\prod_t Z_t} I(H(\vec{x}^j) \geq H(\vec{x})) \tag{13}$$

This difference indicates how much the current ranking function needs to be modified to compensate for the loss incurred by including this example. Note that $I(x \geq 0) \leq e^x$ for $\forall x \in \mathbb{R}$ (Freund et al., 2003). Therefore, the upper bound on ΔL can be written as:

$$\Delta L(\vec{x}, y=1) \leq \sum_{\vec{x}^j,\vec{x}} \frac{exp(2(H(\vec{x}^j) - H(\vec{x})))}{\prod_t Z_t} \tag{14}$$

$\Delta L(\vec{x}, y=-1)$ can be similarly bounded, e.g. $\Delta L(\vec{x}, y=-1) \leq \sum_{\vec{x},\vec{x}^m} \frac{exp(2(H(\vec{x}) - H(\vec{x}^m)))}{\prod_t Z_t}$. Now, the loss difference can be estimated by taking the expectation over the possible rank labels of $\vec{x}$ with respect to the current ranker's posterior, $\hat{P}(y \mid \vec{x})$:

$$E_{\hat{P}}(\Delta L(\vec{x})) = \hat{P}(y=1 \mid \vec{x}) \Delta L(\vec{x}, y=1) + \hat{P}(y=-1 \mid \vec{x}) \Delta L(\vec{x}, y=-1) \tag{15}$$

Note the similarity with Equation 9 in the SVM case. Finally, we select the instance $\vec{x}$ that has the highest expected loss differential, e.g. $\vec{x}^* = \text{argmax}_{\vec{x}} E_{\hat{P}}(\Delta L(\vec{x}))$. For notational clarity, we take the maximum over the upper bound in Equation 14 as follows:

$$\vec{x}^* = \underset{\vec{x} \in U}{\text{argmax}} \Big\{ \hat{P}(y=1 \mid \vec{x}) \Big(\sum_{\vec{x}^j,\vec{x}} exp(2(H(\vec{x}^j) - H(\vec{x}))) \Big) + \hat{P}(y=-1 \mid \vec{x}) \Big(\sum_{\vec{x},\vec{x}^m} exp(2(H(\vec{x}) - H(\vec{x}^m))) \Big) \Big\} \tag{16}$$

For simplicity, we leave out the normalization constant $\prod_t Z_t$ since we are interested in the relative expectation rather than the absolute expectation.

3.7. Final Selection

The sample selection in both RankSVM and RankBoost requires estimating a posterior label distribution. We adopt a sigmoid function to estimate that posterior in the SVM case, as suggested by (Platt, 1999):

$$\hat{P}(y \mid \vec{x}) = \frac{1}{1 + exp(-y * f(\vec{x}) + C)}$$

where $f(\vec{x})$ is the real-valued score of the ranking algorithm, and C is a constant for calibrating the estimate. C is tuned on a separate corpus not used for evaluation in this paper. The final ranking in RankBoost is a sum of weak learners with the corresponding weights. When the weights are too small (or too large), the posterior gets close to the extreme (either 0 or 1) regardless of the example. Hence, we normalize the RankBoost output dividing by the maximum possible

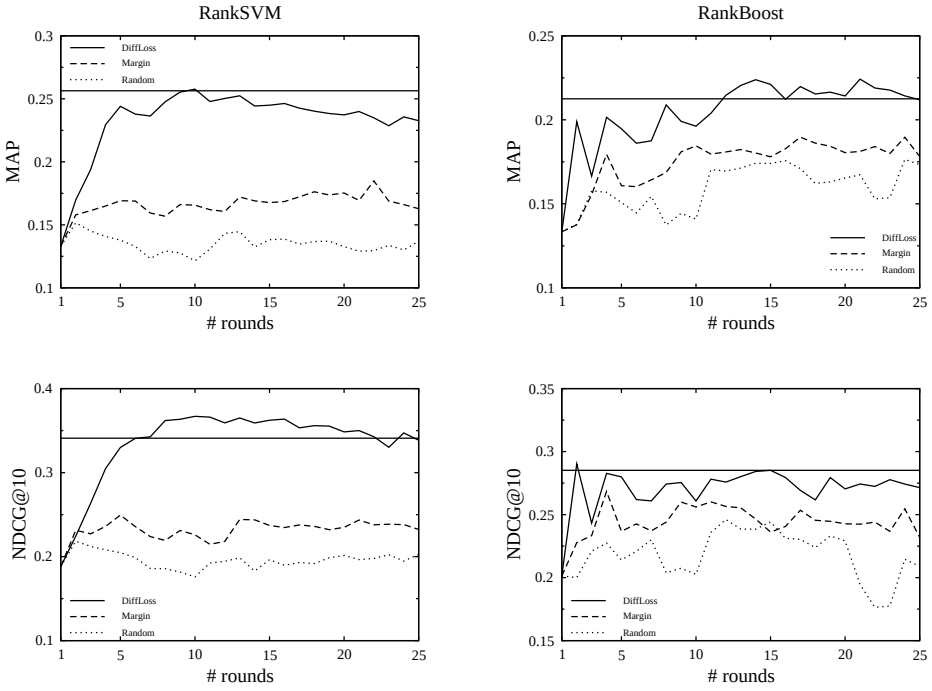


Figure 1. Comparison of different active learners on TREC 2003. The horizontal line indicates the performance when the entire training data is used. Only $\sim 15\%$ of the training data is actively labeled in total by each method.

rank score without changing the rank order:

$$\hat{P}(y \mid \vec{x}) = \frac{1}{1 + exp(-y * \frac{H(\vec{x})}{\sum_{t=1}^{T} \alpha_t} + C)}$$

Note $max_{\vec{x}} H(\vec{x}) = max_{\vec{x}} \sum_{t=1}^{T} \alpha_t h_t(\vec{x}) = \sum_{t=1}^{T} \alpha_t$ since the weak learner $h_t(\vec{x})$ in RankBoost is a {0,1}-valued function defined on the ordering information provided by the corresponding feature (Freund et al., 2003).

4. Evaluation

4.1. Data and Settings

We used two datasets in the experiments: TREC 2003 and 2004 topic distillation tasks in LETOR (Liu et al., 2007). The topic distillation task in TREC is very similar to web search where a page is considered relevant to a query if it is an entry page of some web site relevant to the query. The relevance judgments on the web pages with respect to the queries are binary. There are 44 features, e.g. content and hyperlink features, each of which is extracted from each document-query pair and normalized into $[0, 1]$. There are 50 and 75 queries with 1% and 0.6% relevant documents in TREC03 and TREC04, respectively. The total number of documents per query is ~ 1000 for both datasets. We used the standard train/test splits over 5 folds in LETOR. For each fold, we randomly picked 16 documents including exactly one relevant document per query for initial labeling. The remaining training data is considered as the unlabeled set. We compared our method with the margin-based sampling of (Brinker, 2004; Yu, 2005) and random sampling baselines. Each method selects 5 documents per query for labeling at each round, e.g. our method selects the top 5 documents according to the criteria in Equation 9 and 16. Then, the ranking function is re-trained, and evaluated on the test set. This process is repeated for 25 iterations which corresponds to labeling only $\sim 15\%$ of the entire training data. The reported results are averaged over 5 folds.

We adopted two standard, widely used performance metrics for evaluation, namely the Mean Average Precision (MAP) and the Normalized Discounted Cumulative Gain (NDCG) (Järvelin & Kekäläinen, 2002). For a single query, average precision is defined as the average of the precision as computed at each rank for all relevant documents: $AP = \frac{\sum_{n=1}^{N}(P(r)*rel(r))}{\#\text{ relevant documents for this query}}$ where r is the rank, N is the number of documents retrieved, $rel()$ is a binary function on the relevance of a given rank, and $P(r) = \frac{\#\text{ relevant docs in top r results}}{r}$ is the precision at

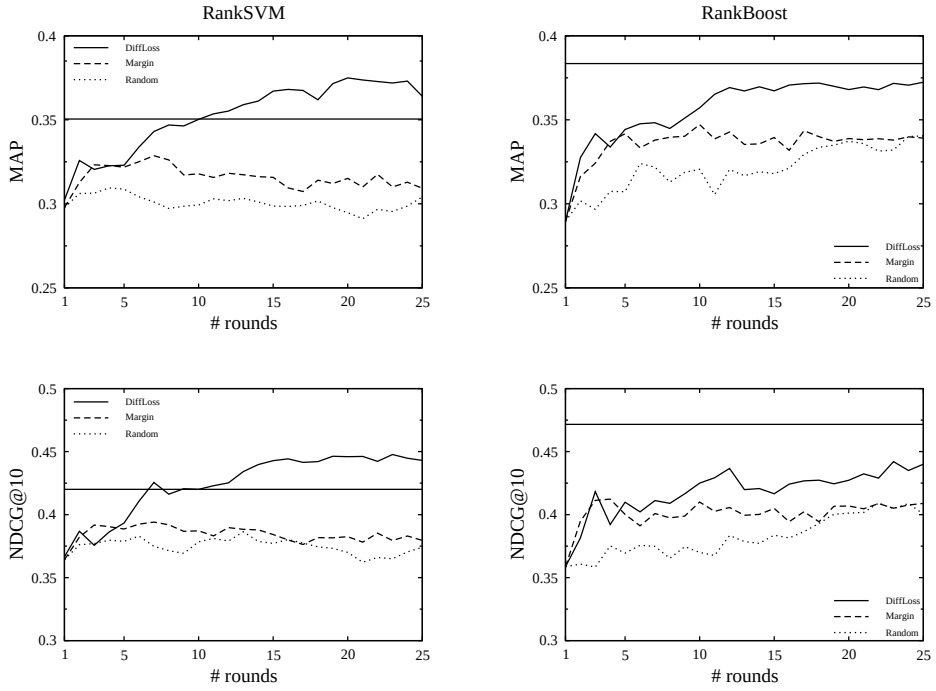


Figure 2. Comparison of different active learners on TREC 2004. The horizontal line indicates the performance when the entire training data is used. Only $\sim 15\%$ of the training data is actively labeled in total by each method.

the rank cut-off r. MAP is obtained by averaging the AP values for all queries. NDCG is cumulative and discounted since the overall utility of a list is measured by the sum of the gain of each relevant document, but the gain is discounted as a function of rank position. The NDCG value of a rank list at position n is given as follows: $NDCG@n = Z_n \sum_{j=1}^{n} \frac{2^{r(j)}-1}{log(1+j)}$ where $r(j)$ is the rank of the j^{th} document in the list, and Z_n is the normalization constant so that a perfect ranking yields an NDCG score of 1.

4.2. Results

Figure 1 and 2 plot the performance of the proposed method (denoted by *DiffLoss*), and as comparative baselines, the margin-based sampling and random sampling strategies on TREC 2003 and 2004 datasets. DiffLoss has a clear advantage over margin-based and random sampling in all cases with respect to different evaluation metrics. The differences over the entire operating range are also statistically significant ($p < 0.0001$) according to a two-sided paired t-test at 95% confidence level. DiffLoss especially achieves 30% relative improvement over the margin-based sampling for RankSVM on TREC 2003 dataset.

The horizontal line in each figure indicates the performance if all the training data was used, which we call the "optimal" performance. The performance of DiffLoss for RankBoost is comparable to the "optimal" on TREC 2003 and 2004 datasets. In case of RankSVM, DiffLoss is close to the "optimal" on TREC 2003, and outperforms it on TREC 2004 dataset. More precisely, DiffLoss using RankSVM reaches the optimal performance (even surpassing it on TREC 2004) after 10 rounds of labeling on average (labeling 5 documents per query at each round). DiffLoss using RankBoost, on the other hand, reaches 95% and 90% of the optimal performance on MAP and NDCG@10, respectively on TREC 2004 dataset after 10 rounds. This suggests that carefully chosen samples might lead to a higher level of accuracy than blindly using large amounts of training data. This is an important development over traditional supervised rank learning since it not only reduces the expensive labeling effort, but also may lead to greater generalization power. As follow-up work, we intend to explore methods that will automatically tell the sampling algorithm when to stop so that maximum gain with minimum cost is obtained, as well as exploring the underlying criteria for measuring the quality of actively selected examples.

We conducted another set of experiments to test the hypothesis that selecting a diverse set of samples might

lead to better results. We adopted the maximal marginal relevance principle of (Carbonell & Goldstein, 1998), originally proposed for text summarization. The idea is to select samples for labeling such that they have both the maximum potential to change the current ranking function and are maximally dissimilar to each other. See (Carbonell & Goldstein, 1998) for more details. However, incorporating this diversity principle into our selection criteria only slightly improved our results at the very beginning of the learning curve, but the improvement vanished afterwards. Thus, we do not report these results here in this paper.

5. Conclusion

We proposed two novel active sampling methods based on SVM rank learning and RankBoost. Our framework relies on the estimated risk of the ranking function on the labeled set after adding a new instance with all possible labels. The samples with the largest expected risk(loss) differential are selected to maximize the degree of learning at the fastest rate. Empirical results on two standard test collections indicate that our method significantly reduces the required number of labeled examples to learn an accurate ranking function. Possible extensions of this work include a study of the risk minimization in terms of direct optimization of ranking performance metrics, such as MAP, NDCG, precision@k, etc. and self-regulating algorithms that can decide when to terminate.

Acknowledgments

This material is based in part upon work supported by the Defense Advanced Projects Research Agency (DARPA) under Contract No. FA8750-07-D-0185.

References

Amini, M., Usunier, N., Laviolette, F., Lacasse, A., & Gallinari, P. (2006). A selective sampling strategy for label ranking. *ECML '06* (pp. 18–29).

Brinker, K. (2004). Active learning of label ranking functions. *ICML '04* (pp. 17–24).

Burges, C., & Crisp, D. (2000). Uniqueness of the svm solution. *NIPS '00* (pp. 223–229).

Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., & Hon, H.-W. (2006). Adapting ranking svm to document retrieval. *Proceedings of the international ACM SIGIR Conference on Research and Development in information retrieval (SIGIR'06)* (pp. 186–193).

Carbonell, J., & Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. *SIGIR '98* (pp. 335–336).

Cauwenberghs, G., & Poggio, T. (2000). Incremental and decremental support vector machine learning. *NIPS '00* (pp. 409–415).

Donmez, P., Carbonell, J., & Bennett, P. (2007). Dual strategy active learning. *Proceedings of the European Conference on Machine Learning* (pp. 116–127).

Freund, Y., Iyer, R., Schapire, R., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, *4*, 933–969.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Transaction on Information Systems*, *20(4)*, 422–446.

Joachims, T. (2002). Optimizing search engines using clickthrough data. *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*.

Liu, T., Xu, J., Qin, T., Xiong, W., Wang, T., & Li, H. (2007). Letor: Benchmark dataset for research on learning to rank for information retrieval. *SIGIR '07 Workshop: Learning to Rank for IR*.

Mitchell, T. (1982). Generalization as search. *Journal of Artificial Intelligence*, *18*, 203–226.

Nguyen, H., & Smeulders, A. (2004). Active learning with pre-clustering. *ICML '04* (pp. 623–630).

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 61–74.

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. *ICML '01* (pp. 441–448).

Tong, S., & Koller, D. (2000). Support vector machine active learning with applications to text classification. *Proceedings of International Conference on Machine Learning* (pp. 999–1006).

Xu, Z., Yu, K., Tresp, V., Xu, X., & Wang, J. (2003). Representative sampling for text classification using support vector machines. *ECIR '03*.

Yu, H. (2005). Svm selective sampling for ranking with application to data retrieval. *SIGKDD '05* (pp. 354–363).

Reinforcement Learning with Limited Reinforcement: Using Bayes Risk for Active Learning in POMDPs

Finale Doshi FINALE@MIT.EDU
Massachusetts Institute of Technology, Boston, USA

Joelle Pineau JPINEAU@CS.MCGILL.CA
McGill University, Montreal, Canada

Nicholas Roy NICKROY@MIT.EDU
Massachusetts Institute of Technology, Boston, USA

Abstract

Partially Observable Markov Decision Processes (POMDPs) have succeeded in planning domains that require balancing actions that increase an agent's knowledge and actions that increase an agent's reward. Unfortunately, most POMDPs are defined with a large number of parameters which are difficult to specify only from domain knowledge. In this paper, we present an approximation approach that allows us to treat the POMDP model parameters as additional hidden state in a "model-uncertainty" POMDP. Coupled with model-directed queries, our planner actively learns good policies. We demonstrate our approach on several POMDP problems.

1. Introduction

Partially Observable Markov Decision Processes (POMDPs) have succeeded in many planning domains because they can reason in the face of uncertainty, optimally trading between actions that gather information and actions that achieve a desired goal. This ability has made POMDPs attractive in real-world problems such as dialog management (Roy et al., 2000), but such problems often require a large number of parameters that are difficult to specify from domain knowledge alone. Recent advances can solve POMDPs with tens of thousands of states (Shani et al., 2007), but learning in POMDPs remains limited to small problems (Jaulmes et al., 2005).

Traditional reinforcement learning approaches (Watkins, 1989; Strehl et al., 2006; Even-Dar et al., 2005) to learning in MDP or POMDP domains require a reinforcement signal to be provided after each of the agent's actions. If learning must occur through interaction with a human expert, the feedback requirement may be undesirable. The traditional approach also does not guarantee the agent's performance during training. We identify and address three limitations in the traditional approach in this work:

1. Gathering sufficient training data for supervised learning may be prohibitively expensive.
2. Most approaches require the agent to experience a large penalty (i.e., make critical mistakes) to discover the consequences of a poor decision.
3. Accurate numerical reward feedback is especially hard to obtain from people, and inverse reinforcement learning (identifying the reward model without explicit reinforcement) poses its own challenges (Ng & Russell, 2000).

Our objective is to propose a framework for simultaneous learning and planning in POMDPs that overcomes the limitations above, allowing us to build agents that behave effectively in domains with model uncertainty.

We now discuss how our approach will address each of these three issues. To address the issue of long training periods, we adopt a Bayesian reinforcement learning approach and express model-uncertainty as additional hidden state. Bayesian methods (Dearden et al., 1999; Strens, 2000; Poupart et al., 2006; Jaulmes et al., 2005) have received recent attention in reinforcement learning because they allow experts to incorporate domain knowledge into priors over models. Thus, the system begins the learning process as a robust, functional (if conservative) agent while learning to adapt online to novel situations. The domain knowledge specified as a prior can also provide the agent with a basic understanding of potential pitfalls. Our work builds on previous Bayesian reinforcement learning approaches in that we provide both practical approximation schemes as well as guarantees on correctness and convergence.

To ensure robustness toward catastrophic mistakes, we develop an active learning scheme that determines when additional training is needed (typically active learning involves

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

asking for a few labels from unlabeled data; in this work, the 'label' corresponds to asking for the optimal action at a particular point in time). If the agent deems that model uncertainty may cause it to take undue risks, it queries an expert regarding what action it should perform. These queries both limit the amount of training required and allow the agent to infer the potential consequences of an action without executing it. Depending on the domain, we can imagine that different forms of information are most readily available. For example, in a navigation task, it may be straight-forward to query a state oracle (i.e., a GPS system) for a location. Similarly, rewards may be easy to measure based on quantities such as energy usage or time to goal. However, in other domains—particularly when working with human-robot interaction and dialog management systems—policy information may be more accurate; a human user may know what he wishes the agent to do, but may be unable to provide the agent with an accurate state representation (which is often complex, for optimization purposes). In these domains, asking for policy information, instead of a traditional reward signal, also side-steps the issue of getting explicit reward feedback from a human user, which can also be inaccurate (Millet, 1998). In this work, we deal exclusively with policy-based queries.

We are still left with the inverse reinforcement learning problem, as the user's response regarding correct actions provides only implicit information about the underlying reward. To date, Bayesian reinforcement learning has succeeded in learning observation and transition distributions (Jaulmes et al., 2005; Poupart et al., 2006), where updates have closed forms (such as updating Dirichlet counts); previous inverse reinforcement learning work (Ng & Russell, 2000) does not extend to the partially observable case. To overcome this issue, we use a non-parametric approach to model distributions over POMDPs; we demonstrate our approach on several standard problems.

We describe two practical contributions. First, we propose an approximation based on minimizing the immediate Bayes risk for choosing actions when transition, observation, and reward models are uncertain. The Bayes risk criterion avoids the computational intractability of solving large, continuous-valued POMDPs; we show it performs well in a variety of problems. Second, to gather information about the model without assuming state observability, we introduce the notion of *meta-queries*. These meta-queries accelerate learning and help the agent to infer the consequences of a potential pitfall without experiencing its effects. They are a powerful way of gaining information, but they make the strong assumption that they will be answered. Fortunately, a number of decision-making problems exist where this assumption is reasonable, particularly in collaborative human-machine tasks (e.g. automated dialogue systems and shared robot control scenarios).

2. The POMDP Model

A POMDP consists of the n-tuple $\{S,A,O,T,\Omega,R,\gamma\}$. S, A, and O are sets of states, actions, and observations. The transition function $T(s'|s,a)$ is a distribution over the states the agent may transition to after taking action a from state s. The observation function $\Omega(o|s,a)$ is a distribution over observations o that may occur in state s after taking action a. The reward function $R(s,a)$ specifies the immediate reward for each state-action pair. The factor $\gamma \in [0,1)$ weighs the importance of current and future rewards.

In the POMDP model, the agent must choose actions based on past observations; the true state is hidden. The belief, a probability distribution over states, is a sufficient statistic for a history of actions and observations. The belief at time $t+1$ can be computed from the previous belief, b_t, the last action a, and observation o, by applying Bayes rule:

$$b_{t+1}^{a,o}(s) = \Omega(o|s,a) \sum_{s' \in S} T(s|s',a) b_t(s') / Pr(o|b,a), \quad (1)$$

where $Pr(o|b,a) = \sum_{s' \in S} \Omega(o|s',a) \sum_{s \in S} T(s'|s,a) b_t(s)$. If the goal is to maximize the expected discounted reward, then the optimal policy is given by:

$$V_t(b) = \max_{a \in A} Q_t(b,a), \quad (2)$$

$$Q_t(b,a) = R(b,a) + \gamma \sum_{o \in O} Pr(o|b,a) V_t(b^{a,o}), \quad (3)$$

where the value function $V(b)$ is the expected discounted reward that an agent will receive if its current belief is b and $Q(b,a)$ is the value of taking action a in belief b. The exact solution to equation 3 is only tractable for tiny problems, so we use a point-based approximation (Pineau et al., 2003).

3. Modeling POMDP Uncertainty

We assume that the sets S, A, and O are known. The POMDP learning problem is to determine the parameters of T, Ω, and R that describe the dynamics and objective of the problem domain. A Bayesian approach is attractive in many real-world settings because we may have strong notions regarding certain parameters, but the value of those parameters may be difficult to specify exactly. We place a prior over the model parameters to express our domain knowledge, and improve upon this prior with experience.

If the state, action, and observation sets are discrete, T and Ω are collections of multinomial distributions. As conjugate priors, Dirichlet distributions are a natural choice of prior for T and Ω. We use a uniform prior over expert-specified ranges for the reward function R. Together these priors specify a distribution over POMDP models. To build a POMDP that incorporates the model parameters into the hidden state, we consider the joint state space $S' = S \times M$, where M is the space of models as described by all valid values for the model parameters. Although S' is continuous and high dimensional, the transition model for M is simple (assuming the true model is static).

The formulation above makes the agent aware of the uncertainty in the model parameters, and by trying various actions, it will be able to reduce uncertainty both in its state and in the parameters. However, the model information provided by the standard actions may be weak, and we would like the agent to be able to explicitly reduce model uncertainty in a safe manner. To allow for active learning, we augment the action space A of our original POMDP with a set of meta-queries $\{q_m\}$. The meta-queries consult an oracle (e.g., a domain expert) for the optimal action at a particular time step. We assume that the expert has access to the history of actions and observations (as does the agent), as well as the true POMDP model, and thus can advise the agent on the optimal action at any particular time. The agent begins by confirming the action it thinks is best:

"I think a_i is the best action. Should I do a_i?"

If the oracle answers to the negative, the agent follows with what it thinks is next best:

"Then I think a_j is best. Is that correct?"

until it receives an affirmative response. The ordered list of actions helps give the expert a sense of the agent's uncertainty; if the agent is uncertain, the expert might advise it to gather information rather than risk an incorrect decision.[1]

Meta-queries may be applied in situations where an expert is available to guide the agent. Unlike the oracle of Jaulmes et al. (2005), the meta-queries ask for policy information, not state information, which can be important if optimization procedures make the state-space unintuitive to the user (e.g., Williams and Young (2005)). In human-robot interaction, it may also simply be more natural to ask "I think you want me to go to the coffee machine. Should I go there?" which may be more natural than "Please enter your most recent statement" or "Please enter our position coordinates."

We can think of these meta-queries simply as additional actions and simply attempt to solve the model-uncertainty POMDP with this augmented action space. However, such an approach quickly becomes intractable. Therefore, we will treat the meta-query as a special action to be taken if the other actions are too risky. We take the cost ξ of querying the user to be a fixed parameter of the problem.

4. Solution Techniques

Table 1 summarizes our two-part approach to solving the model-uncertainty POMDP. First, given a history of actions and observations, the agent must select the next action. Second, the agent must update its distribution over the model parameters given additional interactions with the environment. In the most general case, both steps are intractable via standard POMDP solution techniques.[2]

[1] Our simulations used a shortened meta-query for speed.

[2] Analytic updates are possible if the distributions take certain forms (Poupart & Vlassis, 2008), but even here pruning is needed to keep the solutions to a tractable size.

Table 1. POMDP active learning approach.

ACTIVE LEARNING WITH BAYES RISK

- Sample POMDPs from a prior distribution.
- Complete a task choosing actions based on Bayes risk:
  - Use the POMDP samples to compute the action with minimal Bayes risk (Section 4.1).
  - If the risk is larger than a given ξ, perform a meta-query (Section 4.1).
  - Update each POMDP sample's belief based on the observation received (Section 4.2).
- Once a task is completed, update prior (Section 4.2):
  - Use a kernel incorporating action-observation history to propagate POMDP samples.
  - Weight POMDPs based on meta-query history.

Performance and termination bounds are in 4.3 and 4.4.

4.1. Bayes-Risk Action Selection

Let the loss $L_m(a, a^*; b)$ of taking action a in model m be $Q^*_m(b, a) - Q^*_m(b, a^*)$, where a^* is the optimal action in belief b according to model m. Given a belief $p_M(m)$ over models, the expected loss $E_M[L]$ is the Bayes risk:

$$BR(a) = \int_M (Q^*_m(b_m, a) - Q^*_m(b_m, a^*_m))p_M(m), \quad (4)$$

where M is the space of models, b_m is the current belief according to model m, and a^*_m is the optimal action for the current belief b_m according to model m. Let $a' = \arg\max_{a \in A} BR(a)$ be the action with the least risk. In the passive learning scenario, our agent just performs a'.

If the risk of the least-risky action a' is large, the agent may still incur significant losses. We would like our agent to be sensitive to the absolute magnitude of the risks that it takes. In the active learning scenario, the agent performs a meta-query if $BR(a')$ is less than $-\xi$, that is, if the least expected loss is more than the cost of the meta-query. The series of meta-queries will lead us to choose the correct action and thus incur no risk.

Intuitively, our criterion selects the least risky action now and hopes that the uncertainty over models will be resolved at the next time step. We can rearrange equation 4 to get:

$$BR(a) = \int_M Q(b_m, a)p_M(m) - \int_M Q(b_m, a^*_m)p_M(m). \quad (5)$$

The second term is independent of the action choice; to maximize $BR(a)$, one may simply maximize the first term:

$$V_{BR} = \max \int_M Q(b_m, a)p_M(m). \quad (6)$$

The Bayes risk criterion is similar to the Q_{MDP} heuristic (Littman et al., 1995), which uses the approximation $V(b) = \max \sum_s Q(s, a)b(s)$ to plan in known POMDPs.

In our case, the belief over states $b(s)$ is replaced by a belief over models $p_M(m)$ and the action-value function over states $Q(s,a)$ is replaced by an action-value function over beliefs $Q(b_m,a)$. In the Q_{MDP} heuristic, the agent assumes that the uncertainty over states will be resolved after the next time step. Our Bayes-risk criterion may be viewed as similarly assuming that the next action will resolve the uncertainty over models.

Though similar, the Bayes risk action selection criterion differs from Q_{MDP} in two important ways. First, our actions come from POMDP solutions and thus do fully consider the uncertainty in the POMDP state. Unlike Q_{MDP}, we do not act on the assumption that our state uncertainty will be resolved after taking the next action; our approximation supposes that only the model uncertainty will be resolved. Thus, if the model stochasticity is an important factor, our approach will take actions to reduce state uncertainty. This observation is true regardless of whether the agent is passive (does not ask meta-queries) or active.

In the active learning setting, the second difference is the meta-query. Without the meta-query, while the agent may take actions to resolve state uncertainty, it will never take actions to reduce model uncertainty. However, meta-queries ensure that the agent rarely (with probability δ) takes a less than ξ-optimal action in expectation. Thus the meta-queries make the learning process robust from the start and allow the agent to resolve model uncertainty.

Approximation and bounds: The integral in equation 4 is computationally intractable, so we approximate it with a sum over a sample of POMDPs from the space of models:

$$BR(a) \approx \sum_i (Q(b_i,a) - Q(b_i,a_i^*))p_M(m_i) \qquad (7)$$

There are two main sources of approximation that can lead to error in our computation of the Bayes risk:

- Error due to the Monte Carlo approximation of the integral in equation 4: Note that the maximum value of the $Q(b_i,a) - Q(b_i,a_i^*)$ is trivially upper bounded by $\frac{R_{\max} - \min(R_{\min},\xi)}{1-\gamma}$ and lower bounded by zero. Applying the Hoeffding bound with sampling error ϵ_s and confidence δ, we will require n_m samples:[3]

$$n_m = \frac{(R_{\max} - \min(R_{\min},\xi))^2}{2(1-\gamma)^2\epsilon_s^2} \log \frac{1}{\delta} \qquad (8)$$

- Error due to the point-based approximation of $Q(b_i,a)$: The difference $Q(b_i,a) - Q(b_i,a_i^*)$ may have an error of up to $\epsilon_{PB} = \frac{2(R_{\max} - R_{\min})\delta_B}{(1-\gamma)^2}$, where δ_B is the sampling density of the belief points. This result is directly from the error bound due to Pineau et al. (2003).

[3] An error of ϵ with confidence δ means $Pr[x - \hat{x} > \epsilon] < \delta$.

To obtain a confidence δ when calculating if the Bayes risk is greater than $-\xi$, we combine these bounds, setting $\epsilon_s = \xi - \epsilon_{PB}$, and computing the appropriate number of samples n from equation 8. We note however that the Hoeffding bounds used to derive this approximation are quite loose; for example in the shuttle POMDP problem, we used 200 samples, whereas equation 8 suggested over 3000 samples may have been necessary even with a perfect POMDP solver.

4.2. Updating the Model Distribution

As described in Section 3, we initially placed Dirichlet priors over the transition and observation parameters and uniform priors over the reward parameters. As our agent interacts with the environment, it receives two sources of information to update its prior: a history h of actions and observations and a set of meta-queries (and responses) Q. Given h and Q, the posterior $p_{M|h,Q}$ over models is:

$$p_{M|h,Q}(m|h,Q) \propto p(Q|m)p(h|m)p_M(m), \qquad (9)$$

where Q and h are conditionally independent given m because they are both computed from the model parameters. The history h is the sequence of actions and observations since p_M was last updated. The set Q is the set of *all* meta-queries asked (and the expert's responses). Each source poses a different challenge when updating the posterior.

If the agent were to have access to the hidden underlying state, then it would be straightforward to compute $p_{M|h}(m|h) \propto p(h|m)p_M(m)$; we simply need to add counts to the appropriate Dirichlet distributions. However, when the state sequence is unknown, the problem becomes more difficult; the agent must use its belief over the state sequence to update the posterior. Thus, it is best to perform the update when it is most likely to be accurate. For example, in a robot maze scenario, if the robot is lost, then estimating its position may be inaccurate. However, once the robot reaches the end of the maze, it knows both its start and end position, providing more information to recover its path. We focus on episodic tasks in this work and update the belief over models at the completion of a task.

The meta-query information poses a different challenge: the questions provide information about the policy, but our priors are over the model parameters. The meta-queries truncate the original Dirichlet as models inconsistent with meta-query responses have zero likelihood. We approximate the posterior with a particle filter.

4.2.1. During a task: Updating particle weights

Recall that sequential Monte Carlo techniques let us represent a distribution at time t using a set of samples from time $t-1$ using the following procedure (Moral et al., 2002):

$$m_t \sim K(m_{t-1}, m_t), \qquad (10)$$

$$w_t = w_{t-1} \frac{p_{M,t}(m_t)}{p_{M,t-1}(m_{t-1})K(m_{t-1},m_t)}, \qquad (11)$$

where $K(m, m')$ is an arbitrary transition kernel and p_M is the probability of the model m under the true posterior.

Sampling a new model requires solving a POMDP, which is computationally expensive and thus may be undesirable while an agent is in the process of completing a task. Thus, we do not change our set of samples during a task (that is, $K(m, m') = \delta_m(m')$ where $\delta()$ is the Dirac delta function). We begin at time $t-1$ with a set of models m_i and weights w_i that represent our current belief over models. If a meta-query occurs at time t, then $p_{M,t}(m) \propto p(Q_t|m)p_{M,t-1}(m)$, and the weight update reduces to

$$w_t = w_{t-1}p(Q_t|m). \tag{12}$$

In theory, the $p(Q|m)$ should be a delta function: either the model m produces a policy that is consistent with the meta-query ($p(Q|m) = 1$), or it does not ($p(Q|m) = 0$). In practice, approximation techniques used to compute the model's policy are imperfect (and expert advice can be incorrect) so we do not want to penalize a model that occasionally acts incorrectly. We model the probability of seeing k incorrect responses in n trials as a binomial variable with parameter p_e, where p_e is the probability a model fails a meta-query due to the approximate solver. This value is hard to characterize, of course, and is problem-specific; we used $p_e = 0.3$ in our tests.

4.2.2. Between tasks: Resampling Particles.

Over time, samples taken from the original prior may no longer represent the posterior well. Moreover, if only a few high weight samples remain, the Bayes risk may appear smaller than it really is because most of the samples are in the wrong part of the space. We also need to update the models based on the history information, which we have ignored so far. We do both these steps at the end of a task.

Action-Observation Histories: Dirichlet Update. We first discuss how to update the posterior $p(m|h)$ in closed form. Recall that updating the Dirichlet counts given actions and observations requires knowing the underlying state history, and our agent only has access to history of actions and observations. We therefore update our parameters using an online extension of the standard EM algorithm (Sato, 1999). In the E-step, we estimate a distribution over state sequences in the episode. In the M-step, we use this distribution to update counts on our Dirichlet priors. Online EM guarantees convergence to a local optimum.

For the E-step, we first estimate the true state history. Two sources of uncertainty are present: model stochasticity and unknown model parameters. To compute the expectation with respect to model stochasticity, we use the standard forward-backward algorithm to obtain a distribution over states for each sample. Next, we combine the distributions for each sample based on the sample's weight. For example, suppose a sampled model assigns a probability $p_i(s)$ to being in state s. Then the expected probability $\hat{p}(s)$ of being in state s is $\hat{p}(s) = \sum_i^n w_i p_i(s)$. The samples represent our distribution over models, so this sum approximates an expectation over all models.

Next, we update our Dirichlet counts based on both the probability that a POMDP assigns to a particular state and the probability of that POMDP. Given an action a and observation o corresponding to time t, we would update our Dirichlet count for $\alpha_{o,s,a}$ in the following manner:

$$\alpha_{o,s,a} = \alpha_{o,s,a} + \hat{p}(s) \tag{13}$$

for each state s. This update combines prior knowledge about the parameters—the original value of $\alpha_{o,s,a}$—with new information from the current episode, $\hat{p}(s)$.

Resampling Models. As is standard in sequential Monte Carlo techniques, we begin by resampling models according to their weights w_i. Thus, a model with high weight may get selected many times for inclusion in the resampled set of models, while a model with low weight may disappear from the sample set since it is no longer representative of the posterior. Once resampled, each model has equal weight. Before we begin the next task, we perturb the models with the following transition kernel:

- Draw a sample m' from $p_{M|h}$.
- With probability p_k, replace m with m'
- With probability $1 - p_k$, take the convex combination of the model parameters of m and m' so that $m' = p \cdot m' + (1-p) \cdot m$ with the convexity parameter being chosen uniformly at random on $[0, a]$.

We reduce the probability p_k from 0.9 to 0.4 as the interactions continue, encouraging large exploration earlier on and fine-tuning in later interactions. We set a to 0.2 in our experiments. Based on this sampling procedure, the weight (keeping in mind that after resampling, all models had equal weight) of the transitioned model m' is given by:

$$w_t \propto \frac{p(Q|m')p_{M|h}(m')}{p(Q|m)p_M(m)K(m, m')}, \tag{14}$$

where, $K(m, m') = p_k \cdot p_{M|h}(m')$ if we keep the newly-sampled model and and $K(m, m') = (1-p_k) \cdot p_{M|h}(m')/a$ if we perturb m via a convex combination.

4.3. Performance Bounds

Let V^* be the value of the optimal policy under the true model. From our risk criterion, the expected loss at each action is no more than ξ. However, with probability δ, in the worst case, the agent may choose a bad action that takes it to an absorbing state in which it receives R_{min} forever.

To determine the expected discounted reward, we consider a two-state Markov chain. In state 1, the "normal" state, the agent receives a reward of $R - \xi$, where R is the value the agent would have received under the optimal policy. In state 2, the agent receives $R_{\min}$. Equation 15 describes the

transitions in this simple chain and the values of the states:

$$\begin{vmatrix} V_1 \\ V_2 \end{vmatrix} = \begin{vmatrix} R - \xi \\ R_{\min} \end{vmatrix} + \gamma \begin{vmatrix} 1-\delta & \delta \\ 0 & 1 \end{vmatrix} \begin{vmatrix} V_1 \\ V_2 \end{vmatrix}. \quad (15)$$

Solving this system and noting that the agent begins in state 1 with probability $1 - \delta$ and state 2 with probability δ, the lower bound V' on the expected value is

$$V' = \eta(V^* - \frac{\xi}{1-\gamma}) + (1-\eta)\frac{R_{min}}{1-\gamma}, \quad (16)$$

$$\eta = (1-\delta)(1-\gamma)(1-\gamma(1-\delta))^{-1}. \quad (17)$$

4.4. Model Convergence

Given the algorithm in Table 1, we would like to know if the learner will eventually stop asking meta-queries. We state that the model is *converged* if $BR(a') > -\xi$ for all histories (where ξ is the cost of a meta-query). Our convergence argument involves two steps. First, let us ignore the reward model and consider only the observation and transition models. As long as standard reinforcement learning conditions—periodic resets to a start state and information about all states (via visits or meta-queries)—hold, the prior will peak around some value (perhaps to a local extremum) in a bounded number of interactions from the properties of the online EM algorithm (Sato, 1999). We next argue that once the observation and transition parameters have converged, we can bound the meta-queries required for the reward parameters to converge.

Observation and Transition Convergence. To discuss the convergence of the observation and transition distributions, we apply a weaker sufficient condition than the convergence of the EM algorithm. We note that the number of interactions bounds the number of meta-queries, since we ask at most one meta-query for each normal interaction. We also note that the counts on the Dirichlet priors increase monotonically. Once the Dirichlet parameters are sufficiently large, the variance in the sampled models will be small; even if the mean of the Dirichlet distribution shifts with time, no additional meta-queries will be asked.

The specific convergence rate of the active learning will depend heavily upon the problem. However, we can check if k additional interactions are sufficient such that the probability of asking a meta-query is p_q with confidence δ_q. To do so, we will sample random beliefs and test if less than a p_q-proportion have a Bayes risk greater than ξ.

1. **Sampling a Sufficient Number of Beliefs.** To test if k interactions lead to a probability p_q of additional meta-queries with confidence δ_q, we compute the Bayes risk for n_b beliefs sampled uniformly. If fewer than $n_q = p_q n_b$ beliefs require meta-queries after k interactions, we accept the value of k. We sample from the posterior Dirichlet given k interactions and estimate $\hat{p}_q = n_q/n_b$. We desire $\hat{p}_q$ to be within ϵ_q of $p'_q = p_q - \epsilon_q$ with probability δ_q. Using the Chernoff bound $\delta_q = e^{-n_b p'_q \epsilon_q^2/3}$, we set ϵ_q to $2/3p_q$ to minimize the samples needed:

$$n_b > -27/4 \cdot (p_q)^{-3} \log \delta_q. \quad (18)$$

2. **Computing Bayes Risk from a Conservative Posterior.** We next compute the Bayes risk for each belief given a hypothesized set of k interactions. We do not know *a priori* the response to the interactions, so we use the maximum-entropy Dirichlet posterior to compute the posterior Bayes risk (that is, assign the k counts to assign an equal number of counts to each variable). We compute the Bayes risk of each belief from this posterior and accept k if $\hat{p}_q < p_q$.
3. **Correction for Approximate Bayes Risk.** Recall that we approximate the Bayes risk integral with a sum over sampled POMDP models, and the number of models n_m required is given by equation 8. We must correct for the error induced by this approximation. Section 4.1 tells us if a belief b has risk $BR(a) < -\xi$ with confidence δ. Suppose we sample n_b beliefs, and the true fraction of beliefs in which meta-queries are asked is p_q. Due to misclassifications, however, the expected value we will observe is only $(1-\delta)p_q$. We can then apply a second Chernoff bound to determine that with probability δ, no more than $2(1-\delta)n_b$ beliefs will be misclassified.[4] Let

$$p''_q = p_q(1 - 2(1-\delta)), \quad (19)$$

be the minimum fraction of beliefs queries we expect to observe requiring meta-queries if the true fraction is p_q.

Thus, to test if k interactions lead to a probability of p_q for meta-queries with confidence δ_q, we compute p''_q from equation 19, sample n_b beliefs uniformly from equation 18, update the Dirichlet posteriors to be maximum-entropy posteriors, sample the n_m models from equation 8, and finally compute the posterior Bayes risk for each belief. If less than a p_q-proportion of beliefs require meta-queries, then k is an upper bound on the number of remaining meta-queries with probability p_q and confidence δ_q.

Reward Convergence. The cost of a meta-query limits the reward resolution. Suppose a POMDP P has an optimal policy π with value V. If we adjusted all the rewards by some small ϵ_r, then the value of the same policy π will differ from V by at most $\frac{\epsilon_r}{1-\gamma}$ (since we will receive at worst ϵ_r less reward at each time step). This value is a lower-bound on the optimal policy in the new POMDP. Thus, a POMDP with all its rewards within $(1-\gamma)\xi$ of P will have a policy of value $V \pm \xi$. In this way, the value ξ imposes a minimal level of discretization over the reward space.

The rewards are bounded between $R_{\min}$ and $R_{\max}$. If our reward space has d dimensions, then our discretization will

[4] This bound requires $n_b > \frac{3}{\delta} \log \frac{1}{\delta}$, but we will find that our final bound for n_b is greater than this value.

yield at most $(\frac{R_{\max}-R_{\min}}{(1-\gamma)\xi})^d$ POMDPs. (Intuitively, the discretization involves limiting the precision of the sampled rewards.) Since each meta-query invalidates at least one POMDP, we must eventually stop asking meta-queries.

5. Results

We first solve a discretized model-uncertainty POMDP solved directly to show the utility of meta-queries. We next couple the meta-queries with our Bayes-risk criterion for learning with with continuous-valued unknown parameters.

5.1. Learning Discrete Parameters

In domains where model uncertainty is limited to a few, discrete parameters, we may be able to solve for the complete model-uncertainty POMDP using standard POMDP methods. We consider a simple POMDP-based dialog management task (Doshi & Roy, 2007) where the reward is unknown. We presume the correct reward is one of four (discrete) possible levels and that the meta-query had a fixed associated cost. Figure 1 compares the performance of the optimal policy *with* meta-queries (left column), an optimal policy *without* meta-queries (middle column), and our Bayes risk policy *with* meta-queries (right column). The difference in median performance is small, but the variance reduction from the meta-queries is substantial.[5]

Unfortunately, discretizing the model space does not scale; increasing from 4 to 48 possible reward levels, we could no longer obtain high-quality global solutions using standard techniques. Next, we present results using our Bayes-risk action selection criterion when we no longer discretize the parameter space and instead allow the parameters to take on continuous values within prespecified ranges.

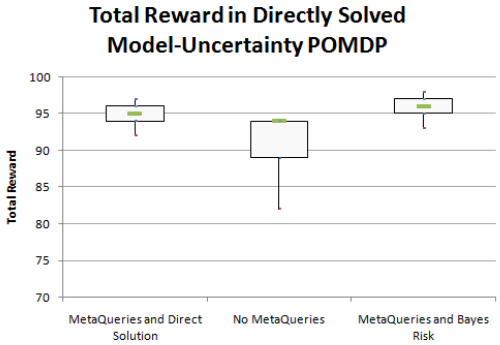


Figure 1. Boxplot of POMDP learning performance with a discrete set of four possible models. The medians of the policies are comparable, but the active learner (left) makes fewer mistakes than the passive learner (center). The Bayes risk action selection criterion (right) does not cause the performance to suffer.

[5]Although the Bayes risk approximation appears higher, the difference in performance in both median and variance is negligible between the optimal policy and the Bayes risk approximation.

5.2. Learning Continuous Parameters

Table 2 shows our approach applied to several standard POMDP problems (Littman et al., 1995). For each problem, between 50-200 POMDP samples were initially taken from the prior over models. The sampled POMDPs were solved very approximately, using relatively few belief points (500) and only 25 partial backups. The policy oracle used a solution to the true model with many more belief points (1000-5000) and 250 full backups. We took this solution to be the optimal policy. During a trial, which continued until either the task was completed or a maximum number of iterations was reached, the agent had the choice of either taking a normal action or asking a meta-query and then taking the supplied optimal action. POMDPs were resampled at the completion of each trial.

The non-learner (control) always used its initial samples to make decisions, using the Bayes-risk criterion to select an action from the policies of the sampled models. Its prior did not change based on the action-observation histories that it experienced, nor did it ask any meta-queries to gain additional information. The passive learner resampled its POMDP set after updating its prior over transitions and observations using the forward-backward algorithm. The active learner used both the action-observation histories and meta-queries for learning. None of the systems received explicit reward information, but the active learner used meta-queries to infer information about the reward model. The Hallway problem was too large for the agent to learn (after 50 repetitions, it still queried the oracle at nearly every step); in these results we provided the agent with possible successor states. The smarter prior seemed reasonable as a map and may be easier to obtain for a new environment than to the robot's dynamics. Depending on the problem, tasks required an average of 7 to 32 actions to complete.

Table 2. Mean difference between optimal (under the true model) and accrued rewards (smaller = better).

Problem	States	Control	Passive	Active
Tiger	2	46.5	50.7	33.3
Shuttle	8	10.0	10.0	2.0
Gridworld-5	26	33.1	102	21.4
Hallway	57	1.0	1.0	0.08

Figure 2 shows the performance of the three agents on the shuttle problem (a medium-sized standard POMDP). In each case, the agent began with observation and transition priors with high variance but peaked toward the correct value (that is, slightly better than uniform). We created these priors by applying a diffusion filter to the ground-truth transition and observation distributions and using the result as our initial Dirichlet parameters. All reward priors were uniform between the minimum and maximum reward values of the ground-truth model. The active learner started (and remained) with good performance because it used meta-queries when initially confused about the model. Thus, its performance was robust throughout.

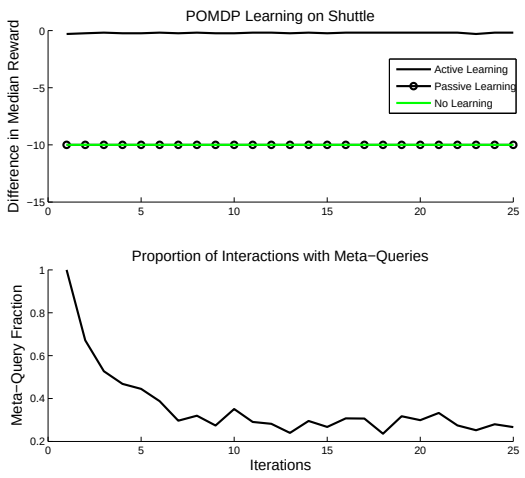


Figure 2. Performance of the non-learner, passive learner, and active learner on the shuttle problem.

6. Discussion and Conclusion

One recent approach to MDP model learning, the Beetle algorithm (Poupart et al., 2006), converts a discrete MDP into a continuous POMDP with state variables for each MDP parameter. However, their analytic solution does not scale to handle the entire model as a hidden state in POMDPs. Also, since the MDP is fully observable, Beetle can easily adjust its prior over the MDP parameters as it acquires experience; in our POMDP scenario, we needed to estimate the possible states that the agent had visited. Recently the authors have extended Beetle to partially observable domains (Poupart & Vlassis, 2008), providing similar analytic solutions to the POMDP case. The work outlines efficient approximations but results are not provided.

Prior work in MDP and POMDP learning has also considered sampling to approximate a distribution over uncertain models. Dearden et. al. (1999) discusses several approaches for representing and updating priors over MDPs using sampling and value function updates. Strens (2000) shows that in the MDPs, randomly sampling only one model from a prior over models, and using that model to make decisions, is guaranteed to converge to the optimal policy if one resamples the MDP sufficiently frequently from an updated prior over models. More recently, in the case of POMDPs, Medusa (Jaulmes et al., 2005) avoids the problem of knowing how to update the prior by occasionally requesting the true state based on model-uncertainty heuristics. It converges to the true model but may make several mistakes before convergence. Our risk-based heuristic and policy queries provide correctness and convergence guarantees throughout the learning process.

We developed an approach for active learning in POMDPs that can robustly determine a near-optimal policy. Meta-queries—questions about actions that the agent is thinking of taking—and a risk-averse action selection criterion allowed our agent to behave robustly even with uncertain knowledge of the POMDP model. We analyzed the theoretical properties of our algorithm, but also included several practical approximations that rendered the method tractable. Finally, we demonstrated the approach on several problems from the POMDP literature. In our future work, we hope to develop more efficient POMDP sampling schemes—as well as heuristics for allocating more computation to more promising solutions—to allow our approach to be deployed on larger, real-time applications.

References

Dearden, R., Friedman, N., & Andre, D. (1999). Model based Bayesian exploration. .

Doshi, F., & Roy, N. (2007). Efficient model learning for dialog management. *Technical Report SS-07-07*. AAAI Press.

Even-Dar, E., Kakade, S. M., & Mansour, Y. (2005). Reinforcement learning in POMDPs without resets. *IJCAI*.

Jaulmes, R., Pineau, J., & Precup, D. (2005). Learning in non-stationary partially observable Markov decision processes. *ECML Workshop on Reinforcement Learning in Non-Stationary Environments*.

Littman, M. L., Cassandra, A. R., & Kaelbling, L. P. (1995). Learning policies for partially observable environments: scaling up. *ICML*.

Millet, I. (1998). The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. *Journal of Multi-Criteria Decision Analysis, 6*.

Moral, P., Doucet, A., & Peters, G. (2002). Sequential Monte Carlo samplers.

Ng, A., & Russell, S. (2000). Algorithms for inverse reinforcement learning. *ICML*.

Pineau, J., Gordon, G., & Thrun, S. (2003). Point-based value iteration: an anytime algorithm for POMDPs. *IJCAI*.

Poupart, P., & Vlassis, N. (2008). Model-based Bayesian reinforcement learning in partially observable domains. *ISAIM*.

Poupart, P., Vlassis, N., Hoey, J., & Regan, K. (2006). An analytic solution to discrete Bayesian reinforcement learning. *ICML*.

Roy, N., Pineau, J., & Thrun, S. (2000). Spoken dialogue management using probabilistic reasoning. *ACL*. Hong Kong.

Sato, M. (1999). Fast learning of on-line EM algorithm. *TR-H-281, ATR Human Information Processing Lab*.

Shani, G., Brafman, R., & Shimony, S. (2007). Forward search value iteration for POMDPs. *IJCAI*.

Strehl, A. L., Li, L., & Littman, M. L. (2006). Incremental model-based learners with formal learning-time guarantees. *UAI*.

Strens, M. (2000). A Bayesian framework for reinforcement learning. *ICML*.

Watkins, C. (1989). *Learning from delayed rewards*. Doctoral dissertation, Cambridge University.

Williams, J., & Young, S. (2005). Scaling up POMDPs for dialogue management: The "summary POMDP" method. *Proceedings of the IEEE ASRU Workshop*.

Confidence-Weighted Linear Classification

Mark Dredze MDREDZE@CIS.UPENN.EDU
Koby Crammer CRAMMER@CIS.UPENN.EDU
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA

Fernando Pereira[1] PEREIRA@GOOGLE.COM
Google, Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043 USA

Abstract

We introduce confidence-weighted linear classifiers, which add parameter confidence information to linear classifiers. Online learners in this setting update both classifier parameters and the estimate of their confidence. The particular online algorithms we study here maintain a Gaussian distribution over parameter vectors and update the mean and covariance of the distribution with each instance. Empirical evaluation on a range of NLP tasks show that our algorithm improves over other state of the art online and batch methods, learns faster in the online setting, and lends itself to better classifier combination after parallel training.

1. Introduction

Online learning algorithms operate on a single instance at a time, allowing for updates that are fast, simple, make few assumptions about the data, and perform well in wide range of practical settings. Online learning algorithms have become especially popular in natural language processing for tasks including classification, tagging, and parsing. In this paper, we revisit the design of linear classifier learning informed by the particularities of natural language tasks. Specifically, feature representations for natural language processing have very high dimension (millions of features derived from words and word combinations are common), and most features are observed on only a small fraction of instances. Nevertheless, those many rare features are important in classifying the instances in which they occur. Therefore, it is worth investigating whether online learning algorithms for linear classifiers could be improved to take advantage of these particularities of natural language data.

We introduce *confidence-weighted* (CW) learning, a new class of online learning methods that maintain a probabilistic measure of confidence in each parameter. Less confident parameters are updated more aggressively than more confident ones. Parameter confidence is formalized with a Gaussian distribution over parameter vectors, which is updated for each new training instance so that the probability of correct classification for that instance under the updated distribution meets a specified confidence. We show superior classification accuracy over state-of-the-art online and batch baselines, faster learning, and new classifier combination methods after parallel training.

We begin with a discussion of the motivating particularities of natural language data. We then derive our algorithm and discuss variants. A series of experiments shows CW learning's empirical benefits. We conclude with a discussion of related work.

2. Online Algorithms and NLP

In natural language classification tasks, many different features, most of which are binary and are infrequently on, can be weakly indicative of a particular class. Therefore, we have both data sparseness, which demands large training sets, and very high dimensional parameter vectors. For certain types of problems, such as structured prediction in tagging or parsing, the size and processing complexity of individual instances make it difficult to keep more than a small number of instances in main memory. These particularities make online algorithms, which process a single instance at a time, a good match for natural-language tasks. Processing large amounts of data is simple for online methods, which require observing each instance once — though in practice several iterations may be

[1]Work done at the University of Pennsylvania.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

necessary — and update parameter vectors using a single instance at a time. In addition, the simple nature of most online updates make them very fast.

However, while online algorithms do well with large numbers of features and instances, they are not designed for the heavy tailed feature distributions characteristic of natural language tasks. This type of feature distribution can have a detrimental effect on learning. With typical linear classifier training algorithms, such as the perceptron or passive-aggressive (PA) algorithms (Rosenblatt, 1958; Crammer et al., 2006), the parameters of binary features are only updated when the features occur. Therefore, frequent features typically receive more updates. Similarly, features that occur early in the data stream take more responsibility for correct prediction than those observed later. The result is a model that could have good parameter estimates for common features and inaccurate values for rare features. However, no distinction is made between these feature types in most online algorithms.

Consider an illustrative example from the problem of sentiment classification. In this task, a product review is represented as n-grams and the goal is to label the review as being positive or negative about the product. Consider a positive review that simply read *"I liked this author."* An online update would increase the weight of both "liked" and "author." Since both are common words, over several examples the algorithm would converge to the correct values, a positive weight for "liked" and zero weight for "author." Now consider a slightly modified negative example: *"I liked this author, but found the book dull."* Since "dull" is a rare feature, the algorithm has a poor estimate of its weight. An update would decrease the weight of both "liked" and "dull." The algorithm does not know that "dull" is rare and the changed behavior is likely caused by the poorly estimated feature ("dull") instead of the common well estimated feature ("liked.") This update incorrectly modified "liked" and does not attribute enough negative weight to "dull," thereby decreasing the rate of convergence.

This example demonstrates how a lack of memory for previous instances — a property that allows online learning — can hurt learning. A simple solution is to augment an online algorithm with additional information, a memory of past examples. Specifically, the algorithm can maintain a confidence parameter for each feature weight. For example, assuming binary features, the algorithm could keep a count of the number of times each feature has been observed, or, for general real-valued features, it could keep the cumulative second moment per feature. The larger the count or second moement, the more confidence in a feature's weight. These estimates are then used to influence parameter updates. Instead of equally updating every feature weight for the features present in an instance, the update favors changing more low-confidence weights than high-confidence ones. At each update, the confidence in all observed features is increased by focusing the update on low confidence features. In the example above, the update would decrease the weight of "dull" but make only a small change to "liked" since the algorithm already has a good estimate of this parameter.

3. Online Learning of Linear Classifiers

Online algorithms operate in rounds. On round i the algorithm receives an instance $\boldsymbol{x}_i \in \mathbb{R}^d$ to which it applies its current prediction rule to produce a prediction $\hat{y}_i \in \{-1, +1\}$ (for binary classification.) It then receives the true label $y_i \in \{-1, +1\}$ and suffers a loss $\ell(y_i, \hat{y}_i)$, which in this work will be the zero-one loss $\ell(y_i, \hat{y}_i) = 1$ if $y_i \neq \hat{y}_i$ and $\ell(y_i, \hat{y}_i) = 0$ otherwise. The algorithm then updates its prediction rule and proceeds to the next round.

Just as in many well known algorithms, such as the perceptron and support vector machines, in this work our prediction rules are linear classifiers

$$f_{\boldsymbol{w}}(\boldsymbol{x}) : f_{\boldsymbol{w}}(\boldsymbol{x}) = \text{sign}(\boldsymbol{x} \cdot \boldsymbol{w}) \ . \tag{1}$$

If we fix the norm of $\boldsymbol{w}$, we can identify $f_{\boldsymbol{w}}$ with $\boldsymbol{w}$, and we will use $\boldsymbol{w}$ in the rest of this work.

The *margin* of an example $(\boldsymbol{x}, y)$ with respect to a specific classifier $\boldsymbol{w}$ is given by $y(\boldsymbol{w} \cdot \boldsymbol{x})$. The sign of the margin is positive iff the classifier $\boldsymbol{w}$ predicts correctly the true label y. The absolute value of the margin $|y(\boldsymbol{w} \cdot \boldsymbol{x})| = |\boldsymbol{w} \cdot \boldsymbol{x}|$ is often thought of as the *confidence* in the prediction, with larger positive values corresponding to more confident correct predictions. We denote the margin at round i by $m_i = y_i(\boldsymbol{w}_i \cdot \boldsymbol{x}_i)$.

A variety of linear classifier training algorithms, including the perceptron and linear support vector machines, restrict $\boldsymbol{w}$ to be a linear combination of the input examples. Online algorithms of that kind typically have updates of the form

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i + \alpha_i y_i \boldsymbol{x}_i \ , \tag{2}$$

for some non-negative coefficients α_i.

In this paper we focus on PA updates (Crammer et al., 2006) for linear classifiers. After predicting with $\boldsymbol{w}_i$ on the ith round and receiving the true label y_i, the

algorithm updates the prediction function such that the example $(\boldsymbol{x}_i, y_i)$ will be classified correctly with a fixed margin (which can always be scaled to 1):

$$\begin{aligned} \boldsymbol{w}_{i+1} = \min_{\boldsymbol{w}} \quad & \frac{1}{2}\left\|\boldsymbol{w}_i - \boldsymbol{w}\right\|^2 \\ \text{s.t.} \quad & y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i) \geq 1 \ . \end{aligned} \tag{3}$$

The dual of (3) gives us the round coefficients for (2):

$$\alpha_i = \max\left\{\frac{1 - y_i\left(\boldsymbol{w}_i\cdot\boldsymbol{x}_i\right)}{\left\|\boldsymbol{x}_i\right\|^2}, 0\right\}$$

Crammer et al. (2006) provide a theoretical analysis of algorithms of this form, and they have been shown to work well in a variety of applications.

4. Distributions over Classifiers

We model parameter confidence for a linear classifier with a diagonal Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and standard deviation $\boldsymbol{\sigma} \in \mathbb{R}^d$. The values μ_j and σ_j represent our knowledge of and confidence in the parameter for feature j. The smaller σ_j, the more confidence we have in the mean parameter value μ_j. For simplicity of presentation, we use a covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$ for the distribution, with diagonal $\boldsymbol{\sigma}$ and zero for off-diagonal elements. Note that while our motivation assumed sparse binary features, the algorithm does not depend on that assumption.

Conceptually, to classify an input instance $\boldsymbol{x}$, we draw a parameter vector $\boldsymbol{w} \sim \mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right)$ and predict the label according to the sign of $\boldsymbol{w}\cdot\boldsymbol{x}$. This multivariate Gaussian distribution over parameter vectors induces a univariate Gaussian distribution over the margin viewed as a random variable:

$$M \sim \mathcal{N}\left(y_i(\boldsymbol{\mu}\cdot\boldsymbol{x}_i) \ , \ \boldsymbol{x}_i^\top\Sigma\boldsymbol{x}_i\right) \ .$$

The mean of the margin is the margin of the averaged parameter vector and its variance is proportional to the length of the projection of $\boldsymbol{x}_i$ on Σ_i. Since a prediction is correct iff the margin is non-negative, the probability of a correct prediction is

$$\Pr_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{\mu},\Sigma)}\left[M \geq 0\right] = \Pr_{\boldsymbol{w}\sim\mathcal{N}(\boldsymbol{\mu},\Sigma)}\left[y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i\right) \geq 0\right] \ .$$

When possible, we omit the explicit dependency on the distribution parameters and write $\Pr\left[y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i\right) \geq 0\right]$.

4.1. Update

On round i, the algorithm adjusts the distribution to ensure that the probability of a correct prediction for training instance i is no smaller than the confidence hyperparameter $\eta \in [0, 1]$:

$$\Pr\left[y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i\right) \geq 0\right] \geq \eta \ . \tag{4}$$

Following the intuition underlying the PA algorithms (Crammer et al., 2006), our algorithm chooses the distribution closest in the KL divergence sense to the current distribution $\mathcal{N}\left(\boldsymbol{\mu}_i, \Sigma_i\right)$. Thus, on round i, the algorithm sets the parameters of the distribution by solving the following optimization problem:

$$\left(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}\right) = \min \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right) \left\| \mathcal{N}\left(\boldsymbol{\mu}_i, \Sigma_i\right)\right.\right) \tag{5}$$

$$\text{s.t. } \Pr\left[y_i\left(\boldsymbol{w}\cdot\boldsymbol{x}_i\right) \geq 0\right] \geq \eta \ . \tag{6}$$

We now develop both the objective and the constraint of this optimization problem following Boyd and Vandenberghe, (2004, page 158). We start with the constraint (6). As noted above, under the distribution $\mathcal{N}\left(\boldsymbol{\mu}, \Sigma\right)$, the margin for $(\boldsymbol{x}_i, y_i)$ has a Gaussian distribution with mean $\mu_M = y_i\left(\boldsymbol{\mu}\cdot\boldsymbol{x}_i\right)$ and variance $\sigma_M^2 = \boldsymbol{x}_i^\top\Sigma\boldsymbol{x}_i$. Thus the probability of a *wrong* classification is

$$\Pr\left[M \leq 0\right] = \Pr\left[\frac{M - \mu_M}{\sigma_M} \leq \frac{-\mu_M}{\sigma_M}\right] \ .$$

Since $\left(M - \mu_M\right)/\sigma_M$ is a normally distributed random variable, the above probability equals $\Phi\left(-\mu_M/\sigma_M\right)$, where Φ is the cumulative function of the normal distribution. Thus we can rewrite (6) as

$$\frac{-\mu_M}{\sigma_M} \leq \Phi^{-1}\left(1 - \eta\right) = -\Phi^{-1}\left(\eta\right) \ .$$

Substituting μ_M and σ_M by their definitions and rearranging terms we obtain:

$$y_i(\boldsymbol{\mu}\cdot\boldsymbol{x}_i) \geq \phi\sqrt{\boldsymbol{x}_i^\top\Sigma\boldsymbol{x}_i} \ ,$$

where $\phi = \Phi^{-1}\left(\eta\right)$.

Unfortunately, this constraint is not convex in Σ, so we linearize it by omitting the square root:

$$y_i(\boldsymbol{\mu}\cdot\boldsymbol{x}_i) \geq \phi\left(\boldsymbol{x}_i^\top\Sigma\boldsymbol{x}_i\right) \ . \tag{7}$$

Conceptually, this is a large-margin constraint, where the value of the margin requirement depends on the example $\boldsymbol{x}_i$ via a quadratic form.

We now study the objective (5). The KL divergence between two Gaussians is given by

$$\begin{aligned} &\mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{\mu}_0, \Sigma_0\right) \left\| \mathcal{N}\left(\boldsymbol{\mu}_1, \Sigma_1\right)\right.\right) = \\ &\quad \frac{1}{2}\left(\log\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right) + \mathrm{Tr}\left(\Sigma_1^{-1}\Sigma_0\right)\right. \\ &\quad \left. + \left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right)^\top\Sigma_1^{-1}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right) - d\right) \ . \end{aligned} \tag{8}$$

Using the foregoing equations and omitting irrelevant constants, we obtain the following revised optimization problem:

$$(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}) = \min \frac{1}{2}\log\left(\frac{\det\Sigma_i}{\det\Sigma}\right) + \frac{1}{2}\mathrm{Tr}\left(\Sigma_i^{-1}\Sigma\right) + \frac{1}{2}\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}\right)^\top \Sigma_i^{-1}\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}\right)$$
$$\text{s.t. } y_i(\boldsymbol{\mu}\cdot\boldsymbol{x}_i) \geq \phi\left(\boldsymbol{x}_i^\top\Sigma\boldsymbol{x}_i\right) . \quad (9)$$

The optimization objective is convex in $\boldsymbol{\mu}$ and Σ simultaneously and the constraint is linear, so any convex optimization solver could be used to solve this problem. We call the corresponding update *Variance-Exact*. However, for efficiency we prefer a closed-form approximate update that we call *Variance*. In this approximation, we allow the solution for Σ_{i+1} in (9) to produce (implicitly) a full matrix, and then project it to a diagonal matrix, where the non-zero off-diagonal entries are dropped. The Lagrangian for this optimization is

$$\mathcal{L} = \frac{1}{2}\log\left(\frac{\det\Sigma_i}{\det\Sigma}\right) + \frac{1}{2}\mathrm{Tr}\left(\Sigma_i^{-1}\Sigma\right) + \frac{1}{2}\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}\right)^\top \Sigma_i^{-1}\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}\right) + \alpha\left(-y_i\left(\boldsymbol{\mu}\cdot\boldsymbol{x}_i\right) + \phi\left(\boldsymbol{x}_i^\top\Sigma\boldsymbol{x}_i\right)\right) . \quad (10)$$

At the optimum, we must have

$$\frac{\partial}{\partial\boldsymbol{\mu}}\mathcal{L} = \Sigma_i^{-1}\left(\boldsymbol{\mu} - \boldsymbol{\mu}_i\right) - \alpha y_i\boldsymbol{x}_i = 0 .$$

Assuming Σ_i is non-singular we get,

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha y_i\Sigma_i\boldsymbol{x}_i . \quad (11)$$

At the optimum, we must also have

$$\frac{\partial}{\partial\Sigma}\mathcal{L} = -\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma_i^{-1} + \phi\alpha\boldsymbol{x}_i\boldsymbol{x}_i^\top = 0 .$$

Solving for Σ^{-1} we obtain

$$\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + 2\alpha\phi\boldsymbol{x}_i\boldsymbol{x}_i^\top . \quad (12)$$

Finally, we compute the inverse of (12) using the Woodbury identity (Petersen & Pedersen, 2007, Eq. 135) and get,

$$\Sigma_{i+1} = \left(\Sigma_i^{-1} + 2\alpha\phi\boldsymbol{x}_i\boldsymbol{x}_i^\top\right)^{-1} = \Sigma_i - \Sigma_i\boldsymbol{x}_i\left(\frac{1}{2\alpha\phi} + \boldsymbol{x}_i^\top\Sigma_i\boldsymbol{x}_i\right)^{-1}\boldsymbol{x}_i^\top\Sigma_i = \Sigma_i - \Sigma_i\boldsymbol{x}_i\frac{2\alpha\phi}{1 + 2\alpha\phi\boldsymbol{x}_i^\top\Sigma_i\boldsymbol{x}_i}\boldsymbol{x}_i^\top\Sigma_i . \quad (13)$$

The KKT conditions for the optimization imply that the either $\alpha = 0$, and no update is needed, or the constraint (7) is an equality after the udpate. Substituting (11) and (13) into the equality version of (7), we obtain:

$$y_i\left(\boldsymbol{x}_i\cdot\left(\boldsymbol{\mu}_i + \alpha y_i\Sigma_i\boldsymbol{x}_i\right)\right) = \phi\left(\boldsymbol{x}_i^\top\left(\Sigma_i - \Sigma_i\boldsymbol{x}_i\frac{2\alpha\phi}{1 + 2\alpha\phi\boldsymbol{x}_i^\top\Sigma_i\boldsymbol{x}_i}\boldsymbol{x}_i^\top\Sigma_i\right)\boldsymbol{x}_i\right) . \quad (14)$$

Rearranging terms we get,

$$y_i\left(\boldsymbol{x}_i\cdot\boldsymbol{\mu}_i\right) + \alpha\boldsymbol{x}_i^\top\Sigma_i\boldsymbol{x}_i = \phi\boldsymbol{x}_i^\top\Sigma_i\boldsymbol{x}_i - \phi\left(\boldsymbol{x}_i^\top\Sigma_i\boldsymbol{x}_i\right)^2\frac{2\alpha\phi}{1 + 2\alpha\phi\boldsymbol{x}_i^\top\Sigma_i\boldsymbol{x}_i} . \quad (15)$$

For simplicity, let $M_i = y_i\left(\boldsymbol{x}_i\cdot\boldsymbol{\mu}_i\right)$ be the mean margin and $V_i = \boldsymbol{x}_i^\top\Sigma_i\boldsymbol{x}_i$ be the margin variance before the update. Substituting these into (15) we get,

$$M_i + \alpha V_i = \phi V_i - \phi V_i^2\frac{2\alpha\phi}{1 + 2\alpha\phi V_i} .$$

It is straightforward to see that this is a quadratic equation in α. Its smaller root is always negative and thus is not a valid Lagrange multiplier. Let γ_i be its larger root:

$$\gamma_i = \frac{-(1+2\phi M_i) + \sqrt{(1+2\phi M_i)^2 - 8\phi\left(M_i - \phi V_i\right)}}{4\phi V_i} . \quad (16)$$

The constraint (7) is satisfied before the update if $M_i - \phi V_i \geq 0$. If $1 + 2\phi M_i \leq 0$, then $M_i \leq \phi V_i$ and from (16) we have that $\gamma_i > 0$. If, instead, $1 + 2\phi M_i \geq 0$, then, again by (16), we have

$$\gamma_i > 0 \Leftrightarrow \sqrt{(1 + 2\phi M_i)^2 - 8\phi\left(M_i - \phi V_i\right)} > (1 + 2\phi M_i) \Leftrightarrow M_i < \phi V_i .$$

From the KKT conditions, either $\alpha_i = 0$ or (9) is satisfied as an equality. In the later case, (14) holds, and thus $\alpha_i = \gamma_i > 0$. To summarize, we have proved the following:

Lemma 1 *The optimal value of the Lagrange multiplier is given by* $\alpha_i = \max\{\gamma_i, 0\}$.

The above derivation yields a full covariance matrix. As noted above, we restrict ourself to diagonal matrices and thus we project the solution into the set of diagonal matrices to get our approximation. In practice, it is equivalent to compute α_i as above but update with the following rule instead of (12).

$$\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + 2\alpha\phi\mathrm{diag}\left(\boldsymbol{x}_i\right) , \quad (17)$$

Algorithm 1 Variance Algorithm (Approximate)

Input: confidence parameter $\phi = \Phi^{-1}(\eta)$
initial variance parameter $a > 0$
Initialize: $\boldsymbol{\mu}_1 = \mathbf{0}$, $\Sigma_1 = aI$
for $i = 1, 2 \ldots$ **do**
Receive $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \{+1, -1\}$
Set the following variables:
α_i as in Lemma 1
$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i y_i \Sigma_i \boldsymbol{x}_i$ (11)
$\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + 2\alpha_i \phi \operatorname{diag}(\boldsymbol{x}_i)$ (17)
end for

comp	`comp.sys.ibm.pc.hardware`
	`comp.sys.mac.hardware`
sci	`sci.electronics`
	`sci.med`
talk	`talk.politics.guns`
	`talk.politics.mideast`

Table 1. 20 Newsgroups binary decision tasks.

where $\operatorname{diag}(\boldsymbol{x}_i)$ is a diagonal matrix with the square of the elements of $\boldsymbol{x}_i$ on the diagonal.

The pesudocode of the algorithm appears in Alg. 1. From the initalization of Σ_1 and the update rule of (12), we conclude that the eigenvalues of Σ_i are shirinking, but never set to zero explicltly, and thus the covariance matrices Σ_i are not singular.

5. Evaluation

We evaluated our Variance and Variance-Exact algorithms on three popular NLP datasets. Each dataset contains several binary classification tasks from which we selected a total of 12 problems, each contains a balanced mixture of instance labels.

20 Newsgroups The 20 Newsgroups corpus contains approximately 20,000 newsgroup messages, partitioned across 20 different newsgroups.[2] The dataset is a popular choice for binary and multi-class text classification as well as unsupervised clustering. Following common practice, we created binary problems from the dataset by creating binary decision problems of choosing between two similar groups, as shown in Table 1. Each message was represented as a binary bag-of-words. For each problem we selected 1800 instances.

Reuters The Reuters Corpus Volume 1 (RCV1-v2/LYRL2004) contains over 800,000 manually categorized newswire stories (Lewis et al., 2004). Each article

[2] `http://people.csail.mit.edu/jrennie/20Newsgroups/`

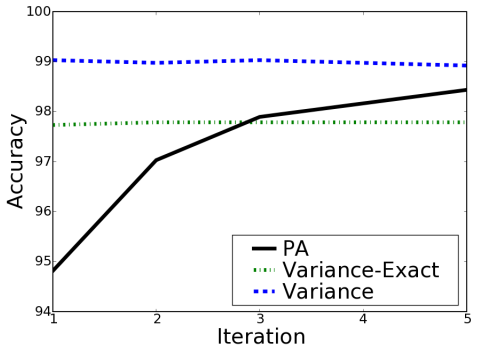


Figure 1. Accuracy on test data after each iteration on the "talk" dataset.

contains one or more labels describing its general topic, industry and region. We created the following binary decision tasks from the labeled documents: Insurance: Life (I82002) vs. Non-Life (I82003), Business Services: Banking (I81000) vs. Financial (I83000), and Retail Distribution: Specialist Stores (I65400) vs. Mixed Retail (I65600). These distinctions involve neighboring categories so they are fairly hard to make. Details on document preparation and feature extraction are given by Lewis et al. (2004). For each problem we selected 2000 instances using a bag of words representation with binary features.

Sentiment We obtained a larger version of the sentiment multi-domain dataset of Blitzer et al. (2007) containing product reviews from 6 Amazon domains (book, dvd, electronics, kitchen, music, video). The goal in each domain is to classify a product review as either positive or negative. Feature extraction follows Blitzer et al. (2007). For each problem we selected 2000 instances using uni/bi-grams with counts.

Each dataset was randomly divided for 10-fold cross validation experiments. Classifier parameters (ϕ for CW and C for PA) were tuned for each classification task on a single randomized run over the data. Results are reported for each problem as the average accuracy over the 10 folds. Statistical significance is computed using McNemar's test.

5.1. Results

We start by examining the performance of the Variance and Variance-Exact versions of our method, discussed in the preceding section, against a PA algorithm. All three algorithms were run on the datasets described above and each training phase consisted of five passes over the training data, which seemed to be

	Task	*PA*	*Variance*	*Variance-Exact*	*SVM*	*Maxent*	*SGD*
20 Newsgroups	comp	8.90	†**6.33**	9.63	∗7.67	∗7.62	7.36
	sci	4.22	†**1.78**	3.3	†3.51	†3.55	†4.77
	talk	1.57	1.09	2.21	**0.91**	**0.91**	1.36
Reuters	Business	17.80	17.65	17.70	⋆15.64	⋆**15.10**	⋆15.85
	Insurance	9.76	∗**8.45**	9.49	9.19	8.59	9.05
	Retail	15.41	†**11.05**	14.14	∗12.80	∗12.30	†14.31
Sentiment	books	19.55	∗**17.40**	20.45	†20.45	†19.91	∗19.41
	dvds	19.71	**19.11**	19.91	20.09	19.26	20.20
	electronics	17.40	†**14.10**	17.44	†16.80	†16.21	†16.81
	kitchen	15.64	∗**14.24**	16.35	15.20	14.94	∗15.60
	music	20.05	∗**18.10**	19.66	19.35	19.45	18.81
	videos	19.86	⋆**17.20**	19.85	†20.70	†19.45	⋆19.65

Table 2. Error on test data using batch training. Statistical significance (McNemar) is measured against PA or the batch method against Variance. (∗ p=.05, ⋆ p=.01, † p=.001)

enough to yield convergence. The average error on the test set for the three algorithms on all twelve datasets is shown in table 2.

Variance-Exact achieved about the same performance as PA, with each method achieving a lower error on half of the datasets. In contrast, Variance (approximate) significantly improves over PA, achieving lower error on all twelve datasets, with statistically significant results on nine of them.

As discussed above, online algorithms are attractive even for batch learning because of their simplicity and ability to operate on extremely large datasets. In the batch setting, these algorithms are run several times over the training data, which yields slower performance than single pass learning (Carvalho & Cohen, 2006). Our algorithm improves on both accuracy and learning speed by requiring fewer iterations over the training data. Such behavior can be seen on the "talk" dataset in Figure 1, which shows accuracy on test data after each iteration of the PA baseline and the two variance algorithms. While Variance clearly improves over PA, it converges very quickly, reaching near best performance on the first iteration. In contrast, PA benefits from multiple iterations over the data; its performance changes significantly from the first to fifth iteration. Across the twelve tasks, Variance yields a 3.7% error reduction while PA gives a 12.4% reduction between the first and fifth iteration, indicating that multiple iterations help PA more. The plot also illustrates Variance-Exact's behavior, which initially beats PA but does not improve. In fact, on eleven of the twelve datasets, Variance-Exact beats PA on the first iteration. The exact update results in aggressive behavior causing the algorithm to converge very quickly, even more so than Variance. It appears that the approximate update in Variance reduces overtraining and yields the best accuracy.

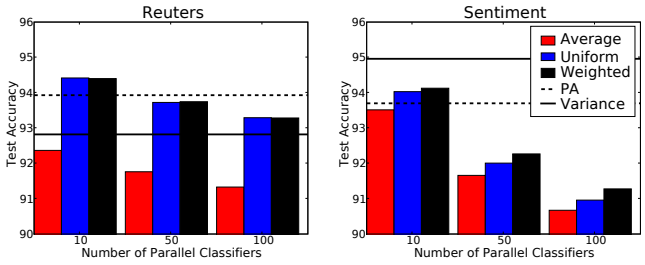


Figure 2. Results for Reuters (800k) and Sentiment (1000k) averaged over 4 runs. Horizontal lines show the test accuracy of a model trained on the entire training set. Vertical bars show the performance of n (10, 50, 100) classifiers trained on disjoint sections of the data as the average performance, uniform combination, or weighted combination. All improvements are statistically significant except between uniform and weighted for Reuters.

5.2. Batch Learning

While online algorithms are widely used, batch algorithms are still preferred for many tasks. Batch algorithms can make global learning decisions by examining the entire dataset, an ability beyond online algorithms. In general, when batch algorithms can be applied they perform better. We compare our new online algorithm (Variance) against two standard batch algorithms: maxent classification (default configuration of the maxent learner in McCallum (2002)) and support vector machines (LibSVM (Chang & Lin, 2001)). We also include stochastic gradient descent (SGD) (Blitzer et al., 2007), which performs well for NLP tasks. Classifier parameters (Gaussian prior for maxent, C for SVM and the learning rate for SGD) were tuned as for the online methods.

Results for batch learning are shown in Table 2. As expected, the batch methods tend to do better than PA,

with SVM doing better 9 times and maxent 11 times. However, in most cases Variance improves over the batch method, doing better than SVM and maxent 10 out of 12 times (7 statistically significant.) These results show that in these tasks, the much faster and simpler online algorithm performs better than the slower more complex batch methods.

We also evaluated the effects of commonly used techniques for online and batch learning, including averaging and TFIDF features, none of which improved accuracy. Although the above datasets are balanced with respect to labels and predictive features, we also evaluated the methods on variant datasets with unbalanced label or feature distributions, and still saw similar benefits from the Variance method.

5.3. Large Datasets

Online algorithms are especially attractive in tasks where training data exceeds available main memory. However, even a single sequential pass over the data can be impractical for extremely large training sets, so we investigate training different models on different portions of the data in parallel and combining the learned classifiers into a single classifier. While this often does not perform as well as a single model trained on all of the data, it is a cost effective way of learning from very large training sets.

Averaging models trained in parallel assumes that each model has an equally accurate estimate of the model parameters. However, our model provides a confidence value for each parameter, allowing for a more intelligent combination of parameters from multiple models. Specifically, we compute the combined model Gaussian that minimizes the total divergence to the set C of individually trained classifiers for some divergence operator D:

$$\min_{\boldsymbol{\mu},\Sigma} \sum_{c \in C} D((\boldsymbol{\mu}, \Sigma) || (\boldsymbol{\mu}_c, \Sigma_c)), \tag{18}$$

If D is the Euclidean distance, this is just the average of the individual models. If D is the KL divergence, the minimization leads to the following weighted combination of individual model means:
$\boldsymbol{\mu} = \left(\sum_{c \in C} \Sigma_c^{-1}\right)^{-1} \sum_{c \in C} \Sigma_c^{-1} \boldsymbol{\mu}_c$, $\Sigma^{-1} = \sum_{c \in C} \Sigma_c^{-1}$.

We evaluate the single model performance of the PA baseline and our method. For our method, we evaluate classifier combination by training n (10, 50, 100) models by dividing the instance stream into n disjoint parts and report the average performance of each of the n classifiers (average), the combined classifier from taking the average of the n sets of parameters (uniform) and the combination using the KL distance (weighted) on the test data across 4 randomized runs.

We evaluated classifier combination on two datasets. The combined product reviews for all the domains in Blitzer et al. (2007) yield one million sentiment instances. While most reviews were from the book domain, the reviews are taken from a wide range of Amazon product types and are mostly positive. From the Reuters corpus, we created a one vs. all classification task for the *Corporate* topic label, yielding 804,411 instances of which 381,325 are labeled corporate. For the two datasets, we created four random splits each with one million training instances and 10,000 test instances. Parameters were optimized by training on 5K random instances and testing on 10K.

The two datasets use very different feature representations. The Reuters data contains 288,062 unique features, for a feature to document ratio of 0.36. In contrast, the sentiment data contains 13,460,254 unique features, a feature to document ratio of 13.33. This means that Reuters features tend to occur several times during training while many sentiment features occur only once.

Average accuracy on the test sets are reported in Figure 2. For Reuters data, the PA single model achieves higher accuracy than Variance, possibly because of the low feature to document ratio. However, combining 10 Variance classifiers achieves the best performance. For sentiment, combining 10 classifiers beats PA but is not as good as a single Variance model. In every case, combining the classifiers using either uniform or weighted improves over each model individually. On sentiment weighted combination improves over uniform combination and in Reuters the models are equivalent.

Finally, we computed the actual run time of both PA and Variance on the large datasets to compare the speed of each model. While Variance is more complex, requiring more computation per instance, the actual speed is comparable to PA; in all tests the run time of the two algorithms was indistinguishable.

6. Related Work

The idea of using parameter-specific variable learning rates has a long history in neural-network learning (Sutton, 1992), although we do not know of a previous model that specifically models confidence in a way that takes into account the frequency of features. The second-order perceptron (SOP) (Cesa-Bianchi et al., 2005) is perhaps the closest to our CW algorithm. Both are online algorithms that maintain a weight vector and some statistics about previous examples. While the SOP models certainty with feature counts,

CW learning models uncertainty with a Gaussian distribution. CW algorithms have a probabilistic motivation, while the SOP is based on the geometric idea of replacing a ball around the input examples with a refined ellipsoid. Shivaswamy and Jebara (2007) used this intuition in the context of batch learning.

Gaussian process classification (GPC) maintains a Gaussian distribution over weight vectors (primal) or over regressor values (dual). Our algorithm uses a different update criterion than the the standard Bayesian updates used in GPC (Rasmussen & Williams, 2006, Ch. 3), avoiding the challenging issues in approximating posteriors in GPC. Bayes point machines (Herbrich et al., 2001) maintain a collection of weight vectors consistent with the training data, and use the single linear classifier which best represents the collection. Conceptually, the collection is a non-parametric distribution over the weight vectors. Its online version (Harrington et al., 2003) maintains a finite number of weight-vectors updated simultaneously.

Finally, with the growth of available data there is an increasing need for algorithms that process training data very efficiently. A similar approach to ours is to train classifiers incrementally (Bordes & Bottou, 2005). The extreme case is to use each example once, without repetitions, as in the multiplicative update method of Carvalho and Cohen (2006).

Conclusion: We have presented confidence-weighted linear classifiers, a new learning method designed for NLP problems based on the notion of parameter confidence. The algorithm maintains a distribution over parameter vectors; online updates both improve the parameter estimates and reduce the distribution's variance. Our method improves over both online and batch methods and learns faster on a dozen NLP datasets. Additionally, our new algorithms allow more intelligent classifier combination techniques, yielding improved performance after parallel learning. We plan to explore theoretical properties and other aspects of CW classifiers, such as multi-class and structured prediction tasks, and other data types.

Acknowledgements: This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185.

References

Blitzer, J., Dredze, M., & Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Association of Computational Linguistics (ACL)*.

Bordes, A., & Bottou, L. (2005). The huller: a simple and efficient online svm. *European Conference on Machine Learning(ECML), LNAI 3720*.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Carvalho, V. R., & Cohen, W. W. (2006). Single-pass online learning: Performance, voting schemes and online feature selection. *KDD-2006*.

Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2005). A second-order perceptron algorithm. *SIAM Journal on Computing*, *34*, 640 – 668.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *JMLR*, *7*, 551–585.

Harrington, E., Herbrich, R., Kivinen, J., Platt, J., & Williamson, R. (2003). Online bayes point machines. *7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*.

Herbrich, R., Graepel, T., & C.Campbell (2001). Bayes point machinesonline passive-aggressive algorithms. *JMLR*, *1*, 245–279.

Lewis, D. D., Yand, Y., Rose, T., & Li., F. (2004). Rcv1: A new benchmark collection for text categorization research. *JMLR*, *5*, 361–397.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Petersen, K. B., & Pedersen, M. S. (2007). *The matrix cookbook*.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. The MIT Press.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psych. Rev.*, *68*, 386–407.

Shivaswamy, P., & Jebara, T. (2007). Ellipsoidal kernel machines. *Artificial Intelligence and Statistics*.

Sutton, R. S. (1992). Adapting bias by gradient descent: an incremental version of delta-bar-delta. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 171–176). MIT Press.

Efficient Projections onto the ℓ_1-Ball for Learning in High Dimensions

John Duchi JDUCHI@CS.STANFORD.EDU
Google, Mountain View, CA 94043

Shai Shalev-Shwartz SHAI@TTI-C.ORG
Toyota Technological Institute, Chicago, IL, 60637

Yoram Singer SINGER@GOOGLE.COM
Tushar Chandra TUSHAR@GOOGLE.COM
Google, Mountain View, CA 94043

Abstract

We describe efficient algorithms for projecting a vector onto the ℓ_1-ball. We present two methods for projection. The first performs exact projection in $O(n)$ expected time, where n is the dimension of the space. The second works on vectors k of whose elements are perturbed outside the ℓ_1-ball, projecting in $O(k \log(n))$ time. This setting is especially useful for online learning in sparse feature spaces such as text categorization applications. We demonstrate the merits and effectiveness of our algorithms in numerous batch and online learning tasks. We show that variants of stochastic gradient projection methods augmented with our efficient projection procedures outperform interior point methods, which are considered state-of-the-art optimization techniques. We also show that in online settings gradient updates with ℓ_1 projections outperform the exponentiated gradient algorithm while obtaining models with high degrees of sparsity.

1. Introduction

A prevalent machine learning approach for decision and prediction problems is to cast the learning task as penalized convex optimization. In penalized convex optimization we seek a set of parameters, gathered together in a vector $\mathbf{w}$, which minimizes a convex objective function in $\mathbf{w}$ with an additional penalty term that assesses the complexity of $\mathbf{w}$. Two commonly used penalties are the 1-norm and the square of the 2-norm of $\mathbf{w}$. An alternative but mathematically equivalent approach is to cast the problem as a *constrained* optimization problem. In this setting we seek a minimizer of the objective function while constraining the solution to have a bounded norm. Many recent advances in statistical machine learning and related fields can be explained as convex optimization subject to a 1-norm constraint on the vector of parameters $\mathbf{w}$. Imposing an ℓ_1 constraint leads to notable benefits. First, it encourages sparse solutions, *i.e* a solution for which many components of $\mathbf{w}$ are zero. When the original dimension of $\mathbf{w}$ is very high, a sparse solution enables easier interpretation of the problem in a lower dimension space. For the usage of ℓ_1-based approach in statistical machine learning see for example (Tibshirani, 1996) and the references therein. Donoho (2006b) provided sufficient conditions for obtaining an optimal ℓ_1-norm solution which is sparse. Recent work on compressed sensing (Candes, 2006; Donoho, 2006a) further explores how ℓ_1 constraints can be used for recovering a sparse signal sampled below the Nyquist rate. The second motivation for using ℓ_1 constraints in machine learning problems is that in some cases it leads to improved generalization bounds. For example, Ng (2004) examined the task of PAC learning a sparse predictor and analyzed cases in which an ℓ_1 constraint results in better solutions than an ℓ_2 constraint.

In this paper we re-examine the task of minimizing a convex function subject to an ℓ_1 constraint on the norm of the solution. We are particularly interested in cases where the convex function is the average loss over a training set of m examples where each example is represented as a vector of high dimension. Thus, the solution itself is a high-dimensional vector as well. Recent work on ℓ_2 constrained optimization for machine learning indicates that gradient-related projection algorithms are more efficient in approaching a solution of good generalization than second-order algorithms when the number of examples and

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

the dimension are large. For instance, Shalev-Shwartz et al. (2007) give recent state-of-the-art methods for solving large scale support vector machines. Adapting these recent results to projection methods onto the ℓ_1 ball poses algorithmic challenges. While projections onto ℓ_2 balls are straightforward to implement in linear time with the appropriate data structures, projection onto an ℓ_1 ball is a more involved task. The main contribution of this paper is the derivation of gradient projections with ℓ_1 domain constraints that can be performed almost as fast as gradient projection with ℓ_2 constraints.

Our starting point is an efficient method for projection onto the probabilistic simplex. The basic idea is to show that, after sorting the vector we need to project, it is possible to calculate the projection exactly in linear time. This idea was rediscovered multiple times. It was first described in an abstract and somewhat opaque form in the work of Gafni and Bertsekas (1984) and Bertsekas (1999). Crammer and Singer (2002) rediscovered a similar projection algorithm as a tool for solving the dual of multiclass SVM. Hazan (2006) essentially reuses the same algorithm in the context of online convex programming. Our starting point is another derivation of Euclidean projection onto the simplex that paves the way to a few generalizations. First we show that the same technique can also be used for projecting onto the ℓ_1-ball. This algorithm is based on sorting the components of the vector to be projected and thus requires $O(n \log(n))$ time. We next present an improvement of the algorithm that replaces sorting with a procedure resembling median-search whose expected time complexity is $O(n)$.

In many applications, however, the dimension of the feature space is very high yet the number of features which attain non-zero values for an example may be very small. For instance, in our experiments with text classification in Sec. 7, the dimension is two million (the bigram dictionary size) while each example has on average one-thousand non-zero features (the number of unique tokens in a document). Applications where the dimensionality is high yet the number of "on" features in each example is small render our second algorithm useless in some cases. We therefore shift gears and describe a more complex algorithm that employs red-black trees to obtain a linear dependence on the number of non-zero features in an example and only logarithmic dependence on the full dimension. The key to our construction lies in the fact that we project vectors that are the sum of a vector in the ℓ_1-ball and a sparse vector—they are "almost" in the ℓ_1-ball.

In conclusion to the paper we present experimental results that demonstrate the merits of our algorithms. We compare our algorithms with several specialized interior point (IP) methods as well as general methods from the literature for solving ℓ_1-penalized problems on both synthetic and real data (the MNIST handwritten digit dataset and the Reuters RCV1 corpus) for batch and online learning. Our projection based methods outperform competing algorithms in terms of sparsity, and they exhibit faster convergence and lower regret than previous methods.

2. Notation and Problem Setting

We start by establishing the notation used throughout the paper. The set of integers 1 through n is denoted by $[n]$. Scalars are denoted by lower case letters and vectors by lower case bold face letters. We use the notation $\mathbf{w} \succ b$ to designate that all of the components of $\mathbf{w}$ are greater than b. We use $\|\cdot\|$ as a shorthand for the Euclidean norm $\|\cdot\|_2$. The other norm we use throughout the paper is the 1-norm of the vector, $\|\mathbf{v}\|_1 = \sum_{i=1}^n |v_i|$. Lastly, we consider order statistics and sorting vectors frequently throughout this paper. To that end, we let $v_{(i)}$ denote the i^{th} order statistic of $\mathbf{v}$, that is, $v_{(1)} \geq v_{(2)} \geq \ldots \geq v_{(n)}$ for $\mathbf{v} \in \mathbb{R}^n$.

In the setting considered in this paper we are provided with a convex function $L : \mathbb{R}^n \to \mathbb{R}$. Our goal is to find the minimum of $L(\mathbf{w})$ subject to an ℓ_1-norm constraint on $\mathbf{w}$. Formally, the problem we need to solve is

$$\underset{\mathbf{w}}{\text{minimize}}\, L(\mathbf{w}) \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \leq z \ . \tag{1}$$

Our focus is on variants of the projected subgradient method for convex optimization (Bertsekas, 1999). Projected subgradient methods minimize a function $L(\mathbf{w})$ subject to the constraint that $\mathbf{w} \in X$, for X convex, by generating the sequence $\{\mathbf{w}^{(t)}\}$ via

$$\mathbf{w}^{(t+1)} = \Pi_X \left(\mathbf{w}^{(t)} - \eta_t \nabla^{(t)} \right) \tag{2}$$

where $\nabla^{(t)}$ is (an unbiased estimate of) the (sub)gradient of L at $\mathbf{w}^{(t)}$ and $\Pi_X(\mathbf{x}) = \operatorname{argmin}_{\mathbf{y}} \{\|\mathbf{x} - \mathbf{y}\| \mid \mathbf{y} \in X\}$ is Euclidean projection of $\mathbf{x}$ onto X. In the rest of the paper, the main algorithmic focus is on the projection step (computing an unbiased estimate of the gradient of $L(\mathbf{w})$ is straightforward in the applications considered in this paper, as is the modification of $\mathbf{w}^{(t)}$ by $\nabla^{(t)}$).

3. Euclidean Projection onto the Simplex

For clarity, we begin with the task of performing Euclidean projection onto the positive simplex; our derivation naturally builds to the more efficient algorithms. As such, the most basic projection task we consider can be formally described as the following optimization problem,

$$\underset{\mathbf{w}}{\text{minimize}}\, \frac{1}{2}\|\mathbf{w} - \mathbf{v}\|_2^2 \quad \text{s.t.} \quad \sum_{i=1}^n w_i \ = \ z \ , \ \ w_i \geq 0 \ . \tag{3}$$

When $z = 1$ the above is projection onto the probabilistic simplex. The Lagrangian of the problem in Eq. (3) is

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\zeta}) \;=\; \frac{1}{2}\|\mathbf{w} - \mathbf{v}\|^2 + \theta\left(\sum_{i=1}^{n} w_i - z\right) - \boldsymbol{\zeta}\cdot\mathbf{w} \;\;,$$

where $\theta \in \mathbb{R}$ is a Lagrange multiplier and $\boldsymbol{\zeta} \in \mathbb{R}^n_+$ is a vector of non-negative Lagrange multipliers. Differentiating with respect to w_i and comparing to zero gives the optimality condition, $\frac{d\mathcal{L}}{dw_i} \;=\; w_i - v_i + \theta - \zeta_i \;=\; 0$. The complementary slackness KKT condition implies that whenever $w_i > 0$ we must have that $\zeta_i = 0$. Thus, if $w_i > 0$ we get that

$$w_i \;=\; v_i - \theta + \zeta_i \;=\; v_i - \theta \;. \tag{4}$$

All the non-negative elements of the vector $\mathbf{w}$ are tied via a single variable, so knowing the indices of these elements gives a much simpler problem. Upon first inspection, finding these indices seems difficult, but the following lemma (Shalev-Shwartz & Singer, 2006) provides a key tool in deriving our procedure for identifying non-zero elements.

Lemma 1. *Let $\mathbf{w}$ be the optimal solution to the minimization problem in Eq. (3). Let s and j be two indices such that $v_s > v_j$. If $w_s = 0$ then w_j must be zero as well.*

Denoting by I the set of indices of the non-zero components of the sorted optimal solution, $I = \{i \in [n] : v_{(i)} > 0\}$, we see that Lemma 1 implies that $I = [\rho]$ for some $1 \le \rho \le n$. Had we known ρ we could have simply used Eq. (4) to obtain that

$$\sum_{i=1}^{n} w_i = \sum_{i=1}^{n} w_{(i)} = \sum_{i=1}^{\rho} w_{(i)} = \sum_{i=1}^{\rho} \left(v_{(i)} - \theta\right) = z$$

and therefore

$$\theta = \frac{1}{\rho}\left(\sum_{i=1}^{\rho} v_{(i)} - z\right) \;\;. \tag{5}$$

Given θ we can characterize the optimal solution for $\mathbf{w}$ as

$$w_i \;=\; \max\left\{v_i - \theta \;,\; 0\right\} \;\;. \tag{6}$$

We are left with the problem of finding the optimal ρ, and the following lemma (Shalev-Shwartz & Singer, 2006) provides a simple solution once we sort $\mathbf{v}$ in descending order.

Lemma 2. *Let $\mathbf{w}$ be the optimal solution to the minimization problem given in Eq. (3). Let $\boldsymbol{\mu}$ denote the vector obtained by sorting $\mathbf{v}$ in a descending order. Then, the number of strictly positive elements in $\mathbf{w}$ is*

$$\rho(z, \boldsymbol{\mu}) \;=\; \max\left\{j \in [n] \;:\; \mu_j - \frac{1}{j}\left(\sum_{r=1}^{j}\mu_r - z\right) > 0\right\} \;\;.$$

The pseudo-code describing the $O(n \log n)$ procedure for solving Eq. (3) is given in Fig. 1.

INPUT: A vector $\mathbf{v} \in \mathbb{R}^n$ and a scalar $z > 0$

Sort $\mathbf{v}$ into $\boldsymbol{\mu} : \mu_1 \ge \mu_2 \ge \ldots \ge \mu_p$

Find $\rho = \max\left\{j \in [n] : \mu_j - \frac{1}{j}\left(\sum_{r=1}^{j}\mu_r - z\right) > 0\right\}$

Define $\theta = \frac{1}{\rho}\left(\sum_{i=1}^{\rho}\mu_i - z\right)$

OUTPUT: $\mathbf{w}$ s.t. $w_i = \max\left\{v_i - \theta \;,\; 0\right\}$

Figure 1. Algorithm for projection onto the simplex.

4. Euclidean Projection onto the ℓ_1-Ball

We next modify the algorithm to handle the more general ℓ_1-norm constraint, which gives the minimization problem

$$\underset{\mathbf{w}\in\mathbb{R}^n}{\text{minimize}} \;\; \|\mathbf{w} - \mathbf{v}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \le z \;\;. \tag{7}$$

We do so by presenting a reduction to the problem of projecting onto the simplex given in Eq. (3). First, we note that if $\|\mathbf{v}\|_1 \le z$ then the solution of Eq. (7) is $\mathbf{w} = \mathbf{v}$. Therefore, from now on we assume that $\|\mathbf{v}\|_1 > z$. In this case, the optimal solution must be on the boundary of the constraint set and thus we can replace the inequality constraint $\|\mathbf{w}\|_1 \le z$ with an equality constraint $\|\mathbf{w}\|_1 = z$. Having done so, the sole difference between the problem in Eq. (7) and the one in Eq. (3) is that in the latter we have an additional set of constraints, $\mathbf{w} \ge 0$. The following lemma indicates that each non-zero component of the optimal solution $\mathbf{w}$ shares the sign of its counterpart in $\mathbf{v}$.

Lemma 3. *Let $\mathbf{w}$ be an optimal solution of Eq. (7). Then, for all i, $w_i\, v_i \ge 0$.*

Proof. Assume by contradiction that the claim does not hold. Thus, there exists i for which $w_i\, v_i < 0$. Let $\hat{\mathbf{w}}$ be a vector such that $\hat{w}_i = 0$ and for all $j \ne i$ we have $\hat{w}_j = w_j$. Therefore, $\|\hat{\mathbf{w}}\|_1 = \|\mathbf{w}\|_1 - |w_i| \le z$ and hence $\hat{\mathbf{w}}$ is a feasible solution. In addition,

$$\begin{aligned}\|\mathbf{w} - \mathbf{v}\|_2^2 - \|\hat{\mathbf{w}} - \mathbf{v}\|_2^2 \;&=\; (w_i - v_i)^2 - (0 - v_i)^2 \\ &=\; w_i^2 - 2w_i v_i > w_i^2 > 0 \;\;.\end{aligned}$$

We thus constructed a feasible solution $\hat{\mathbf{w}}$ which attains an objective value smaller than that of $\mathbf{w}$. This leads us to the desired contradiction. □

Based on the above lemma and the symmetry of the objective, we are ready to present our reduction. Let $\mathbf{u}$ be a vector obtained by taking the absolute value of each component of $\mathbf{v}$, $u_i = |v_i|$. We now replace Eq. (7) with

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^n}{\text{minimize}} \;\; \|\boldsymbol{\beta} - \mathbf{u}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_1 \le z \;\text{ and }\; \boldsymbol{\beta} \ge 0 \;\;. \tag{8}$$

Once we obtain the solution for the problem above we construct the optimal of Eq. (7) by setting $w_i = \text{sign}(v_i)\,\beta_i$.

INPUT A vector $\mathbf{v} \in \mathbb{R}^n$ and a scalar $z > 0$
INITIALIZE $U = [n]$ $s = 0$ $\rho = 0$
WHILE $U \neq \phi$
  PICK $k \in U$ at random
  PARTITION U:
    $G = \{j \in U \mid v_j \geq v_k\}$
    $L = \{j \in U \mid v_j < v_k\}$
  CALCULATE $\Delta\rho = |G|$; $\Delta s = \sum_{j \in G} v_j$
  IF $(s + \Delta s) - (\rho + \Delta\rho)v_k < z$
    $s = s + \Delta s$; $\rho = \rho + \Delta\rho$; $U \leftarrow L$
  ELSE
    $U \leftarrow G \setminus \{k\}$
  ENDIF
SET $\theta = (s - z)/\rho$
OUTPUT $\mathbf{w}$ s.t. $v_i = \max\{v_i - \theta\ ,\ 0\}$

Figure 2. Linear time projection onto the simplex.

5. A Linear Time Projection Algorithm

In this section we describe a more efficient algorithm for performing projections. To keep our presentation simple and easy to follow, we describe the projection algorithm onto the simplex. The generalization to the ℓ_1 ball can straightforwardly incorporated into the efficient algorithm by the results from the previous section (we simply work in the algorithm with a vector of the absolute values of $\mathbf{v}$, replacing the solution's components w_i with $\text{sign}(v_i) \cdot w_i$).

For correctness of the following discussion, we add another component to $\mathbf{v}$ (the vector to be projected), which we set to 0, thus $v_{n+1} = 0$ and $v_{(n+1)} = 0$. Let us start by examining again Lemma 2. The lemma implies that the index ρ is the largest integer that still satisfies $v_{(\rho)} - \frac{1}{\rho}\left(\sum_{r=1}^{\rho} v_{(r)} - z\right) > 0$. After routine algebraic manipulations the above can be rewritten in the following somewhat simpler form:

$$\sum_{i=1}^{\rho} \left(v_{(i)} - v_{(\rho)}\right) < z \ \text{ and } \ \sum_{i=1}^{\rho+1} \left(v_{(i)} - v_{(\rho+1)}\right) \geq z. \quad (9)$$

Given ρ and $v_{(\rho)}$ we slightly rewrite the value θ as follows,

$$\theta = \frac{1}{\rho}\left(\sum_{j:v_j \geq v_{(\rho)}} v_j - z\right) \ . \quad (10)$$

The task of projection can thus be distilled to the task of finding θ, which in turn reduces to the task of finding ρ and the pivot element $v_{(\rho)}$. Our problem thus resembles the task of finding an order statistic with an additional complicating factor stemming from the need to compute summations (while searching) of the form given by Eq. (9). Our efficient projection algorithm is based on a modification of the randomized median finding algorithm (Cormen et al., 2001). The algorithm computes partial sums just-in-time and has expected linear time complexity.

The algorithm identifies ρ and the pivot value $v_{(\rho)}$ without sorting the vector $\mathbf{v}$ by using a divide and conquer procedure. The procedure works in rounds and on each round either eliminates elements shown to be strictly smaller than $v_{(\rho)}$ or updates the partial sum leading to Eq. (9). To do so the algorithm maintains a set of unprocessed elements of $\mathbf{v}$. This set contains the components of $\mathbf{v}$ whose relationship to $v_{(\rho)}$ we do not know. We thus initially set $U = [n]$. On each round of the algorithm we pick at random an index k from the set U. Next, we partition the set U into two subsets G and L. G contains all the indices $j \in U$ whose components $v_j > v_k$; L contains those $j \in U$ such that v_j is smaller. We now face two cases related to the current summation of entries in $\mathbf{v}$ greater than the hypothesized $v_{(\rho)}$ (i.e. v_k). If $\sum_{j:v_j \geq v_k}(v_j - v_k) < z$ then by Eq. (9), $v_k \geq v_{(\rho)}$. In this case we know that all the elements in G participate in the sum defining θ as given by Eq. (9). We can discard G and set U to be L as we still need to further identify the remaining elements in L. If $\sum_{j:v_j \geq v_k}(v_j - v_k) \geq z$ then the same rationale implies that $v_k < v_{(\rho)}$. Thus, all the elements in L are smaller than $v_{(\rho)}$ and can be discarded. In this case we can remove the set L and v_k and set U to be $G \setminus \{k\}$. The entire process ends when U is empty.

Along the process we also keep track of the sum and the number of elements in $\mathbf{v}$ that we have found thus far to be no smaller than $v_{(\rho)}$, which is required in order not to recalculate partial sums. The pseudo-code describing the efficient projection algorithm is provided in Fig. 2. We keep the set of elements found to be greater than $v_{(\rho)}$ only *implicitly*. Formally, at each iteration of the algorithm we maintain a variable s, which is the sum of the elements in the set $\{v_j : j \notin U, v_j \geq v_{(\rho)}\}$, and overload ρ to designate the cardinality of the this set throughout the algorithm. Thus, when the algorithms exits its main while loop, ρ is the maximizer defined in Lemma 1. Once the while loop terminates, we are left with the task of calculating θ using Eq. (10) and performing the actual projection. Since $\sum_{j:v_j \geq \mu_\rho} v_j$ is readily available to us as the variable s, we simply set θ to be $(s - z)/\rho$ and perform the projection as prescribed by Eq. (6).

Though omitted here for lack of space, we can also extend the algorithms to handle the more general constraint that $\sum a_i|w_i| \leq z$ for $a_i \geq 0$.

6. Efficient Projection for Sparse Gradients

Before we dive into developing a new algorithm, we remind the reader of the iterations the minimization algorithm takes from Eq. (2): we generate a sequence $\{\mathbf{w}^{(t)}\}$

```
INPUT A balanced tree T and a scalar z > 0
INITIALIZE v* = ∞, ρ* = n + 1, s* = z
CALL PIVOTSEARCH(root(T), 0, 0)
PROCEDURE PIVOTSEARCH(v, ρ, s)
  COMPUTE  ρ̂ = ρ + r(v) ; ŝ = s + σ(v)
  IF ŝ < vρ̂ + z      // v ≥ pivot
    IF v* > v
      v* = v ; ρ* = ρ̂ ; s* = ŝ
    ENDIF
    IF  leaf_T(v)
      RETURN  θ = (s* − z)/ρ*
    ENDIF
    CALL PIVOTSEARCH(left_T(v), ρ̂, ŝ)
  ELSE                // v < pivot
    IF  leaf_T(v)
      RETURN  θ = (s* − z)/ρ*
    ENDIF
    CALL PIVOTSEARCH(right_T(v), ρ, s)
  ENDIF
ENDPROCEDURE
```

Figure 3. Efficient search of pivot value for sparse feature spaces.

by iterating

$$\mathbf{w}^{(t+1)} = \Pi_W \left(\mathbf{w}^{(t)} + \mathbf{g}^{(t)} \right)$$

where $\mathbf{g}^{(t)} = -\eta_t \nabla^{(t)}$, $W = \{\mathbf{w} \mid \|\mathbf{w}\|_1 \leq z\}$ and Π_W is projection onto this set.

In many applications the dimension of the feature space is very high yet the number of features which attain a non-zero value for each example is very small (see for instance our experiments on text documents in Sec. 7). It is straightforward to implement the gradient-related updates in time which is proportional to the number of non-zero features, but the time complexity of the projection algorithm described in the previous section is linear in the dimension. Therefore, using the algorithm verbatim could be prohibitively expensive in applications where the dimension is high yet the number of features which are "on" in each example is small. In this section we describe a projection algorithm that updates the vector $\mathbf{w}^{(t)}$ with $\mathbf{g}^{(t)}$ and scales linearly in the number of non-zero entries of $\mathbf{g}^{(t)}$ and only *logarithmically* in the total number of features (*i.e.* non-zeros in $\mathbf{w}^{(t)}$).

The first step in facilitating an efficient projection for sparse feature spaces is to represent the projected vector as a "raw" vector $\mathbf{v}$ by incorporating a global shift that is applied to each non-zero component. Specifically, each projection step amounts to deducting θ from each component of $\mathbf{v}$ and thresholding the result at zero. Let us denote by θ_t the shift value used on the t^{th} iteration of the algorithm and by Θ_t the cumulative sum of the shift values, $\Theta_t = \sum_{s \leq t} \theta_s$. The representation we employ enables us to perform the step in which we deduct θ_t from all the elements of the vector *implicitly*, adhering to the goal of performing a sub-linear number of operations. As before, we assume that the goal is to project onto the simplex. Equipped with these variables, the j^{th} component of the projected vector after t projected gradient steps can be written as $\max\{v_j - \Theta_t, 0\}$.

The second substantial modification to the core algorithm is to keep only the *non-zero* components of the weight vector in a red-black tree (Cormen et al., 2001). The red-black tree facilitates an efficient search for the pivot element ($v_{(\rho)}$) in time which is logarithmic in the dimension, as we describe in the sequel. Once the pivot element is found we implicitly deduct θ_t from all the non-zero elements in our weight vector by updating Θ_t. We then remove all the components that are less than $v_{(\rho)}$ (i.e. less than Θ_t); this removal is efficient and requires only logarithmic time (Tarjan, 1983).

The course of the algorithm is as follows. After t projected gradient iterations we have a vector $\mathbf{v}^{(t)}$ whose non-zero elements are stored in a red-black tree $\mathcal{T}$ and a global deduction value Θ_t which is applied to each non-zero component just-in-time, *i.e.* when needed. Therefore, each non-zero weight is accessed as $v_j - \Theta_t$ while $\mathcal{T}$ does not contain the zero elements of the vector. When updating $\mathbf{v}$ with a gradient, we modify the vector $\mathbf{v}^{(t)}$ by adding to it the gradient-based vector $\mathbf{g}^{(t)}$ with k non-zero components. This update is done using k deletions (removing v_i from $\mathcal{T}$ such that $g_i^{(t)} \neq 0$) followed by k re-insertions of $v'_i = (v_i + g_i^{(t)})$ into $\mathcal{T}$, which takes $O(k \log(n))$ time. Next we find in $O(\log(n))$ time the value of θ_t. Fig. 3 contains the algorithm for this step; it is explained in the sequel. The last step removes all elements of the new raw vector $\mathbf{v}^{(t)} + \mathbf{g}^{(t)}$ which become zero due to the projection. This step is discussed at the end of this section.

In contrast to standard tree-based search procedure, to find θ_t we need to find a pair of consecutive values in $\mathbf{v}$ that correspond to $v_{(\rho)}$ and $v_{(\rho+1)}$. We do so by keeping track of the smallest element that satisfies the left hand side of Eq. (9) while searching based on the condition given on the right hand side of the same equation. $\mathcal{T}$ is keyed on the values of the un-shifted vector $\mathbf{v}_t$. Thus, all the children in the left (right) sub-tree of a node v represent values in $\mathbf{v}_t$ which are smaller (larger) than v. In order to efficiently find θ_t we keep at each node the following information: (a) The value of the component, simply denoted as v. (b) The number of elements in the right sub-tree rooted at v, denoted $r(v)$, including the node v. (c) The sum of the elements in the right sub-tree rooted at v, denoted $\sigma(v)$, including the value v itself. Our goal is to identify the pivot element $v_{(\rho)}$ and its index ρ. In the previous section we described a simple condition for checking whether an element in $\mathbf{v}$ is greater or smaller than the pivot value. We now rewrite this expression yet one more time. A component with value v is *not*

smaller than the pivot iff the following holds:

$$\sum_{j:v_j \geq v} v_j > |\{j : v_j \geq v\}| \cdot v + z \; . \tag{11}$$

The variables in the red-black tree form the infrastructure for performing efficient recursive computation of Eq. (11). Note also that the condition expressed in Eq. (11) still holds when we do *not* deduct Θ_t from all the elements in $\mathbf{v}$.

The search algorithm maintains recursively the number ρ and the sum s of the elements that have been shown to be greater or equal to the pivot. We start the search with the root node of $\mathcal{T}$, and thus initially $\rho = 0$ and $s = 0$. Upon entering a new node v, the algorithm checks whether the condition given by Eq. (11) holds for v. Since ρ and s were computed for the parent of v, we need to incorporate the number and the sum of the elements that are larger than v itself. By construction, these variables are $r(v)$ and $\sigma(v)$, which we store at the node v itself. We let $\hat{\rho} = \rho + r(v)$ and $\hat{s} = s + \sigma(v)$, and with these variables handy, Eq. (11) distills to the expression $\hat{s} < v\hat{\rho} + z$. If the inequality holds, we know that v is either larger than the pivot or it may be the pivot itself. We thus update our current hypothesis for μ_ρ and ρ (designated as $v^\star$ and $\rho^\star$ in Fig. 3). We continue searching the left sub-tree ($\text{left}_{\mathcal{T}}(v)$) which includes all elements smaller than v. If inequality $\hat{s} < v\hat{\rho} + z$ does not hold, we know that $v < \mu_\rho$, and we thus search the right subtree ($\text{right}_{\mathcal{T}}(v)$) and keep ρ and s intact. The process naturally terminates once we reach a leaf, where we can also calculate the correct value of θ using Eq. (10).

Once we find θ_t (if $\theta_t \geq 0$) we update the global shift, $\Theta_{t+1} = \Theta_t + \theta_t$. We need to discard all the elements in $\mathcal{T}$ smaller than Θ_{t+1}, which we do using Tarjan's (1983) algorithm for splitting a red-black tree. This step is logarithmic in the total number of non-zero elements of $\mathbf{v}_t$. Thus, as the additional variables in the tree can be updated in constant time as a function of a node's child nodes in $\mathcal{T}$, each of the operations previously described can be performed in logarthmic time (Cormen et al., 2001), giving us a total update time of $O(k \log(n))$.

7. Experiments

We now present experimental results demonstrating the effectiveness of the projection algorithms. We first report results for experiments with synthetic data and then move to experiments with high dimensional natural datasets.

In our experiment with synthetic data, we compared variants of the projected subgradient algorithm (Eq. (2)) for ℓ_1-regularized least squares and ℓ_1-regularized logistic regression. We compared our methods to a specialized coordinate-descent solver for the least squares problem due to Friedman et al. (2007) and to very fast interior point

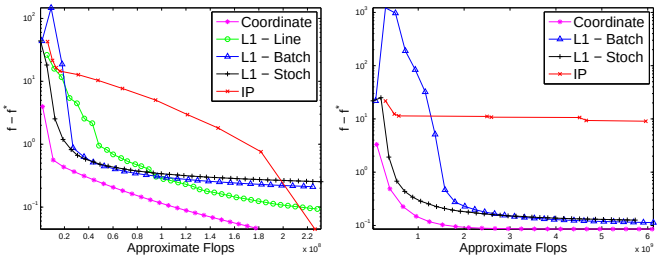


Figure 4. Comparison of methods on ℓ_1-regularized least squares. The left has dimension $n = 800$, the right $n = 4000$

methods for both least squares and logistic regression (Koh et al., 2007; Kim et al., 2007). The algorithms we use are batch projected gradient, stochastic projected subgradient, and batch projected gradient augmented with a backtracking line search (Koh et al., 2007). The IP and coordinate-wise methods both solve regularized loss functions of the form $f(\mathbf{w}) = L(\mathbf{w}) + \lambda\|\mathbf{w}\|_1$ rather than having an ℓ_1-domain constraint, so our objectives are not directly comparable. To surmount this difficulty, we first minimize $L(\mathbf{w}) + \lambda\|\mathbf{w}\|_1$ and use the 1-norm of the resulting solution $\mathbf{w}^*$ as the constraint for our methods.

To generate the data for the least squares problem setting, we chose a $\mathbf{w}$ with entries distributed normally with 0 mean and unit variance and randomly zeroed 50% of the vector. The data matrix $X \in \mathbb{R}^{m \times n}$ was random with entries also normally distributed. To generate target values for the least squares problem, we set $\mathbf{y} = X\mathbf{w} + \boldsymbol{\nu}$, where the components of $\boldsymbol{\nu}$ were also distributed normally at random. In the case of logistic regression, we generated data X and the vector $\mathbf{w}$ identically, but the targets y_i were set to be $\text{sign}(\mathbf{w} \cdot \mathbf{x}_i)$ with probability 90% and to $-\text{sign}(\mathbf{w} \cdot \mathbf{x}_i)$ otherwise. We ran two sets of experiments, one each for $n = 800$ and $n = 4000$. We also set the number of examples m to be equal to n. For the subgradient methods in these experiments and throughout the remainder, we set $\eta_t = \eta_0/\sqrt{t}$, choosing η_0 to give reasonable performance. (η_0 too large will mean that the initial steps of the gradient method are not descent directions; the noise will quickly disappear because the step sizes are proportional to $1/\sqrt{t}$).

Fig. 4 and Fig. 5 contain the results of these experiments and plot $f(\mathbf{w}) - f(\mathbf{w}^*)$ as a function of the number of floating point operations. From the figures, we see that the projected subgradient methods are generally very fast at the outset, getting us to an accuracy of $f(\mathbf{w}) - f(\mathbf{w}^*) \leq 10^{-2}$ quickly, but their rate of convergence slows over time. The fast projection algorithms we have developed, however, allow projected-subgradient methods to be very competitive with specialized methods, even on these relatively small problem sizes. On higher-dimension data sets interior point methods are infeasible or very slow. The rightmost graphs in Fig. 4 and Fig. 5 plot $f(\mathbf{w}) - f(\mathbf{w}^*)$ as functions of floating point operations for least squares and logistic re-

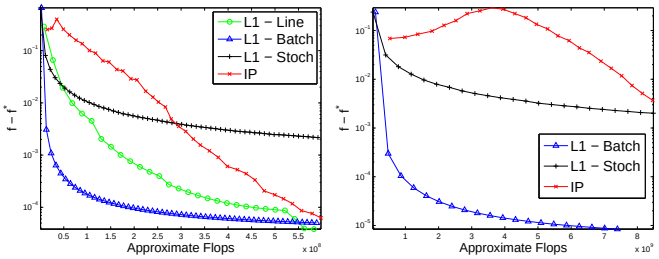


Figure 5. Comparison of methods on ℓ_1-regularized logistic regression. The left has dimension $n = 800$, the right $n = 4000$

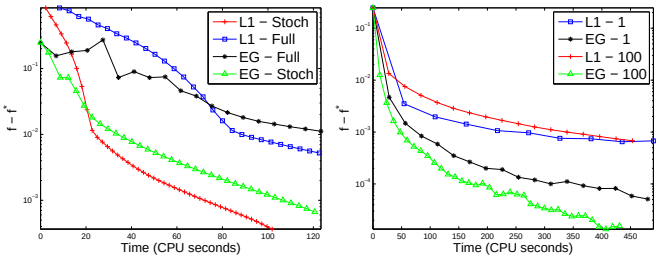


Figure 6. EG and projected subgradient methods on RCV1.

gression with dimension $n = 4000$. These results indicate that in high dimensional feature spaces, the asymptotically faster convergence of IP methods is counteracted by their quadratic dependence on the dimension of the space.

We also ran a series of experiments on two real datasets with high dimensionality: the Reuters RCV1 Corpus (Lewis et al., 2004) and the MNIST handwritten digits database. The Reuters Corpus has 804,414 examples; with simple stemming and stop-wording, there are 112,919 unigram features and 1,946,684 bigram features. With our preprocessing, the unigrams have a sparsity of 1.2% and the bigrams have sparsity of .26%. We performed ℓ_1-constrained binary logistic regression on the CCAT category from RCV1 (classifying a document as corporate/industrial) using unigrams in a batch setting and bigrams in an online setting. The MNIST dataset consists of 60,000 training examples and a 10,000 example test set and has 10-classes; each image is a gray-scale 28×28 image, which we represent as $\mathbf{x}_i \in \mathbb{R}^{784}$. Rather than directly use the input $\mathbf{x}_i$, however, we learned weights $\mathbf{w}_j$ using the following Kernel-based "similarity" function for each class $j \in \{1, \ldots, 10\}$:

$$k(\mathbf{x}, j) = \sum_{i \in S} w_{ji} \sigma_{ji} K(\mathbf{x}_i, \mathbf{x}), \; \sigma_{ji} = \begin{cases} 1 & \text{if } y_i = j \\ -1 & \text{otherwise.} \end{cases}$$

In the above, K is a Gaussian kernel function, so that $K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/25)$, and S is a 2766 element support set. We put an ℓ_1 constraint on each $\mathbf{w}_j$, giving us the following multiclass objective with dimension 27,660:

$$\begin{array}{ll} \text{minimize}_{\mathbf{W}} & \frac{1}{m} \sum_{i=1}^m \log \left(1 + \sum_{r \neq y_i} e^{k(\mathbf{x}_i, r) - k(\mathbf{x}_i, y_i)}\right) \\ \text{s.t.} & \|\mathbf{w}_j\|_1 \leq z, \mathbf{w}_j \succeq 0. \end{array} \tag{12}$$

As a comparison to our projected subgradient methods on real data, we used a method known in the literature as either entropic descent, a special case of mirror descent (Beck & Teboulle, 2003), or exponentiated gradient (EG) (Kivinen & Warmuth, 1997). EG maintains a weight vector $\mathbf{w}$ subject to the constraint that $\sum_i w_i = z$ and $\mathbf{w} \succeq 0$; it can easily be extended to work with negative weights under a 1-norm constraint by maintaining two vectors $\mathbf{w}^+$ and $\mathbf{w}^-$. We compare against EG since it works well in very high dimensional spaces, and it very quickly identifies and shrinks weights for irrelevant features (Kivinen & Warmuth, 1997). At every step of EG we update

$$w_i^{(t+1)} = \frac{w_i^{(t)} \exp\left(-\eta_t \nabla_i f(\mathbf{w}^{(t)})\right)}{Z_t} \tag{13}$$

where Z_t normalizes so that $\sum_i w_i^{(t+1)} = z$ and $\nabla_i f$ denotes the i^{th} entry of the gradient of f, the function to be minimized. EG can actually be viewed as a projected subgradient method using generalized relative entropy ($D(\mathbf{x}\|\mathbf{y}) = \sum_i x_i \log \frac{x_i}{y_i} - x_i + y_i$) as the distance function for projections (Beck & Teboulle, 2003). We can replace $\nabla_i f$ with $\hat{\nabla}_i f$ in Eq. (13), an unbiased estimator of the gradient of f, to get stochastic EG. A step size $\eta_t \propto 1/\sqrt{t}$ guarantees a convergence rate of $O(\sqrt{\log n / T})$. For each experiment with EG, however, we experimented with learning rates proportional to $1/t$, $1/\sqrt{t}$, and constant, as well as different initial step-sizes; to make EG as competitive as possible, we chose the step-size and rate for which EG performed best on each individual test..

Results for our batch experiments learning a logistic classifier for CCAT on the Reuters corpus can be seen in Fig. 6. The figure plots the binary logistic loss of the different algorithms minus the optimal log loss as a function of CPU time. On the left side Fig. 6, we used projected gradient descent and stochastic gradient descent using 25% of the training data to estimate the gradient, and we used the algorithm of Fig. 2 for the projection steps. We see that ℓ_1-projections outperform EG both in terms of convergence speed and empirical log-loss. On the right side of the figure, we performed stochastic descent using only 1 training example or 100 training examples to estimate the gradient, using Fig. 3 to project. When the gradient is sparse, updates for EG are $O(k)$ (where k is the number of non-zeros in the gradient), so EG has a run-time advantage over ℓ_1-projections when the gradient is very sparse. This advantage can be seen in the right side of Fig. 6.

For MNIST, with dense features, we ran a similar series of tests to those we ran on the Reuters Corpus. We plot the multiclass logistic loss from Eq. (12) over time (as a function of the number gradient evaluations) in Fig. 7. The left side of Fig. 7 compares EG and gradient descent using

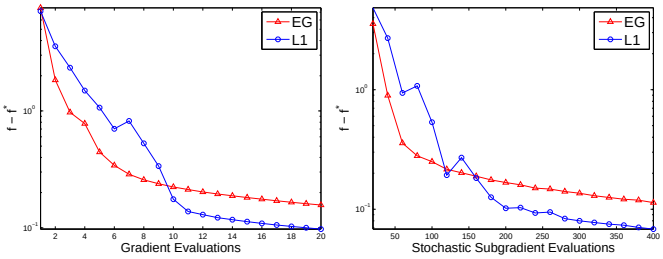


Figure 7. MNIST multiclass logistic loss as a function of the number of gradient steps. The left uses true gradients, the right stochastic subgradients.

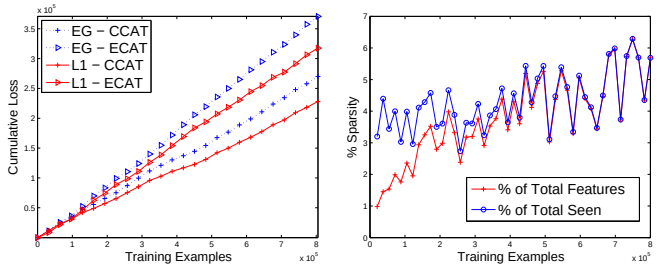


Figure 8. Online learning of bigram classifier on RCV1. Left is the cumulative loss, right shows sparsity over time.

the true gradient while the right figure compares stochastic EG and stochastic gradient descent using only 1% of the training set to estimate the gradient. On top of outperforming EG in terms of convergence rate and loss, the ℓ_1-projection methods also gave sparsity, zeroing out between 10 and 50% of the components of each class vector $\mathbf{w}_j$ in the MNIST experiments, while EG gives no sparsity.

As a last experiment, we ran an online learning test on the RCV1 dataset using bigram features, comparing ℓ_1-projections to using decreasing step sizes given by Zinkevich (2003) to exponential gradient updates. The ℓ_1-projections are computationally feasible because of algorithm 3, as the dimension of our feature space is nearly 2 million (using the expected linear-time algorithm of Fig. 2 takes 15 times as long to compute the projection for the sparse updates in online learning). We selected the bound on the 1-norm of the weights to give the best online regret of all our experiments (in our case, the bound was 100). The results of this experiment are in Fig. 8. The left figure plots the cumulative log-loss for the CCAT and ECAT binary prediction problems as a function of the number of training examples, while the right hand figure plots the sparsity of the ℓ_1-constrained weight vector both as a function of the dimension and as a function of the number of features actually seen. The ℓ_1-projecting learner maintained an active set with only about 5% non-zero components; the EG updates have no sparsity whatsoever. Our online ℓ_1-projections outperform EG updates in terms of the online regret (cumulative log-loss), and the ℓ_1-projection updates also achieve a classification error rate of 11.9% over all the examples on the CCAT task and 14.9% on ECAT (versus more than 15% and 20% respectively for EG).

Acknowledgments

We thank the anonymous reviewers for their helpful and insightful comments.

References

Beck, A., & Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, *31*, 167–175.

Bertsekas, D. (1999). *Nonlinear programming*. Athena Scientific.

Candes, E. J. (2006). Compressive sampling. *Proc. of the Int. Congress of Math., Madrid, Spain*.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms*. MIT Press.

Crammer, K., & Singer, Y. (2002). On the learnability and design of output codes for multiclass problems. *Machine Learning*, *47*.

Donoho, D. (2006a). Compressed sensing. *Technical Report, Stanford University*.

Donoho, D. (2006b). For most large underdetermined systems of linear equations, the minimal ℓ_1-norm solution is also the sparsest solution. *Comm. Pure Appl. Math. 59*.

Friedman, J., Hastie, T., & Tibshirani, R. (2007). Pathwise coordinate optimization. *Annals of Applied Statistics*, *1*, 302–332.

Gafni, E., & Bertsekas, D. P. (1984). Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, *22*, 936–964.

Hazan, E. (2006). Approximate convex optimization by online game playing. Unpublished manuscript.

Kim, S.-J., Koh, K., Lustig, M., Boyd, S., & Gorinevsky, D. (2007). An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, *4*, 606–617.

Kivinen, J., & Warmuth, M. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, *132*, 1–64.

Koh, K., Kim, S.-J., & Boyd, S. (2007). An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, *8*, 1519–1555.

Lewis, D., Yang, Y., Rose, T., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*, 361–397.

Ng, A. (2004). Feature selection, l_1 vs. l_2 regularization, and rotational invariance. *Proceedings of the Twenty-First International Conference on Machine Learning*.

Shalev-Shwartz, S., & Singer, Y. (2006). Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research, 7 (July)*, 1567–1599.

Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for SVM. *Proceedings of the 24th International Conference on Machine Learning*.

Tarjan, R. E. (1983). *Data structures and network algorithms*. Society for Industrial and Applied Mathematics.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, *58*, 267–288.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the Twentieth International Conference on Machine Learning*.

Pointwise Exact Bootstrap Distributions of Cost Curves

Charles Dugas DUGAS@DMS.UMONTREAL.CA
David Gadoury GADOURY@DMS.UMONTREAL.CA
Department of Mathematics and Statistic, Université de Montréal, Canada

Abstract

Cost curves have recently been introduced as an alternative or complement to ROC curves in order to visualize binary classifiers performance. Of importance to both cost and ROC curves is the computation of confidence intervals along with the curves themselves so that the reliability of a classifier's performance can be assessed. Computing confidence intervals for the difference in performance between two classifiers allows the determination of whether one classifier performs *significantly* better than another. A simple procedure to obtain confidence intervals for costs or the difference between two costs, under various *operating conditions*, is to perform bootstrap resampling of the test set. In this paper, we derive *exact* bootstrap distributions for these values and use these dstributions to obtain confidence intervals, under various operating conditions. Performances of these confidence intervals are measured in terms of coverage accuracies. Simulations show excellent results.

1. Introduction

A cost curve (Drummond & Holte, 2000; Drummond & Holte, 2006) is a plot of a classifier's expected cost as a function of operating conditions, i.e. misclassification costs and class probabilities. Performance assessment in terms of expected cost is paramount but cannot be visualized through ROC analysis although knowledge of the distribution of a classifier's total misclassification error cost is often among the enduser's interests.

Cost curve analysis can be enhanced if dispersion measures of the curve are provided along with the curve itself, thereby allowing the enduser to assess the reliability of the estimated performance of the classifier considered for implementation. In order to obtain confidence intervals from a single test set, resampling methods such as the bootstrap (Efron & Tibshirani, 1993) technique can be used: from the test set, a certain number of samples are drawn with remplacement and from these samples, a distribution of the cost can be obtained. In certain cases, the bootstrap technique lends itself to analytic derivations for the limit case where the number of samples tends to infinity. Distributions thus obtained are referred to as *exact bootstrap* distributions. The purpose of this paper is to derive exact bootstrap distributions for a classifier's total cost of misclassification errors as well as the difference between two classifiers' total costs, for varying operating conditions.

Except for Drummond & Holte (2006), little attention has been given to developing and evaluating the performance of confidence intervals for cost curves. ROC curves have received much more attention. Arguably, the recency of cost curves explains in part this situation. Recent literature on the derivations of confidence intervals for ROC curves can be segmented in three categories: parametric, semi-parametric or empirical. Semi-parametric methods mainly refer to kernel-based methods (Hall & Hyndman, 2003; Hall et al., 2004; Lloyds, 1998; Lloyds & Wong, 1999). Bootstrap resampling has been used for ROC curves as an empirical method but to date, exact bootstrap distributions for the ROC curve have not been presented.

A technical difficulty arises from the fact that, when sampling from the entire test set, a procedure we shall refer to as *full* sampling, relative proportions of classes will vary from one sample to another. Mathematical derivations of exact bootstrap distributions, in the context of full sampling, are thus more complicated. In this paper, we first use a procedure referred to as *stratified sampling* according to which proportions of positive and negative instances of each bootstrap sam-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

ple are fixed as equal to those of the original test set. Here, an *instance* is an element of the test set. Instances of the class for which the event has (not) taken place are called *positive* (*negative*). For example, for a credit card fraud detection application, fraudulous transactions would be labelled as positive whereas legitimate transactions would be labelled as negative. Within the stratified sampling framework, each sample is obtained from the combination of two independent bootstrap samples: one drawn from the set of positive instances and the other drawn from the set of negative instances. This procedure has previously been used in the context of ROC (Bandos, 2005) as well as cost curves (Drummond & Holte, 2006). After obtaining results under this simplified stratified sampling approach, we derive exact bootstrap distributions for the *full* sampling approach.

From the user's perspective, the two sampling procedures, stratified and full, provide different information so that the difference between the two approaches reaches beyond mere mathematical derivations. According to stratified sampling, the user is provided with a cost distribution conditional on the operating conditions that will eventually prevail once the model is implemented. We refer to these as the *deployment* conditions. This corresponds to the view of Drummond & Holte (2006) who argue in favor of plotting cost curves in terms of all possible values of the unknown future deployment conditions. Within the stratified sampling approach, cost dispersion measures obtained for a specific value of the deployment conditions make no provision for uncertainty around expected deployment conditions. On the other hand, according to the full sampling approach, class proportions are implicitly assumed to be binomially distributed around those of the test set so that cost dispersion measures incorporate uncertainty around class proportions. Since the two approaches provide different information that may both be of interest, both are treated in this paper.

The rest of the paper is as follows: in section 2, we briefly review the main aspects of ROC and cost curves. Then, mathematical derivations are presented in section 3 for stratified sampling and in section 4 for full sampling. In section 5, we perform simulations and measure coverage accuracies of the confidence intervals. Limitations of the proposed approach are discussed in section 6. Finally, we conclude in section 7.

2. ROC and cost curves

An ROC curve is a plot of the probability of correctly identifying a positive instance (a true positive) against the probability of mistakenly identifying a negative instance as positive (a false positive), for various threshold values. Fawcett (2004) provides an excellent introduction to ROC curves along with descriptions of the essential elements of ROC graph analysis. Classifier performance assessment in terms of expected total error cost cannot be done using ROC curves and for this reason (and others (Drummond & Holte, 2006)), cost curves have been introduced as an alternative (or a complement) to ROC curves.

The main objective of cost curves is to visualize classifier performance in terms of expected cost rather than through a tradeoff between misclassification error probabilities. Expected cost is plotted against operating conditions where, as mentionned above, operating conditions include two factors: class probabilities and misclassification costs. Once these values are fixed, all possibly attainable true and false positive rates pairs are considered. Given class probabilities, misclassification costs, and true and false positive rates, a cost is obtained. The pair that minimizes the cost is selected. It is assumed that given certain operating conditions, the enduser would select the cost minimizing pair and set the classifier's threshold accordingly. In order to obtain a cost curve, this optimization process is repeated for all possible operating conditions values. As shown below, a set of operating conditions can be summarized through a single normalized scalar value ranging between 0 and 1. Figure 1 illustrates this process.

Cost curves are obtained assuming the enduser selects the threshold that minimizes expected cost, given operating conditions, *based on the test set*. One approach to obtain cost distributions is to draw bootstrap samples from the test set, obtain a cost curve for each of the samples and derive a distribution for the cost from these cost curves. Now consider a specific set of values for the operating conditions. Each of the samples will lead a possibly different optimal threshold for this set of operating conditions. This can be viewed in Figure 1 by comparing the left- and right-hand columns.

Thus, averaging cost curves (fixed operating conditions but varying thresholds) in order to obtain an estimate of the expected cost would correspond to the enduser being able to select the optimal thresholds, depending on the actually observed sample of instances. In other words, the enduser would be required to have knowledge of the test set *before* deciding on a threshold value, something that can't be done in practice. Ob-

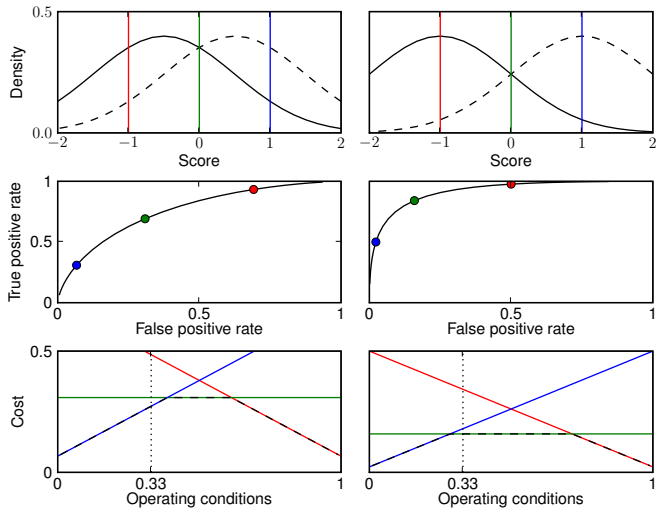


Figure 1. Derivation of ROC and cost curves for two classifiers with relatively low (left) and high (right) discrimination power. Top: score distributions for negative (solid) and positive (dashed) instances. Three colored vertical lines represent possible thresholds, from low (red) to high (blue). Middle: ROC curves associated to top row distributions. Three true and false positive pairs are identified with colored dots. Bottom: each dot of the middle row plotted in ROC space is uniquely associated to a line in cost curve space. Given specific operating conditions, the cost minimizing threshold may vary from one curve to another. Here, with $w = 0.33$, the optimal threshold is the highest (blue) of the three considered value on the left-hand side. On the right-hand side, it is the second largest threshold value (green) that leads to the lowest expected cost.

viously, thresholds must somehow be selected *prior* to test set cost measurements. This can be done through the standard machine learning process of splitting the data in three sets: training, validation and test. In our simulations, we assume the user chooses the optimal *theoretical* thresholds for all operating conditions, thus implicitely assuming an infinite sized validation set. The impact of this assumption is discussed later, in section 6. Our approach can therefore be considered as a form of threshold averaging of the costs. But since both operating conditions (abscissa values) and thresholds are fixed for each computed distribution, then the approach could be considered as vertical averaging as well. We now turn to more formal derivations of the cost curves and associated exact bootstrap distributions.

3. Stratified sampling

Consider a test set consisting of n instances from which stratified bootstrap samples are drawn. In this paper, we shall assume bootstrap samples are of the same size as the test set itself, a common procedure. Let n^+ and n^- be the numbers of positive and negative instances in the test set. According to the stratified bootstrap procedure and since we assume sample size equals test set size, the numbers of sampled positive and negative instances are fixed for all samples and also equal to n^+ and n^-, respectively. Let n_t^+ denote the number of instances, among the n^+ positive instances of the test set, with score greater or equal to the threshold $t = t(w)$ associated to operating conditions w, where w will be defined shortly. The corresponding value for a set of sampled positive instances is noted N_t^+ and follows binomial distribution with parameters n_t^+/n^+ and n^+ which we note as $N_t^+ \sim \text{Bin}(n_t^+/n^+, n^+)$. The random variable for the true positive rate, at threshold t, is denoted $TP_t^+ = N_t^+/n^+$. Similarly for negative instances, n_t^- refers to the number of instances with score greater or equal to t among the n^- negative instances of the test set, N_t^- is the random variable for the corresponding number of sampled instances and $FP_t^- = N_t^-/n^-$ is the false positive rate, at threshold t, with $N_t^- \sim \text{Bin}(n_t^-/n^-, n^-)$. Note that, according to the stratified sampling procedure, samples from positive and negative instances are drawn independently so that TP_t^+ and FP_t^- are independent as well.

Let us now formalize the above mentionned operating conditions and define w. Let $p_+ = n^+/n$ and $p_- = n^-/n$ represent class probabilities for positive and negative instances, respectively. Misclassification costs are noted $c_{+/-}$ and $c_{-/+}$ for false positive and false negative errors, respectively. Total cost is therefore given by the following:

$$C_t^T = p_+ c_{-/+}(1 - TP_t^+) + p_- c_{+/-} FP_t^- . \quad (1)$$

Drummond & Holte (2006) divide the total cost by its maximum possible value, in order to obtain a normalized cost with maximum value of one. This maximum total cost value is reached when $1 - TP_t^+ = FP_t^- = 1$ and the total cost is then equal to $p_+ c_{-/+} + p_- c_{+/-}$. Defining w as

$$w = \frac{p_+ \cdot c_{-/+}}{p_+ \cdot c_{-/+} + p_- \cdot c_{+/-}}, \quad (2)$$

the normalized cost is given by

$$C_t^N = w(1 - TP_t^+) + (1 - w) FP_t^- \quad (3)$$

with $w \in [0, 1]$. As mentionned above, true and false positive rates are independent when stratified sampling is used. Thus, the expected value and variance

of C_t^N follow as:

$$E[C_t^N] = w(1-n_t^+/n^+)+(1-w)n_t^-/n^-. \quad (4)$$

$$V[C_t^N] = w^2 n_t^+/n^+(1-n_t^+/n^+) + (1-w)^2 n_t^-/n^-(1-n_t^-/n^-). \quad (5)$$

We use these expectation and variance of the distribution of C_t^N to fit a gaussian distribution from which confidence intervals are easily obtained.

Now, in order to assess the statistical significance of the difference in performance of two classifiers, we need to obtain the distribution of the difference in their normalized costs:

$$\Delta C^N_{t_1,t_2} = C^N_{t_2} - C^N_{t_1} = w(TP^+_{t_1} - TP^+_{t_2}) + (1-w)(FP^-_{t_2} - FP^-_{t_1}) \quad (6)$$

where we use subscripts 1 and 2 to differentiate values obtained for the two classifiers. The values of $C^N_{t_2}$ and $C^N_{t_1}$ cannot be assumed independent since it is possible that the scores assigned by two different classifiers are correlated: for example, obvious fraudulous transactions will likely obtain high scores on all classifiers. Also note that only instances that are falsely labelled by one and only one of the two classifiers will affect the difference in costs. Errors made by both classifiers will offset each other when computing cost differences. Let $n^+_{t_1}$ represent the number of positive test set instances labelled as positive by the first classifier and negative by the second classifier, given operating conditions w. Similarly, let $n^+_{t_2}$ represent the number of positive test set instances labelled as positive by classifier 2 and negative by classifier 1. Note that thresholds $t_1 = t_1(w)$ and $t_2 = t_2(w)$ associated to operating conditions w may differ from one classifier to the other since score distributions and scales may vary from one classifier to the other. Values $n^-_{t_1}$ and $n^-_{t_2}$ are defined similarly for negative instances, given the same operating conditions value w.

Let $N^+_{t_1}$, $N^+_{t_2}$, $N^-_{t_1}$, and $N^-_{t_2}$ be the associated random variables for the number of instances in a bootstrap sample. Values $N^+_{t_1}$ and $N^+_{t_2}$ jointly follow a multinomial distribution. This also applies to $N^-_{t_1}$ and $N^-_{t_2}$. Accordingly, moments of $\Delta C^N_{t_1,t_2}$ are easily obtained:

$$E[\Delta C^N_{t_1,t_2}] = w\left(\frac{n^+_{t_1}-n^+_{t_2}}{n^+}\right) + (1-w)\left(\frac{n^-_{t_2}-n^-_{t_1}}{n^-}\right) \quad (7)$$

$$V[\Delta C^N_{t_1,t_2}] = w^2\left(\frac{n^+_{t_1}+n^+_{t_2}-\frac{(n^+_{t_1}-n^+_{t_2})^2}{n^+}}{(n^+)^2}\right) + (1-w)^2\left(\frac{n^-_{t_1}+n^-_{t_2}-\frac{(n^-_{t_1}-n^-_{t_2})^2}{n^-}}{(n^-)^2}\right). \quad (8)$$

Let us now evaluate the computational time required to obtain confidence intervals for the performance of a single classifier and for the difference between the performances of two classifiers. Here, we assume the number of different operating conditions considered, i.e. the number of different values for w is proportional to n. Also, as explained above, we assume the thresholds associated to each of these operating conditions have previously been determined through a validation process. For the case of a single classifier performance, we first need to sort instances with respect to their score, which requires time $O(n \ln n)$. Then, values of n_t^+ and n_t^- are easily obtained in linear time. There remains to compute expectations and variances, using equations (4) and (5), and derive confidence intervals using these values. This is realized in constant time for each value of w, thus overall linear time. Globally, the entire process is therefore dominated by the sorting phase and total computational time is $O(n \ln n)$. Confidence intervals for the difference in performance between two classifiers can be obtained in $O(n \ln n)$ computational time as well, although less trivially. Naive solutions lead to quadratic time but, given careful sorting preprocessing, values $n^+_{t_1}, n^+_{t_2}, n^-_{t_1}$, and $n^-_{t_2}$ are computed in linear time. Then, moments and confidence intervals for $\Delta C^N_{t_1,t_2}$ are obtained in linear time (for all values of w) using equations (7) and (8).

4. Full sampling

Within the framework of full sampling, the proportions of positive and negative instances vary from one sample to another. Whereas with stratified sampling, the number of positive and negative instances in each sample, n^+ and n^-, were set as equal to those of the test set, we now consider these numbers as random variables, and accordingly use capital notation N^+ and N^-. Here again, these values follow binomial distributions: $N^+ \sim \text{Bin}(n^+/n, n)$. Thus, full sampling implicitly assumes a binomial distribution for the observed class proportions $P_+ = N^+/n$ and $P_- = N^-/n$ but this distribution could easily be replaced.

Equation (1) still holds in the case of full sampling, but with the difference that P_+ and P_- are now treated as random variables. In the previous section the normalized version of the total cost was obtained by dividing the total cost by the largest possible cost: $p_+c_{-/+}+p_-c_{+/-}$, a weighted average between misclassification costs $c_{-/+}$ and $c_{+/-}$. Since P_+ and P_- are

no longer fixed, we must consider the largest possible weighted average which simply is the maximum of the two misclassification costs, $c_{\max} = \max[c_{-/+}, c_{+/-}]$. The case where $C_t^T = c_{-/+}$ is obtained when $N^+ = n$ and $TP_t^+ = 0$. Similarly, we have $C_t^T = c_{+/-}$ if $N^- = n$ and $FP_t^- = 1$. Thus, for full sampling, the normalized cost can be written as

$$C_t^N = \frac{N^+ \cdot c_{-/+} \cdot (1 - TP_t^+) + N^- \cdot c_{+/-} \cdot FP_t^-}{n \cdot c_{\max}}.$$

Then, expected normalized cost and normalized cost variance are obtained through iterated expectations:

$$\begin{aligned} E[C_t^N] &= E_{N^+}\{E[C_t^N|N^+]\} \\ &= \frac{c_{-/+}(n^+ - n_t^+) + c_{+/-} \cdot n_t^-}{n \cdot c_{\max}} \end{aligned} \quad (9)$$

$$\begin{aligned} V[C_t^N] &= V_{N^+}\{E[C_t^N|N^+]\} + E_{N^+}\{V[C_t^N|N^+]\} \\ &= \frac{c_{-/+}^2 \alpha_t^+ + c_{+/-}^2 \alpha_t^- + \delta_t^2}{(n \cdot c_{\max})^2} \end{aligned} \quad (10)$$

where

$$\begin{aligned} \alpha_t^+ &= n_t^+ - \frac{(n_t^+)^2}{n^+} \\ \alpha_t^- &= n_t^- - \frac{(n_t^-)^2}{n^-} \\ \delta_t^2 &= \left(c_{-/+} \frac{n^+ - n_t^+}{n^+} - c_{+/-} \frac{n_t^-}{n^-} \right)^2 \frac{n^+ \cdot n^-}{n} \end{aligned}$$

Here again, equations (9) and (10) can be used to obtain a fitted gaussian distribution from which confidence intervals are easily derived.

Let us now turn to the difference in performance between two classifiers. In the case of full sampling, this difference is

$$\Delta C_{t_1,t_2}^N = \frac{c_{-/+}(N_{t_1}^+ - N_{t_2}^+) + c_{+/-}(N_{t_2}^- - N_{t_1}^-)}{n \cdot c_{\max}} \quad (11)$$

Again, expected normalized cost and normalized cost variance are obtained through iterated expectations:

$$\begin{aligned} E[\Delta C_{t_1,t_2}^N] &= E_{N^+}\{E[\Delta C_{t_1,t_2}^N|N^+]\} \\ &= \frac{c_{-/+}(n_{t_1}^+ - n_{t_2}^+) + c_{+/-} \cdot (n_{t_2}^- - n_{t_1}^-)}{n \cdot c_{\max}} \end{aligned} \quad (12)$$

$$\begin{aligned} V[\Delta C_{t_1,t_2}^N] &= V_{N^+}\{E[\Delta C_{t_1,t_2}^N|N^+]\} \\ &\quad + E_{N^+}\{V[\Delta C_{t_1,t_2}^N|N^+]\} \\ &= \frac{c_{-/+}^2 \alpha_{t_1,t_2}^+ + c_{+/-}^2 \alpha_{t_1,t_2}^- + \delta_{t_1,t_2}^2}{(n \cdot c_{\max})^2} \end{aligned} \quad (13)$$

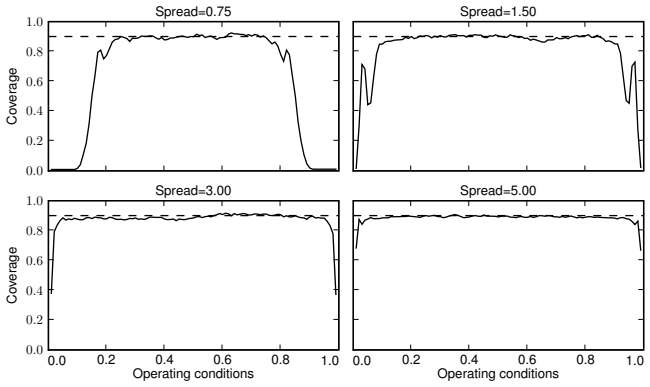


Figure 2. Effect of spread between distributions on coverage. Stratified sampling is used. Confidence intervals are derived for a classifier's cost. Location (spread) parameter for positive instances is set equal to 0.75 (up and left), 1.50 (up and right), 3.00 (down and left), and 5.00 (down and right). Sample size is 1,000. Confidence intervals are built with significance level $\alpha = 10\%$. Coverage proportion (solid) for 1,000 simulations and target coverage of 90% (dashed) are plotted against operating conditions.

where

$$\begin{aligned} \alpha_{t_1,t_2}^+ &= n_{t_1}^+ + n_{t_2}^+ - \frac{(n_{t_1}^+ - n_{t_2}^+)^2}{n^+} \\ \alpha_{t_1,t_2}^- &= n_{t_1}^- + n_{t_2}^- - \frac{(n_{t_1}^- - n_{t_2}^-)^2}{n^-} \\ \delta_{t_1,t_2}^2 &= \left(c_{-/+} \frac{n_{t_1}^+ - n_{t_2}^+}{n^+} - c_{+/-} \frac{n_{t_2}^- - n_{t_1}^-}{n^-} \right)^2 \frac{n^+ \cdot n^-}{n} \end{aligned}$$

This completes mathematical derivations. A total of four distributions have been obtained. For all four distributions, computation of confidence intervals is dominated by the need to sort instances so that computational time is $O(n \ln n)$ in all cases. Note that such time efficiency is obtained because we rely on the gaussian fitting of the variables' distributions. Computing true exact bootstrap distributions would lead to higher computational time orders. But as we show in the next section, results obtained with gaussian fitting are already excellent.

5. Numerical results

In this section, we conduct a series of experiments in order to assess the performance of the confidence intervals derived in sections 3 and 4. Performance is measured in terms of coverage accuracy of confidence intervals.

The first experiment is based on the framework used

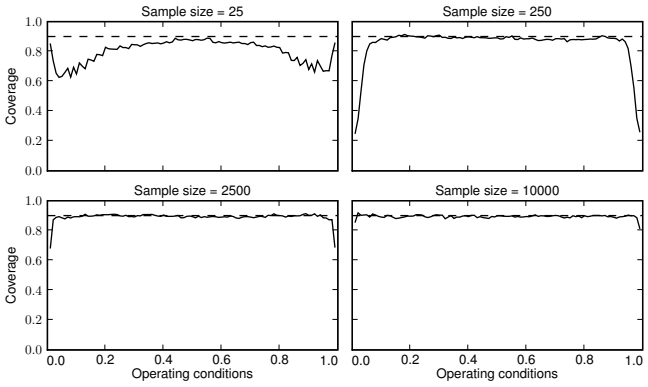


Figure 3. Effect of sample size on coverage. Stratified sampling is used. Confidence intervals are derived for a classifier's cost. Sample sizes of 25 (up and left), 250 (up and right), 2,500 (down and left), and 1,000, (down and right) are considered. Confidence intervals are built for significance level $\alpha = 10\%$. Location parameter for positive instances is set to $\theta = 3.0$. Coverage proportion (solid) for 1,000 simulations and target coverage of 90% (dashed) are plotted against operating conditions.

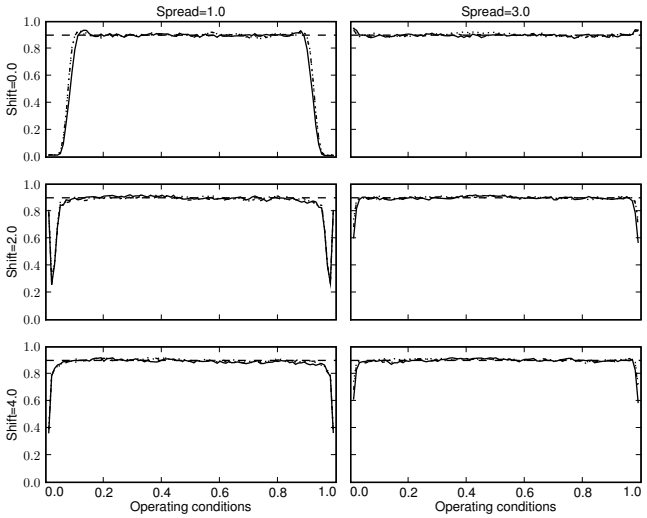


Figure 4. Coverage accuracy of confidence intervals for the difference in performance between two classifiers. Stratified sampling is used. Sample size is 1000, and significance level is $\alpha = 10\%$. Location parameter for positive instances of first classifier is set to $\theta = 1.0$ (left) and $\theta = 3.0$ (right). Location parameter for the score of positive instances according to the second classifier is θ (top), $\theta + 2.00$ (middle) and $\theta + 4.00$ (bottom). Within each plot, correlation factor is equal to 0.3 (dotted), 0.6 (dash-dotted) and 0.9 (solid). Coverage proportions for 1,000 simulations and target coverage of 90% (dashed) are plotted against operating conditions.

by Macskassy et al. (2005) in which four methods for obtaining pointwise confidence intervals for ROC curves are compared: threshold averaging, vertical averaging, kernel smoothing (Hall et al., 2004) and Working-Hotelling bounds. Positive and negative instance scores follow normal distributions but with various parameter values. We set the scale parameter to 3.00 for both positive and negative instances scores. The location parameter θ for positive instances varies within the set $\{0.75, 1.5, 3.0, 5.0\}$ and the location parameter for negative instances is set equal to $-\theta$. Sample size is set to 1,000, i.e. a set of 1,000 instances is drawn from the positive instances distribution and another set of 1,000 negative instances is drawn from the negative instances distribution. The sampling procedure is repeated 1,000 times, i.e. 1,000 simulations are performed for each value of θ. We shall refer to this experiment as the *spread* experiment. Confidence intervals are obtained for a significance level of 10%.

Figure 2 provides simulation results which clearly show that better results are obtained when score distributions of positive and negative instances have few overlap, i.e. for high values of θ. Breaks in coverage accuracy appear as w is close to 0 or 1. This recurring pattern is discussed in section 6.

As a second experiment, we consider the effect of sample size on coverage accuracy. This experiment is everywhere similar to the previous one except for two modifications: (1) the location parameter not longer varies: it is set to $\theta = 3.0$ and (2) the sample size takes values in $\{25; 250; 2,500; 10,000\}$ instead of being fixed at 1,000. We shall refer to this experiment as the *size* experiment. Simulation results appear in Figure 3. As the sample size increases, the range of operating condition values with good coverage accuracy widens. For sample sizes of 25, only a very narrow range of operating condition values lead to a coverage rate that is on target.

Our third experiment addresses the modeling of the difference in performance between two classifiers. The experiment design is similar to the ones used for the previous two experiments, i.e. the spread and size experiments. Scores are distributed according to a binormal distribution with scale of 3.00. Confidence intervals are obtained for a significance level of $\alpha = 10\%$. The location parameters are set as follows: for positive instances of the first classifier, we consider two values: $\theta \in \{1.0, 3.0\}$. For negative instances of both classifiers the parameter is set equal to $-\theta$. Finally, for positive instances of the second classifier we consider three values: θ, $\theta + 2.0$ and $\theta + 4.0$. The difference between the location parameters of the two classifiers'

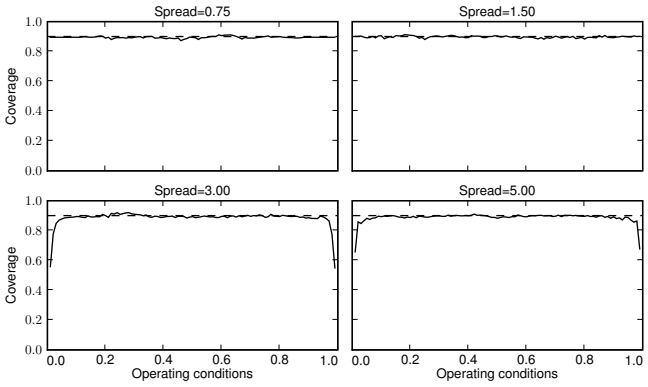


Figure 5. Effect of spread between distributions on coverage. Full sampling is used. Confidence intervals are derived for a classifier's cost. Location parameter for positive instances is set equal to 0.75 (up and left), 1.50 (up and right), 3.00 (down and left), and 5.00 (down and right). Sample size is 1,000. Confidence intervals are built with significance level $\alpha = 10\%$. Coverage proportion (solid) for 1,000 simulations and target coverage of 90% (dashed) are plotted against operating conditions.

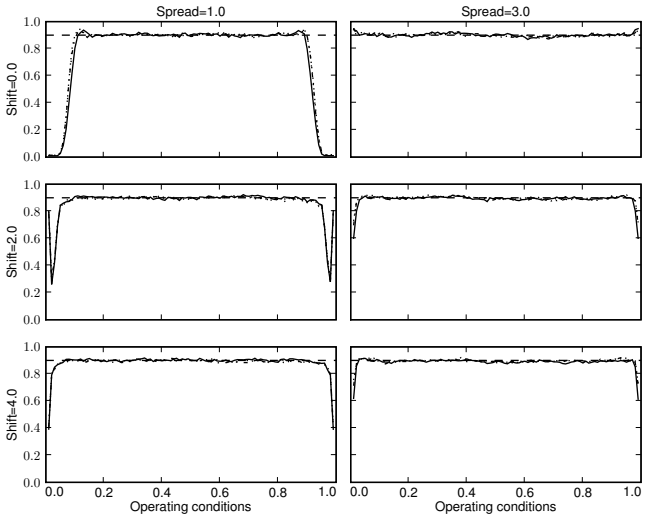


Figure 6. Coverage accuracy of confidence intervals for the difference in performance between two classifiers. Full sampling is used. Sample size is 1000, and significance level is $\alpha = 10\%$. Location parameter for positive instances of first classifier is set to $\theta = 1.0$ (left) and $\theta = 3.0$ (right). Location parameter for the score of positive instances according to the second classifier is θ (top), $\theta + 2.00$ (middle) and $\theta + 4.00$ (bottom). Within each plot, correlation factor is equal to 0.3 (dotted), 0.6 (dash-dotted) and 0.9 (solid). Coverage proportions for 1,000 simulations and target coverage of 90% (dashed) are plotted against operating conditions.

positive instances distributions, either 0.0, 2.0 or 4.0, is referred to as the *shift* parameter. In order to include some form of dependency between the scores of the two classifiers, three values of a correlation factor are considered: $\rho \in \{0.3, 0.6, 0.9\}$. We shall refer to this experiment as the *difference* experiment.

Results appear in Figure 4. As in the previous two experiments, coverage accuracy breaks for very low or high total positive rates. Comparing curves on the left of Figure 4 with those on the right, we see the spread parameter θ has some impact: higher values of θ cause the range of total positive rate values with good coverage accuracy to widen. With $\theta = 1.0$, higher shift parameter values lead to better coverage accuracy whereas with $\theta = 3.0$, the shift parameter has the opposite, but less pronounced, effect. The correlation coefficient seems to have very little effect on coverage accuracy which is a welcome property: the performances of the confidence intervals seem independent of the level of correlation between the scores of two models.

Figure 5 and 6 repeat the spread (first) and diffrence (third) experiments described above, but with the use of full sampling. Looking at Figure 5, it it clear that full sampling leads to better coverage accuracy than stratified sampling for low values of the spread parameter ($\theta = 0.75$). In fact, the effect of the spread parameter seems to have reversed although performance at $\theta = 5.00$ is better than with $\theta = 3.0$. Finally, Figure 6 indicates that both stratified and full sampling perform equally well for modeling the difference between two classifiers' performances.

6. Limitations of the approach

A first consideration is whether actually performing a certain number of bootstrap resamplings of the test set instances would allow us to reach coverage accuracy similar to that obtained in the previous experiments, using exact bootstrap distributions. Let b be the number of empirical bootstrap samples drawn. Computational time is dominated by the need to sort instances, as a preprocessing, for each sample and is thus within $O(b\,n \ln n)$. Obtaining confidence intervals through empirical bootstrap is therefore both an order of magnitude slower and less precise than using the exact bootstrap approach. Obviously, coverage accuracies similar to those presented here could be obtained with a large number of resamples but at high computational cost.

Another issue is the presence of breaks in coverage accuracy for extreme values of operating conditions.

When considering operating conditions close to 0 or 1, optimal thresholds are likely to lie outside the range of observed scores of the simulated test sets. For such thresholds and simulations, variances are either zero (equations 5,8, and 13) in which case coverage is impossible or very close to zero (equation 10) in which case coverage is very unlikely. Coverage accuracy breaks appear as the probability that the optimal threshold is outside the range of observed score values rises. Also, as is apparent from Figure 1, the expected value of the cost (thus the cost difference as well) drops to zero as operating conditions reach extreme values.

Finally, we may wonder how the assumption of optimal threshold selection impacts the results presented in this paper. Instead of assuming optimal threshold selection, consider selecting the thresholds, for each simulation of the previous experiments, based on a randomly generated finite-sized validation set which leading to suboptimal thresholds. Of course, expected costs are, by definition, higher for suboptimal thresholds than for optimal thresholds but what is of interest here is whether we can develop reliable confidence intervals for the cost, at the chosen thresholds, whether optimal or not. Given certain operating conditions, the selected suboptimal threshold follows a distribution centered around the optimal value so that coverage accuracy, given these operating conditions, is the expected coverage accuracy where the expectation is taken over the distribution of the suboptimal threshold. This results in a smoothing of the coverage accuracy breaks observed in the experiments above.

7. Conclusion

In this paper, we have derived exact bootstrap distributions for the (normalized) cost of the misclassification errors of a classifier's decisions. We have also derived exact bootstrap distributions for the difference between the costs of two classifiers. The first and second moments of these distributions have been used to fit gaussian distributions and thus approximate the true exact bootstrap distributions. From these approximated distributions, we were able to obtain confidence intervals for the variables of interest. Table 1 summarizes these results. All confidence intervals can be derived in $O(n \ln n)$ time.

Results obtained in this paper are excellent but limited to a few simulations. In a few cases, severe breaks in coverage accuracy appear when operating conditions values close to 0 or 1. These breaks can be avoided if cost distribution computations are limited to thresholds within the range of sampled scores. Another possibility is to extrapolate score distributions beyond observed values, an area for future work.

Sampling	Variable	Equations	Figures
Stratified	C_t^N	(4), (5)	2,3
	$\Delta C_{t_1,t_2}^N$	(7), (8)	4
Full	C_t^N	(9), (10)	5
	$\Delta C_{t_1,t_2}^N$	(12), (13)	6

Table 1. Summary of the paper's main results.

References

Bandos, A. (2005). *Nonparametric methods in comparing two correlated ROC curves.* Doctoral dissertation, Graduate School of Public Health, University of Pittsburgh.

Drummond, C., & Holte, R. (2000). Explicitly representing expected cost: an alternative to ROC representation. *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 198–207). ACM.

Drummond, C., & Holte, R. (2006). Cost curves: an improved method for visualizing classifier performance. *Machine Learning*, *65*, 95–130.

Efron, B., & Tibshirani, R. (1993). *An introduction to the bootstrap.* No. 57 in Monographs on Statistics and Probability. Chapman & Hall.

Fawcett, T. (2004). *ROC graphs: Notes and practical considerations for researchers* (Technical Report). HP Laboratories.

Hall, P., & Hyndman, R. (2003). Improved methods for bandwidth selection when estimating ROC curves. *Statistics & Probability Letters*, *64*, 181–189.

Hall, P., Hyndman, R., & Fan, Y. (2004). Nonparametric confidence intervals for receiver operating characteristic curves. *Biometrika*, *91*, 743–750.

Lloyds, C. (1998). The use of smoothed ROC curves to summarise and compare diagnostic systems. *Journal of the American Statistical Association*, *93*, 1356–1364.

Lloyds, C., & Wong, Z. (1999). Kernel estimators of the ROC curve are better than empirical. *Statistics & Probability Letters*, *44*, 221–228.

Macskassy, S., Provost, F., & Rosset, S. (2005). Pointwise ROC confidence bounds: An empirical evaluation. *Proceedings of the ICML 2005 workshop on ROC Analysis in Machine Learning.* Bonn, Germany.

Polyhedral Classifier for Target Detection A Case Study: Colorectal Cancer

M. Murat Dundar MURAT.DUNDAR@SIEMENS.COM
Matthias Wolf MWOLF@SIEMENS.COM
Sarang Lakare SARANG.LAKARE@SIEMENS.COM
Marcos Salganicoff MARCOS.SALGANICOFF@SIEMENS.COM
Vikas C. Raykar VIKAS.RAYKAR@SIEMENS.COM
CAD & Knowledge Solutions, Siemens Medical Solutions Inc., Malvern, PA 19355 USA

Abstract

In this study we introduce a novel algorithm for learning a polyhedron to describe the target class. The proposed approach takes advantage of the limited subclass information made available for the negative samples and jointly optimizes multiple hyperplane classifiers each of which is designed to classify positive samples from a subclass of the negative samples. The flat faces of the polyhedron provides robustness whereas multiple faces contributes to the flexibility required to deal with complex datasets. Apart from improving the prediction accuracy of the system, the proposed polyhedral classifier also provides run-time speedups as a by-product when executed in a cascaded framework in real-time. We evaluate the performance of the proposed technique on a real-world Colon dataset both in terms of prediction accuracy and online execution speed.

1. Problem Specification

In target detection the objective is to determine whether or not a given example is from a target class. Obtaining ground truth for the target class usually involves a tedious process of manual labeling. If samples belonging to the target class are labeled as positive, then negative class covers everything else. Due to the nature of the problem and the labeling process, the number of samples representing the target class is usually scarce whereas abundant data is potentially available to represent the negative class. In other words the data is highly unbalanced between classes favoring the negative class.

In this process the actual labels of the counter-examples are ignored and the negative class is formed by pooling samples of potentially different characteristics together within a single class. In other words samples of the negative class do not cluster well since they can belong to different subclasses.

One promising approach that has been heavily explored in this domain is the one-class classifiers. One-class classification simply omits the negative class (if it exists) and aims to learn a model with the positive examples only. Several techniques have been proposed in this direction. Support vector domain description technique aims to fit a tight hyper-sphere in the feature space to include most of the positive training samples and reject outliers (Tax & Duin, 1999). In this approach the nonlinearity of the data can be addressed implicitly through the kernel evaluation of the technique. One-class SVM generates an artificial point through kernel transformation for representing the negative class and then using relaxation parameters it aims to separate the image of the one-class from the origin (Scholkopf et al., 1999). Compression Neural Network constructs a three-layer feed-forward neural network and trains this network with a standard back-propogation algorithm to learn the identity function on the positive examples (Manevitz & Yousef, 2001).

Discriminative techiques such as Support Vector Machines (Vapnik, 1995), Kernel Fisher Discriminant (Mika et al., 2000), Relevance Vector Machines (Tipping, 2000) to name few are also used in this domain. These techniques deal with the unbalanced nature of the data by assigning different cost factors to the negative and positive samples in the objective function. The kernel evaluation of these techniques yields nonlinear decision boundaries suitable for classifying multi-mode data from the target class.

In this study we aim to learn a polyhedron in the feature

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

space to describe the positive training samples. Polyhedral decision boundaries such as boundaries that are drawn parallel to the axes of the feature space as in decision trees or skewed decision boundaries (Murth et al., 1994) have existed for quite some time. Our approach is similar in some sense to the Support vector domain description technique but there are two major differences. First instead of a hypersphere, a polyhedron is used to fit positive training samples. Second, positive and negative samples are used together in this process. The target polyhedron is learned through joint optimization of multiple hyperplane classifiers, each of which is designed to classify positive samples from a subgroup of negative samples. The number of such hyperplane classifiers is equivalent to the number of subclasses identified in the negative class. The proposed technique requires labeling of a small portion of the negative samples to collect training data for the subclasses that exist in the negative class.

Our approach does not intend to precisely identify each and every subclass in the dataset. By manual labeling we aim to identify major subclasses. Subclasses with similar characteristics or with only few labeled samples can be grouped together. During annotation one may also encounter positive look alikes, i.e. samples do not appear as negative but not yet confirmed as positive. A new subclass can be introduced for these samples.

In Figure 1 the proposed algorithm is demonstrated with a toy example. Positive samples are depicted by the dark circles in the middle, whereas negative samples are depicted with the numbers with each number corresponding to a different subclass. All eight classifiers are optimized simultaneously and polygon shown with dark lines is obtained as a decision boundary that classifies positive samples from the negative ones.

Kernel-based classifiers have the capacity to learn higly nonlinear decision boundaries allowing great flexibility. However it is well-known that in real-world applications where feature noise and redundancy is a problem, too much capacity usually hurts the generalizability of a classifier by enabling the classifier to easily overfit the training data. The proposed approach is capable of addressing nonlinearities by fitting the positive class through a series of linear hyperplanes, all of which are optimized jointly to form a polyhedral decision boundary. The flat faces provides robustness whereas multiple faces contributes to the flexibility.

The problem described above is commonly encountered in areas like content-base image retrieval (Chen et al., 2001), document classification (Manevitz & Yousef, 2001) and speech recognition (Brew et al., 2007). A similar scheme is also observed in Computer Aided Detection (CAD). In this study we explore the proposed idea for a CAD application, namely Colon CAD.

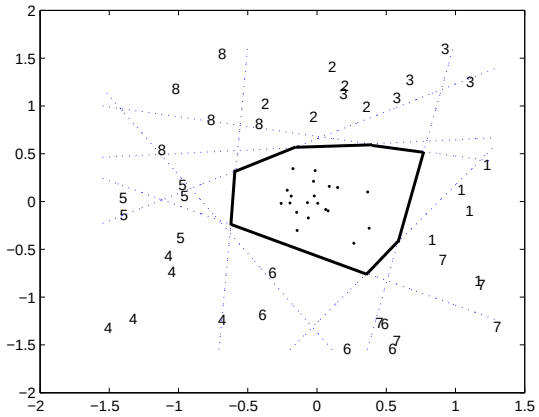


Figure 1. A Toy example demonstrating the proposed algorithm. Dark circles depicting positive samples, numbers representing negative samples. The decision boundary is shown with the solid lines.

2. Hyperplane Classifiers

We are given a training dataset $\{(x_i, y_i)\}_{i=1}^{\ell}$, where $x_i \in \Re^d$ are input variables and $y_i \in \{-1, 1\}$ are class labels. We consider a class of models of the form $f(x) = \alpha^T x$, with the sign of $f(x)$ predicting the label associated with the point x. An hyperplane classifier with hinge loss can be designed by minimizing the following cost function.

$$\mathcal{J}(\alpha) = \Phi(\alpha) + \sum_{i=1}^{\ell} w_i \left(1 - \alpha^T y_i x_i\right)_+ \qquad (1)$$

where the function $\Phi : \Re^{(d)} \Rightarrow \Re$ is a regularization function or regularizer on the hyperplane coefficients and $(k)_+ = max(0, k)$ represents the hinge loss, and $\{w_i : w_i \geq 0, \forall i\}$ is the weight preassigned to the loss associated with x_i. For balanced data usually $w_i = w$, but for unbalanced data it is a common practice to weight positive and negative classes differently, i.e. $\{w_i = w_+,\ \forall i \in C^+\}$ and $\{w_i = w_-,\ \forall i \in C^-\}$ where C^+ and C^- are the corresponding sets of indices for the positive and negative classes respectively.

The function $\left(1 - \alpha^T y_i x_i\right)_+$ is a convex function. The weighted sum of convex functions is also convex. Therefore for a convex function $\Phi(\alpha)$ (1) is also convex. The problem in (1) can be formulated as a mathematical pro-

gramming problem as follows:

$$\begin{array}{rcl} \min\limits_{(\alpha,\xi)\in R^{d+\ell}} & \Phi(\alpha) + \sum_{i=1}^{\ell} w_i\xi_i & \\ \text{s.t.} & \xi_i \geq 1-\alpha^T y_i x_i & \\ & \xi_i \geq 0,\ \forall i & \end{array} \tag{2}$$

For $\Phi(\alpha) = \|\alpha\|_2^2$, where $\|.\|_2$ is the 2-norm, (2) results in the conventional Quadratic-Programming-SVM, and for $\Phi(\alpha) = |\alpha|$, where $|.|$ is the 1-norm it yields the sparse Linear-Programming-SVM.

3. Polyhedral Decision Boundaries

3.1. Training a Classifier with an AND Structure

We aim to optimize the following cost function

$$\begin{aligned} \mathcal{J}(\alpha_1,\dots\alpha_K) &= \sum_{k=1}^{K}\Phi_k(\alpha_k) \\ &+ \nu_1\sum_{k=1}^{K}\sum_{i\in C_k^-}(e_{ik})_+ \\ &+ \nu_2\sum_{i\in C^+}\max(0,e_{i1},\dots,e_{iK}) \end{aligned} \tag{3}$$

where $e_{ik} = 1+\alpha_k^T x_{ik}$ and $(e_{ik})_+$ defines the hinge loss of the i-th training example $\{(x_{ik}, y_{ik})\}$ in subclass-k induced by classifier k. C_k^- is the set of indices of the negative samples in subclass-k. Note that classifier k is designed to classify positive examples from the negative examples in the subclass-k. The first term in (3) is a summation of the regularizers for each of the classifiers in the cascade and the second and third terms accounts for the losses induced by the negative and positive samples respectively. Unlike (1) the loss function here is different for the positive samples. The loss induced by a positive sample i, $i \in C^+$ is zero only if $\forall k : 1-\alpha_k^T x_i \leq 0$, which corresponds to the "AND" operation. The problem (3) can be formulated as follows

$$\begin{array}{rcl} \min\limits_{(\alpha,\xi)\in R^{Kd+\ell}} & \sum_{k=1}^{K}\Phi_k(\alpha_k) + \nu_1\sum_{k=1}^{K}\sum_{i\in C_k^-}\xi_{ik} & \\ & + \nu_2\sum_{i\in C^+}\xi_i & \\ \text{s.t.} & \xi_{ik} \geq 1+\alpha_k^T x_{ik} & \\ & \xi_{ik} \geq 0 & \\ & \xi_i \geq 1-\alpha_k^T x_i & \\ & \xi_i \geq 0 & \end{array} \tag{4}$$

where the first two constraints are imposed for $\forall i \in C_k^-$, $k = 1,...,K$ and the last two constraints are imposed for $\forall i \in C^+$, $k = 1,...,K$. Note that for a convex function $\Phi(\alpha)$ the problem in (4) is convex. In a nutshell we designed K classifiers, one for each of the binary classification problems, i.e. positive class vs subclass-k of the negative class. Then we construct a learning algorithm to jointly optimize these classifiers such that the cost induced by a positive sample is zero if and only if all of the K classifiers classifies this sample correctly, i.e. $\forall k : 1-\alpha_k^T x_i \leq 0$. Since each negative sample is only used once for training the classifier k, the cost induced for a negative sample is zero as long as it is classified correctly by the corresponding classifier k, i.e. $1+\alpha_k^T x_{ik} \leq 0$. Each classifier can use an arbitrary subset of the original feature set. This provides run time advantages in real-time when the classification architecture is implemented in a cascaded framework. This will be explained later in the paper. For now to keep the notation clean and tractable we assumed each classifier uses the entire feature set in the formulation (4) above.

3.2. Training a Classifier with an AND-OR Structure

The AND algorithm is developed with the assumption that the negative class is fully labeled. That is to say, the subclass membership of each of the negative sample is known apriori. For most real world applications this is not a very realistic scenario as it is almost impractical to label all of the negative samples due to the time limitations. However one can label a small subset of the negative class to discover different type of subclasses as well as pool the training data for each subgroup. To accommodate for the unlabeled samples we modify the equation (3) for the unlabeled negative samples as follows.

$$\begin{aligned} \mathcal{J}(\alpha_1,\dots\alpha_K) &= \sum_{k=1}^{K}\Phi_k(\alpha_k) \\ &+ \nu_1\sum_{k=1}^{K}\sum_{i\in C_k^-}(e_{ik})_+ \\ &+ \nu_1\sum_{i\in \acute{C}^-}\prod_{k=1}^{K}(e_{ik})_+ \\ &+ \nu_2\sum_{i\in C^+}\max(0,e_{i1},\dots,e_{iK}) \end{aligned}$$

where $\acute{C}^-$ is the set of indices of the unlabeled negative samples. The first term in (5) requires a labeled sample from a subclass-k to be correctly classified by the classifier k. In other words if a sample is known to be a member of subclass-k, ideally it should be classified as negative by the corresponding classifier k. On the other hand the second term requires an unlabeled negative sample to be correctly classified by any of the classifiers. As long as an unlabeled sample is classified as negative it does not matter which classifier does it, i.e. $\exists k : 1-\alpha_k^T x_{ik} \leq 0$ which corresponds to a "OR" operation. The third term requires a positive sample is classified as positive by all of the K classifiers, i.e. $\forall k : 1-\alpha_k^T x_i \leq 0$, which corresponds to the "AND" operation.

Due to the product operation in the objective function for unlabeled samples, unlike equation (3), equation (5) can not be cast as a convex programming problem. In the next section we propose an efficient alternating optimization algorithm to solve this problem.

4. Cyclic Optimization of AND-OR Algorithm

We develop an iterative algorithm which, at each iteration, carries out K steps, each aiming to optimize one classifier at a time. This type of algorithms is usually called alternating or cyclic optimization approaches. At any iteration, we fix all of the classifiers but the classifier k. The fixed terms have no effect on the optimization of the problem once they are fixed. Hence solving (5) is equivalent to solving the following problem by dropping the fixed terms in (5):

$$\begin{aligned}\mathcal{J}(\alpha_k) &= \Phi_k(\alpha_k) \\ &+ \nu_1 \sum_{i\in C_k^-} (e_{ik})_+ \\ &+ \nu_1 \sum_{i\in \acute{C}^-} w_i\ (e_{ik})_+ \\ &+ \nu_2 \sum_{i\in C^+} \max(0, e_{i1}, \ldots, e_{ik}, \ldots, e_{iK})\end{aligned}$$

where $w_i = \prod_{k=1,k\neq k}^{K} (e_{ik})_+$. This can be cast into a constrained problem as follows

$$\begin{aligned}\min_{(\alpha_k,\xi)\in R^{d+\ell_k+\ell_+}} \quad & \Phi_k(\alpha_k) + \nu_1 \textstyle\sum_{i\in C_k^-} \xi_i \\ & + \nu_1 \textstyle\sum_{i\in \acute{C}^-} w_i\xi_i \\ & + \nu_2 \textstyle\sum_{i\in C^+} \xi_i \\ \text{s.t.} \quad & \xi_i \geq e_{ik},\ \forall i \\ & \xi_i \geq 0,\ \forall i \in \acute{C}^- \cup C_k^- \\ & \xi_i \geq \gamma_i,\ \forall i \in C^+\end{aligned} \tag{5}$$

where $\gamma_i = max\left(0, e_{i1}, \ldots, e_{i(k-1)}, e_{i(k+1)}, \ldots, e_{iK}\right)$ and ℓ_k is the number of samples in subclass-k. The subproblem in (5) is a convex problem and differs from the problem in (2) by two small changes. First the weight assigned to the loss induced by the negative samples is now adjusted by the term $w_i = \prod_{k=1,k\neq k}^{K} (e_{ik})_+$. This term multiplies out to zero for negative samples correctly classified by one of the other classifiers. For these samples $e_{ik} < 0$ and $\xi_i = 0$ making the constraints on ξ_i in (5) redundant. As a result there is no need to include these samples when training for the *classifier-k*, which yields significant computational advantages. Second the lower bound for ξ is now $max\left(0, e_{i1}, \ldots, e_{i(k-1)}, e_{i(k+1)}, \ldots, e_{iK}\right)$. This implies that if any of the classifiers in the cascade misclassifies x_{ik} the lower bound on ξ is no longer zero relaxing the constraint on x_{ik}.

4.1. An Algorithm for AND-OR Learning

(0) Initialize e_{ik} in (5) such that all candidates are classified as positive, i.e. 100% sensitivity, 0% specificity. Set counter $c = 1$.

(i) Fix all the classifiers in the cascade except classifier k and solve (5) for α_k^c using the training dataset $\left\{\left(x_i^k, y_i\right)\right\}_{i=1}^{\ell}$. Repeat this for all $k = 1, \ldots, K$.

(ii) Compute $J^c(\alpha_1, \ldots, \alpha_K)$ by replacing α_k^{c-1} by α_k^c in (5), for all $k = 1, \ldots, K$.

(iii) Stop if $J^c - J^{c-1}$ is less than some desired tolerance. Else replace α_k^{c-1} by α_k^c for all $k = 1, \ldots, K$, c by $c + 1$ and go to *step i*.

The initial objective function in (5) is neither convex nor twice differentiable due to the product of the hinge loss term. Therefore the convergence theorem introduced in (Bezdek & Hathaway, 2003) for cyclic optimization does not hold here. On the other hand the subproblem in (5) is convex and hence at each iteration $J^c <= J^{c-1}$ holds and also (5) is bounded below. These guarantee convergence of the algorithm from any initial point to the set of suboptimal solutions. The solution is suboptimal if the objective function J can not be further improved following any directions. For a more detailed discussion on this topic please see (Dundar & Bi, 2007).

An unseen sample x can be classified as positive if $max(1 - \alpha_1^T x, \ldots, 1 - \alpha_K^T x) \leq \tau$ and as negative if vice versa for a threshold τ. The receiver operating characteristics (ROC) curve can be plotted by varying the value of τ.

5. Cascade Design for Run-Time Speedups

In Figure 2 a cascade classification scheme is shown. The key insight here is to reduce the computation time and speed-up online learning. This is achieved by designing simpler classifiers in the earlier stages of the cascade to reject as many negative candidates as possible before calling upon classifiers with more features to further reduce the false positive rate. A positive result from the first classifier activates the second classifier and a positive result from the second classifier activates the third classifier, and so on (Viola & Jones., 2004). A negative outcome for a candidate at any stage in the cascade leads to an immediate rejection of that candidate. Under this scenario $T_{k-1} = T_k \cup F_k$ and $T_0 = T_K \cup \bigcup_1^K F_k$ where T_k and F_k are the sets of candidates labeled as positive and negative respectively by classifier k.

The proposed algorithm learns a polyhedron through jointly optimizing a series of sparse linear classifiers. Since

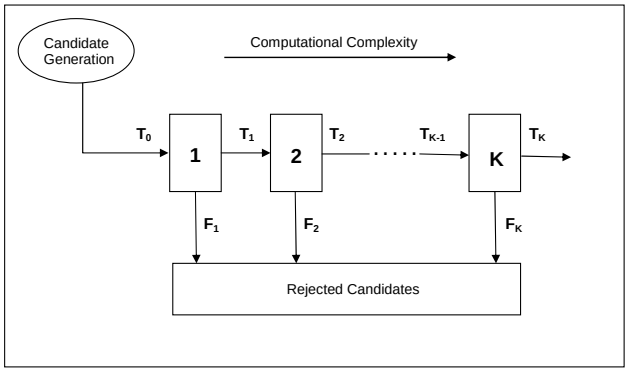


Figure 2. A general cascade framework used for online classification

the order these classifiers are executed in real-time does not matter in terms of the overall prediction accuracy of the system, we can arrange the execution sequence of these classifiers in a way to optimize the run-time. Let Γ_k be the set of indices of nonzero coefficients for *classifier-k*, t_i be the time required to compute *feature i* for a given candidate, $i = 1, \ldots, d$ and n_{k-1} is the number of samples in set T_{k-1}, then the total time required for the online execution of the algorithm is

$$\mathcal{T} = \sum_{k=1}^{K} n_{k-1} \sum_{i \in \left(\Gamma_{k-1} \setminus \bigcup_{i=0}^{k-2} \Gamma_i\right)} t_i \qquad (6)$$

The sets Γ_k are learned during offline training of the polyhedral classifier and are fixed for online execution. However the sets T_k is a function of the *classifier 1* through *classifier k-1*. Therefore this is a combinatorial optimization problem with $K!$ different outcomes, where $K!$ is the factorial of K. For small K one can try the exhaustive number of orderings between classifier to find the optimum sequence. However when K is large we can start with the *most sparse* classifier, i.e. the linear classifier with the least number of nonzero coefficients and choose the next classifier as the one that will require computing least number of additional features.

6. Computer Aided Detection

The goal of a Computer Aided Detection (CAD) system is to detect potentially malignant tumors and lesions in medical images (CT scans, X-ray, MRI etc). In an almost universal paradigm for CAD algorithms, this problem is addressed by a 3 stage system: A typical CAD system consists of a candidate generation phase, a feature extraction module and a classifier. The task of the candidate generation module is to create a list of potential polyps with a high sensitivity but low specificity. Features are then extracted for each candidate and eventually passed to a classifier where each candidate is labeled as normal or diseased.

In order to train a CAD system, a set of medical images (eg CT scans, MRI, X-ray etc) is collected from archives of community hospitals that routinely screen patients, *e.g.* for colon cancer. These medical images are then read by expert radiologists; the regions that they consider unhealthy are marked as ground-truth in the images. After the data collection stage, a CAD algorithm is designed to learn to diagnose images based on the expert opinions of the radiologists on the database of training images. First, domain knowledge engineering is employed to (a) identify all potentially suspicious regions in a candidate generation stage, and (b) to describe each such region quantitatively using a set of medically relevant features based on for example, texture, shape, intensity and contrast. Then, a classifier is trained using the features computed for each candidate in the training data and the corresponding ground truth.

When training a classifier for a CAD system for detection of colonic polyps, the only information that is usually available is the location of polyps, since radiologists only mark unhealthy regions when they are reading cases. This, of course, is very important for training a CAD system. But for all other structures that are picked up during candidate generation phase that are not pointing to a known lesion there is no other information available and they all have to be treated equally as negative examples. This introduces two complications. First all the negative candidates are clustered in one group although variation in size and shape among them is very big and valuable information about those candidates, e.g. type category, is not used. Second, radiologists only mark lesions of clinical importance, i.e. polyps greater than 6mm in colon. It is also possible that some lesions are overlooked during clinical evaluation. So potentially there are unidentified lesions in our dataset with no matching ground truth. If the candidate generation algorithm generates candidates for these lesions then these candidates are also marked as negative together with all the other candidates with no corresponding radiologist marks. In other words negative class may also contain unidentified samples of the target class.

In the rest of this section we will discuss some motivation for the proposed algorithm within the scope of a CAD system designed to detect colorectal cancer. Colorectal cancer is the second leading cause of cancer-related death in the western world (Jemal et al., 2004). Early detection of polyps through colorectal screening can help to prevent colon cancer by removing the polyps before they can turn malignant.

Typical examples of different polyp morphologies are given in Figure 3.

The commonly encountered false positive types in colon are fold, stool, tagged stool, meniscus, illeocecal valve, rectum etc. Some of these are shown in Figure 4. Ideally we can label all of the negative candidates in the training data and use the proposed AND algorithm in (4) to jointly train classifiers one for each of the subclasses of negative samples. However an exhaustive annotation of all negative examples is not feasible. Therefore we first select a very small subset of the negative candidates and annotate them manually through visual inspection. Then this set together with the positive samples and the remaining negative samples, i.e. unlabeled samples is used in the proposed $AND-OR$ framework to train the classifiers.

7. Experimental Results

We validate the proposed polyhedral classifier *(*polyhedral) with respect to its generalization performance and run-time efficiency. We compared our algorithm to a Support Vector Domain Description technique *(*svdd) (Tax & Duin, 1999), nonlinear SVM with Radial Basis Function *(*rbf), and one-norm SVM *(*sparse). To achieve sparseness we set the $\Phi_k(\alpha_k) = |\alpha_k|$ for the polyhedral classifier.

7.1. Data and Experimental Settings

The database of high-resolution CT images used in this study were obtained from two different sites across US. The 370 patients were randomly partitioned into two groups: training (n=167) and test (n=199). The test group was sequestered and only used to evaluate the performance of the final system.

Training Data Patient and Polyp Info: There were 167 patients with 316 volumes. The candidate generation (CG) algorithm identifies a total of 226 polyps at the volume level across all sizes while generating a total of 64890 candidates or an average of 205 false positives per volume. *Testing Data Patient and Polyp Info:* There were 199 patients with 385 volumes. The candidate generation (CG) algorithm identifies a total of 245 polyps at the volume level across all sizes while generating an average of 75946 samples or 194 false positives per volume (fp/vol).

A total of 98 features are extracted to capture shape and intensity characteristics of each candidate. The proposed algorithm requires a small set of false positives annotated. Rather than labeling false positives randomly across a dataset with 64890 samples we used the output of the most recent prototype classifier for labeling. This classifier is trained using a naive SVM and optimized for the 0-5 fp/vol range. This way we only focus on the most challenging false positives. This classifier marks a total of 1432 candidates as positive. Out of these candidates 1249 are false positives. A small subset of the volumes from this set is chosen for labeling and a total of 177 false positives (out of 1249) are annotated and ten different subcategories are identified.

7.2. Performance Evaluation

The classifiers are trained with the combination of 1249 false positives generated by the prototype classifier and all the polyps the candidade generation detects in the training data. A total of 1560 candidates are used for training. Classifiers are evaluated on the 1920 candidates the prototype classifier marked as positive in the testing data. The corresponding Receiver Operating Characteristics (ROC) curves for each algorithm is plotted in Figure 5.

The classifier parameters are estimated using a 10-fold patient cross validation from a set of discrete values using the training data. These are namely the width of the kernel (γ=[0.01 0.03 0.05 0.1 0.3 0.5 1 5]) for *rbf*, *svdd*, the cost factor (c=[5 10 15 20 25 50 75 100]) for *rbf*, *polyhedral*, *sparse* and the ν=[0.001 0.005 0.01 0.05 0.1 0.2] parameter for *svdd*. The desired tolerance value for Algorithm 4.1 is set to 0.001. The algorithm converged in less than 10 iterations.

As shown in Figure 5 the ROC curve corresponding to the proposed *polyhedral* classifier is consistently dominating all the other curves. The curve associated with the *sparse SVM* is almost linear implying a random behavior. This is not surprising to a greater extent as both the training and testing data sets used in this experiment are derived from the initial datasets via a linear SVM classifier. In other words the datasets are composed of samples marked as positive by the linear SVM, a significant portion of which are false positives. The *one-class SVM* is only slightly better than the *sparse SVM*. Even though the *rbf SVM* is the best of the three competitor algorithms, the difference in sensitivity between the *rbf SVM* and the proposed *polyhedral* classifier can be as high as 5% (10 polyps) in favor of the proposed algorithm.

7.3. Run-time Evaluation

As stated earlier in the paper, run-time speedups can be achieved as a by-product of the proposed algorithm when the real-time classification is implemented in a cascaded framework as in Figure 2. For a more detailed discussion of cascade classifiers in the CAD domain we refer the interested readers to these recent works (Dundar & Bi, 2007), (Bi et al., 2006).

To avoid any delays in the workflow of a physician the CAD results should be ready by the time physician completes his own review of the case. Therefore there is a run-time requirement a CAD system needs to satisfy. Among the stages involved during online processing of a volume,

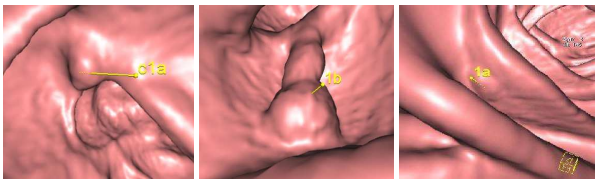


Figure 3. Polyp morphologies (from left to right): Sessile, pedunculated, and flat polyp

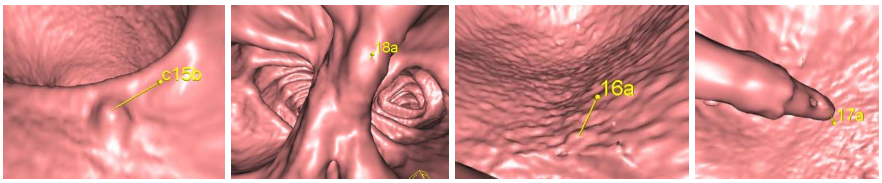


Figure 4. Negative examples (from left to right): stool, fold, noisy data and rectal tube

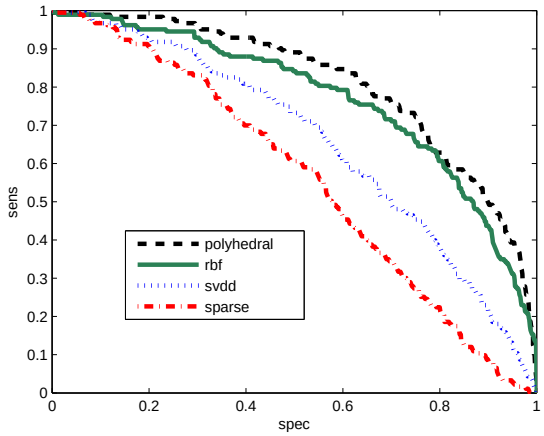


Figure 5. ROC curves obtained by the four classifiers on the test data.

feature computation is by far the most computational stage of online processing. A cascaded framework for executing the classifier in the order of increasing feature complexity may bring significant computational advantages in this case. In this framework the cascade is designed so as to execute classifiers with less number of features earlier in the sequence. This way the additional features required for the succeeding classifiers will only be computed on the remaining candidates, i.e. candidates marked as positive by all of the previous classifiers.

In this section we evaluate the speedups achieved by the proposed classifier. We set the operating point at 60% specificity, around 2.2fp/vol. At this specificity the proposed polyhedral classifier yields 85% sensitivity (see Figure 5). Assuming each feature takes on the average t seconds per candidate to compute we came up with the table in 1. This table shows the aggregate number of features used, feature computation time and number of candidates rejected at each stage in the sequence.

Feature computation for the proposed approach on average takes 452t secs per volume. On the other hand for *svdd* and *rbf*, which require computation of all the features at once, this stage takes 595t secs. This represents a roughly 25% improvement in run-time execution speed of the system. For the one-norm SVM *sparse*, this time is 437t secs. However the corresponding sensitivity at this operating point for one-norm SVM is around 40% vs 85% for the proposed technique.

8. Conclusions

In this study we have presented a methodology to take advantage of the subclass information information available in the negative class to achieve a more robust description of the target class. The subclass information which is neglected in conventional binary classifiers provides a better insight of the dataset and when incorporated into the learning mechanism acts as an implicit regularizer on the classifier coefficients. We believe this is an important contribution for applications where feature noise is prevalent. Highly nonlinear kernel classifiers provides flexibility for modeling complex data but they tend to overfit when there are too many redundant and irrelevant features in the data. Linear classifiers on the other hand do not have enough capacity to model complex data but they are more robust when there is noise. The polyhedral classifier is proposed as a midway solution. The linear faces of the polyhedron achieves robustness whereas multiple faces provides flexibility.

The order in which the classifiers are executed during online execution does not matter. Even though finding the globally optimum sequence is an open research problem for a large number of subclasses, the ordering of the clas-

sequence order	1	2	3	4	5	6	7	8	9
aggregate number of features	48	67	73	75	76	78	81	84	87
aggregate number of rejected candidates (avg. per volume)	1.08	2.37	2.57	2.82	2.90	2.93	3.06	3.07	3.40
aggregate feature computation time in t (avg. per volume)	291	386	408	414	418	424	434	443	452

Table 1. Run-time Results obtained for the Polyhedral classifier. The classifiers are executed in the order of increasing number of features required by each classifier.

sifiers can be arranged in a cascaded manner to reduce the total run-time of the system.

References

Bezdek, J., & Hathaway, R. (2003). Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, *11*, 351–368.

Bi, J., Periaswamy, S., Okada, K., Kubota, T., Fung, G., Salganicoff, M., & Rao, R. B. (2006). Computer aided detection via asymmetric cascade of sparse hyperplane classifiers. *Proceedings of the Twelfth Annual SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 837–844). Philadelphia, PA.

Brew, A., Grimaldi, M., & Cunningham, P. (2007). *An evaluation of one-class classification techniques for speaker verification* (Technical Report UCD-CSI-2007-8). University College Dublin.

Chen, Y., Zhou, X. S., & Huang, T. S. (2001). One-class svm for learning in image retrieval. *ICIP (1)* (pp. 34–37).

Dundar, M., & Bi, J. (2007). Joint optimization of cascaded classifiers for computer aided detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–8).

Jemal, D., Tiwari, R., Murray, T., Ghafoor, A., Saumuels, A., Ward, E., Feuer, E., & Thun, M. (2004). Cancer statistics.

Manevitz, L. M., & Yousef, M. (2001). One-class svms for document classification. *Journal of Machine Learning Research*, *2*, 139–154.

Mika, S., Rätsch, G., & Müller, K.-R. (2000). A mathematical programming approach to the kernel fisher algorithm. *NIPS* (pp. 591–597).

Murth, S. K., Kasif, S., & Salzberg, S. (1994). A system for induction of obligue decision trees. *Journal of Artificial Intelligence Research*, *2*, 1–33.

Scholkopf, B., Platt, O., Shawe-Taylor, J., Smola, A., & Williamson, R. (1999). Estimating the support of a high-dimensional distribution.

Tax, D. M. J., & Duin, R. P. W. (1999). Support vector domain description. *Pattern Recognition Letters*, *20*, 1191–1199.

Tipping, M. E. (2000). The relevance vector machine. In S. Solla, T. Leen and K.-R. Muller (Eds.), *Advances in neural information processing systems 12*, 652–658. Cambridge, MA: MIT Press.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer.

Viola, P., & Jones., M. (2004). Robust real-time face detection. *International Journal of Computer Vision*, *57*.

Active Reinforcement Learning

Arkady Epshteyn AEPSHTEY@GOOGLE.COM
Google Inc., 4720 Forbes Ave, Pittsburgh, PA 15213 USA

Adam Vogel ACVOGEL@STANFORD.EDU
Gerald DeJong MREBL@UIUC.EDU
Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA

Abstract

When the transition probabilities and rewards of a Markov Decision Process (MDP) are known, an agent can obtain the optimal policy without any interaction with the environment. However, exact transition probabilities are difficult for experts to specify. One option left to an agent is a long and potentially costly exploration of the environment. In this paper, we propose another alternative: given initial (possibly inaccurate) specification of the MDP, the agent determines the sensitivity of the optimal policy to changes in transitions and rewards. It then focuses its exploration on the regions of space to which the optimal policy is most sensitive. We show that the proposed exploration strategy performs well on several control and planning problems.

1. Introduction

When the transition probabilities and rewards of an MDP are known, the optimal policy can be computed offline. However, it is unrealistic to expect a domain expert to accurately specify thousands of MDP parameters. The optimal policy computed offline in an imperfectly modeled world may turn out to be suboptimal when executed in the actual environment. To fix this problem in practice, both rewards and transition probabilities are tweaked by domain experts until the desired performance is achieved. An alternative approach is to allow the agent to explore the world in a model-free fashion using reinforcement learning (RL). However, reinforcement learning in the actual environment is time-consuming, expensive, and sometimes dangerous (Abbeel and Ng (2005), for example, describe a helicopter crash which occurred during an overly aggressive exploration).

In this work, we introduce an approach called active reinforcement learning which combines the strengths of offline planning and online exploration. In particular, our framework allows domain experts to specify possibly inaccurate models of the world offline. However, instead of using this model for planning, our algorithm uses it as a blueprint for exploration. Our approach is based on the observation that, while all of the transition probabilities and rewards in the model may be misspecified, it is not important to know all of them to determine the optimal policy. Consider a surveillance helicopter flying agent. Does it make a difference if it crashes with probability 0.9 or 0.95 when it flies close to the ground? It seems unlikely that the optimal policy would be very sensitive to this value. However, the probability of the agent taking a good photograph of its target from a given viewing angle is extremely important. Therefore, the primary goal of the agent's experimentation, given a description of the problem, should be to determine the probabilities of capturing a photo of the target as opposed to trying to determine the exact probability of crashing. Active reinforcement learning enables this type of exploration. It uses sensitivity analysis to determine how the optimal policy in the expert-specified MDP is affected by changes in transition probabilities and rewards of individual actions. This analysis guides the exploration process by forcing the agent to sample the most sensitive actions first. We will present experimental results demonstrating the effectiveness of active RL. In addition, we will show that, while our algorithm is approximate, it produces near-optimal results in polynomial time for a special class of MDPs.

2. Related Work

Many strategies have been proposed to address the difficulty of specifying MDPs offline. Givan's bounded-parameter MDP framework (Givan et al., 2000) allows

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

the designer to specify uncertainty intervals around MDP's transition probabilities and rewards. The agent then finds the best policy in a game against adversarial nature which picks the worst possible world in which to evaluate it. (Alternative specifications of prior uncertainty in the same framework are given in Nilim and Ghaoui (2003).) While intuitively appealing, this approach may pick an overly conservative policy which is far from optimal for a given environment. Moreover, it places an excessive demand on the designer to quantify not only the prior model, but also his uncertainty about its transition probabilities.

In the online RL setting, there are plenty of reinforcement learning approaches that use optimistic exploration in face of uncertainty, such as E^3 (Kearns & Singh, 2002), $R-MAX$ (Brafman & Tennenholtz, 2002), and model-based interval exploration (Strehl & Littman, 2005). The main idea of these algorithms is to explore the actions the agent has experienced the fewest number of times in the past. This criterion does not apply to prior knowledge. In this paper, we focus on the problem of determining which states are worth exploring based solely on the prior MDP specification.

An approach similar in spirit to ours is Bayesian reinforcement learning (Dearden et al., 1999), which imposes a prior distribution over possible worlds and updates it based on interactions with the environment. However, this approach makes use of unrealistic assumptions on the shapes of probability distributions and approximate sampling to ensure tractability. The largest problem to which it was applied is two orders of magnitude smaller than the problems we solve in this work. In addition, we present an approximate version of our algorithm which is able to handle much larger (possibly continuous) state/action spaces.

The idea of using a prior MDP specification to reduce the amount of exploration in RL has also been explored by Abbeel et al. (2006). However, they only handle deterministic environments and their exploration is driven by the perceived optimal policy, not sensitivity analysis.

3. Preliminaries

The Markov decision process is defined by a tuple $(S, A, T, Next, R, \alpha)$, where $S = \{1, .., |S|\}$ is a finite set of states, A is a finite set of actions, $R(s,a)$ is a reward function, $T(s'|s,a)$ is a transition probability function, $\alpha \in (0,1)$ is the discount factor. $Next(s,a) = \{s' : T(s'|s,a) > 0\}$ defines a set of states reachable in one step with nonzero probability after taking action $a \in A$ in a state $s \in S$. Since transition probabilities of all the states in $Next(s,a)$ are constrained to lie in the probability simplex $\Delta(Next(s,a))$, $T(\cdot|s,a)$ is a function with $|Next(s,a)|-1$ degrees of freedom. To make this explicit, let $\underline{Next}(s,a)$ denote an arbitrary state such that $T(\underline{Next}(s,a)|s,a) = 1 - \sum_{s' \in \overline{Next}(s,a)} T(s'|s,a)$, where $\overline{Next}(s,a) = Next(s,a) \backslash \underline{Next}(s,a)$ is the set of states in $Next(s,a)$ other than the state $\underline{Next}(s,a)$, and let $T|_{\overline{s,a}}$ denote the restriction of $T(\cdot|s,a)$ to $\overline{Next}(s,a)$.

Let $\pi(s)$ define a deterministic policy which maps states to actions. Let $T^\pi(s,s') = T(s'|s,\pi(s))$ be the $|S| \times |S|$ transition probability matrix and $R^\pi(s) = R(s,\pi(s))$ be the $|S| \times 1$ reward vector under π. Then the value matrix $V^\pi(T,R)$ is given by the Bellman equation $V^\pi = \alpha T^\pi V^\pi + R^\pi$. V^π can be computed efficiently via iterative application of the Bellman equation, known as policy evaluation.

The utility of a policy π, $U^\pi(T,R)$, is given by the expected discounted rewards: $U^\pi(T,R) = E_{s_0 \sim D} V^\pi(s_0; T, R)$, with initial state s_0 drawn from the distribution D. *The utility of a policy explicitly depends on the transition and reward model of the MDP.*

We need one more piece of notation to describe the algorithm. We want to be able to take a transition probability function T and replace the transition probabilities $T(\cdot|\hat{s},\hat{a})$ of a fixed state/action pair $\hat{s},\hat{a}$ with a given probability distribution $X \in \Delta(Next(\hat{s},\hat{a}))$, leaving the rest of the probabilities the same. To do this, we define the replacement function $W_{\hat{s},\hat{a}}[T,X](s'|s,a) \triangleq \begin{cases} X(s'), & \text{if } s,a = \hat{s},\hat{a} \\ T(s'|s,a), & \text{otherwise} \end{cases}$. Similarly, the function $Y_{\hat{s},\hat{a}}[R,r](s,a) \triangleq \{r \text{ if } s,a = \hat{s},\hat{a} \text{ and } R(s,a) \text{ otherwise}\}$ replaces the reward of the chosen state/action pair with r.

4. Active RL Algorithm

In this section, we give a general overview of the active reinforcement learning algorithm.

Let T_0, R_0 be the user-supplied model of transition probabilities and rewards for an MDP. We can use Taylor's approximation to model the local sensitivity of $U^\pi(T_0,R_0)$ as the transition probabilities $\mathbf{X} \in \Delta(Next(\hat{s},\hat{a}))$ are perturbed around some specified value $\mathbf{T_1}$ for a *single* state/action pair $\hat{s},\hat{a}$:

$$\hat{U}^\pi_{T_1}(W_{\hat{s},\hat{a}}[T_0,\mathbf{X}]) \approx U^\pi(W_{\hat{s},\hat{a}}[T_0,\mathbf{T_1}],R_0) + \nabla_{\mathbf{X}|_{\overline{\hat{s},\hat{a}}}} U^\pi(W_{\hat{s},\hat{a}}[T_0,\mathbf{T_1}],R_0)(\mathbf{X}|_{\overline{\hat{s},\hat{a}}} - \mathbf{T_1}|_{\overline{\hat{s},\hat{a}}})$$

Transition probabilities for all the actions other than the action $\hat{a}$ in state $\hat{s}$ are held fixed at the values

defined by the user-supplied model T_0. Similarly, the rewards of all the actions are held fixed at the user-supplied values R_0. Sensitivity of the utility function to changes in rewards $R(s, a)$ of individual actions is modeled in an analogous fashion.

The above approximation fixes the transition probabilities for all the actions except one, and approximates the utility of the best policy around any specified point $\mathbf{T_1}$ in the transition probability simplex of that one action. In this section, we assume that it is possible to compute this Taylor's expansion efficiently and explain the main idea of our algorithm, deferring the details of computing the gradient to Section 5.

Taylor's approximation makes it possible to determine how the payoff from following a fixed policy is affected by the changes in the MDP parameters. However, even large changes in payoffs do not necessarily mean that the agent is acting suboptimally. An extreme illustration of this is a gridworld agent who is rewarded only upon getting to the goal state. Even if the agent is wrong about the magnitude of the reward, its optimal policy remains the same: always move towards the goal. Thus, an agent could be completely wrong about the environment and still act optimally. The key goal of the sensitivity analysis is to determine how the *optimal policy* changes in response to the changes in the transition probabilities and rewards. One way to measure this sensitivity is by asking the question: how much do transition probabilities/rewards of a given action have to change before the currently optimal policy becomes suboptimal?

To make this question precise, let us first focus on the sensitivity to transition probabilities. We will use $\Pi_{T_1;\hat{s},\hat{a}} = \arg\max_\pi U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{T_1}], R_0)$ to denote the optimal policy in the MDP in which all transitions except for those of action $\hat{a}$ in state $\hat{s}$ are held fixed at T_0, and transitions of $\hat{s}, \hat{a}$ are given by $\mathbf{T_1}$. Let $C = \{T : U^{\Pi_{T_0;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{T}], R_0) \geq U^{\Pi_{T;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{T}], R_0)\}$ define a region in the transition probability space in which the optimal policy Π_{T_0} for the user-specified MDP dominates every other policy. The goal of sensitivity analysis is to find the radius of the largest ball which we can position at $\mathbf{T_0}$ and expand without leaving the confines of C along the dimensions $T(\cdot|\hat{s}, \hat{a})$ corresponding to the transitions for a given action $\hat{a}$ in $\hat{s}$. The larger the ball, the more robust the optimal policy is to the changes in the transition probabilities of the given action $\hat{a}$. However, it is not obvious how to compute this quantity exactly. Instead, we approximate it via a variant of Newton's root-finding method which starts out at some point T_0' in the transition probability space outside of C and converges to a point on the boundary of C. The method is illustrated in Figure 1. Each application of the method consists of starting out at some point T_0', replacing the utility function $U^{\Pi_{T_0';\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{T_0'}], R_0)$ of the best policy at T_0' with its tangent, finding the "zero" of this tangent, i.e., a point T_1' closest to T_0 in the intersection of this tangent and the tangent to the utility function at T_0, replacing the initial estimate T_0' with the new estimate T_1', and iterating.

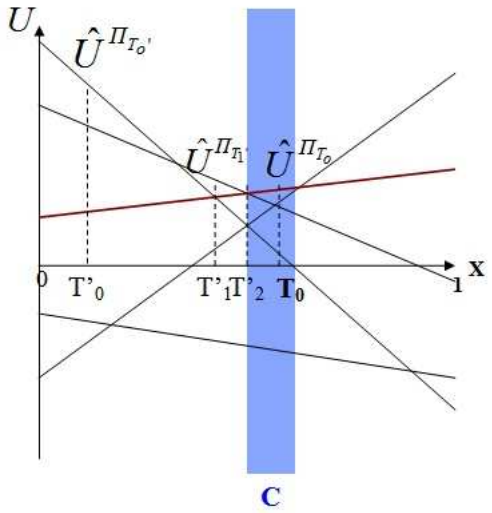


Figure 1. Newton's method for sensitivity analysis of MDPs. The $\mathbf{X}$-axis is the probability of transition to s' for a single chosen $\hat{s}, \hat{a}$ pair that leads to one of two states s' or s'' (the probability of transition to s'' is $1 - \mathbf{X}$). The utilities of policies are linear functions of $\mathbf{X}$. Function $U^{\Pi_T}(\mathbf{X})$ is the utility of the policy which is optimal when $\mathbf{X} = T$. It is given by the upper envelope of the set of all value functions. C is the (blue) region where the optimal policy at T_0 (red line) dominates every other policy. The algorithm starts at T_0' and converges to T_2' on the boundary of C.

The pseudocode for this algorithm consists of the following steps:

1. Determine the optimal policy $\Pi_{T_0;\hat{s},\hat{a}}$ for the user-provided model using any MDP solver.
2. Determine the Taylor approximation of the utility of this policy $\hat{U}_{T_0}^{\Pi_{T_0;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$ as a function of transition probabilities $\mathbf{X} \in \Delta(Next(\hat{s}, \hat{a}))$.
3. Select the starting point $T_0' \in \Delta(Next(\hat{s}, \hat{a}))$ and let $i \leftarrow 0$.
4. Using any MDP solver, determine the optimal policy $\Pi_{T_i';\hat{s},\hat{a}}$ for the user-provided model with transition probabilities of action $\hat{a}$ in state $\hat{s}$ replaced by T_i'.

5. Determine the Taylor approximation of the utility of this policy $\hat{U}_{T_i'}^{\Pi_{T_i';\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$ as a function of transition probabilities $\mathbf{X} \in \Delta(Next(\hat{s}, \hat{a}))$.

6. Let T'_{i+1} be the point in the intersection of $\hat{U}_{T_0}^{\Pi_{T_0;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$ and $\hat{U}_{T_i'}^{\Pi_{T_i';\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$ closest to the user-specified model T_0. Find T'_{i+1} by solving the following second-order cone program[1]:

$$T'_{i+1} = \arg\min_{\mathbf{X}} \left\| \mathbf{X}|\overline{\hat{s},\hat{a}} - \mathbf{T_0}|\overline{\hat{s},\hat{a}} \right\|$$

$$\text{s.t.} \hat{U}_{T_0}^{\Pi_{T_0;\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}]) = \hat{U}_{T_i'}^{\Pi_{T_i';\hat{s},\hat{a}}}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}])$$

$$[\mathbf{X}|\overline{\hat{s},\hat{a}}] \succeq \mathbf{0}; [\mathbf{X}|\overline{\hat{s},\hat{a}}]^T \boldsymbol{e} \leq 1,$$

where $\boldsymbol{e}$ is a vector of all 1's

The last set of constraints ensures that T'_{i+1} is a valid probability distribution.

7. Let $i \leftarrow i+1$ and repeat steps 4-7 while $\Pi_{T_i;\hat{s},\hat{a}} \neq \Pi_{T_0;\hat{s},\hat{a}}$.

8. Return $\left\| T_i'|\overline{\hat{s},\hat{a}} - T_0|\overline{\hat{s},\hat{a}} \right\|$

The algorithm returns an estimate of the maximum-radius sensitivity ball about the user-specified transition model T_0. The estimate is the minimum over a set of restarts of Newton's method, initializing T_0' to each vertex of the probability simplex $\Delta(Next(\hat{s}, \hat{a}))$. The quality of the estimate is limited both by the finite number of iterations of Newton's method and by the limited number of restarts. The algorithm is executed for every state/action pair $\hat{s}, \hat{a}$ to find which actions are most sensitive to changes in transition probabilities of the user-supplied model. We confirmed experimentally that the estimates of the radius of the sensitivity ball produced by our algorithm are very close to the true values when the dimensionality of the probability simplex is small.

An analogous algorithm is used to determine the sensitivity of the MDP to perturbations in individual rewards. We have not yet described how to compute the Taylor approximation of the utility function. We will do so in the next section.

5. Sensitivity of a Policy

By definition of the gradient and linearity of expectation, the gradient of the utility function is given by $\nabla_{\mathbf{X}|\overline{\hat{s},\hat{a}}} U^{\pi}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0) =$ $[E_{s_0 \sim D} \frac{\partial V^{\pi}(s_0; W_{\hat{s},\hat{a}}[T_0,\mathbf{X}], R_0)}{\partial X(s'_j)}]_{j=1}^{|\overline{Next}(\hat{s},\hat{a})|}; s'_j \in \overline{Next}(\hat{s}, \hat{a})$. To compute the gradient of the value function, we need the following lemma:

Lemma 5.1. *For a given policy π, a [state,action,next state] tuple $[\hat{s}, \hat{a}, s'] : s' \in \overline{Next}(\hat{s}, \hat{a})$, a transition function for the given $[\hat{s}, \hat{a}]$: $\mathbf{X} \in \Delta(Next(s, a))$, reward function R_0, let transition function $T = W_{\hat{s},\hat{a}}[T_0, \mathbf{X}]$. Then $\frac{\partial V^{\pi}(s_0;T,R_0)}{\partial X(s')} \triangleq 0$ for $\hat{a} \neq \pi(\hat{s})$. For $\hat{a} = \pi(\hat{s})$, let V^{π} be the policy value function which satisfies the Bellman equation and let an $|S| \times |S|$ matrix L^{π} define the directional vector for the derivative: $L^{\pi}(s_i, s_j) \triangleq \begin{cases} 1, & \text{if } s_j = s' \\ -1, & \text{if } s_j = \underline{Next}(\hat{s}, \hat{a}) \\ 0, & \text{otherwise} \end{cases}$. Then the partial derivative $\frac{\partial V^{\pi}(s_0;T,R_0)}{\partial X(s')}$ can be computed from the recurrence $\frac{\partial V^{\pi}(T,R_0)}{\partial X(s')} = \alpha T^{\pi} \frac{\partial V^{\pi}(T,R_0)}{\partial X(s')} + \alpha L^{\pi} V^{\pi}$. The form of this recurrence is exactly the same as that of the Bellman equation, with the value function replaced by its derivative and the reward function replaced by $\alpha L^{\pi} V^{\pi}$. Therefore, policy evaluation can be used to compute $\frac{\partial V^{\pi}(s_0;T,R_0)}{\partial X(s')}$.*

Proof. (sketch) A slight modification of the analysis given in (Cao, 2003) shows that $\frac{\partial V^{\pi}(T,R_0)}{\partial X(s')} = \alpha(I - \alpha T^{\pi})^{-1} L^{\pi} V^{\pi}$, where I is the $|S| \times |S|$ identity matrix. In order to compute the directional derivatives efficiently, note that this equation can be rewritten as the above recurrence. □

Applying a similar analysis to calculate the derivative of the utility function with respect to the reward x for a given state/action pair $[\hat{s}, \hat{a}]$, we obtain (letting the reward function $R = Y_{\hat{s},\hat{a}}(R_0, x)$): $\frac{\partial V^{\pi}(T_0,R)}{\partial x} = \alpha T_0^{\pi} \frac{\partial V^{\pi}(T_0,R)}{\partial x} + M$ where $M(s_i) \triangleq \begin{cases} 1, \text{ if } s_i = \hat{s} \\ 0, \text{ otherwise} \end{cases}$ for $\hat{a} = \pi(\hat{s})$ and $\frac{\partial V^{\pi}(T,R)}{\partial x} \triangleq 0$ for $\hat{a} \neq \pi(\hat{s})$.

6. Convergence and Complexity

In this section, we consider the algorithm's convergence and complexity. The geometric structure of our algorithm is similar to policy iteration which has well-known connections to Newton's method (Puterman, 1994; Madani, 2000). Just like in policy iteration, the known local quadratic convergence of Newton's method does not ensure global polynomial time complexity. Unlike policy iteration, we cannot rely on properties of contractions to establish convergence. However, we can establish convergence for MDPs whose structures (given by transitions with nonzero probabilities) are directed acyclic graphs (DAGs). For

[1] Second-order cone programs (SOCPs) are a special case of semidefinite programs which can be solved more efficiently, see (Lobo et al., 1998) for an overview.

such MDPs, the values $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)$ are linear functions of $\mathbf{X}|\hat{s},\hat{a}$ for single state/action pairs $[\hat{s},\hat{a}]$. We have the following result which follows from the Intermediate Value Theorem:

Theorem 6.1. *Suppose that for a given $\hat{s},\hat{a}$ and for every policy π, $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}]; R_0)$ is a linear function of $\mathbf{X}|\hat{s},\hat{a}$. Then, for any initial point $T'_0 \in \Delta(Next(\hat{s},\hat{a}))$, the sequence $\{T'_i\}$ generated by the active RL algorithm converges to the boundary of C in a finite number of steps.*

For DAG-structured MDPs such that the maximum number of $Next$ states for any state/action pair is two, much stronger guarantees are available. As pointed out in (Madani, 2000), any DAG-structured MDP can be converted in polynomial time into an MDP of this form by introducing extra states and transitions as necessary. The significance of these MDPs is that, since $\overline{Next}(\hat{s},\hat{a})$ is a singleton, $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)$ is a linear function of a single variable $\mathbf{X}|\hat{s},\hat{a}$. The following theorem shows that our algorithm can determine the radius of the region C in logarithmic time in this degenerate case:

Theorem 6.2. *Define the distance between two policies π and π' as $max_{\mathbf{X}}|U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0) - U^{\pi'}(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)|$. Let γ be the smallest distance larger than 0 between any policy and $\Pi_{T_0;\hat{s},\hat{a}}$. Let the MDP have bounded rewards: $|R_0(s,a)| \leq M$ for $\forall s,a$. Then, after $t = O(\log(\frac{M}{\gamma\epsilon(1-\alpha)}))$ iterations, the iterates $T'_{i\geq t}$ of Newton's method are within ϵ of the limit point on the boundary of C.*

Proof. (sketch) The proof follows from a known fact that one-dimensional Newton's method makes progress in each iteration by either exponentially decreasing the height or exponentially increasing the slope of the function $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)$. (Madani, 2000). □

If the structure of an MDP is not a DAG, then $U^\pi(W_{\hat{s},\hat{a}}[T_0, \mathbf{X}], R_0)$ is not linear in $\mathbf{X}|\hat{s},\hat{a}$ and the above convergence results no longer apply. However, as the discount factor $\alpha \to 0$, the value function for any MDP will become approximately linear as the influence of distant rewards becomes negligible. Thus, Newton's method offers a way to find an approximate solution to our problem which becomes more accurate as the discount factor decreases. For general root-finding problems, Newton's method need not converge (i.e., it may cycle or diverge to infinity). It is possible that our variant may also exhibit this undesirable behavior when applied to arbitrary MDPs. However, our experimental results in Section 8 demonstrate that our method works well in practice on a wide range of problems even when strong assumptions required to obtain theoretical guarantees for convergence are significantly violated.

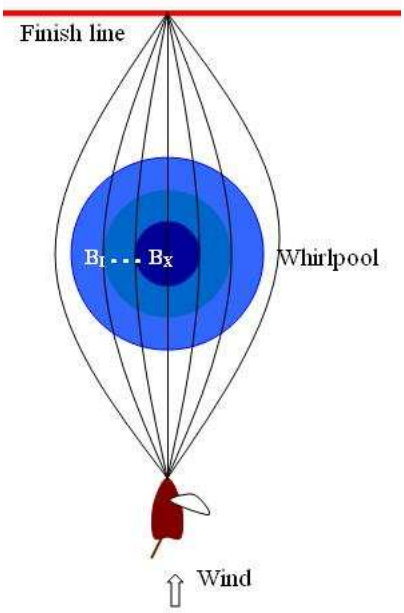


Figure 2. Sailboat Domain. Black lines indicate the paths of the boat for a set of possible initial policies. The boat is controlled by the rudder and the sail.

7. Approximate Active RL

In worlds with very large or continuous state/action spaces, the exact Active RL algorithm of Section 4 is intractable. Moreover, in continuous state spaces, sensitivity of individual state/action pairs no longer applies. Instead, we consider the case where the state space is partitioned into regions $\mathbb{B}$ with the uncertainty in our transition model for each region generated by a random variable.

Formally, we assume that the world is governed by the control model $s_{t+1} = f(s_t, a, d(\mathbb{B}(s_t))$. The agent's state s at time $t+1$ is a (possibly nonlinear) function f of the agent's state at time t, its action a, and the disturbance input d. In every region $\hat{B} \in \mathbb{B}$, the disturbance $d(\hat{B})$ is a random variable with a probability distribution $T \in \Delta(Next(\hat{B}))$ defined on a discrete set $Next(\hat{B}) = \underline{Next}(\hat{B}) \cup \overline{Next}(\hat{B})$. The approximate active RL procedure determines which regions $\hat{B} \in \mathbb{B}$ affect the optimal policy the most.

Consider, for example, the sailing problem illustrated in Figure 2. The agent's goal in this domain is to get

the sailboat to the finish line. The sailboat is controlled by the rudder and the sail. The problem is complicated by a probabilistic whirlpool which could aid the agent by increasing its speed or detain it by deviating the boat from its course. In this case, the function f describes the sailboat dynamics given the boat's position, the rudder/sail action, the deterministic wind, and the whirlpool current d. The strength of the current and its direction are given by a distribution T which depends on the band $\hat{B}$ inside the whirlpool in which the agent finds itself. Approximate active RL helps us find a small number of bands in which the current determines which one of the policies shown in Figure 2 is optimal.

The approximate active RL procedure is the same as its exact variant, except that: 1) local policy search is used in place of an MDP solver[2], 2) the utility $\widetilde{U}^{\pi}(T,R)$ of any policy π is approximated by Markov Chain Monte Carlo, and 3) the Taylor approximation of a policy's utility is given by $\hat{U}^{\pi}_{T_1}(W_{\hat{B}}[T_0,\mathbf{X}]) \approx \widetilde{U}^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1}],R_0) + \nabla_{\mathbf{X}|\overline{\hat{B}}} U^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1}],R_0)(\mathbf{X}|\overline{\hat{B}} - \mathbf{T_1}|\overline{\hat{B}})$, where $W_{\hat{B}}[T_0;\mathbf{T_1}]$ is a world in which the disturbance distribution in band $\hat{B}$ is replaced with $\mathbf{T_1}$, and the gradient of the utility function is approximated linearly by perturbing the transition probabilities by a small value ϵ in each dimension of the probability simplex: $\nabla_{\mathbf{X}|\overline{\hat{B}}} U^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1}],R_0) \approx \frac{1}{\epsilon}[\widetilde{U}^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1^{j;\hat{B}}}],R_0) - \widetilde{U}^{\pi}(W_{\hat{B}}[T_0,\mathbf{T_1}],R_0)]_{j=1}^{\overline{Next}(\hat{B})}$ where $\mathbf{T_1^{j;\hat{B}}}$ denotes the world in which the transitions in region $\hat{B}$ are perturbed by ϵ as follows:

$$\mathbf{T_1^{j;\hat{B}}}(d') \triangleq \begin{cases} \mathbf{T_1}(d') + \epsilon, & \text{if } d' = d_j \\ \mathbf{T_1}(d') - \epsilon, & \text{if } d' = \underline{Next}(\hat{B}) \\ \mathbf{T_1}(d'), & \text{otherwise} \end{cases}$$

8. Experiments

Exact RL experiments were performed on the following domains:

- In the mountain-car task (231 states), the problem is to drive a car up a steep mountain (Sutton & Barto, 1998). The engine power is a uniform random variable on the interval $[.15,.3]$.
- The task in the cart-pole problem (5832 states) is to balance a pole on a moving cart. The power of the cart is random, uniformly distributed in the interval $[20,75]$.
- Windy gridworld is a simple 10×7 gridworld with agent's movement affected by stochastic wind (Sutton & Barto, 1998).
- Pizza delivery problem (4769 states) is based on the racetrack example (Sutton & Barto, 1998). The agent's goal is to drive a car to the finish line, while delivering as many pizzas and avoiding as many randomly placed potholes as possible.
- The drunkard's walk problems are two 10×10 gridworlds with random rewards and penalties. When the agent moves in some direction, it is equally likely to deviate diagonally from it.

In all the setups, $\alpha = 0.9$ was used. The structure of the MDPs varies widely from one problem to another (and none of them are DAGs). In the first set of experiments, the effectiveness of active reinforcement learning for transition probabilities was evaluated. The system was provided with an initial description of the problem, as given above. It then performed the sensitivity analysis and sorted state/action pairs based on their sensitivity values. A different problem specification was then generated by randomly perturbing all the transition probabilities. This new specification represented the actual world in which transition probabilities are different from the expert-provided MDP specification. The agent was allowed to sample one action at a time in this actual world[3], replace user-specified transition probabilities with their maximum likelihood estimates (based on 10,000 samples), use an MDP solver to find the optimal policy in this "corrected" MDP, and evaluate this policy in the actual world. We tested two different ways of selecting the order in which actions were tested: 1) the active RL agent which samples the actions in order of decreasing sensitivity, with sensitivities computed by the algorithm in Section 4, and 2) the random agent which samples actions randomly. For comparison, we also tested two agents applying fixed policies: 1) the prior agent which applies the optimal policy for the expert-provided MDP specification, and 2) the omniscient agent which knows the transition probabilities in the actual test world and selects the optimal policy for this world. In addition, we tested the Q-learning agent with $\epsilon-$greedy exploration ($\epsilon = 0.1$) and full backups (full backups means that after taking an action in a state, the agent gets full information about all the

[2] Any local policy search algorithm, such as policy gradient or dynamic programming, can be used for this procedure.

[3] This exploration strategy (sampling with resets) assumes that the agent can execute any action in any state and observe its outcome without having to plan how to get to that state.

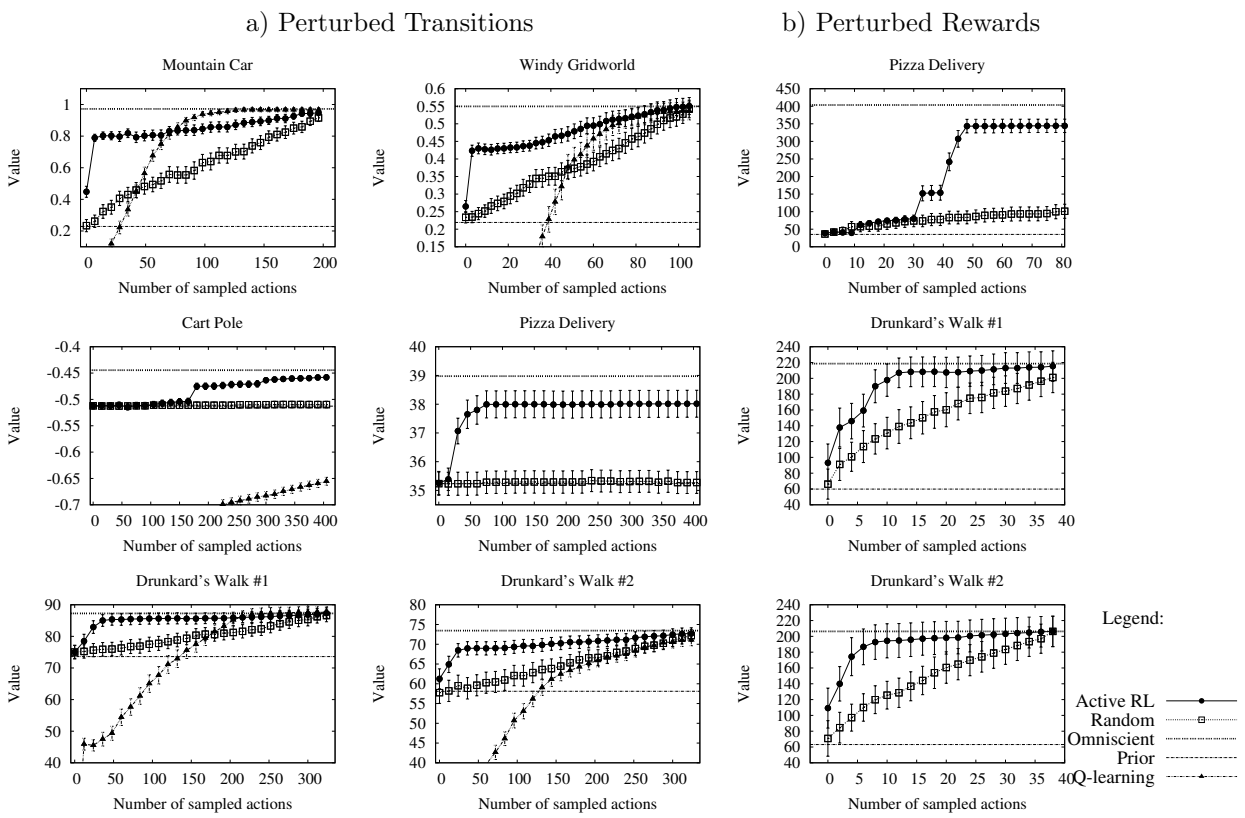


Figure 3. Evaluation of exploration strategies on perturbed MDPs. Error bars are based on 95% confidence intervals.

transition probabilities for all the possible next states for that action - this model was used to make the comparison between the Q-learning agent and the Active RL agent fair). The utility of the policy of each of these five agents appears in the plots in Figure 3-(a) as a function of the number of actions tested. The results are averaged over 100 different randomly generated actual worlds[4]. In each domain, the active RL agent outperforms the random sampling agent. The prior agent which relies solely on the expert's specification performs poorly, indicating the need for exploration. As expected, the Q-learning agent performs poorly initially since it has no prior knowledge, but improves with experience. It rarely catches up with the Active RL agent because Q-learning is forced to explore without resets. These experiments indicate that integrating Active RL with Q-learning may result in improvement in the Q-learning agent's performance. This is an important future extension of our work.

In the next experiment, the random worlds were generated by perturbing the rewards rather than transition probabilities of the MDP[5]. Performance of the four exploration strategies on the three domains with non-trivial reward structure appear in Figure 3-(b). Once again, the active RL agent significantly outperforms the random sampling agent.

Finally, we experimented with approximate active RL in the sailboat simulation, in which the agent must navigate a whirlpool of water current to reach the finish line. The whirlpool was modeled by ten concentric bands based on the distance from the center of the vertex, and the magnitude of the current varied proportionally to this distance. The expert-specified world reflected uncertainty about the direction of the whirlpool current: the direction of the current was counterclockwise with probability 0.1, clockwise with probability 0.1, and there was no current with prob-

[4]The test worlds were generated by perturbing transition probabilities of each action uniformly in the probability simplex within a radius of 0.6 around the expert-specified values T_0. A random variable $T \in \mathbb{R}^n$ uniformly distributed on the n-dimensional probability simplex can be generated from $n-1$ random variables $X_1, .., X_{n-1} \sim Uniform(0,1)$ by sorting them into $X_{(0)} \triangleq 0, X_{(1)}, .., X_{(n-1)}, X_{(n)} \triangleq 1$, and letting $T_i = X_{(i)} - X_{(i-1)}$ (Devroye, 1986). Rejection sampling is then used to ensure that $\|T - T_0\| \leq 0.6$.

[5]The test worlds were generated by perturbing all the nonzero rewards uniformly in the interval $[-70, 70]$ around the expert-specified values.

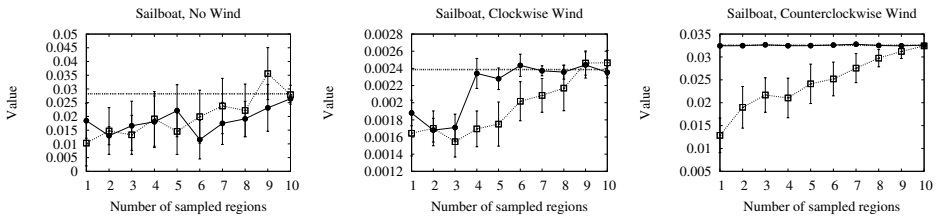


Figure 4. Evaluation of exploration strategies in the approximation architecture. The legend is the same as in figure 3, but with Prior strategy not shown (its value is too low to appear on the plots). Error bars are based on 95% confidence intervals.

ability 0.8. In each iteration of the active RL algorithm, local policy search was performed from each of the seven policies shown in Figure 2, and the best policy was selected. Approximate Active RL was tested in three actual worlds: one with a deterministic clockwise current, one with a deterministic counterclockwise current, and one with no current. The results appear in Figure 4. In the two worlds with current, the algorithm which samples bands according to the active RL-prescribed order outperforms the algorithm which samples bands according to a random order. In the world with no current, the performance of the two algorithms is similar.

9. Conclusions

In this paper, we presented a new algorithm for combining exploration with prior knowledge in reinforcement learning. We demonstrated that our algorithm can be implemented efficiently using policy iteration and a standard SOCP solver. We also introduced an approximate version of active RL to be applied in domains with large state spaces. In addition to being useful for exploration, the active RL algorithm can be used by an MDP designer to determine which regions of the state space require most precision in specifying transition probabilities and rewards. An important future extension of this work is designing a policy which explores the sensitive regions of the state space without resets.

References

Abbeel, P., & Ng, A. Y. (2005). Exploration and apprenticeship learning in reinforcement learning. *ICML*.

Abbeel, P., Quigley, M., & Ng, A. (2006). Using inaccurate models in reinforcement learning. *ICML*.

Brafman, R. I., & Tennenholtz, M. (2002). R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Cao, X. (2003). From perturbation analysis to markov decision processes and reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, *13*, 9–39.

Dearden, R., Friedman, N., & Andre, D. (1999). Model based bayesian exploration. *Uncertainty in Artificial Intelligence*.

Devroye, L. (1986). *Non-uniform random variate generation*. New York, NY: Springer-Verlag.

Givan, R., Leach, S., & Dean, T. (2000). Bounded-parameter markov decision processes. *Artificial Intelligence*, *122*, 71–109.

Kearns, M., & Singh, S. (2002). Near optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

Lobo, M. S., Vandenberghe, L., Boyd, S., & Lebret, H. (1998). Applications of second-order cone programming. *Linear Algebra and its Applications*, *284*, 193–228.

Madani, O. (2000). *Complexity results for infinite-horizon markov decision processes*. Doctoral dissertation, University of Washington.

Nilim, A., & Ghaoui, L. E. (2003). Robustness in markov decision problems with uncertain transition matrices. *NIPS*.

Puterman, M. (1994). *Markov decision processes—discrete stochastic dynamic programming*. New York, NY: John Wiley & Sons, Inc.

Strehl, A. L., & Littman, M. L. (2005). A theoretical analysis of model-based interval estimation. *ICML*.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Training Structural SVMs when Exact Inference is Intractable

Thomas Finley TOMF@CS.CORNELL.EDU
Thorsten Joachims TJ@CS.CORNELL.EDU
Cornell University, Department of Computer Science, Upson Hall, Ithaca, NY 14853 USA

Abstract

While discriminative training (e.g., CRF, structural SVM) holds much promise for machine translation, image segmentation, and clustering, the complex inference these applications require make exact training intractable. This leads to a need for approximate training methods. Unfortunately, knowledge about how to perform efficient and effective approximate training is limited. Focusing on structural SVMs, we provide and explore algorithms for two different classes of approximate training algorithms, which we call undergenerating (e.g., greedy) and overgenerating (e.g., relaxations) algorithms. We provide a theoretical and empirical analysis of both types of approximate trained structural SVMs, focusing on fully connected pairwise Markov random fields. We find that models trained with overgenerating methods have theoretic advantages over undergenerating methods, are empirically robust relative to their undergenerating brethren, and relaxed trained models favor non-fractional predictions from relaxed predictors.

1. Introduction

Discriminative training methods like conditional random fields (Lafferty et al., 2001), maximum-margin Markov networks (Taskar et al., 2003), and structural SVMs (Tsochantaridis et al., 2005) have substantially improved prediction performance on a variety of structured prediction problems, including part-of-speech tagging (Altun et al., 2003), natural language parsing (Tsochantaridis et al., 2005), sequence alignment (Yu et al., 2007), and classification under multivariate loss functions (Joachims, 2005). In the context of structural SVMs, in all these problems, both the inference problem (i.e., computing a prediction) and the separation oracle required in the cutting-plane training algorithm can be solved exactly. This leads to theoretical guarantees of training procedure convergence and solution quality.

However, in many important problems (e.g., clustering (Culotta et al., 2007; Finley & Joachims, 2005), multi-label classification, image segmentation, machine translation) exact inference and the separation oracle are computationally intractable. Unfortunately, use of approximations in these settings abandons many of the existing theoretical guarantees of structural SVM training, and relatively little is known about discriminative training using approximations.

This paper explores training structural SVMs on problems where exact inference is intractable. A pairwise fully connected Markov random field (MRF) serves as a representative class of intractable models. This class includes natural formulations of models for multi-label classification, image segmentation, and clustering. We identify two classes of approximation algorithms for the separation oracle in the structural SVM cutting-plane training algorithm, namely undergenerating and overgenerating algorithms, and we adapt loopy belief propagation (LBP), greedy search, and linear-programming and graph-cut relaxations to this problem. We provide a theoretical and empirical analysis of using these algorithms with structural SVMs.

We find substantial differences between different approximate algorithms in training and inference. In particular, much of the existing theory can be extended to overgenerating though not undergenerating methods. In experimental results, intriguingly, our structural SVM formulations using the overgenerating linear-programming and graph-cut relaxations successfully learn models in which relaxed inference is "easy" (i.e., the relaxed solution is mostly integral), leading to robust and accurate models. We conclude that the relaxation formulations are preferable over the formulations involving LBP and greedy search.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Algorithm 1 Cutting plane algorithm to solve OP 1.

1: Input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, C, ϵ
2: $S_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
3: **repeat**
4: **for** $i = 1, \ldots, n$ **do**
5: $H(\mathbf{y}) \equiv \Delta(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) - \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i)$
6: compute $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$
7: compute $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$
8: **if** $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$ **then**
9: $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
10: $\mathbf{w} \leftarrow$ optimize primal over $\bigcup_i S_i$
11: **end if**
12: **end for**
13: **until** no S_i has changed during iteration

2. Structured Output Prediction

Several discriminative structural learners were proposed in recent years, including conditional random fields (CRFs) (Lafferty et al., 2001), Perceptron HMMs (Collins, 2002), max-margin Markov networks (M^3Ns) (Taskar et al., 2003), and structural SVMs (SSVMs) (Tsochantaridis et al., 2005). Notational differences aside, these methods all learn (kernelized) linear discriminant functions, but differ in how they choose model parameters.

2.1. Structural SVMs

Structural SVMs minimize a particular trade-off between model complexity and empirical risk. From a training set $\mathcal{S} = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n))$, an SSVM learns a hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$ to map inputs $\mathbf{x} \in \mathcal{X}$ to outputs $\mathbf{y} \in \mathcal{Y}$. Hypotheses take the form $h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}, \mathbf{y})$ with discriminant function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, where $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$. The $\mathbf{\Psi}$ combined feature vector function relates inputs and outputs, and $\mathbf{w}$ are model parameters. The loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ indicates how far $h(\mathbf{x}_i)$ is from true output $\mathbf{y}_i$. To find $\mathbf{w}$ balancing model complexity and empirical risk $R^{\Delta}_{\mathcal{S}}(h) = \frac{1}{n} \sum_{i=1}^{n} \Delta(\mathbf{y}_i, h(\mathbf{x}_i))$, SSVMs solve this quadratic program (QP) (Tsochantaridis et al., 2005):

Optimization Problem 1. (Structural SVM)

$$\min_{\mathbf{w}, \xi \geq \mathbf{0}} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \quad (1)$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \quad (2)$$

Introducing a constraint for every wrong output is typically intractable. However, OP 1 can be solved by the cutting plane algorithm in Algorithm 1. This iteratively constructs a sufficient subset $\bigcup_i S_i$ of constraints and solves the QP only over this subset (line 10). The algorithm employs a separation oracle to find the next constraint to include (line 6). It finds the currently most violated constraint (or, a constraint that is violated by at least the desired precision ϵ). If a polynomial time separation oracle exists, OP 1 and Algorithm 1 have three theoretical guarantees (Tsochantaridis et al., 2005):

Polynomial Time Termination: Algorithm 1 terminates in a polynomial number of iterations, and thus overall polynomial time.

Correctness: Algorithm 1 solves OP 1 accurate to a desired precision ϵ, since Algorithm 1 terminates only when all constraints in OP 1 are respected within ϵ (lines 8 and 13).

Empirical Risk Bound: Since each ξ_i upper bounds training loss $\Delta(\mathbf{y}_i, h(\mathbf{x}_i))$, $\frac{1}{n} \sum_{i=1}^{n} \xi_i$ upper bounds empirical risk.

Unfortunately, proofs of these properties rely on the separation oracle (line 6) being exactly solvable, and do not necessarily hold with approximations. We will later analyze which properties are retained.

2.2. Markov Random Fields in SSVMs

A special case of structural SVM that we will examine throughout this paper is M^3N (Taskar et al., 2003). In this, $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is constructed from an MRF

$$f(\mathbf{x}, \mathbf{y}) = \sum_{k \in \text{cliques}(G)} \phi_k(y_{\{k\}}) \quad (3)$$

with graph structure $G = (V, E)$ and the loss function is restricted to be linearly decomposable in the cliques, i.e., $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{k \in \text{cliques}(G)} \delta_k(y_{\{k\}}, \hat{y}_{\{k\}})$. Here, $\mathbf{y}$ is the value assignment to variables, δ_k are sub-component local loss functions, and ϕ_k are potential functions representing the fitness of variable assignment $y_{\{k\}}$ to clique k. The network potential $f(\mathbf{x}, \mathbf{y})$ serves as a discriminant function representing the variable assignment $\mathbf{y}$ in the structural SVM, and $h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}, \mathbf{y})$ serves as the maximum a posteriori (MAP) prediction.

OP 1 requires we express (3) in the form $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$. First express potentials as $\phi_k(y_{\{k\}}) = \mathbf{w}^T \psi(\mathbf{x}, y_{\{k\}})$. The feature vector functions ψ_k relate $\mathbf{x}$ and label assignments $y_{\{k\}}$. Then, $f(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ where $\mathbf{\Psi}(\mathbf{x}, \mathbf{y}) = \sum_{k \in \text{cliques}(G)} \psi_k(\mathbf{x}, y_{\{k\}})$.

In the following, we use a particular linearly decomposable loss function that simply counts the percentage proportion of different labels in $\mathbf{y}$ and $\hat{\mathbf{y}}$, i.e.,

$\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \|100 \cdot \mathbf{y} - \hat{\mathbf{y}}\|_0 / |V|$. Further, in our applications, labels are binary (i.e., each $y_u \in \mathbb{B} = \{0, 1\}$), and we allow only $\phi_u(1)$ and $\phi_{uv}(1, 1)$ potentials to be non-zero. This latter restriction may seem onerous, but any pairwise binary MRF with non-zero $\phi_u(0), \phi_{uv}(0, 0), \phi_{uv}(0, 1), \phi_{uv}(1, 0)$ has an equivalent MRF where these potentials are zero.

To use Algorithm 1 for MRF training and prediction, one must solve two argmax problems:

Prediction: $\text{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$

Separation Oracle: $\text{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y})$

The prediction problem is equivalent to MAP inference. Also, we can state the separation oracle as MAP inference. Taking the MRF we would use to solve $\text{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$, we include $\Delta(\mathbf{y}_i, \mathbf{y})$ in the argmax by incrementing the node potential $\phi_u(y)$ by $\frac{100}{|V|}$ for each wrong value y of u, since each wrong variable assignment increases loss by $\frac{100}{|V|}$. Thus, we may express the separation oracle as MAP inference.

3. Approximate Inference

Unfortunately, MAP inference is $\#P$-complete for general MRFs. Fortunately, a variety of approximate inference methods exist. For prediction and the separation oracle, we explore two general classes of approximate inference methods, which we call undergenerating and overgenerating approximations.

3.1. Undergenerating Approximations

Undergenerating methods approximate $\text{argmax}_{\mathbf{y} \in \mathcal{Y}}$ by $\text{argmax}_{\mathbf{y} \in \underline{\mathcal{Y}}}$, where $\underline{\mathcal{Y}} \subseteq \mathcal{Y}$. We consider the following undergenerating methods in the context of MRFs:

Greedy iteratively changes the single variable value y_u that would increase network potential most.

LBP is loopy belief propagation (Pearl, 1988).

Combine picks the assignment $\mathbf{y}$ with the highest network potential from both greedy and LBP.

We now theoretically characterize undergenerating learning and prediction. All theorems generalize to any learning problem, not just MRFs. Due to space constraints, provided proofs are proof skeletons.

Since undergenerating approximations can be arbitrarily poor, we must restrict our consideration to a subclass of undergenerating approximations to make meaningful theoretical statements. This analysis focuses on ρ-approximation algorithms, with $\rho \in (0, 1]$. What is a ρ-approximation? In our case, for predictive inference, if $\mathbf{y}^* = \text{argmax}_{\mathbf{y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is the true optimum and $\mathbf{y}'$ the ρ-approximation output, then

$$\rho \cdot \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}^*) \leq \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}') \tag{4}$$

Similarly, for our separation oracle, for $\mathbf{y}^* = \text{argmax}_{\mathbf{y}} \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y})$ as the true optimum, and if $\mathbf{y}'$ corresponds to the constraint found by our ρ-approximation, we know

$$\rho\big[\mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}^*) + \Delta(\mathbf{y}_i, \mathbf{y}^*)\big] \leq \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}, \mathbf{y}') + \Delta(\mathbf{y}_i, \mathbf{y}') \tag{5}$$

For simplicity, this analysis supposes $\mathcal{S}$ contains exactly one training example $(\mathbf{x}_0, \mathbf{y}_0)$. To generalize, one may view n training examples as 1 example, where inference consists of n separate processes with combined outputs, etc. Combined ρ-approximation outputs may be viewed as a single ρ-approximation output.

Theorem 1. (Polynomial Time Termination) *If $\bar{R} = \max_{i, \mathbf{y} \in \mathcal{Y}} \|\Psi(\mathbf{x}_i, \mathbf{y})\|$, $\bar{\Delta} = \max_{i, \mathbf{y} \in \mathcal{Y}} \|\Delta(\mathbf{y}_i, \mathbf{y})\|$ are finite, an undergenerating learner terminates after adding at most $\epsilon^{-2}(C\bar{\Delta}^2\bar{R}^2 + n\bar{\Delta})$ constraints.*

Proof. The original proof holds as it does not depend upon separation oracle quality (Algorithm 1, l.6). □

Lemma 1. *After line 6 in Algorithm 1, let $\mathbf{w}$ be the current model, $\hat{\mathbf{y}}$ the constraint found with the ρ-approximation separation oracle, and $\hat{\xi} = H(\hat{\mathbf{y}})$ the slack associated with $\hat{\mathbf{y}}$. Then, $\mathbf{w}$ and slack $\hat{\xi} + \frac{1-\rho}{\rho}\left[\mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_0, \hat{\mathbf{y}}) + \Delta(\mathbf{y}_0, \hat{\mathbf{y}})\right]$ is feasible in OP 1.*

Proof. If we knew the true most violated constraint $\mathbf{y}^*$, we would know the minimum ξ^* such that $\mathbf{w}, \xi^*$ was feasible in OP 1. The proof upper bounds ξ^*. □

Theorem 2. *When iteration ceases with the result $\mathbf{w}, \xi$, if $\hat{\mathbf{y}}$ was the last found most violated constraint, we know that the optimum objective function value v^* for OP 1 lies in the interval*

$$\tfrac{1}{2}\|\mathbf{w}\|^2 + C\xi \leq v^* \leq \tfrac{1}{2}\|\mathbf{w}\|^2 + C\Big[\tfrac{1}{\rho}\big[\mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_0, \hat{\mathbf{y}}) + \Delta(\mathbf{y}_0, \hat{\mathbf{y}})\big] - \mathbf{w}^T \mathbf{\Psi}(\mathbf{x}_0, \mathbf{y}_0)\Big]$$

Proof. Lemma 1 applied to the last iteration. □

So, even with ρ-approximate separation oracles, one may bound how far off a final solution is from solving OP 1. Sensibly, the better the approximation, i.e., as ρ approaches 1, the tighter the solution bound.

The last result concerns empirical risk. The SVM margin attempts to ensure that high-loss outputs have a low discriminant function value, and ρ-approximations produce outputs within a certain factor of optimum.

Theorem 3. (ρ-Approximate Empirical Risk) *For $\mathbf{w}, \xi$ feasible in OP 1 from training with single example $(\mathbf{x}_0, \mathbf{y}_0)$, the empirical risk using ρ-approximate prediction has upper bound $(1-\rho)\mathbf{w}^T\mathbf{\Psi}(\mathbf{x}_0, \mathbf{y}_0) + \xi$.*

Proof. Take the $\mathbf{y}' = h(\mathbf{x}_0)$ associated constraint, then apply known bounds to its $\mathbf{w}^T\mathbf{\Psi}(\mathbf{x}_0, \mathbf{y}')$ term. □

If also using undergenerating ρ-approximate training, one may employ Theorem 2 to get a feasible ξ.

3.2. Overgenerating Approximations

Overgenerating methods approximate $\text{argmax}_{\mathbf{y}\in\mathcal{Y}}$ by $\text{argmax}_{\mathbf{y}\in\overline{\mathcal{Y}}}$, where $\overline{\mathcal{Y}} \supseteq \mathcal{Y}$. We consider the following overgenerating methods:

LProg is an expression of the inference problem as a relaxed integer linear program (Boros & Hammer, 2002). We first add $y_{uv} \in \mathbb{B}$ values indicating if $y_u = y_v = 1$ to linearize the program:

$$\max_{\mathbf{y}} \sum_{u\in\{1..|V|\}} y_u\phi_u(1) + \sum_{u,v\in\{1..|V|\}} y_{uv}\phi_{uv}(1,1) \quad (6)$$

$$\text{s.t.} \quad \forall u, v. \quad y_u \geq y_{uv} \qquad y_v \geq y_{uv} \quad (7)$$

$$y_u + y_v \leq 1 + y_{uv} \qquad y_u, y_{uv} \in \mathbb{B} \quad (8)$$

We relax $\mathbb{B}$ to $[0, 1]$ to admit fractional solutions. Importantly, there is always some optimal solution where all $y_u, y_{uv} \in \{0, \frac{1}{2}, 1\}$ (Hammer et al., 1984).

Cut is quadratic pseudo-Boolean optimization using a graph-cut (Kolmogorov & Rother, 2004). This is a different relaxation where, instead of $\mathbf{y} \in \mathbb{B}^{|V|}$, we have $\mathbf{y} \in \{0, 1, \varnothing\}^{|V|}$.

The LProg and Cut approximations share two important properties (Boros & Hammer, 2002; Hammer et al., 1984): *Equivalence* says that maximizing solutions of the Cut and LProg formulations are transmutable. One proof defines this transmutation procedure, where $\varnothing$ (in cuts optimization) and $\frac{1}{2}$ (in LP optimization) variable assignments are interchangeable (Boros & Hammer, 2002). The important practical implication of equivalence is both approximations return the same solutions. *Persistence* says unambiguous labels (i.e., not fractional or $\varnothing$) are optimal labels.

As a final detail, in the case of LProg, $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{|V|}\sum_{u\in\{1..|V|\}} |y_u - \hat{y}_u|$ and $\mathbf{\Psi}(\mathbf{x}, \mathbf{y}) = \sum_{u\in\{1..|V|\}} y_u\psi_u(1) + \sum_{u,v\in\{1..|V|\}} y_{uv}\psi_{uv}(1,1)$. Cut's functions have similar formulations.

Theorem 4. (Polynomial Time Termination) *If $\bar{R} = \max_{i,\mathbf{y}\in\overline{\mathcal{Y}}} \|\Psi(\mathbf{x}_i, \mathbf{y})\|$, $\bar{\Delta} = \max_{i,\mathbf{y}\in\overline{\mathcal{Y}}} \|\Delta(\mathbf{y}_i, \mathbf{y})\|$ are finite ($\overline{\mathcal{Y}}$ replacing $\mathcal{Y}$ in the overgenerating case), an overgenerating learner terminates after adding at most $\epsilon^{-2}(C\bar{\Delta}^2\bar{R}^2 + n\bar{\Delta})$ constraints.*

Proof. The original proof holds as an overgenerating learner is a straightforward structural learning problem on a modified output range $\overline{\mathcal{Y}}$. □

Theorem 5. (Correctness) *An overgenerating Algorithm 1 terminates with $\mathbf{w}, \xi$ feasible in OP 1.*

Proof. The learner considers a superset of outputs $\overline{\mathcal{Y}} \supseteq \mathcal{Y}$, so constraints in OP 1 are respected within ϵ. □

With these "extra" constraints from overgenerating inference, Algorithm 1's solution may be suboptimal w.r.t. the original OP 1. Further, for undergenerating methods correctness does not hold, as Algorithm 1 may not find violated constraints present in OP 1.

Theorem 6. (Empirical Risk Bound) *If prediction and the separation oracle use the same overgenerating algorithm, Algorithm 1 terminates with $\frac{1}{n}\sum_i \xi_i$ upper bounding empirical risk $R^{\Delta}_{\mathcal{S}}(h)$.*

Proof. Similar to the proof of Theorem 4.

3.3. Related Work

In prior work on discriminative training using approximate inference, structural SVMs have learned models for correlation clustering, utilizing both greedy and LP relaxed approximations (Finley & Joachims, 2005). For M^{3}Ns, Anguelov et al. (Anguelov et al., 2005) proposed to directly fold a linear relaxation into OP 1. This leads to a very large QP, and is inapplicable to other inference methods like LBP or cuts. Furthermore, we will see below that the linear-program relaxation is the slowest method. With CRFs, likelihood training requires computing the partition function in addition to MAP inference. Therefore, the partition function is approximated (Culotta et al., 2007; He et al., 2004; Kumar & Hebert, 2003; Vishwanathan et al., 2006), or the model is simplified to make the partition function tractable (Sutton & McCallum, 2005), or CRF max-likelihood training is replaced with Perceptron training (Roth & Yih, 2005).

The closest work to ours is a theoretical analysis of MRF structural learning with LBP and LP-relaxation approximations (Kulesza & Pereira, 2007). It defines the concepts *separable* (i.e., there exists $\mathbf{w}$ such that $\forall(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{S}, \mathbf{y} \in \mathcal{Y}, \mathbf{w}^T\mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T\mathbf{\Psi}(\mathbf{x}_i, \mathbf{y})$), *algorithmically separable* (i.e., there exists $\mathbf{w}$ so that empirical risk under the inference algorithm is 0), and *learnable* (i.e., the learner using the inference method finds a separating $\mathbf{w}$). The paper illustrates that using

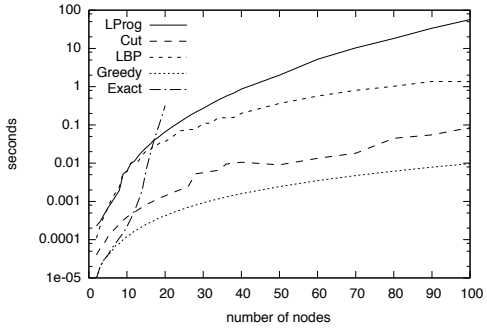


Figure 1. Runtime comparison. Average inference time for different methods on random problems of different sizes.

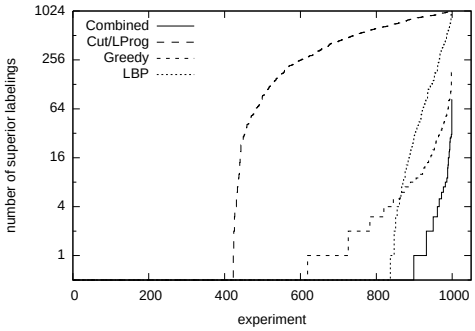


Figure 2. Quality comparison. Inference on 1000 random 18 label problems. Lower curves are better.

approximate inference, these concepts are not equivalent. Our work's major differences are our analysis handles non-zero training error, generalizes to any structural problem, uses structural SVMs, and we have an empirical analysis.

4. Experiments: Approximate Inference

Before we move into learning experiments, it helps to understand the runtime and quality performance characteristics of our MAP inference algorithms.

For runtime, Figure 1 illustrates each approximate inference method's average time to solve a single pairwise fully connected MRF with random potentials as the number of nodes increases.[1] Note that cuts are substantially faster than LBP, and several orders of magnitude faster than the linear relaxation while maintaining equivalence.

For evaluating solution quality, we generate 1000 random problems, ran the inference methods, and exhaustively count how many labelings with higher discriminant value exist. The resulting curve for 10-node MRFs is shown in Figure 2. For cut, $\varnothing$ labels are randomly assigned to 0 or 1. The lower the curve, the better the inference method. LBP finds "perfect" labelings more often than Greedy, but also tends to fall into horrible local maxima. Combined does much better than either alone; apparently the strengths of Greedy and LBP are complimentary.

Finally, note the apparent terrible performance of Cut, which is due to assigning many $\varnothing$ labels. At first glance, persistence is an attractive property since we *know* unambiguous labels are correct, but on the other hand, classifying only when it is certain leads it to leave many labels ambiguous.

[1]Implementation details: The methods were C-language Python extension modules. LProg was implemented in GLPK (see `http://www.gnu.org/software/glpk/glpk.html`). Cut was implemented with Maxflow software (Boykov & Kolmogorov, 2004). Other methods are home-spun. Experiments were run on a 2.6 GHz P4 Linux box.

5. Experiments: Approximate Learning

Our goal in the following experiments is to gain insight about how different approximate MRF inference methods perform in SSVM learning and classification. Our evaluation uses multi-label classification using pairwise fully connected MRFs as an example application.

Multi-label classification bears similarity to multi-class classification, except classes are not mutually exclusive, e.g., a news article may be about both "Iraq" and "oil." Often, incorporating inter-label dependencies into the model can improve performance (Cesa-Bianchi et al., 2006; Elisseeff & Weston, 2002).

How do we model this labeling procedure as an MRF? For each input $\mathbf{x}$, we construct an MRF with a vertex for each possible label, with values from $\mathbb{B} = \{0, 1\}$ (1 indicates $\mathbf{x}$ has the corresponding label), and an edge for each vertex pair (i.e., complete graph MRF).

What are our potential functions? In these problems, inputs $\mathbf{x} \in \mathbb{R}^m$ are feature vectors. Each of the ℓ possible labels u is associated with a weight vector $\mathbf{w}_u \in \mathbb{R}^m$. The resulting vertex potentials are $\phi_u(1) = {\mathbf{w}_u}^T\mathbf{x}$. Edge potentials $\phi_{uv}(1,1)$ come from individual values in $\mathbf{w}$, one for each label pair. Thus, the overall parameter vector $\mathbf{w} \in \mathbb{R}^{\ell m + \binom{\ell}{2}}$ has ℓm weights for the ℓ different $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_\ell$ sub-component weight vectors, and $\binom{\ell}{2}$ parameters for edge potentials. In terms of ψ functions, $\psi_u(\mathbf{x}, 1)$ vectors contain an offset version of $\mathbf{x}$ to "select out" $\mathbf{w}_u$ from $\mathbf{w}$, and $\psi_{uv}(\mathbf{x}, 1, 1)$ vectors have a single 1 entry to "select" the appropriate element from the end of $\mathbf{w}$.

5.1. Datasets and Model Training Details

We use six multi-label datasets to evaluate performance. Table 1 contains statistics on these datasets.

Table 1. Basic statistics for the datasets, including number of labels, training and test set sizes, number of features, and parameter vector **w** size, and performance on baseline trained methods and a default model.

Dataset	Labels	Train	Test	Feats.	**w** Size	Baseline	Default
Scene	6	1211	1196	294	1779	11.43±.29	18.10
Yeast	14	1500	917	103	1533	20.91±.55	25.09
Reuters	10	2916	2914	47236	472405	4.96±.09	15.80
Mediamill	10	29415	12168	120	1245	18.60±.14	25.37
Synth1	6	471	5045	6000	36015	8.99±.08	16.34
Synth2	10	1000	10000	40	445	9.80±.09	10.00

Four real datasets, **Scene** (Boutell et al., 2004), **Yeast** (Elisseeff & Weston, 2002), **Reuters** (the RCV1 subset 1 data set) (Lewis et al., 2004), and **Mediamill** (Snoek et al., 2006), came from the LIBSVM multi-label dataset collection (Chang & Lin, 2001). **Synth1** is a synthetic dataset of 6 labels. Labels follow a simple probabilistic pattern: label i is on half the time label $i-1$ is on and never otherwise, and label 1 is always on. Also, each label has 1000 related binary features (the learner does not know a priori which feature belong to each label): if i is on, a random 10 of its 1000 are set to 1. This hypothesis is learnable without edge potentials, but exploiting label dependency structure may result in better models. **Synth2** is a synthetic dataset of 10 labels. In this case, each example has exactly one label on. There are also 40 features. For an example, if label i is on, $4i$ randomly chosen features are set to 1. Only models with edge potentials can learn this concept.

We used 10-fold cross validation to choose C from 14 possible values $\{1{\cdot}10^{-2}, 3{\cdot}10^{-2}, 1{\cdot}10^{-1}, \ldots, 3{\cdot}10^{4}\}$. This C was then used when training a model on all training data. A separate C was chosen for each dataset and separation oracle.

5.2. Results and Analysis

Table 2 reports loss on the test set followed by standard error. For each dataset, we present losses for each combination of separation oracle used in learning (the rows) and of predictive inference procedure used in classification (the columns). This lets us distinguish badly learned models from bad inference procedures as explanations for inferior performance.

We also employ three additional methods as a point of comparison. Our **Baseline** is an MRF with no edge potentials, and our **Default** classifier always predicts the best-performing single labeling; results for these appear in Table 1. The **Exact** classifier is one which exhaustively searches for the argmax; to enable comparisons on Reuters and Mediamill, we pruned these datasets to the 10 most frequent labels.

Cut is omitted from Table 2. Its equivalence to LProg means the two are interchangeable and always produce the same results, excepting Cut's superior speed.

Table 3. Percentage of "ambiguous" labels in relaxed inference. Columns represent different data sets. Rows represent different methods used as separation oracles in training.

	Scene	Yeast	Reuters	Mediamill	Synth1	Synth2
Greedy	0.43%	17.02%	31.28%	20.81%	0.00%	31.17%
LBP	0.31%	0.00%	0.00%	0.00%	0.00%	0.00%
Combine	2.90%	91.42%	0.44%	4.27%	0.00%	29.11%
Exact	0.95%	84.30%	0.67%	65.58%	0.00%	27.92%
LProg	0.00%	0.43%	0.32%	1.30%	0.00%	1.48%

In all datasets, some edged model always exceeds the performance of the edgeless model. On Mediamill and Reuters, selecting only the 10 most frequent labels robs the dataset of many dependency relationships, which may explain the relatively lackluster performance.

5.2.1. The Sorry State of LBP, but Relax

Let's first examine the diagonal entries in Table 2. Models trained with LBP separation oracles yield generally poor performance. What causes this? LBP's tendency to fall into horrible local maxima (as seen in Section 4) misled Algorithm 1 to believe its most violated constraint was not violated, leading it to early termination, mirroring the result in (Kulesza & Pereira, 2007). The combined method remedies some of these problems; however, LProg still gives significantly better/worse performance on 3 vs. 1 datasets.

How does LProg training compare against exact training? Table 2 shows that both methods give similar performance. Exact-trained models significantly outperform relaxed-trained models on two datasets, but they also lose on two datasets.

5.2.2. Relaxation in Learning and Prediction

Observe that relaxation used *in prediction* performs well when applied to models trained with relaxation. However, on models trained with non-relaxed methods (i.e., models that do not constrain fractional solutions), relaxed inference often performs quite poorly. The most ludicrous examples appear in Yeast, Reuters, Mediamill, and Synth2. Table 3 suggests an explanation for this effect. The table lists the percentage of ambiguous labels from the relaxed classifier (fractional in LProg, $\varnothing$ in Cut). Ignoring degenerate LBP-trained models, the relaxed predictor *always* has the fewest ambiguous judgments. Apparently, SSVMs with relaxed separation oracles produce models that disfavor non-integer solutions. In retrospect this is unsurprising: ambiguous labels always incur loss during training. Minimizing loss during training therefore not only reduces training error, but also encourages parameterizations that favor integral (i.e., exact) solutions.

Table 2. Multi-labeling loss on six datasets. Results are grouped by dataset. Rows indicate separation oracle method. Columns indicate classification inference method.

	Greedy	LBP	Combine	Exact	LProg	Greedy	LBP	Combine	Exact	LProg
	Scene Dataset					Mediamill Dataset				
Greedy	10.67±.28	10.74±.28	10.67±.28	10.67±.28	10.67±.28	23.39±.16	25.66±.17	24.32±.17	24.92±.17	27.05±.18
LBP	10.45±.27	10.54±.27	10.45±.27	10.42±.27	10.49±.27	22.83±.16	22.83±.16	22.83±.16	22.83±.16	22.83±.16
Combine	10.72±.28	11.78±.30	10.72±.28	10.77±.28	11.20±.29	19.56±.14	20.12±.15	19.72±.14	19.82±.14	20.23±.15
Exact	10.08±.26	10.33±.27	10.08±.26	10.06±.26	10.20±.26	19.07±.14	27.23±.18	19.08±.14	18.75±.14	36.83±.21
LProg	10.55±.27	10.49±.27	10.49±.27	10.49±.27	10.49±.27	18.50±.14	18.26±.14	18.26±.14	18.21±.14	18.29±.14
	Yeast Dataset					Synth1 Dataset				
Greedy	21.62±.56	21.77±.56	21.58±.56	21.62±.56	24.42±.61	8.86±.08	8.86±.08	8.86±.08	8.86±.08	8.86±.08
LBP	24.32±.61	24.32±.61	24.32±.61	24.32±.61	24.32±.61	13.94±.12	13.94±.12	13.94±.12	13.94±.12	13.94±.12
Combine	22.33±.57	37.24±.77	22.32±.57	21.82±.56	42.72±.81	8.86±.08	8.86±.08	8.86±.08	8.86±.08	8.86±.08
Exact	23.38±.59	21.99±.57	21.06±.55	20.23±.53	45.90±.82	6.89±.06	6.86±.06	6.86±.06	6.86±.06	6.86±.06
LProg	20.47±.54	20.45±.54	20.47±.54	20.48±.54	20.49±.54	8.94±.08	8.94±.08	8.94±.08	8.94±.08	8.94±.08
	Reuters Dataset					Synth2 Dataset				
Greedy	5.32±.09	13.38±.21	5.06±.09	5.42±.09	16.98±.26	7.27±.07	27.92±.20	7.27±.07	7.28±.07	19.03±.15
LBP	15.80±.25	15.80±.25	15.80±.25	15.80±.25	15.80±.25	10.00±.09	10.00±.09	10.00±.09	10.00±.09	10.00±.09
Combine	4.90±.09	4.57±.08	4.53±.08	4.49±.08	4.55±.08	7.90±.07	26.39±.19	7.90±.07	7.90±.07	18.11±.15
Exact	6.36±.11	5.54±.10	5.67±.10	5.59±.10	5.62±.10	7.04±.07	25.71±.19	7.04±.07	7.04±.07	17.80±.15
LProg	6.73±.12	6.41±.11	6.38±.11	6.38±.11	6.38±.11	5.83±.05	6.63±.06	5.83±.05	5.83±.05	6.29±.06

Undergenerating and exact training do not control for this, leading to relaxed inference yielding many ambiguous labelings.

On the other hand, observe that models trained with the relaxed separation oracle have relatively consistent performance, irrespective of the classification inference procedure; even LBP never shows the catastrophic failure it does with other training approximations and even exact training (e.g., Mediamill, Synth2). Why might this occur? Recall the persistence property from Section 3: unambiguous labels are optimal labels. In some respects this property is attractive, but Section 4 revealed its dark side: relaxation predictors are very conservative, delivering unambiguous labels only when they are *certain*. By making things "obvious" for the relaxed predictors (which are the most conservative w.r.t. what they label), it appears they simultaneously make things obvious for all predictors, explaining the consistent performance of relaxed-trained models regardless of prediction method.

SSVM's ability to train models to "adapt" to the weakness of overgenerating predictors is an interesting complement with Searn structural learning (Daumé III et al., 2006), which trains models to adapt to the weaknesses of undergenerating search based predictors.

5.2.3. Known Approximations

How robust is SSVM training to an increasingly poor approximate separation oracle? To evaluate this, we built an artificial ρ-approximation separation oracle: for example $(\mathbf{x}_i, \mathbf{y}_i)$ we exhaustively find the optimal $\mathbf{y}^* = \text{argmax}_{\mathbf{y}\in\mathcal{Y}} w^T\mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y})$, but we return the labeling $\hat{\mathbf{y}}$ such that $f(\mathbf{x}, \hat{\mathbf{y}}) \approx \rho f(\mathbf{x}, \mathbf{y}^*)$. In this way, we build an approximate undergenerating MRF inference method with known quality.

Table 4 details these results. The first column indicates the approximation factor used in training each model for each dataset. The remaining columns show train and test performance using exact inference.

What is promising is that test performance does not drop precipitously as we use increasingly worse approximations. For most problems, the performance remains reasonable even for $\rho = 0.9$.

6. Conclusion

This paper theoretically and empirically analyzed two classes of methods for training structural SVMs on models where exact inference is intractable. Focusing on completely connected Markov random fields, we explored how greedy search, loopy belief propagation, a linear-programming relaxation, and graph-cuts can be used as approximate separation oracles in structural SVM training. In addition to a theoretical comparison of the resulting algorithms, we empirically compared performance on multi-label classification problems. Relaxation approximations distinguish themselves as preserving key theoretical properties of structural SVMs, as well as learning robust predictive models. Most significantly, structural SVMs appear to train models to avoid relaxed inference methods' tendency to yield fractional, ambiguous solutions.

Acknowledgments

This work was supported under NSF Award IIS-0713483 and through a gift from Yahoo! Inc.

References

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden Markov support vector machines. *ICML* (pp. 3–10).

Table 4. Known ρ-approximations table, showing performance change as we use increasingly inferior separation oracles.

ρ Approx.	Scene		Yeast		Reuters		Mediamill		Synth1		Synth2	
Factor	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
1.000	4.97	10.06	18.91	20.23	4.30	5.59	17.65	18.75	0.00	6.86	4.57	7.04
0.990	4.36	10.87	19.35	21.06	4.01	5.39	17.19	18.13	0.00	8.61	5.20	7.36
0.975	3.95	11.45	19.27	20.56	3.55	4.99	17.68	18.40	3.64	12.72	4.43	6.76
0.950	9.06	10.72	19.90	20.98	3.97	5.68	18.09	19.66	0.32	6.64	5.35	7.90
0.900	3.96	10.74	18.72	20.14	3.90	5.51	17.10	17.84	2.55	13.19	6.21	8.84
0.850	5.67	11.32	20.04	21.35	3.88	5.21	18.15	19.97	1.45	9.08	6.74	8.57
0.800	5.15	10.59	19.37	21.04	4.93	6.41	19.25	20.86	2.72	14.09	8.83	11.02
0.700	6.32	11.08	24.24	26.26	5.22	6.28	29.24	30.01	0.60	8.69	9.56	11.57
0.600	19.01	20.00	19.00	20.80	4.44	5.44	19.57	20.26	4.21	15.23	12.90	15.48
0.500	10.83	12.28	21.09	22.31	4.65	5.69	29.89	30.42	4.07	10.92	11.85	13.68
0.000	71.80	71.00	45.78	45.36	58.48	58.65	33.00	34.75	36.62	36.84	49.38	50.01

Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., & Ng, A. (2005). Discriminative learning of Markov random fields for segmentation of 3D scan data. *CVPR*. IEEE Computer Society.

Boros, E., & Hammer, P. L. (2002). Pseudo-boolean optimization. *Discrete Appl. Math.*, *123*, 155–225.

Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*, 1757–1771.

Boykov, Y., & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, *26*, 1124–1137.

Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Hierarchical classification: combining Bayes with SVM. *ICML*. Pittsburgh, Pennsylvanias.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM : A library for support vector machines. Software at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. *ACL-EMNLP*.

Culotta, A., Wick, M., & McCallum, A. (2007). First-order probabilistic models for coreference resolution. *NAACL-HLT* (pp. 81–88).

Daumé III, H., Langford, J., & Marcu, D. (2006). Searn in practice. Tech Report.

Elisseeff, A., & Weston, J. (2002). A kernel method for multi-labelled classification. *NIPS*.

Finley, T., & Joachims, T. (2005). Supervised clustering with support vector machines. *ICML*. Bonn, Germany.

Hammer, P. L., Hansen, P., & Simeone, B. (1984). Roof-duality, complementation, and persistency in quadratic 0–1 optimization. *Math. Program.*, *28*, 121–155.

He, X., Zemel, R. S., & Carreira-Perpinan, M. A. (2004). Multiscale conditional random fields for image labeling. *cvpr*, *02*, 695–702.

Joachims, T. (2005). A support vector method for multivariate performance measures. *ICML* (pp. 377–384). New York, NY, USA: ACM Press.

Kolmogorov, V., & Rother, C. (2004). Minimizing non-submodular functions with graph cuts – a review. *PAMI*, *26*, 147–159.

Kulesza, A., & Pereira, F. (2007). Structured learning with approximate inference. *NIPS*.

Kumar, S., & Hebert, M. (2003). Discriminative fields for modeling spatial dependencies in natural images. *NIPS*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, *5*, 361–397.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Roth, D., & Yih, W. (2005). Integer linear programming inference for conditional random fields. *Proc. of the International Conference on Machine Learning (ICML)*.

Snoek, C. G. M., Worring, M., van Gemert, J. C., Geusebroek, J.-M., & Smeulders, A. W. M. (2006). The challenge problem for automated detection of 101 semantic concepts in multimedia. *ACM-MULTIMEDIA*.

Sutton, C., & McCallum, A. (2005). *Fast, piecewise training for discriminative finite-state and parsing models* (Technical Report IR-403). Center for Intelligent Information Retrieval.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin Markov networks. In *NIPS 16*.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *JMLR*, *6*, 1453–1484.

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., & Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. *ICML*.

Yu, C.-N., Joachims, T., Elber, R., & Pillardy, J. (2007). Support vector training of protein alignment models. *RECOMB*.

An HDP-HMM for Systems with State Persistence

Emily B. Fox EBFOX@MIT.EDU
Department of EECS, Massachusetts Institute of Technology, Cambridge, MA 02139

Erik B. Sudderth SUDDERTH@EECS.BERKELEY.EDU
Department of EECS, University of California, Berkeley, CA 94720

Michael I. Jordan JORDAN@EECS.BERKELEY.EDU
Department of EECS and Department of Statistics, University of California, Berkeley, CA 94720

Alan S. Willsky WILLSKY@MIT.EDU
Department of EECS, Massachusetts Institute of Technology, Cambridge, MA 02139

Abstract

The hierarchical Dirichlet process hidden Markov model (HDP-HMM) is a flexible, nonparametric model which allows state spaces of unknown size to be learned from data. We demonstrate some limitations of the original HDP-HMM formulation (Teh et al., 2006), and propose a *sticky* extension which allows more robust learning of smoothly varying dynamics. Using DP mixtures, this formulation also allows learning of more complex, multimodal emission distributions. We further develop a sampling algorithm that employs a truncated approximation of the DP to jointly resample the full state sequence, greatly improving mixing rates. Via extensive experiments with synthetic data and the NIST speaker diarization database, we demonstrate the advantages of our sticky extension, and the utility of the HDP-HMM in real-world applications.

1. Introduction

Hidden Markov models (HMMs) have been a major success story in many applied fields; they provide core statistical inference procedures in areas as diverse as speech recognition, genomics, structural biology, machine translation, cryptanalysis and finance. Even after four decades of work on HMMs, however, significant problems remain. One lingering issue is the choice of the hidden state space's cardinality. While standard parametric model selection methods can be adapted to the HMM, there is little understanding of the strengths and weaknesses of such methods in this setting.

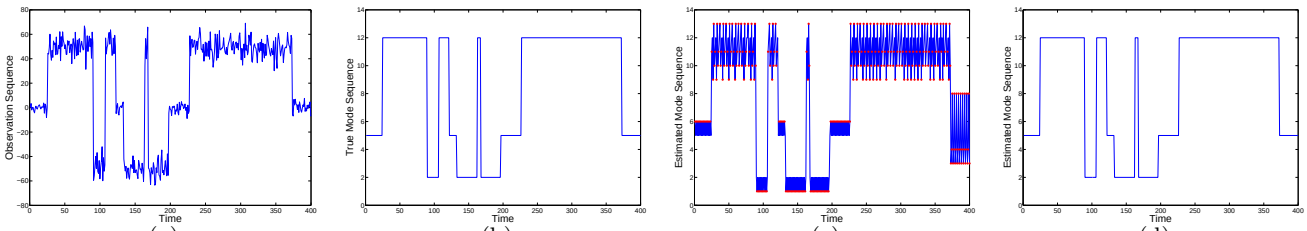
Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Recently, Teh et al. (2006) presented a nonparametric Bayesian approach to HMMs in which a stochastic process, the *hierarchical Dirichlet process* (HDP), defines a prior distribution on transition matrices over countably infinite state spaces. The resulting *HDP-HMM* leads to data–driven learning algorithms which infer posterior distributions over the number of states. This posterior uncertainty can be integrated out when making predictions, effectively averaging over models of varying complexity. The HDP-HMM has shown promise in a variety of applications, including visual scene recognition (Kivinen et al., 2007) and the modeling of genetic recombination (Xing & Sohn, 2007).

One serious limitation of the standard HDP-HMM is that it inadequately models the temporal persistence of states. This problem arises in classical finite HMMs as well, where semi-Markovian models are often proposed as solutions. However, the problem is exacerbated in the nonparametric setting, where the Bayesian bias towards simpler models is insufficient to prevent the HDP-HMM from learning models with unrealistically rapid dynamics, as demonstrated in Fig. 1.

To illustrate the seriousness of this issue, let us consider a challenging application that we revisit in Sec. 5. The problem of *speaker diarization* involves segmenting an audio recording into time intervals associated with individual speakers. This application seems like a natural fit for the HDP-HMM, as the number of true speakers is typically unknown, and may grow as more data is observed. However, this is not a setting in which model averaging is the goal; rather, it is critical to infer the number of speakers as well as the transitions among speakers. As we show in Sec. 5, the HDP-HMM's tendency to rapidly switch among redundant states leads to poor speaker diarization performance.

In contrast, the methods that we develop in this paper yield a state-of-the-art speaker diarization method, as



(a) (b) (c) (d)

Figure 1. Sensitivity of the HDP-HMM to within-state variations in the observations. (a) Observation sequence; (b) true state sequence; estimated state sequence after 100 Gibbs iterations for the (c) original and (d) sticky HDP-HMM, with errors indicated in red. Without an extra self–transition bias, the HDP-HMM rapidly transitions among redundant states.

well as a general solution to the problem of state persistence in HDP-HMMs. The approach is easily stated—we simply augment the HDP-HMM to include a parameter for self-transition bias, and place a separate prior on this parameter. The challenge is to consistently execute this idea in a nonparametric Bayesian framework. Earlier papers have also proposed self-transition parameters for HMMs with infinite state spaces (Beal et al., 2002; Xing & Sohn, 2007), but did not formulate general solutions that integrate fully with nonparametric Bayesian inference.

While the HDP-HMM treats the state transition distribution nonparametrically, it is also desirable to allow more flexible, nonparametric emission distributions. In classical applications of HMMs, finite Gaussian mixtures are often used to model multimodal observations. Dirichlet process (DP) mixtures provide an appealing alternative which avoids fixing the number of observation modes. Such emission distributions are not identifiable for the standard HDP-HMM, due to the tendency to rapidly switch between redundant states. With an additional self-transition bias, however, we show that a fully nonparametric HMM leads to effective learning algorithms. In particular, we develop a blocked Gibbs sampler which leverages forward–backward recursions to jointly resample the state and emission assignments for all observations.

In Sec. 2, we begin by presenting background material on the HDP. Sec. 3 then links these nonparametric methods with HMMs, and extends them to account for state persistence. We further augment the model with multimodal emission distributions in Sec. 4, and present results using synthetic data and the NIST speaker diarization database in Sec. 5.

2. Background: Dirichlet Processes

A Dirichlet process (DP), denoted by $\mathrm{DP}(\gamma, H)$, is a distribution over countably infinite random measures

$$G_0(\theta) = \sum_{k=1}^{\infty} \beta_k \delta(\theta - \theta_k) \qquad \theta_k \sim H \tag{1}$$

on a parameter space Θ. The weights are sampled via a *stick-breaking construction* (Sethuraman, 1994):

$$\beta_k = \beta'_k \prod_{\ell=1}^{k-1} (1 - \beta'_\ell) \qquad \beta'_k \sim \mathrm{Beta}(1, \gamma) \tag{2}$$

We denote this distribution by $\beta \sim \mathrm{GEM}(\gamma)$.

The DP is commonly used as a prior on the parameters of a mixture model of unknown complexity, resulting in a *DPMM* (see Fig. 2(a)). To generate observations, we choose $\bar{\theta}_i \sim G_0$ and $y_i \sim F(\bar{\theta}_i)$. This sampling process is often described via a discrete variable $z_i \sim \beta$ indicating which component generates $y_i \sim F(\theta_{z_i})$.

The *hierarchical Dirichlet process* (HDP) (Teh et al., 2006) extends the DP to cases in which groups of data are produced by related, yet unique, generative processes. Taking a hierarchical Bayesian approach, the HDP places a global Dirichlet process prior $\mathrm{DP}(\alpha, G_0)$ on Θ, and then draws group specific distributions $G_j \sim \mathrm{DP}(\alpha, G_0)$. Here, the base measure G_0 acts as an "average" distribution ($E[G_j] = G_0$) encoding the frequency of each shared, global parameter:

$$G_j(\theta) = \sum_{t=1}^{\infty} \tilde{\pi}_{jt} \delta(\theta - \tilde{\theta}_{jt}) \qquad \tilde{\pi}_j \sim \mathrm{GEM}(\alpha) \tag{3}$$

$$= \sum_{k=1}^{\infty} \pi_{jk} \delta(\theta - \theta_k) \qquad \pi_j \sim \mathrm{DP}(\alpha, \beta) \tag{4}$$

Because G_0 is discrete, multiple $\tilde{\theta}_{jt} \sim G_0$ may take identical values θ_k. Eq. (4) aggregates these probabilities, allowing an observation y_{ji} to be directly associated with the unique global parameters via an indicator random variable $z_{ji} \sim \pi_j$. See Fig. 2(b).

We can alternatively represent this generative process via indicator variables $t_{ji} \sim \tilde{\pi}_j$ and $k_{jt} \sim \beta$, as in Fig. 2(c). The stick-breaking priors on these mixture weights can be analytically marginalized, yielding simple forms for the predictive distributions of assignments. The resulting distribution on partitions is sometimes described using the metaphor of a *Chinese restaurant franchise* (CRF). There are J restaurants (groups), each with infinitely many tables (clusters) at

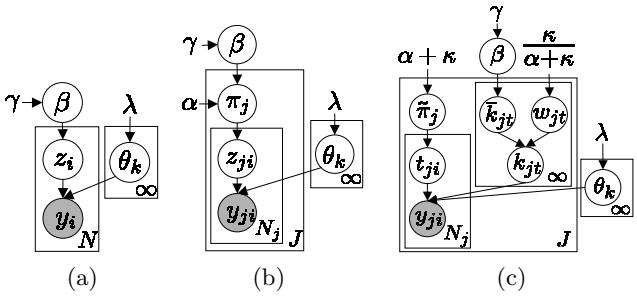


Figure 2. (a) DPMM in which $\beta \sim \text{GEM}(\gamma)$, $\theta_k \sim H(\lambda)$, $z_i \sim \beta$, and $y_i \sim f(y \mid \theta_{z_i})$. (b) HDP mixture model with $\beta \sim \text{GEM}(\gamma)$, $\pi_j \sim \text{DP}(\alpha, \beta)$, $\theta_k \sim H(\lambda)$, $z_{ji} \sim \pi_j$, and $y_{ji} \sim f(y \mid \theta_{z_{ji}})$. (c) CRF with loyal customers. Customers y_{ji} sit at table $t_{ji} \sim \tilde{\pi}_j$ which *considers* dish $\bar{k}_{jt} \sim \beta$, but override variables $w_{jt} \sim \text{Ber}(\kappa/\alpha + \kappa)$ can force the *served* dish k_{jt} to be j. The original CRF, as described in Sec. 2, has $\kappa = 0$ so that $k_{jt} = \bar{k}_{jt}$.

which customers (observations) sit. Upon entering the j^{th} restaurant, customer y_{ji} sits at currently occupied tables t_{ji} with probability proportional to the number of currently seated customers, or starts a new table $\tilde{t}$ with probability proportional to α. Each table chooses a dish (parameter) $\tilde{\theta}_{jt} = \theta_{k_{jt}}$ with probability proportional to the number of other tables in the franchise that ordered that dish, or orders a new dish $\theta_{\tilde{k}}$ with probability proportional to γ. Observation y_{ji} is then generated by global parameter $\theta_{z_{ji}} = \tilde{\theta}_{jt_{ji}} = \theta_{k_{jt_{ji}}}$.

An alternative, non–constructive characterization of samples $G_0 \sim \text{DP}(\gamma, H)$ from a Dirichlet process states that for every finite partition $\{A_1, \ldots, A_K\}$ of Θ,

$$(G_0(A_1), \ldots, G_0(A_K)) \sim \text{Dir}(\gamma H(A_1), \ldots, \gamma H(A_K)). \quad (5)$$

Using this expression, it can be shown that the following finite, hierarchical mixture model converges in distribution to the HDP as $L \to \infty$ (Ishwaran & Zarepour, 2002; Teh et al., 2006):

$$\begin{aligned} \beta &\sim \text{Dir}(\gamma/L, \ldots, \gamma/L) \\ \pi_j &\sim \text{Dir}(\alpha\beta_1, \ldots, \alpha\beta_L). \end{aligned} \quad (6)$$

Later sections use this *weak limit* approximation to develop efficient, blocked sampling algorithms.

3. The Sticky HDP-HMM

The HDP can be used to develop an HMM with an unknown, potentially infinite state space (Teh et al., 2006). For this HDP-HMM, each HDP group-specific distribution, π_j, is a state-specific transition distribution and, due to the infinite state space, there are infinitely many groups. Let z_t denote the state of the Markov chain at time t. For Markov chains $z_t \sim \pi_{z_{t-1}}$, so that z_{t-1} indexes the group to which y_t is assigned. The current HMM state z_t then indexes the parameter θ_{z_t} used to generate observation y_t (see Fig. 3).

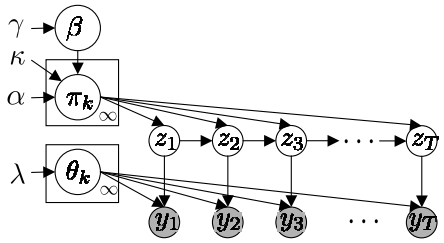


Figure 3. Graph of the sticky HDP-HMM. The state evolves as $z_{t+1} \sim \pi_{z_t}$, where $\pi_k \sim \text{DP}(\alpha + \kappa, (\alpha\beta + \kappa\delta_k)/(\alpha + \kappa))$ and $\beta \sim \text{GEM}(\gamma)$, and observations are generated as $y_t \sim F(\theta_{z_t})$. The original HDP-HMM has $\kappa = 0$.

By sampling $\pi_j \sim \text{DP}(\alpha, \beta)$, the HDP prior encourages states to have similar transition distributions ($E[\pi_{jk}] = \beta_k$). However, it does not differentiate self–transitions from moves between states. When modeling systems with state persistence, the flexible nature of the HDP-HMM prior allows for state sequences with unrealistically fast dynamics to have large posterior probability. For example, with Gaussian emissions, as in Fig. 1, a good explanation of the data is to divide an observation block into two small–variance states with slightly different means, and then rapidly switch between them (see Fig. 1). In such cases, many models with redundant states may have large posterior probability, thus impeding our ability to identify a single dynamical model which best explains the observations. The problem is compounded by the fact that once this alternating pattern has been instantiated by the sampler, its persistence is then reinforced by the properties of the Chinese restaurant franchise, thus slowing mixing rates. Furthermore, when observations are high-dimensional, this fragmentation of data into redundant states may reduce predictive performance. In many applications, one would thus like to be able to incorporate prior knowledge that slow, smoothly varying dynamics are more likely.

To address these issues, we propose to instead sample transition distributions π_j as follows:

$$\pi_j \sim \text{DP}\left(\alpha + \kappa, \frac{\alpha\beta + \kappa\delta_j}{\alpha + \kappa}\right). \quad (7)$$

Here, $(\alpha\beta + \kappa\delta_j)$ indicates that an amount $\kappa > 0$ is added to the j^{th} component of $\alpha\beta$. The measure of π_j over a finite partition $(Z_1, \ldots, Z_K)$ of the positive integers $\mathbb{Z}_+$, as described by Eq. (5), adds an amount κ only to the arbitrarily small partition containing j, corresponding to a self-transition. When $\kappa = 0$ the original HDP-HMM is recovered. Because positive κ values increase the prior probability $E[\pi_{jj}]$ of self–transitions, we refer to this extension as the *sticky* HDP-HMM.

In some ways, this κ parameter is reminiscent of the infinite HMM's self-transition bias (Beal et al., 2002).

However, that paper relied on a heuristic, approximate Gibbs sampler. The full connection between the infinite HMM and an underlying nonparametric Bayesian prior, as well as the development of a globally consistent inference algorithm, was made in Teh et al. (2006), but without a treatment of a self-transition parameter.

3.1. A CRF with Loyal Customers

We further abuse the Chinese restaurant metaphor by extending it to the sticky HDP-HMM, where our franchise now has restaurants with loyal customers. Each restaurant has a specialty dish with the same index as that of the restaurant. Although this dish is served elsewhere, it is more popular in the dish's namesake restaurant. We see this increased popularity from the fact that a table's dish is now drawn as

$$k_{jt} \sim \frac{\alpha\beta + \kappa\delta_j}{\alpha + \kappa}. \tag{8}$$

We will refer to z_t as the parent and z_{t+1} as the child. The parent enters a restaurant j determined by its parent (the grandparent), $z_{t-1} = j$. We assume there is a bijective mapping of indices $f : t \to ji$. The parent then chooses a table $t_{ji} \sim \tilde{\pi}_j$ and that table is served a dish indexed by $k_{jt_{ji}}$. Noting that $z_t = z_{ji} = k_{jt_{ji}}$, the increased popularity of the house specialty dish implies that children are more likely to eat in the same restaurant as their parent and, in turn, more likely to eat the restaurant's specialty dish. This develops family loyalty to a given restaurant in the franchise. However, if the parent chooses a dish other than the house specialty, the child will then go to the restaurant where this dish is the specialty and will in turn be more likely to eat this dish, too. One might say that for the sticky HDP-HMM, children have similar tastebuds to their parents and will always go the restaurant that prepares their parent's dish best. Often, this keeps many generations eating in the same restaurant.

The inference algorithm is simplified if we introduce a set of auxiliary random variables $\bar{k}_{jt}$ and w_{jt} as follows:

$$\bar{k}_{jt} \sim \beta, \qquad w_{jt} \sim \text{Ber}\left(\frac{\kappa}{\alpha+\kappa}\right), \qquad k_{jt} = \begin{cases} \bar{k}_{jt}, & w_{jt} = 0; \\ j, & w_{jt} = 1, \end{cases} \tag{9}$$

where $\text{Ber}(p)$ represents the Bernoulli distribution. The table first chooses a dish $\bar{k}_{jt}$ without taking the restaurant's specialty into consideration (i.e., the original CRF.) With some probability, this *considered* dish is overridden (perhaps by a waiter's suggestion) and the table is served the specialty dish j. Thus, k_{jt} represents the *served* dish. We refer to w_{jt} as the *override* variable. For the original HDP-HMM, when $\kappa = 0$, the considered dish is always the served dish since $w_{jt} = 0$ for all tables. See Fig. 2(c).

3.2. Sampling via Direct Assignments

In this section we describe a modified version of the direct assignment Rao-Blackwellized Gibbs sampler of Teh et al. (2006) which circumvents the complicated bookkeeping of the CRF by sampling indicator random variables directly. Throughout this section, we refer to the variables in the graph of Fig. 3. For this sampler, a set of auxiliary variables m_{jk}, $\bar{m}_{jk}$, and w_{jt} must be added (as illustrated in Fig. 2(c)).

Sampling z_t The posterior distribution factors as:

$$p(z_t = k \mid z_{\setminus t}, y_{1:T}, \beta, \alpha, \kappa, \lambda) \propto p(z_t = k \mid z_{\setminus t}, \beta, \alpha, \kappa) p(y_t \mid y_{\setminus t}, z_t = k, z_{\setminus t}, \lambda). \tag{10}$$

The properties of the Dirichlet process dictate that on the finite partition $\{1, \ldots, K, \tilde{k}\}$ we have the following form for the group-specific transition distributions:

$$\pi_j \sim \text{Dir}(\alpha\beta_1, \ldots, \alpha\beta_j + \kappa, \ldots, \alpha\beta_K, \alpha\beta_{\tilde{k}}). \tag{11}$$

We use the above definition of π_j and the Dirichlet distribution's conjugacy to the multinomial observations z_t to marginalize π_j and derive the following conditional distribution over the states assignments:

$$p(z_t = k \mid z_{\setminus t}, \beta, \alpha, \kappa) \propto (\alpha\beta_k + n^{-t}_{z_{t-1}k} + \kappa\delta(z_{t-1}, k)) \left(\frac{\alpha\beta_{z_{t+1}} + n^{-t}_{kz_{t+1}} + \kappa\delta(k, z_{t+1}) + \delta(z_{t-1}, k)\delta(k, z_{t+1})}{\alpha + n^{-t}_{k\cdot} + \kappa + \delta(z_{t-1}, k)} \right). \tag{12}$$

This formula is more complex than that of the standard HDP sampler due to potential dependencies in the marginalization of $\pi_{z_{t-1}}$ and π_{z_t}. For a detailed derivation, see Fox et al. (2007). The notation n_{jk} represents the number of Markov chain transitions from state j to k, $n_{j\cdot} = \sum_k n_{jk}$, and n^{-t}_{jk} the number of transitions from state j to k not counting the transition z_{t-1} to z_t or z_t to z_{t+1}. Intuitively, this expression chooses a state k with probability depending on how many times we have seen other z_{t-1} to k and k to z_{t+1} transitions. Note that there is a dependency on whether either or both of these transitions correspond to a self-transition, which is strongest when $\kappa > 0$.

As in Teh et al. (2006), by placing a conjugate prior on the parameter space, there is a closed analytic form for the likelihood component $p(y_t \mid y_{\setminus t}, z_t = k, z_{\setminus t}, \lambda)$.

Sampling β Assume there are currently $\bar{K}$ unique dishes being *considered* and take a finite partition $\{\theta_1, \theta_2, \ldots, \theta_{\bar{K}}, \theta_{\tilde{k}}\}$ of Θ, where $\theta_{\tilde{k}} = \Theta \setminus \bigcup_{k=1}^{\bar{K}} \{\theta_k\}$. Since $\tilde{\theta}_{jt} \sim G_0$ and $\bar{m}_{\cdot k}$ tables are considering dish θ_k, the properties of the Dirichlet distribution dictate:

$$p((\beta_1, \ldots, \beta_{\bar{K}}, \beta_{\tilde{k}}) \mid \bar{\boldsymbol{k}}, \gamma) \propto \text{Dir}(\bar{m}_{\cdot 1}, \ldots, \bar{m}_{\cdot \bar{K}}, \gamma). \tag{13}$$

From the above, we see that $\{\bar{m}_{\cdot k}\}_{k=1}^{\bar{K}}$ is a set of sufficient statistics for resampling β on this partition.

However, this requires sampling two additional variables, m_{jk} and w_{jt}, corresponding to the number of tables in restaurant j *served* dish k and the corresponding overwrite variables. We jointly sample from

$$p(\boldsymbol{m}, \boldsymbol{w}, \bar{\boldsymbol{m}} \mid z_{1:T}, \beta, \alpha, \kappa) = p(\bar{\boldsymbol{m}} \mid \boldsymbol{m}, \boldsymbol{w}, z_{1:T}, \beta, \alpha, \kappa) \\ p(\boldsymbol{w} \mid \boldsymbol{m}, z_{1:T}, \beta, \alpha, \kappa) p(\boldsymbol{m} \mid z_{1:T}, \beta, \alpha, \kappa). \quad (14)$$

We start by examining $p(\boldsymbol{m} \mid z_{1:T}, \beta, \alpha, \kappa)$. Having the state index assignments $z_{1:T}$ effectively partitions the data (customers) into both restaurants and dishes, though the table assignments are unknown since multiple tables can be served the same dish. Thus, sampling m_{jk} is in effect equivalent to sampling table assignments for each customer *after* knowing the dish assignment. This conditional distribution is given by:

$$p(t_{ji} = t \mid k_{jt} = k, \boldsymbol{t}^{-ji}, \boldsymbol{k}^{-jt}, y_{1:T}, \beta, \alpha, \kappa) \\ \propto \begin{cases} \tilde{n}_{jt}^{-ji}, & t \in \{1, \ldots, T_j\}; \\ \alpha\beta_k + \kappa\delta(k, j), & t = \tilde{t}_j, \end{cases} \quad (15)$$

where $\tilde{n}_{jt}^{-ji}$ is the number of customers at table t in restaurant j, not counting y_{ji}. The form of Eq. (15) implies that a customer's table assignment conditioned on a dish assignment k follows a DP with concentration parameter $\alpha\beta_k + \kappa\delta(k, j)$ and may be sampled by simulating the associated Chinese restaurant process.

We now derive the conditional distribution for the override variables w_{jt}. The table counts provide that m_{jk} tables are serving dish k in restaurant j. If $k \neq j$, we automatically have m_{jk} tables with $w_{jt} = 0$ since the served dish is not the house specialty. Otherwise,

$$p(w_{jt} \mid k_{jt} = j, \beta, \rho) \propto \begin{cases} \beta_j(1-\rho), & w_{jt} = 0; \\ \rho, & w_{jt} = 1, \end{cases} \quad (16)$$

where $\rho = \frac{\kappa}{\alpha+\kappa}$ is the prior probability that $w_{jt} = 1$. Observing served dish $k_{jt} = j$ makes it more likely that the considered dish $\bar{k}_{jt}$ was overridden than the prior suggests. We draw m_{jj} samples of w_{jt} from Eq. (16).

Given m_{jk} for all j and k and w_{jt} for each of these instantiated tables, we can now deterministically compute $\bar{m}_{jk}$. Any table that was overridden is an uninformative observation for the posterior of $\bar{m}_{jk}$ so that

$$\bar{m}_{jk} = \begin{cases} m_{jk}, & j \neq k; \\ m_{jj} - w_{j\cdot}, & j = k. \end{cases} \quad (17)$$

Sampling Hyperparameters Rather than fixing the sticky HDP-HMM's hyperparameters, we place vague gamma priors on γ and $(\alpha + \kappa)$, and a beta prior on $\kappa/(\alpha+\kappa)$. As detailed in Fox et al. (2007), the auxiliary variables introduced in the preceding section then allow tractable resampling of these hyperparameters. This allows the number of occupied states, and the degree of self–transition bias, to be strongly influenced by the statistics of observed data, as desired.

3.3. Blocked Sampling of State Sequences

The HDP-HMM direct assignment sampler can exhibit slow mixing rates since global state sequence changes are forced to occur coordinate by coordinate. This is explored in Scott (2002) for the finite HMM. Although the sticky HDP-HMM reduces the posterior uncertainty caused by fast state-switching explanations of the data, the self-transition bias can cause two continuous and temporally separated sets of observations of a given state to be grouped into two states. If this occurs, the high probability of self-transition makes it challenging for the sequential sampler to group those two examples into a single state.

A variant of the HMM forward-backward procedure (Rabiner, 1989) allows us to harness the Markov structure and jointly sample the state sequence $z_{1:T}$ given the observations $y_{1:T}$, transitions probabilities π_j, and model parameters θ_k. To take advantage of this procedure, we now must sample the previously marginalized transition distributions and model parameters. In practice, this requires approximating the theoretically countably infinite transition distributions. One approach is the degree L *weak limit approximation* to the DP (Ishwaran & Zarepour, 2002),

$$\mathrm{GEM}_L(\alpha) \triangleq \mathrm{Dir}(\alpha/L, \ldots, \alpha/L), \quad (18)$$

where L is a number that exceeds the total number of expected HMM states. This approximation encourages the learning of models with fewer than L components while allowing the generation of new components, upper bounded by L, as new data are observed.

The posterior distributions of β and π_j are given by:

$$\beta \sim \mathrm{Dir}(\gamma/L + \bar{m}_{\cdot 1}, \ldots, \gamma/L + \bar{m}_{\cdot L}) \quad (19)$$
$$\pi_j \sim \mathrm{Dir}(\alpha\beta_1 + n_{j1}, \ldots, \alpha\beta_j + \kappa + n_{jj}, \ldots, \alpha\beta_L + n_{jL}).$$

Depending on the form of the emission distribution and base measure on the parameter space Θ, we sample parameters for each of the currently instantiated states from the updated posterior distribution:

$$\theta_j \sim p(\theta \mid \{y_t \mid z_t = j\}, \lambda). \quad (20)$$

Now that we are sampling θ_j directly, we can use a non-conjugate base measure.

We block sample $z_{1:T}$ by first computing backward messages $m_{t,t-1}(z_{t-1}) \propto p(y_{t:T} | z_{t-1}, \boldsymbol{\pi}, \boldsymbol{\theta})$ and then recursively sampling each z_t conditioned on z_{t-1} from

$$p(z_t \mid z_{t-1}, y_{1:T}, \boldsymbol{\pi}, \boldsymbol{\theta}) \propto \\ p(z_t \mid \pi_{z_{t-1}}) p(y_t \mid \theta_{z_t}) m_{t+1,t}(z_t). \quad (21)$$

A similar sampler has been used for learning HDP hidden Markov trees (Kivinen et al., 2007). However, this work did not consider the complications introduced by multimodal emissions, as we explore next.

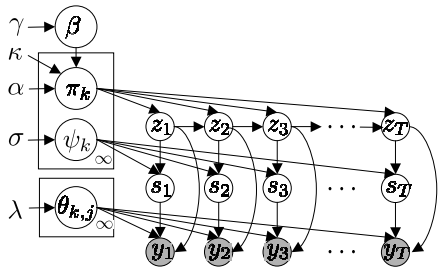


Figure 4. Sticky HDP-HMM with DP emissions, where s_t indexes the state-specific mixture component generating observation y_t. The DP prior dictates that $s_t \sim \psi_{z_t}$ for $\psi_k \sim \text{GEM}(\sigma)$. The j^{th} Gaussian component of the k^{th} mixture density is parameterized by $\theta_{k,j}$ so $y_t \sim F(\theta_{z_t,s_t})$.

4. Multimodal Emission Distributions

For many application domains, the data associated with each hidden state may have a complex, multimodal distribution. We propose to approximate such emission distributions nonparametrically, using an infinite DP mixture of Gaussians. This formulation is related to the nested DP (Rodriguez et al., 2006). The bias towards self-transitions allow us to distinguish between the underlying HDP-HMM states. If the model were free to both rapidly switch between HDP-HMM states and associate multiple Gaussians per state, there would be considerable posterior uncertainty. Thus, it is only with the sticky HDP-HMM that we can effectively learn such models.

We augment the HDP-HMM state z_t with a term s_t indexing the mixture component of the z_t^{th} emission density. For each HDP-HMM state, there is a unique stick-breaking distribution $\psi_k \sim \text{GEM}(\sigma)$ defining the mixture weights of the k^{th} emission density so that $s_t \sim \psi_{z_t}$. The observation y_t is generated by the Gaussian component with parameter θ_{z_t,s_t}. See Fig. 4.

To implement blocked resampling of $(z_{1:T}, s_{1:T})$, we use weak limit approximations to both the HDP-HMM and Dirichlet process emissions, approximated to levels L and L', respectively. The posterior distributions of β and π_k remain unchanged; that of ψ_k is given by:

$$\psi_k \sim \text{Dir}(\sigma/L' + n'_{k1}, \ldots, \sigma/L' + n'_{kL'}), \qquad (22)$$

where n'_{kl} are the number of observations assigned to the l^{th} mixture component of the k^{th} HMM state. The posterior distribution for each Gaussian's mean and covariance, $\theta_{k,j}$, is determined by the observations assigned to this component, namely,

$$\theta_{k,j} \sim p(\theta \mid \{y_t \mid (z_t = k, s_t = j)\}, \lambda). \qquad (23)$$

The augmented state (z_t, s_t) is sampled from

$$p(z_t, s_t \mid z_{t-1}, y_{1:T}, \boldsymbol{\pi}, \boldsymbol{\psi}, \boldsymbol{\theta}) \propto$$
$$p(z_t \mid \pi_{z_{t-1}}) p(s_t \mid \psi_{z_t}) p(y_t \mid \theta_{z_t,s_t}) m_{t+1,t}(z_t). \qquad (24)$$

Since the Markov structure is only on the z_t component of the augmented state, the backward message $m_{t,t-1}(z_{t-1})$ from (z_t, s_t) to (z_{t-1}, s_{t-1}) is solely a function of z_{t-1}. These messages are given by:

$$m_{t,t-1}(z_{t-1}) \propto \sum_{z_t} \sum_{s_t} p(z_t \mid \pi_{z_{t-1}}) p(s_t \mid \psi_{z_t})$$
$$p(y_t \mid \theta_{z_t,s_t}) m_{t+1,t}(z_t). \qquad (25)$$

5. Results

Synthetic Data We generated test data from a three-state Gaussian emission HMM with: 0.97 probability of self-transition; means 50, 0, and -50; and variances 50, 10, and 50 (see Fig. 1(a).) For the blocked sampler, we used a truncation level of $L = 15$.

Fig. 5 shows the clear advantage of considering a sticky HDP-HMM with blocked sampling. The Hamming distance error is calculated by greedily mapping the indices of the estimated state sequence to those maximizing overlap with the true sequence. The apparent slow convergence of the sticky HDP-HMM direct assignment sampler (Fig. 5(b)) can be attributed to the sampler splitting temporally separated segments of a true state into multiple, redundant states. Although not depicted due to space constraints, both sticky HDP-HMM samplers result in estimated models with significantly larger likelihoods of the true state sequence than those of the original HDP-HMM.

To test the model of Sec. 4, we generated data from a two-state HMM, where each state had a two-Gaussian mixture emission distribution with equally weighted components defined by means $(0, 10)$ and $(-7, 7)$, and variances of 10. The probability of self-transition was set to 0.98. The resulting observation and true state sequences are shown in Fig. 6(a) and (b).

Fig. 6(e)-(h) compares the performance of the sticky and original HDP-HMM with single and infinite Gaussian mixture emissions. All results are for the blocked sampler with truncation levels $L = L' = 15$. Intuitively, when constrained to single Gaussian emissions, the best explanation of the data is to associate each true mixture component with a separate state and then quickly switch between these states, resulting in the large Hamming distances of Fig. 6(g)-(h). Although not the desired effect in this scenario, this behavior, as depicted in Fig. 6(c), demonstrates the flexibility of the sticky HDP-HMM: if the best explanation of the data according to the model is fast state-switching, the sticky HDP-HMM still allows for this by learning a small bias towards self-transitions. The sticky HDP-HMM occasionally has more accurate state sequence estimates by grouping a true state's Gaussian mixture components into a single Gaussian with large variance. By far the best performance is

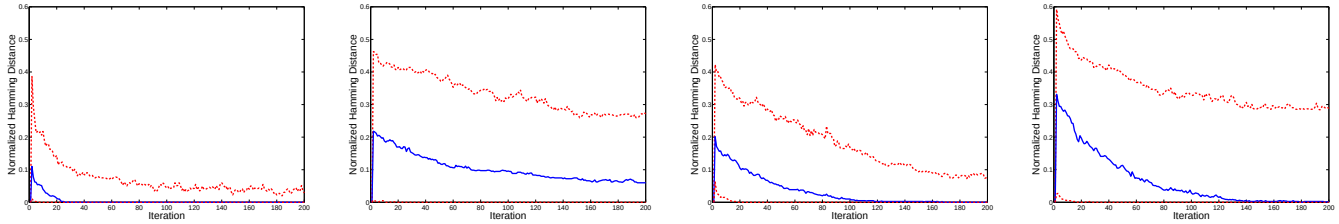


Figure 5. Hamming distance between true and estimated state sequences over 100 iterations for the sticky HDP-HMM (a) blocked and (b) direct assignment samplers and the original HDP-HMM (c) blocked and (d) direct assignment samplers. These plots show the median (solid blue) and 10^{th} and 90^{th} quantiles (dashed red) from 200 initializations.

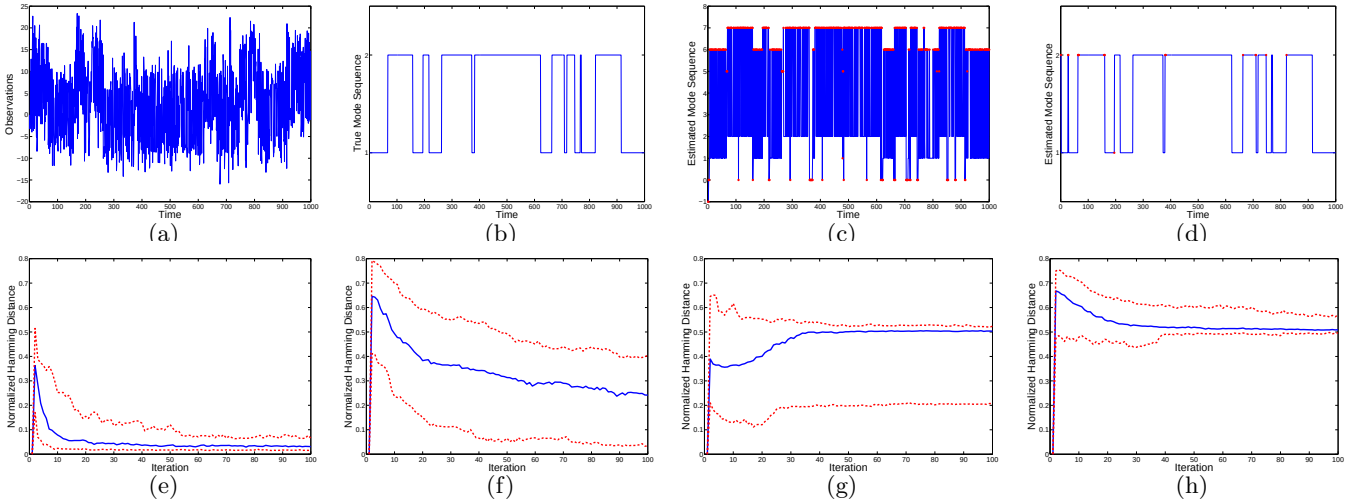
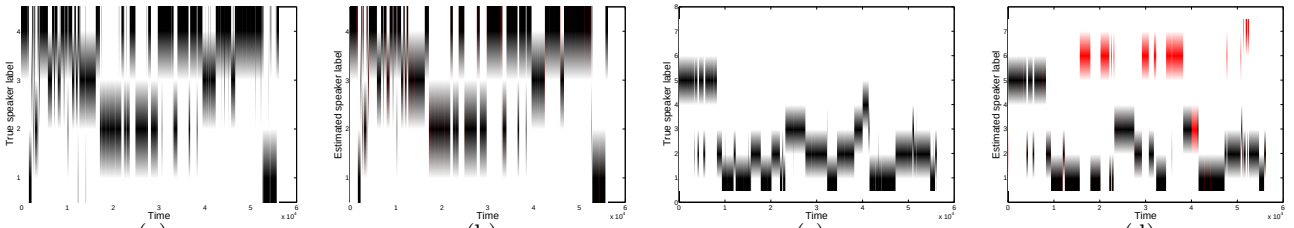


Figure 6. Performance of inference on data generated by an HMM with Gaussian mixture emissions. (a) Observation sequence; (b) true HMM state sequence; estimated HMM state sequence using the sticky HDP-HMM model with (c) single and (d) infinite Gaussian mixture emissions. Errors are indicated by red markers. The bottom row contains Hamming distance plots, as in Fig. 5, for infinite Gaussian mixture emissions and the (e) sticky HDP-HMM and (f) original HDP-HMM, and single Gaussian emissions for the (g) sticky HDP-HMM and (h) original HDP-HMM.

achieved by the sticky HDP-HMM with infinite Gaussian mixture emissions (see Fig. 6(e) and (d)); comparing to Fig. 6(f), we see that the gain can be attributed to modeling rather than just improved mixing rates.

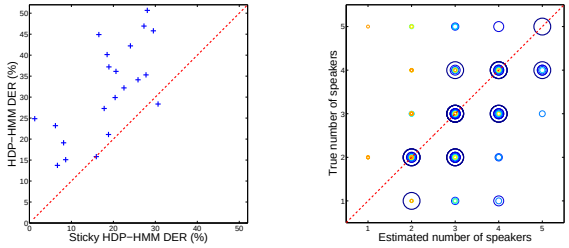
Speaker Diarization Data The *speaker diarization* task involves segmenting an audio recording into speaker-homogeneous regions, while simultaneously identifying the number of speakers. We tested the utility of the sticky HDP-HMM for this task on the data distributed by NIST as part of the Rich Transcription 2004-2007 meeting recognition evaluations (NIST, 2007). We use the first 19 Mel Frequency Cepstral Coefficients (MFCCs), computed over a 30ms window every 10ms, as our feature vector. When working with this dataset, we discovered that: (1) the high frequency content of these features contained little discriminative information, and (2) without a minimum speaker duration, the sticky HDP-HMM learned within speaker dynamics in addition to global speaker changes. To jointly address these issues, we instead model feature averages computed over 250ms, non–overlapping blocks. A minimum speaker duration of 500ms is set by associating two average features with each hidden state. We also tie the covariances of within–state mixture components. We found single–Gaussian emission distributions to be less effective.

For each of 21 meetings, we compare 10 initializations of the original and sticky HDP-HMM blocked samplers. In Fig. 8(a), we report the official NIST diarization error rate (DER) of the run with the largest observation sequence likelihood, given parameters estimated at the 1000th Gibbs iteration. The sticky HDP-HMM's temporal smoothing provides substantial performance gains. Fig 8(b) plots the estimated versus true number of speakers who talk for more than 10% of the meeting time, and shows our model's ability to adapt to a varying number of speakers. As a further comparison, the ICSI team's algorithm (Wooters & Huijbregts, 2007), by far the best performer at the 2007 competition, has an overall DER of 18.37%, simi-



(a) (b) (c) (d)

Figure 7. True state sequences for meetings (a) AMI_20041210-1052 and (c) VT_20050304-1300, with the corresponding most likely state estimates shown in (b) and (d), respectively, with incorrect labels shown in red.



(a) (b)

Figure 8. For the 21 meeting database: (a) plot of sticky vs. original HDP-HMM most likely sequence DER; and (b) plot of true vs. estimated number of speakers for samples drawn from 10 random initializations of each meeting (larger circles have higher likelihood).

lar to our 19.04%. Our best and worst DER are 1.26% and 31.42%, respectively, compared to their 4.39% and 32.23%. We use the same non-speech pre-processing, so that the differences are due to changes in the identified speakers. As depicted in Fig. 7, a significant proportion of our errors can be attributed to splitting or merging speakers. The ICSI team's algorithm uses agglomerative clustering, and requires significant tuning of parameters on representative training data. In contrast, our hyperparameters are automatically set meeting-by-meeting, so that each component's expected mean and covariance are that of the entire feature sequence. Note that the selected runs plotted in Fig. 8 are not necessarily those with the smallest DER. For example, the run depicted in Fig. 7(d) had 24.06% DER, while another run on the same meeting had 4.37% (versus ICSI's 22.00%.) There is inherent posterior uncertainty in this task, and our sampler has the advantage of giving several interpretations. When considering the best per-meeting DER for the five most likely samples, our overall DER drops to 15.14%; we hope to explore automated ways of combining multiple samples in future work. Regardless, our results demonstrate that the sticky HDP-HMM provides an elegant and empirically effective speaker diarization method.

6. Discussion

We have demonstrated the considerable benefits of an extended HDP-HMM in which a separate parameter captures state persistence. We have also shown that this sticky HDP-HMM allows a fully nonparametric treatment of multimodal emissions, disambiguated by its bias towards self-transitions, and presented efficient sampling techniques with mixing rates that improve on the state-of-the-art. Results on synthetic data, and a challenging speaker diarization task, clearly demonstrate the practical importance of our extensions.

Acknowledgments

We thank O. Vinyals, G. Friedland, and N. Morgan for helpful discussions about the NIST dataset. This research was supported in part by DARPA contract NBCHD030010, and MURIs funded through ARO Grant W911NF-06-1-0076 and AFOSR Grant FA9550-06-1-0324. E.B.F. was partially funded by an NDSEG fellowship.

References

Beal, M. J., Ghahramani, Z., & Rasmussen, C. E. (2002). The infinite hidden Markov model. *NIPS* (pp. 577–584).

Fox, E., Sudderth, E., Jordan, M., & Willsky, A. (2007). A tempered HDP-HMM for systems with state persistence. *MIT LIDS, TR #2777.*

Ishwaran, H., & Zarepour, M. (2002). Exact and approximate sum–representations for the Dirichlet process. *Can. J. Stat.*, *30*, 269–283.

Kivinen, J. J., Sudderth, E. B., & Jordan, M. I. (2007). Learning multiscale representations of natural scenes using Dirichlet processes. *ICCV* (pp. 1–8).

NIST (2007). Rich transcriptions database. *http://www.nist.gov/speech/tests/rt/.*

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, *77*, 257–286.

Rodriguez, A., Dunson, D., & Gelfand, A. (2006). The nested Dirichlet process. *Duke ISDS, TR #06-19.*

Scott, S. (2002). Bayesian methods for hidden Markov models: Recursive computing in the 21st century. *J. Amer. Stat. Assoc.*, *97*, 337–351.

Sethuraman, J. (1994). A constructive definition of Dirichlet priors. *Stat. Sinica*, *4*, 639–650.

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet processes. *J. Amer. Stat. Assoc.*, *101*, 1566–1581.

Wooters, C., & Huijbregts, M. (2007). The ICSI RT07s speaker diarization system. *To appear in LNCS.*

Xing, E., & Sohn, K.-A. (2007). Hidden Markov Dirichlet process: Modeling genetic inference in open ancestral space. *Bayes. Analysis*, *2*, 501–528.

Optimized Cutting Plane Algorithm for Support Vector Machines

Vojtěch Franc VOJTECH.FRANC@FIRST.FRAUNHOFER.DE
Soeren Sonnenburg SOEREN.SONNENBURG@FIRST.FRAUNHOFER.DE
Fraunhofer Institute FIRST, Kekulestr. 7, 12489 Berlin, Germany

Abstract

We have developed a new Linear Support Vector Machine (SVM) training algorithm called OCAS. Its computational effort scales linearly with the sample size. In an extensive empirical evaluation OCAS significantly outperforms current state of the art SVM solvers, like SVM$^{\text{light}}$, SVM$^{\text{perf}}$ and BMRM, achieving speedups of over 1,000 on some datasets over SVM$^{\text{light}}$ and 20 over SVM$^{\text{perf}}$, while obtaining the same precise Support Vector solution. OCAS even in the early optimization steps shows often faster convergence than the so far in this domain prevailing approximative methods SGD and Pegasos. Effectively parallelizing OCAS we were able to train on a dataset of size 15 million examples (itself about 32GB in size) in just 671 seconds — a competing string kernel SVM required 97,484 seconds to train on 10 million examples subsampled from this dataset.

1. Introduction

Many applications in e.g. Bioinformatics, IT-Security and Text-Classification come with *huge* amounts (e.g. millions) of data points, which are indeed *needed* to obtain state-of-the-art results. They therefore require computationally extremely efficient methods capable of dealing with ever growing data sizes. Support Vector Machines (SVM) e.g. (Cortes & Vapnik, 1995; Cristianini & Shwawe-Taylor, 2000) have proven to be powerful tools for a wide range of different data analysis problems. Given labeled training examples $\{(\boldsymbol{x}_1, y_1), \dots (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{-1, +1\})^m$ and a regularization constant $C > 0$ they learn a linear classification rule $h(\boldsymbol{x}) = \text{sgn}(\langle \boldsymbol{w}^*, \boldsymbol{x} \rangle + b^*)$ by solving the quadratic SVM primal optimization problem (P) or its dual formulation (D) allowing the use of *kernels*.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

$$(\mathbf{P}) \quad \min_{\boldsymbol{w},\boldsymbol{\xi},b} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i, \text{for } \boldsymbol{w} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}_+^m, b \in \mathbb{R}$$
$$\text{s.t.} \quad y_i\left(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b\right) \geq 1 - \xi_i, \quad i = 1, \dots, m$$

$$(\mathbf{D}) \max_{\boldsymbol{\alpha} \in \mathbb{R}^m} \quad \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$
$$\text{s.t.}: \ \sum_{i=1}^m \alpha_i y_i = 0, \ 0 \leq \alpha_i \leq \frac{C}{m}, \ i = 1 \dots m$$

Due to the central importance of SVMs, many techniques have been proposed to solve the SVM problem. As in practice only limited precision solutions to (P) and (D) can be obtained they may be categorized into *approximative* and *accurate*.

Approximative Solvers make use of heuristics (e.g. learning rate, number of iterations) to obtain (often crude) approximations to the QP-solution. They have very low per-iteration cost and low total training time. Especially for large scale problems, they are claimed to be sufficiently precise while delivering the best performance vs. training time trade-off (Bottou & Bousquet, 2008), which may be attributed to the robust nature of *large margin* SVM solutions. However while they are fast in the beginning they often fail to achieve precise solution. Among the to-date most efficient solvers are Pegasos (Shwartz et al., 2007) and SGD (Bottou & Bousquet, 2008), which are based on stochastic (sub-)gradient descent.

Accurate Solvers In contrast to approximative solvers, accurate methods solve a QP up to a given precision ε, where ε commonly denotes the violation of the relaxed KKT conditions (Joachims, 1999) or the (relative) duality gap. Accurate methods often have good asymptotic convergence properties, and thus for small ε converge to very precise solutions being limited only by numerical precision. Classical examples are off-the-shelf optimizers (e.g. MINOS, CPLEX, LOQO). However it is usually infeasible to use standard optimization tools for solving the SVM training problems (D) on datasets containing more than a few thousand examples. So-called decomposition techniques as chunking (e.g. used in (Joachims, 1999)), or SMO (used in

(Chang & Lin, 2001)) overcome this limitation by exploiting the special structure of the SVM problem. The key idea of decomposition is to freeze all but a small number of optimization variables (*working set*) and to solve a sequence of constant-size problems (subproblems of the SVM dual). While decomposition based solvers are very flexible as they are working in the dual and thus allow the use of kernels they become computationally intractable with a few hundred thousand examples. This limitation can be explained as follows: Decomposition methods exploit the fact that the optimal solution of (P) does not change if inactive constraints at the optimum are removed, they are therefore only efficient if the number of active constraints is reasonably small. Unfortunately, the number of active constraints is lower bound by the portion of misclassified examples, which is proportional to the number of examples m. Thus decomposition methods are computationally prohibitive for large-scale problems (empirically about 10%-30% of the training points become active constraints).

This poses a challenging task for even current state-of-the-art SVM solvers such as SVM$^{\text{light}}$ (Joachims, 1999), Gradient Projection-based Decomposition Technique-SVM (GPDT-SVM) (Zanni et al., 2006), LibSVM (Chang & Lin, 2001). As improving training times using the dual formulation is hard, the research focus has shifted back to the original SVM primal problem. The importance of being able to efficiently solve the primal problem for large datasets is documented by a number of very recently developed methods, e.g. SVMLin (Sindhwani & Keerthi, 2007; Chapelle, 2007), LibLinear (Lin et al., 2007), SVM$^{\text{perf}}$ (Joachims, 2006) and BMRM (Teo et al., 2007).

In the following we will focus on finding *accurate* solutions of the unconstrained linear SVM primal problem[1]

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} F(\boldsymbol{w}) := \left[\tfrac{1}{2}\|\boldsymbol{w}\|^2 + CR(\boldsymbol{w})\right], \quad (1)$$

$$\text{where} \quad R(\boldsymbol{w}) = \tfrac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle\} \quad (2)$$

is a convex risk approximating the training error.

Among the up to date most efficient *accurate* SVM primal problem (1) solvers are the Cutting Plane Algorithm (CPA) based methods put forward in (Joachims, 2006; Teo et al., 2007) and implemented in SVM$^{\text{perf}}$ and BMRM. The idea of CPAs is to approximate the risk R by a piece-wise linear function defined as the maximum over a set of linear underestimators, in CPA terminology called *cutting planes*. In (Joachims, 2006; Teo et al., 2007) it was shown that their number does not depend on the number of training examples m and that very few such cutting planes are needed in practice to sufficiently approximate (1).

[1]Note that we focus on the linear rule without a bias. The bias can be included by adding a constant feature to each training example $\boldsymbol{x}_i$.

In this work we propose a new method, called the Optimized Cutting Plane Algorithm for SVMs (OCAS). We empirically show that OCAS converges on a wide variety of large-scale datasets even considerably faster than SVM$^{\text{perf}}$, BMRM and SVM$^{\text{light}}$, achieving speedups of several orders of magnitude on some problems. We also demonstrate that OCAS even in the early optimization steps shows faster convergence than the so far in this domain dominating approximative methods. Finally we critically analyze all solvers w.r.t. classification performance in an extensive model selection study.

The report is organized as follows. CPA is described in Section 2. In Section 3, we point out a source of inefficiency of CPA and propose a new method, OCAS, to alleviate the problem and prove linear convergence. An extensive empirical evaluation is given in Section 4 and concludes the paper.

2. Cutting Plane Algorithm

Recently, the Cutting Plane Algorithm (CPA) based large-scale solvers, SVM$^{\text{perf}}$ (Joachims, 2006) and BMRM (Teo et al., 2007), have been proposed. SVM$^{\text{perf}}$ implements CPA specifically for the linear SVM problem (1). Decoupling regularizer and loss function, BMRM generalizes SVM$^{\text{perf}}$ to a wide range of losses and regularizers making it applicable to many machine learning problems, like classification, regression, structure learning etc. It should be noted that BMRM using the two norm regularizer $\|.\|_2$ and hinge loss (i.e. SVM problem (1)) coincides with SVM$^{\text{perf}}$. It was shown that SVM$^{\text{perf}}$ and BMRM by far outperform the decomposition methods like SVM$^{\text{light}}$ on large-scale problems. The rest of this section describes the idea behind CPA for the standard SVM setting (1) in more detail.

In CPA terminology, the original problem (1) is called the master problem. Using the approach of (Teo et al., 2007) one may define a reduced problem of (1) which reads

$$\boldsymbol{w}_t = \operatorname*{argmin}_{\boldsymbol{w}} F_t(\boldsymbol{w}) := \left[\frac{1}{2}\|\boldsymbol{w}\|^2 + CR_t(\boldsymbol{w})\right]. \quad (3)$$

Problem (3) is obtained from the master problem (1) by substituting a piece-wise linear approximation R_t for the risk R while leaving the regularization term unchanged, i.e. only the complex part of the objective F is approximated. The approximation R_t is derived as follows. Since the risk R is a convex function, it can be approximated at any point $\boldsymbol{w}'$ by a linear under estimator

$$R(\boldsymbol{w}) \geq R(\boldsymbol{w}') + \langle \boldsymbol{a}', \boldsymbol{w} - \boldsymbol{w}'\rangle, \qquad \forall \boldsymbol{w} \in \mathbb{R}^n, \quad (4)$$

where $\boldsymbol{a}'$ is any subgradient of R at the point $\boldsymbol{w}'$. We will use a shortcut $b' = R(\boldsymbol{w}') - \langle \boldsymbol{a}', \boldsymbol{w}'\rangle$ to abbreviate (4) as $R(\boldsymbol{w}) \geq \langle \boldsymbol{a}', \boldsymbol{w}\rangle + b'$. In CPA terminology, $\langle \boldsymbol{a}', \boldsymbol{w}\rangle + b' = 0$

is called a cutting plane. A subgradient $\boldsymbol{a}'$ of R at the point $\boldsymbol{w}'$ can be obtained as

$$\boldsymbol{a}' = -\frac{1}{m}\sum_{i=1}^{m}\pi_i y_i \boldsymbol{x}_i, \quad \pi_i = \begin{cases} 1 & \text{if} \quad y_i\langle \boldsymbol{w}', \boldsymbol{x}_i\rangle \leq 1\,, \\ 0 & \text{if} \quad y_i\langle \boldsymbol{w}', \boldsymbol{x}_i\rangle > 1\,. \end{cases} \tag{5}$$

To get a better approximation of the risk R than a single cutting plane, one may use a collection of cutting planes $\{\langle \boldsymbol{a}_i, \boldsymbol{w}\rangle + b_i = 0 \mid i = 1, \ldots, t\}$ at t distinct points $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t\}$ and take their point-wise maximum

$$R_t(\boldsymbol{w}) = \max\left\{0, \max_{i=1,\ldots,t}\left(\langle \boldsymbol{a}_i, \boldsymbol{w}\rangle + b_i\right)\right\}. \tag{6}$$

The zero cutting plane is added to the maximization as the risk R is always greater or equal to zero. It follows directly from (4) that the approximation R_t lower bounds R and thus also F_t lower bounds F.

To select the cutting planes, CPA starts from $t = 0$ (no cutting plane) and then it iterates two steps:

1. Compute $\boldsymbol{w}_t$ by solving the reduced problem (3), which can be cast as a standard QP with t variables.
2. Add a new cutting plane $(\boldsymbol{a}_{t+1}, b_{t+1})$ to approximate the risk R at the current solution $\boldsymbol{w}_t$.

A natural stopping condition for CPA is based on evaluating the ε-optimality condition $F(\boldsymbol{w}_t) - F_t(\boldsymbol{w}_t) \leq \varepsilon$ which, if satisfied, guarantees that $F(\boldsymbol{w}_t) - F(\boldsymbol{w}^*) \leq \varepsilon$ holds. [2] (Joachims, 2006) proved that for arbitrary $\varepsilon > 0$ CPA converges to the ε-optimal solution after $\mathcal{O}(\frac{1}{\varepsilon^2})$ iterations, i.e. it does not depend on the number of examples m. An improved analysis of the CPA published recently (Teo et al., 2007) shows that the number of iterations scales only with $\mathcal{O}(\frac{1}{\varepsilon})$. More important, in practice CPA usually requires only tens of iterations to reach a sufficiently precise solution.

3. Optimized Cutting Plane Algorithm for SVMs (OCAS)

We first point out a source of inefficiency appearing in CPA and then propose a new method to alleviate the problem.

CPA selects a new cutting plane such that the reduced problem objective function $F_t(\boldsymbol{w}_t)$ monotonically increases with w.r.t. the number of iterations t. However, there is no such guarantee for the master problem objective $F(\boldsymbol{w}_t)$. Even though it will ultimately converge to the minimum $F(\boldsymbol{w}^*)$, its value can heavily fluctuate between iterations. The reason for these fluctuations is the following. CPA selects at each iteration t the cutting plane which perfectly approximates the master objective F *at the current solution* $\boldsymbol{w}_t$. However, there is no guarantee that such cutting plane will be an active constraint in the vicinity of the optimum $\boldsymbol{w}^*$, nor must the new solution $\boldsymbol{w}_{t+1}$ of the reduced problem improve the master objective. In fact it often occurs that $F(\boldsymbol{w}_{t+1}) > F(\boldsymbol{w}_t)$.

To speed up the convergence of CPA, we propose a new method which we call the Optimized Cutting Plane Algorithm for SVMs (OCAS). Unlike standard CPA, OCAS aims at simultaneously optimizing the master and reduced problem's objective functions F and F_t, respectively. In addition, OCAS tries to select such cutting planes that have higher chance to actively contribute to the approximation of the master objective function F around the optimum $\boldsymbol{w}^*$. In particular, we propose the following three changes to CPA.

Change 1 We maintain the best so far solution $\boldsymbol{w}_t^{\mathrm{b}}$ obtained during the first t iterations, i.e. $F(\boldsymbol{w}_1^{\mathrm{b}}), \ldots, F(\boldsymbol{w}_t^{\mathrm{b}})$ forms a monotonically decreasing sequence.

Change 2 The new best so far solution $\boldsymbol{w}_t^{\mathrm{b}}$ is found by searching along a line starting at the previous best solution $\boldsymbol{w}_{t-1}^{\mathrm{b}}$ crossing the reduced problem's solution $\boldsymbol{w}_t$, i.e. ,

$$\boldsymbol{w}_t^{\mathrm{b}} = \min_{k\geq 0} F(\boldsymbol{w}_{t-1}^{\mathrm{b}}(1-k) + \boldsymbol{w}_t k)\,, \tag{7}$$

which can be solved exactly in $\mathcal{O}(m \log m)$ time (see Appendix A).

Change 3 The new cutting plane is selected to approximate the master objective F at a point $\boldsymbol{w}_t^{\mathrm{c}}$ which lies in a vicinity of the best so far solution $\boldsymbol{w}_t^{\mathrm{b}}$. In particular, the point $\boldsymbol{w}_t^{c}$ is computed as

$$\boldsymbol{w}_t^{c} = \boldsymbol{w}_t^{\mathrm{b}}(1-\lambda) + \boldsymbol{w}_t\lambda\,, \tag{8}$$

where $\lambda \in (0, 1]$ is a prescribed parameter. Having the point $\boldsymbol{w}_c$, the new cutting plane is computed using Equation (5) such that $F(\boldsymbol{w}_t^{\mathrm{c}}) = F_{t+1}(\boldsymbol{w}_t^{\mathrm{c}})$. Note that although the theoretical bound on the number of iterations (see Theorem 1) does not depend on λ its value has impact on the convergence speed in practice. We found that the value $\lambda = 0.1$ works consistently well in all experiments.

Algorithm 1 describes the proposed OCAS. Figure 1 shows the impact of the proposed changes to the convergence. OCAS generates a monotonically decreasing sequence of master objective values $F(\boldsymbol{w}_1^{\mathrm{b}}), \ldots, F(\boldsymbol{w}_t^{\mathrm{b}})$ and a monotonically strictly increasing sequence of reduced objective values $F_1(\boldsymbol{w}_1), \ldots, F_t(\boldsymbol{w}_t)$. Similar to CPA, a natural stopping condition for OCAS reads

$$F(\boldsymbol{w}_t^{\mathrm{b}}) - F_t(\boldsymbol{w}_t) \leq \varepsilon\,, \tag{9}$$

where $\varepsilon > 0$ is a prescribed precision parameter. Satisfying the condition (9) guarantees that $F(\boldsymbol{w}_t^{\mathrm{b}}) - F(\boldsymbol{w}^*) \leq \varepsilon$ holds.

[2] An alternative stopping condition advocated in (Joachims, 2006) halts the algorithm when $R(\boldsymbol{w}_t) - R_t(\boldsymbol{w}_t) \leq \hat{\varepsilon}$. It can be seen that both the stopping conditions become equivalent if we set $\varepsilon = C\hat{\varepsilon}$.

Algorithm 1 Optimized Cutting Plane Algorithm

1: Set $t = 0$ (i.e. there is no cutting plane at the beginning) and $\boldsymbol{w}_0^{\mathrm{b}} = \mathbf{0}$.
2: **repeat**
3: Compute $\boldsymbol{w}_t$ by solving the reduced problem (3).
4: Compute a new best so far solution $\boldsymbol{w}_t^{\mathrm{b}}$ using the line-search (7).
5: Add a new cutting plane which approximates the risk R at the point $\boldsymbol{w}_t^{\mathrm{c}}$ given by (8), i.e. ,

$$\begin{aligned} \boldsymbol{a}_{t+1} &= -\tfrac{1}{m}\textstyle\sum_{i=1}^{m} \pi_i y_i \boldsymbol{x}_i \,, \\ b_{t+1} &= R(\boldsymbol{w}_t^{\mathrm{c}}) - \langle \boldsymbol{a}_{t+1}, \boldsymbol{w}_t^{\mathrm{c}} \rangle \,, \\ \text{where} \qquad \pi_i &= \begin{cases} 1 & \text{if} \quad y_i \langle \boldsymbol{w}_t^{\mathrm{c}}, \boldsymbol{x}_i \rangle \leq 1 \,, \\ 0 & \text{if} \quad y_i \langle \boldsymbol{w}_t^{\mathrm{c}}, \boldsymbol{x}_i \rangle > 1 \,. \end{cases} \end{aligned}$$

6: $t := t + 1$
7: **until** a stopping condition is satisfied

Theorem 1 *For any $\varepsilon > 0$, $C > 0$, $\lambda \in (0, 1]$, and any training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, Algorithm 1 satisfies the stopping condition (9) after at most*

$$\max\left\{\frac{2C}{\varepsilon}, \frac{8C^3Q^2}{\varepsilon^2}\right\}, \tag{10}$$

iterations where $Q = \max_{i=1,\ldots,m} \|\boldsymbol{x}_i\|$.

Proof The proof is along the lines of the convergence analysis of the standard CPA (Joachims, 2006). First, it can be shown that violated condition (9) guarantees that adding a new cutting plane $(\boldsymbol{a}_t, b_t)$ leads to an improvement of the reduced objective $\Delta_t = F_{t+1}(\boldsymbol{w}_{t+1}) - F_t(\boldsymbol{w}_t)$ which is not less than $\min\left\{\frac{\varepsilon}{2}, \frac{\varepsilon^2}{8Q^2}\right\}$. Second, by exploiting that $0 \leq F_t(\boldsymbol{w}_t) \leq F(\boldsymbol{w}^*)$ and $F(\boldsymbol{w}^*) \leq F(\mathbf{0}) = C$ one can conclude that the sum of improvements $\sum_{i=0}^{t} \Delta_t$ cannot be greater than C. Combining these two results gives immediately the bound (10). For more details we refer to our technical report (Franc & Sonnenburg, 2007).

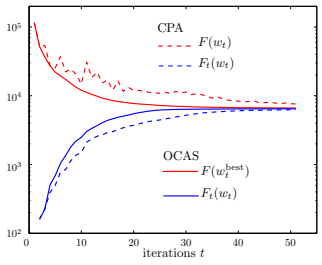


Figure 1. Convergence behaviour of the standard CPA vs. OCAS.

The bound on the maximal number of iterations of OCAS coincides with the bound for CPA given in (Joachims, 2006). Despite the same theoretical bounds, in practice OCAS converges significantly faster compared to CPA (cf. Table 2 in the experiments section).

3.1. Time Complexity and Parallelization

By Theorem 1 the number of iterations of OCAS does not depend on the number of examples m. Hence the overall time complexity is given by the effort required per iteration which is $\mathcal{O}(mn + m \log m) \approx \mathcal{O}(mn)$ (in practice $\log(m) \ll n$, where n is the dimensionality of the data). The per-iteration complexity of the subtasks and the way how they can be effectively parallelized is detailed below:

Output computation involves computation of the dot products $\langle \boldsymbol{w}_t, \boldsymbol{x}_i \rangle$, $i = 1, \ldots, m$, which requires $\mathcal{O}(s)$ time, where s equals the number of non-zero elements in the training examples. Distributing the computation equally to p processor leads to $\mathcal{O}(\frac{s}{p})$ time.

Line-search The dominant part is sorting $|K|$ numbers ($K \leq m$, see Appendix A for details) which can be done in $\mathcal{O}(|K| \log |K|)$. A speedup can be achieved by parallelizing the sorting function to using p processors, reducing complexity to $\mathcal{O}\big(\frac{|K| \log |K|}{p}\big)$. Note that our implementation of OCAS uses quicksort, whose worst case complexity is $\mathcal{O}(|K|^2)$, although its *expected* run-time is $\mathcal{O}(|K| \log |K|)$.

Cutting plane computation The dominant part requires computing the sum $-\frac{1}{m}\sum_{i=1}^{m} \pi_i y_i \boldsymbol{x}_i$ which can be done in $\mathcal{O}(s_\pi)$, where s_π is the number of non-zero elements in the training examples for which π_i is non-zero. Using p processors leads to $\mathcal{O}(\frac{s_\pi}{p})$ time.

Reduced problem The size of the reduced problem (3) is upper bound by the number of iterations which is invariant against the dataset size, hence it requires $\mathcal{O}(1)$ time. Though solving the reduced problem cannot be easily parallelized, it does not constitute the bottleneck as the number of iterations required in practice is small (cf. Table 2).

4. Experiments

We now compare current state-of-the-art SVM solvers (SGD, Pegasos, SVM$^{\text{light}}$, SVM$^{\text{perf}}$, BMRM[3] on a variety of datasets with the proposed method (OCAS) using 5 carefully crafted experiments measuring:

1. Training time and objective for optimal C
2. Speed of convergence (time vs. objective)
3. Time to perform a full model selection
4. Scalability w.r.t. dataset size
5. Effects of parallelization

To this end we implemented OCAS and the standard CPA[4] in C. We use the very general compressed sparse column

[3] SGD version 1.1 (svmsgd2) `http://leon.bottou.org/projects/sgd`, SVM$^{\text{light}}$ 6.01 and SVM$^{\text{perf}}$ 2.1 `http://svmlight.joachims.org`, pegasos `http://ttic.uchicago.edu/~shai/code/`, BMRM version 0.01 `http://users.rsise.anu.edu.au/~chteo/BMRM.html`.

[4] To not measure implementation specific effects (solver, dot-product computation) etc.

(CSC) representation to store the data. Here each element is represented by an index and a value (each 64bit). To solve the reduced problem (3), we use our implementation of improved SMO (Fan et al., 2005).

4.1. Experimental Setup

The datasets used throughout the experiments are summarized in Table 1. We augmented the Cov1, CCAT, Astro datasets from (Joachims, 2006) by the MNIST, a artificial dense and two larger bioinformatics splice datasets for worm and human. The artificial dataset was generated

Dataset	Examples	Dim	Sp	Split
MNIST	70,000	784	19	77/09/14
Astro	99,757	62,369	0.08	43/05/52
Artificial	150,000	500	100	33/33/33
Cov1	581,012	54	22	81/09/10
CCAT	804,414	47,236	0.16	87/10/03
Worm	1,026,036	804	25	80/05/15
Human	15,028,326	564	25	

Table 1. Datasets used in the experimental evaluation. Sp denotes the average number of non-zero elements of a dataset in percent. Split describes the size of the train/validation/test sets in percent. Datasets are available from the following urls: MNIST `http://yann.lecun.com/exdb/mnist/`, Cov1 `http://kdd.ics.uci.edu/databases/covertype/covertype.html`, CCAT `http://www.daviddlewis.com/resources/testcollections/rcv1/`, Worm and Human `http://www.fml.tuebingen.mpg.de/raetsch/projects/lsmkl`

from two Gaussians with different diagonal covarience matrices of multiple scale. If not otherwise stated experiments were performed on a 2.4GHz AMD Opteron Linux machine. We disabled the bias term in the comparison. As stopping conditions we use the defaults: $\varepsilon_{light} = \varepsilon_{gpdt} = 0.001$, $\varepsilon_{perf} = 0.1$ and $\varepsilon_{bmrm} = 0.001$. For OCAS we used the same stopping condition which is implemented in SVM$^{\text{perf}}$, i.e., $\frac{F(\boldsymbol{w})-F_t(\boldsymbol{w})}{C} \leq \frac{\varepsilon_{perf}}{100} = 10^{-3}$. Note that these ε have a very different meaning denoting the maximum KKT violation for SVM$^{\text{light}}$, the maximum tolerated violation of constraints for SVM$^{\text{perf}}$ and for the BMRM the relative duality gap. For SGD we fix the number of iterations to 10 and for Pegasos we use $100/\lambda$, as suggested in (Shwartz et al., 2007). For the regularization parameter C and λ we use the following relations: $\lambda = 1/C$, $C_{perf} = C/100$, $C_{bmrm} = C$ and $C_{light} = Cm$. Throughout experiments we use C as a shortcut for C_{light}.[5]

[5]The exact cmdlines are: `svm_perf_learn -l 2 -m 0 -t 0 --b 0 -e 0.1 -c` C_{perf}, `pegasos -lambda` λ `-iter` $100/\lambda$ `-k 1`, `svm_learn -m 0 -t 0 -b 0 -e 1e-3 -c` C_{light}, `bmrm-train -r 1 -m 10000 -i 999999 -e 1e-3 -c` C_{bmrm}, `svmsgd2 -lambda` λ `-epochs 10`

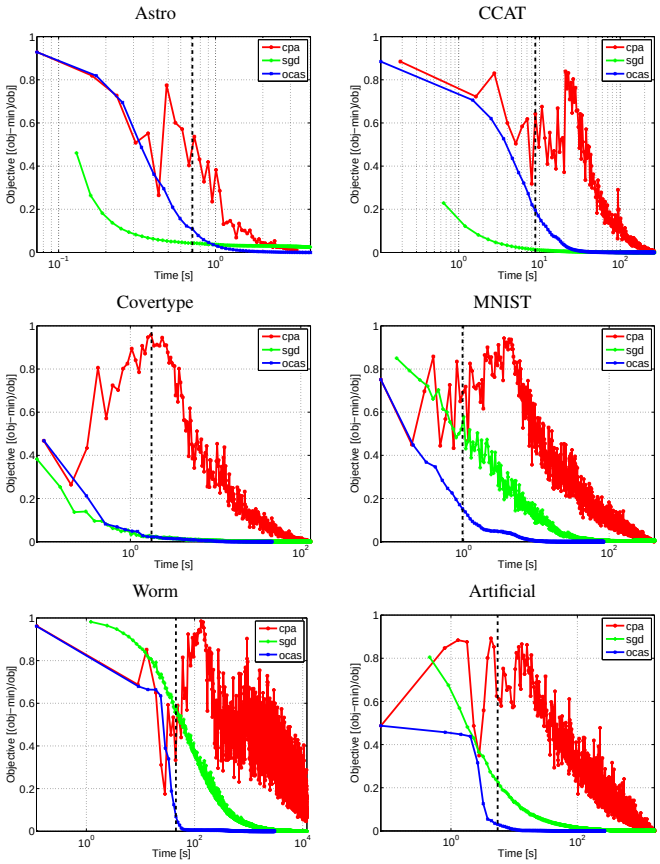


Figure 2. Objective value vs. Training time of CPA (red), SGD (green) and OCAS (blue) measured for a different number of training examples.The dashed line shows the time required to run SGD for 10 iterations. OCAS was stopped when the precision fell below 10^{-6} or the training time for CPA was achieved. In all cases OCAS achieves the minimal objective value and is even on half of the datasets already in the beginning outperforming all other methods including SGD.

4.2. Evaluation

In the following paragraphs we run and evaluate the aforementioned experiments 1–5.

Training time and objective for optimal C We trained all methods on all except the human splice dataset using the training data and measured training time (in seconds) and computed the unconstrained objective value $F(\boldsymbol{w})$

The obtained results are displayed in Table 2. The proposed method – OCAS – consistently outperforms all its competitors of the *accurate solver* category on all benchmark datasets in terms of training time while obtaining a comparable (often the best) objective value. BMRM and SVM$^{\text{perf}}$ implement the same CPA algorithm but due to implementation specific details results can be different. Our implementation of CPA gives very similar results (not shown).[6] Note that for SGD, Pegasos (and SVM$^{perf2.0}$ –

[6]In contrast to SVM$^{\text{perf}}$, BMRM and our implementation of

	astro	ccat	cov1	mnist	worm	artificial
svmlight	**2.0939e+03**	**8.1235e+04**	**2.5044e+06**	**6.7118e+05**		
	2972 22	77429 5295	1027310 41531	622391 2719	-	
svmperf	**2.1180e+03**	**8.1744e+04**	**2.5063e+06**	**6.7245e+05**	**3.2224e+04**	**1.3186e+02**
	38 2	228 228	520 152	1295 228	2029 4436	709 162
bmrm	**2.1152e+03**	**8.1682e+04**	**2.5060e+06**	**6.7250e+05**		
	42 2	327 248	678 225	2318 4327	-	
ocas	**2.1103e+03**	**8.1462e+04**	**2.5045e+06**	**6.7158e+05**	**3.1920e+04**	**1.3172e+02**
	21 1	48 25	80 10	137 10	125 258	76 13
pegasos	**2.1090e+03**	**8.1564e+04**	**2.5060e+06**	**Error**	**4.6212e+04**	**1.3120e+03**
	2689K 4	70M 127	470M 460	270M 647	82M 213	25K 1
sgd	**2.2377e+03**	**8.2963e+04**	**2.6490e+06**	**1.3254e+06**	**2.1299e+05**	**1.8097e+02**
	10 1	10 4	10 1	10 1	10 9	10 2

Table 2. Comparison of OCAS against other SVM solvers. "-" means not converged, blank not attempted. Shown in bold is the unconstrained SVM objective value Eq. (1). The two numbers below the objective value denote the number of iterations (left) and the training time in seconds (right). Lower timing and objective values mean "better." All methods solve the unbiased problem. As convergence criteria the standard settings described in Section 4.1 are used. On MNIST pegasos ran into numerical problems. OCAS clearly outperforms all of its competitors in the *accurate solver* category by a large margin achieving similar and often lowest objective value. The objective value obtained by SGD and Pegasos is often far away from the optimal solution, cf. text for a further discussion.

not shown) the objective value sometimes deviates significantly from the true objective. As a result the learned classifier may differ substantially from the optimal parameter $\boldsymbol{w}^*$. However as training times for SGD are significantly below all others it remains unclear whether SGD achieves the same precision using less time when run for further iterations. An answer to this question is given in the next paragraph.

Speed of convergence (time vs. objective) To address this problem we re-ran the best methods CPA, OCAS and SGD, recording *intermediate* progress, i.e. while optimization record time and objective for several time points. The results are shown in Figure 2. Ocas was stopped when reaching the maximum time or a precision of $1 - F(\boldsymbol{w}^*)/F(\boldsymbol{w}) \leq 10^{-6}$ and was in all cases achieving the minimum objective. In three of the six datasets OCAS not only as expected at a later time point achieves the best objective but already from the very beginning. Further analysis made clear that OCAS wins over SGD in cases where *large* C were used and thus the optimization problem is more difficult. Still plain SGD outcompetes even CPA. One may argue that practically the true objective is not the unconstrained SVM-primal value (1), but the performance on a validation set, i.e. optimization is stopped when the validation error won't change.

One should however note that one in this case does not obtain an SVM but *some classifier* instead. Then a comparison should not be limited to SVM solvers but should be open to any other large scale approach, like on-line algorithms (e.g. perceptrons) too. We argue that to compare *SVM solvers* in a fair way one needs to compare objective values. As it is still interesting to see how the methods perform w.r.t. classification performance we analyze them under this criterion in the next paragraph.

Time to perform a full model selection When using SVMs in practice, their C parameter needs to be tuned in model selection. We therefore train all methods using different settings[7] for C on the training part of all datasets, evaluate them on the validation set and choose the best model to do predictions on the test set. As performance measure we use the area under the receiver operator characteristic curve (auROC) (Fawcett, 2003). Again among the *accurate methods* OCAS outperforms its competitors by a large margin, followed by SVM$^{\text{perf}}$. Note that for all *accurate methods* the performance is very similar and has little variance. Except for the artificial dataset plain SGD is clearly fastest while achieving a similar accuracy. However the optimal parameter settings for accurate SVMs and SGD are different. Accurate SVM solvers use a larger C constant than SGD. For lower C the objective function is dominated by the regularization term $\|w\|$. A potential explanation is that SGDs update rule puts more emphasize on the regularization term and SGD when not run for a large number of iterations does imply early stopping.

Scalability w.r.t. Dataset Size In this section, we investigate how computational time of OCAS, CPA and SGD scales with the number of examples on the worm splice dataset, for sizes 100 to $1,026,036$. We again use our implementation of CPA that shares essential sub-routines with OCAS. Results are shown and discussed in Figure 3.

CPA did not converge for large C on worm even after 5000 iterations. Most likely the core solver of SVM$^{\text{perf}}$ is more robust.

[7] For Worm and Artificial we used $C = 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5,$ for CCAT, Astro, Cov1 $C = 0.1, 0.5, 1, 5, 10$ and for MNIST $C = 1, 5, 10, 50, 100$.

	astro		ccat		cov1		mnist		worm		artificial	
avg svm perf	**98.15 ± 0.00**		**98.51 ± 0.01**		**83.92 ± 0.01**		**95.86 ± 0.01**		**99.45 ± 0.00**		**86.38 ± 0.02**	
svmlight	1	152	1	124700	10	282703	10	9247	-			
svmperf	1	13	1	1750	5	781	10	887	1	22983	0.005	24520
bmrm	1	17	1	2735	10	1562	10	20278	-			
ocas	1	4	1	163	50	51	10	35	0.1	1438	0.005	6740
pegasos	**98.15**		**98.51**		**83.89**		**95.84**		**99.27**		**78.35**	
	1	59	1	2031	5	731	5	2125	5	1438	5	201
sgd	**98.13**		**98.52**		**83.88**		**95.71**		**99.43**		**80.88**	
	0.5	1	1	20	1	5	1	3	0.005	69	0.005	7

Table 3. Comparison of OCAS against other SVM solvers. "-" means not converged, blank not attempted. Shown in bold is the area under the receiver operator characteristic curve (auROC) obtained for the best model obtained via model selection over a wide range of regularization constants C. In each cell, numbers on the left denote the optimal C, numbers on the right the training time in seconds to perform the whole model selection. As there is little variance, for accurate SVM solvers only the mean and standard deviation of the auROC are shown. SGD is clearly fastest achieving similar performance for all except for the artificial dataset. However often a smaller C than the ones chosen by accurate SVMs is selected — an indication that the learned decision function is only remotely SVM-like. Among the accurate solvers OCAS clearly outperforms its competitors. It should be noted that training times for all accurate methods are dominated by training for large C (see Table 2 for training times for the optimal C). For further discussion see the text.

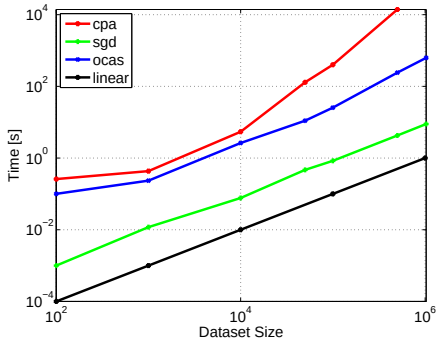


Figure 3. This figure displays how the methods scale with dataset size on the worm splice dataset. The slope of the "lines" in this figure denotes the exponent e in $\mathcal{O}(m^e)$, where the black line denotes linear effort $\mathcal{O}(m)$. Both OCAS and SGD scale about linearly. Note that SGD is much faster (as it runs for a fixed number of iterations and thus does early stopping).

Effects of Parallelization As OCAS training times are very low on the above datasets, we also apply OCAS to the 15 million human splice dataset. Using a 2.4GHz 16-Core AMD Opteron Linux machine we run OCAS using $C = 0.0001$ on 1 to 16 CPUs and show the accumulated times for each of the subtasks, the total training time and the achieved speedup w.r.t. the single CPU algorithm in Table 4. Also shown is the time accumulated for each of the threads. As can be seen — except for the line search — computations distribute nicely. Using 8 CPU cores the speedup saturates at a factor of 4.5, most likely as memory access becomes the bottleneck (for 8 CPUs output computation creates a load of 28GB/s just on memory reads).

5. Conclusions

We have developed a new Linear SVM solver called OCAS, which outperforms current state of the art SVM solvers by several orders of magnitude. OCAS even in the early optimization steps shows often faster convergence than the so far in this domain dominating approximative methods. By parallelizing the subtasks of the algorithm, OCAS gained additional speedups of factors up to 4.6 on a multi-core multiprocessor machine. Using OCAS we were able to train on a dataset of size 15 million examples (itself about 32GB in size) in just 671 seconds. As extensions to one and multi-class are straight forward, we plan to implement them in the near future. Furthermore OCAS can be extended to work with a bias term. Finally it will be future work to investigate how the kernel framework can be incorporated into OCAS and how the $\mathcal{O}(\frac{1}{\varepsilon})$ result of (Teo et al., 2007) can be applied to OCAS. An implementation of OCAS is available within the shogun toolbox `http://www.shogun-toolbox.org` and as a separate library from `http://ida.first.fraunhofer.de/~franc/ocas`.

CPUs	1	2	4	8	16
speedup	1	1.77	3.09	4.5	4.6
line search (s)	238	184	178	139	117
$\boldsymbol{a}_t$ (s)	270	155	80	49	45
output (s)	2476	1300	640	397	410
total (s)	3087	1742	998	684	671

Table 4. Speedups due to parallelizing OCAS achieved on 15 million human splice dataset.

Acknowledgements

The authors gratefully acknowledge partial support from the PASCAL Network of Excellence (EU 506778). VF was supported by Marie Curie Intra-European Fellowship grant SCOLES (MEIF-CT-2006-042107). We thank A. Zien, G. Rätsch and G. Blanchard for great discussions.

References

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. In *NIPS 20*. MIT Press.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for svms*. `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chapelle, O. (2007). Training a Support Vector Machine in the Primal. *Neural Comp.*, *19*, 1155–1178.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297.

Cristianini, N., & Shwawe-Taylor, J. (2000). *An introduction to support vector machines*. Cambridge, UK: CUP.

Fan, R.-E., Chen, P.-H., & Lin, C.-J. (2005). Working set selection using second order information for training svm. *Journal of Machine Learning Research*, *6*, 1889–1918.

Fawcett, T. (2003). *Roc graphs: Notes and practical considerations for data mining researchers*. Technical Report HPL-2003-4). HP Laboratories, Palo Alto, CA, USA.

Franc, V., & Sonnenburg, S. (2007). *Optimized cutting plane algorithm for SVMs*. Research Report; Electronic Publication 1). Fraunhofer Institute FIRST.

Joachims, T. (1999). Making large–scale SVM learning practical. *Advances in Kernel Methods — Support Vector Learning* (pp. 169–184). Cambridge, MA, USA: MIT Press.

Joachims, T. (2006). Training linear svms in linear time. *KDD'06*.

Lin, C.-J., Weng, R. C., & Keerthi, S. S. (2007). Trust region newton methods for large-scale logistic regression. *ICML '07* (pp. 561 – 568). ACM New York.

Shwartz, S.-S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. *ICML '07* (pp. 807–814). ACM Press.

Sindhwani, V., & Keerthi, S.-S. (2007). Newton methods for fast solution of semi-supervised linear svms. In *Large scale kernel machines*. MIT Press.

Teo, C. H., Le, Q., Smola, A., & Vishwanathan, S. (2007). A scalable modular convex solver for regularized risk minimization. *KDD'07*.

Zanni, L., Serafini, T., & Zanghirati, G. (2006). Parallel software for training. *JMLR*, *7*, 1467–1492.

A. Computing Line-search efficiently

The line-search (7) is an essential procedure of OCAS which is called at every iteration. We show that the line-search can be solved exactly in $\mathcal{O}(m \log m)$ time. First, we introduce a more compact notation for the objective function of the line-search problem (7) $F(\boldsymbol{w}_{t-1}^{\mathrm{b}}(1-k) + \boldsymbol{w}_t k)$ by $G(k) = g_0(k) + \sum_{i=1}^m g_i(k)$ where $g_0(k) = \frac{1}{2}k^2 A_0 + kB_0 + C_0$, $g_i(k) = \max\{0, kB_i + C_i\}$, $A_0 = \|\boldsymbol{w}_{t-1}^{\mathrm{b}} - \boldsymbol{w}_t\|^2$, $B_0 = \langle \boldsymbol{w}_{t-1}^{\mathrm{b}}, \boldsymbol{w}_t - \boldsymbol{w}_{t-1}^{\mathrm{b}} \rangle$, $C_0 = \frac{1}{2}\|\boldsymbol{w}_{t-1}^{\mathrm{b}}\|^2$, $B_i = \frac{C}{m} y_i \langle \boldsymbol{x}_i, \boldsymbol{w}_{t-1}^{\mathrm{b}} - \boldsymbol{w}_t \rangle$ and $C_i = \frac{C}{m}(1 - y_i \langle \boldsymbol{x}_i, \boldsymbol{w}_{t-1}^{\mathrm{b}} \rangle)$. Hence the line-search (7) involves solving $k^* = \operatorname{argmin}_{k \geq 0} G(k)$ and computing $\boldsymbol{w}_t^{\mathrm{b}} = \boldsymbol{w}_{t-1}^{\mathrm{b}}(1 - k^*) + \boldsymbol{w}_t k^*$. As function G is convex the unconstrained minimum of G is attained at the point k^* at which the subdifferential $\partial G(k)$ contains zero, i.e. $0 \in \partial G(k^*)$. The subdifferential of G is $\partial G(k) = kA_0 + B_0 + \sum_{i=1}^m \partial g_i(k)$,

$$\partial g_i(k) = \begin{cases} 0 & \text{if} \quad kB_i + C_i < 0\,, \\ B_i & \text{if} \quad kB_i + C_i > 0\,, \\ [0, B_i] & \text{if} \quad kB_i + C_i = 0\,. \end{cases}$$

Note that the subdifferential is not a function as there ex-

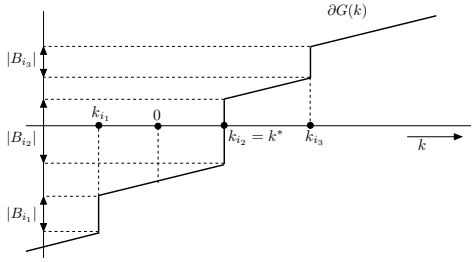


Figure 4. Illustration of the subgradient $\partial G(k)$ of the objective function $G(k)$ minimized in the line-search.

ist k for which $\partial G(k)$ is an interval. The first term of the subdifferential $\partial G(k)$ is an ascending linear function $kA_0 + B_0$ since A_0 must be greater than zero (A_0 is zero only if the algorithm has converged but then the line-search is not invoked). The term $\partial g_i(k)$ is either constantly zero, if $B_i = 0$, or it is a step-like jump whose value changes at the point $k_i = -\frac{C_i}{B_i}$. The value of $\partial g_i(k)$ w.r.t. k is summarized in Table 5. Hence the subdifferential $\partial G(k)$ is a

	$k < k_i$	$k = k_i$	$k > k_i$
$B_i = 0$	0	0	0
$B_i < 0$	B_i	$[B_i, 0]$	0
$B_i > 0$	0	$[0, B_i]$	B_i

Table 5. The value of $\partial g_i(k)$ with respect to k.

monotonically increasing function as is illustrated in Figure 4. To solve $k^* = \operatorname{argmin}_{k \geq 0} G(k)$ we proceed as follows: If $\max(\partial G(0))$ is strictly greater than zero then the unconstrained minimum $\operatorname{argmin}_k G(k)$ is at a point less or equal to 0. Thus the constrained minimum is attained at the point $k^* = 0$.

If $\max(\partial G(0))$ is less then zero then the optimum k^* corresponds to the unconstrained optimum $\operatorname{argmin}_k G(k)$ attained at the intersection between the graph of $\partial G(k)$ and the x-axis. This point can be found efficiently by sorting $K = \{k_i \mid k_i > 0, i = 1, \ldots, m\}$ and checking the condition $0 \in G(k)$ for $k \in K$ and for k in the intervals which split the domain $(0, \infty)$ in the points K. These computation are dominated by sorting the numbers K which takes $\mathcal{O}(|K| \log |K|)$.

Stopping Conditions for Exact Computation of Leave-One-Out Error in Support Vector Machines

Vojtěch Franc[1] VOJTECH.FRANC@FIRST.FRAUNHOFER.DE
Pavel Laskov[1,2] PAVEL.LASKOV@FIRST.FRAUNHOFER.DE
Klaus-Robert Müller[1,3] KLAUS@FIRST.FRAUNHOFER.DE

[1]Fraunhofer Institute FIRST, Kekulestr. 7, 12489 Berlin, Germany
[2]University of Tuebingen, Wilhelm Schickard Instutute for Computer Science, Sand 13, 72070 Tuebingen, Germany
[3]Technical University of Berlin, Dept. of Computer Science, Franklinstr. 28/29, 10587 Berlin, Germany

Abstract

We propose a new stopping condition for a Support Vector Machine (SVM) solver which precisely reflects the objective of the Leave-One-Out error computation. The stopping condition guarantees that the output on an intermediate SVM solution is identical to the output of the optimal SVM solution with one data point excluded from the training set. A simple augmentation of a general SVM training algorithm allows one to use a stopping criterion equivalent to the proposed sufficient condition. A comprehensive experimental evaluation of our method shows consistent speedup of the exact LOO computation by our method, up to the factor of 13 for the linear kernel. The new algorithm can be seen as an example of constructive guidance of an optimization algorithm towards achieving the best attainable expected risk at optimal computational cost.

1. Introduction

The interrelation between a computational complexity and a generalization ability of learning algorithms has seldom been considered in machine learning. Since the solutions to a majority of learning problems are obtained by iterative optimization algorithms, solution accuracy plays an important role in the estimation of expected risk (Bartlett & Mendelson, 2006). In practice, the available computational resources necessitate a tradeoff between approximation accuracy determined by the choice of a class of functions, estimation error determined by a finite set of examples, and an optimization error determined by the accuracy of a solver attainable within a given time budget (Bottou & Bousquet, 2008).

The asymptotic analysis in (Bottou & Bousquet, 2008) provides upper bounds on the time required to reach a certain expected risk by a given algorithm. From the practical point of view, it is desirable to have *constructive* influence over a learning algorithms, by choosing its parameters, such as e.g. learning rate or stopping conditions, to reach the best attainable expected risk. The present contribution provides an example of such a constructive mechanism by developing optimal stopping conditions for SVM training using a particular estimator of an expected risk – the leave-one-out (LOO) error. Although exact computation of a LOO error is hardly used for large-scale learning due to its computational burden, our method is feasible for "small-scale" learning with "expensive" data (e.g. in bioinformatics or finance), especially when accurate estimation of expected risk is required.

The LOO is known to provide an unbiased estimator of the generalization error (Lunts & Brailovskiy, 1967). The naive computation of the LOO error, i.e. by explicit relearning after exclusion of each single example, is in all but the simplest cases impractical. The problem of speeding up a computation of a LOO error has received significant attention. The following approaches exist:

- LOO bounds provide an estimate of the LOO error given an optimal solution of the SVM training problem ((Joachims, 2000; Vapnik & Chapelle, 2000; Jaakkola & Haussler, 1999; Zhang, 2001)). These bounds are computationally efficient but imprecise. In practice, if an accurate estimate of the classification accuracy is needed, exact computation of the LOO error is unavoidable.
- Incremental SVM (Cauwenberghs & Poggio, 2000) allows one to exactly determine for each candidate

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

training point – *after* obtaining the optimal SVM solution – whether or not it will be a LOO error. This approach avoids explicit re-training, but incremental unlearning of points is complicated and requires special organization of matrix operations (Laskov et al., 2007).

- Loose stopping conditions based on the ε-KKT allow one to speed up the LOO computation *before* obtaining an optimal solution. Such methods, (e.g. (Lee & Lin, 2000; Martin et al., 2004)) use fairly simple heuristics, but lack theoretical justification that would connect the ε to a precision of the LOO computation. As it is illustrated in the examples in Section 2, these methods can also be imprecise.

In this contribution we propose a new stopping condition for an SVM solver which *precisely reflects* the objective of the LOO error computation. Our main result, given in Theorem 1, provides a sufficient condition for which the output on an intermediate SVM solution is identical to the output of the optimal SVM solution with one data point excluded from the training set. Although this sufficient condition cannot be computed in practice, we propose a simple augmentation of a general SVM training algorithm which allows one to use a stopping criterion equivalent to the proposed sufficient condition.

2. Leave-One-Out Error Estimate For Support Vector Machines Classifier

Let $\mathcal{X}$ be a set of inputs and $\mathcal{Y} = \{-1, +1\}$ a set of labels of an analyzed object. Let further $\mathcal{T}_{XY} = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (\mathcal{X} \times \mathcal{Y})^m$ be a finite training set i.i.d. sampled from unknown $P(x, y)$. The goal is to learn a classifier $f\colon \mathcal{X} \to \mathcal{Y}$ minimizing the probability of misclassification $R[f] = \int V(y, f(x))dP(x, y)$ where $V(y, y') = 1$ for $y \neq y'$ and $V(y, y') = 0$ otherwise.

The SVMs represent the input states in the Reproducing Kernel Hilbert Space (RKHS) via a map $\mathbf{\Phi}\colon \mathcal{X} \to \mathcal{H}$ which is implicitly defined by a kernel function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (Schölkopf & Smola, 2002). The classifier is assumed to be linear, i.e., $f(x; \boldsymbol{w}, b) = \langle \boldsymbol{w}, \mathbf{\Phi}(x) \rangle + b$, where $\boldsymbol{w} \in \mathcal{H}$, $b \in \mathbb{R}$ are unknown parameters and $\langle \cdot, \cdot \rangle$ denotes an inner product in RKHS. Because $R[f]$ cannot be minimized directly due to the unknown $P(x, y)$, the SVMs replace $R[f]$ by a regularized risk its minimization leads to

$$(\boldsymbol{w}^*, b^*) = \operatorname*{argmin}_{\boldsymbol{w} \in \mathcal{H}, b \in \mathbb{R}} \left(\frac{1}{2} \|\boldsymbol{w}\|^2_{\mathcal{H}} + C \sum_{i \in \mathcal{I}} \hat{V}(y_i, f(x_i; \boldsymbol{w}, b)) \right) \tag{1}$$

where $C \in \mathbb{R}^+$ is a regularization constant, $\hat{V}(y_i, f(x_i; \boldsymbol{w},)) = \max(0, 1 - y_i f(x_i; \boldsymbol{w}, b))$ is a convex piece-wise linear approximation of $V(y_i, f(x_i; \boldsymbol{w}, b))$ and $\mathcal{I} = \{1, \ldots, m\}$. By the Representer theorem (Schölkopf & Smola, 2002), the optimal SVM classifier $f(x; \boldsymbol{w}^*, b^*)$ can be expressed in the form

$$f(x; \boldsymbol{\alpha}, b) = \begin{cases} +1 & \text{if} \quad \sum\limits_{i \in \mathcal{I}} \alpha_i y_i k(x, x_i) + b \geq 0 \,, \\ -1 & \text{if} \quad \sum\limits_{i \in \mathcal{I}} \alpha_i y_i k(x, x_i) + b < 0 \,, \end{cases} \tag{2}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^T \in \mathbb{R}^m$, $b \in \mathbb{R}$. Substituting (2) to (1) allows to find the optimal SVM classifier by solving a convex QP task

$$(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*) = \operatorname*{argmin}_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \in \mathcal{A}} F(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \tag{3}$$

where the convex objective function reads

$$F(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) = \frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + C \sum_{i \in \mathcal{I}} \xi_i \,,$$

and the convex feasible set $\mathcal{A}$ contains all $(\boldsymbol{\alpha} \in \mathbb{R}^m, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}^m)$ satisfying

$$\begin{aligned} y_i \left(\sum_{j \in \mathcal{I}} \alpha_j y_j k(x_i, x_j) + b \right) &\geq 1 - \xi_i \,, \quad i \in \mathcal{I} \,, \\ \xi_i &\geq 0 \,, \quad i \in \mathcal{I} \,. \end{aligned}$$

Minimizing the regularized risk (1) (or QP task (3) respectively) allows to find parameters $(\boldsymbol{\alpha}, b)$ of the SVM classifier provided the hyper-parameters, i.e., the kernel function k and the regularization constant C, are known. This is not the case in practice and the hyper-parameters (C, k) must be optimized as well. A common approach is to select the best (C, k) from a given finite set Θ by minimizing some performance measure. The set Θ is usually created by reasonably discretizing the hyper-parameters space. As the performance measure, the LOO error $R_{LOO}[f]$ is a common choice.

Let $(\boldsymbol{\alpha}^*(r), b^*(r), \boldsymbol{\xi}^*(r))$ denote the optimal solution of the primal QP task (3) with r-th example removed from the training set which is equivalent to solving the task

$$(\boldsymbol{\alpha}^*(r), b^*(r), \boldsymbol{\xi}^*(r)) = \operatorname*{argmin}_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \in \mathcal{A}(r)} F(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \,, \tag{4}$$

where $\mathcal{A}(r) = \mathcal{A} \cap \{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \mid \alpha_r = 0\}$, i.e., $\mathcal{A}(r)$ denotes the original feasible set $\mathcal{A}$ enriched by an additional constraint $\alpha_r = 0$. The LOO error estimator is defined as

$$R_{LOO}[f(\cdot; \boldsymbol{\alpha}^*, b^*)] = \frac{1}{m} \sum_{r \in \mathcal{I}} V(y_r, f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))) \,. \tag{5}$$

The major practical disadvantage of the LOO error is its high computational cost. A naive approach to compute LOO requires solving m different QP tasks (4). In some

cases, however, the value $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ can be immediately derived from the optimal solution $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$ computed from the entire training set. Table 1 summarizes the known sufficiency checks; the implication 1 is generally known and the implications 2, 3 are due to (Joachims, 2000).

1. If $\alpha_r^* = 0$ then $y_r = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$.
2. If $y_r \neq f(x_r; \boldsymbol{\alpha}^*, b^*)$ then $y_r \neq f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$.
3. Let R^2 be an upper bound on $k(x,x) - k(x,x')$, $\forall x, x' \in \mathcal{X}$, and let $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$ be a stable solution which means that there exist at least one $i \in \mathcal{I}$ such that $0 < \alpha_i^* < C$. In this case, if $2\alpha_r^* R^2 + \xi_r^* < 1$ then $y_r = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$.

Table 1. Sufficiency checks for computing $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ directly from $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$.

A portion of the training examples for which the sufficiency checks apply depends on the problem at hand (for empirical study see (Martin et al., 2004)). (Joachims, 2000) proposed using the sufficiency checks to compute an upper bound on the LOO error called $\xi\alpha$-estimator. It has been empirically shown, that in general the $\xi\alpha$-estimator is not sufficiently precise for the hyper-parameter tuning (Duan et al., 2003). Algorithm 1 shows a standard procedure of computing the LOO error exactly with the use of the sufficiency checks to reduced the number of cases when the solution of the QP task (4) is required.

Algorithm 1 Computation of the LOO Error

1: Solve the QP task (3) to obtain $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$.
2: Apply the sufficiency checks from Table 1 to compute $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ from $(\boldsymbol{\alpha}^*, b^*, \boldsymbol{\xi}^*)$.
3: For examples unresolved in Step 2 solve the QP task (4) to obtain $(\boldsymbol{\alpha}^*(r), b^*(r))$.
4: Compute the LOO error by (5) using $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$, $r \in \mathcal{I}$ obtained in Steps 2 and 3.

Algorithm 1 requires the use of an optimization method solving the QP tasks (3) and (4) exactly, i.e., producing the optimal solutions. Although such optimization methods exist, they are applicable only for very small problems. In practice, the QP tasks are solved only approximately via their dual representation which is more suitable for optimization due to a simpler feasible set. In particular, the minimizer $\boldsymbol{\alpha}^*$ of the primal QP task (3) can be equivalently computed by solving the dual QP task

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathcal{B}} \left(\sum_{i \in \mathcal{I}} \alpha_i - \frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right), \tag{6}$$

where $\mathcal{B}$ is a convex feasible set which contains all $\boldsymbol{\alpha} \in \mathbb{R}^m$ satisfying

$$\sum_{i \in \mathcal{I}} \alpha_i y_i = 0\,, \quad \text{and} \quad 0 \leq \alpha_i \leq C\,, i \in \mathcal{I}\,.$$

We will use $G(\boldsymbol{\alpha})$ to denote the objective function of (6). Having $\boldsymbol{\alpha}^*$ computed, the remaining primal variables $(b^*, \boldsymbol{\xi}^*)$ can be obtained easily from the Karush-Kuhn-Tucker (KKT) optimality conditions (e.g., (Boyd & Vandenberghe, 2004)). Similarly, the minimizer $\boldsymbol{\alpha}^*(r)$ of the QP task (4) is obtained by solving the dual

$$\boldsymbol{\alpha}^*(r) = \operatorname*{argmax}_{\boldsymbol{\alpha} \in \mathcal{B}(r)} G(\boldsymbol{\alpha})\,, \tag{7}$$

where $\mathcal{B}(r) = \mathcal{B} \cap \{\boldsymbol{\alpha} \mid \alpha_r = 0\}$ and the primal variables $(b^*(r), \boldsymbol{\xi}^*(r))$ can be again obtained by the KKT conditions. From the optimization point of view, the QP tasks (6) and (7) are equivalent since the latter can be converted to the former simply by excluding the r-th variable. Thus we now concentrate only on the optimization of the QP task (6).

Algorithm 2 Commonly used iterative QP solver

1: Initialize $t := 0$ and $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}$.
2: $t := t + 1$.
3: Update $\boldsymbol{\alpha}^{(t-1)} \rightarrow \boldsymbol{\alpha}^{(t)}$, i.e., find $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}$ such that $G(\boldsymbol{\alpha}^{(t-1)}) < G(\boldsymbol{\alpha}^{(t)})$.
4: If $\boldsymbol{\alpha}^{(t)}$ satisfies the ε-KKT conditions (8) halt otherwise go to 2.

A framework of a commonly used QP solver optimizing (6) describes Algorithm 2. Among the most popular methods which fit to the framework of Algorithm 2 belong the Sequential Minimal Optimizer (SMO) (Platt, 1998), SVM ^{light} (Joachims, 1998) and other decomposition methods (e.g. (Vapnik, 1995; Osuna et al., 1997)). All these solvers iteratively increase the dual criterion $G(\boldsymbol{\alpha})$ until the solution satisfies stopping conditions. A relaxed version of the KKT optimality conditions is the most frequently used stopping criterion: let $\varepsilon \geq 0$ be a prescribed number and $\nabla_i(\boldsymbol{\alpha}) = 1 - y_i \sum_{j \in \mathcal{I}} \alpha_j y_j k(x_i, x_j)$; then a vector $\boldsymbol{\alpha} \in \mathcal{B}$ satisfies the relaxed KKT conditions (e.g. (Keerthi et al., 2001)) if there exist $b \in \mathbb{R}$ such that

$$\begin{array}{rcll} \nabla_i(\boldsymbol{\alpha}) + by_i & \leq & \varepsilon\,, & \text{if} \quad \alpha_i = 0\,, \\ -\nabla_i(\boldsymbol{\alpha}) + by_i & \leq & \varepsilon\,, & \text{if} \quad \alpha_i = C\,, \\ |\, -\nabla_i(\boldsymbol{\alpha}) + by_i \,| & \leq & \varepsilon\,, & \text{if} \quad 0 < \alpha_i < C\,. \end{array} \tag{8}$$

The tightness of the stopping conditions (8) is controlled by $\varepsilon \geq 0$; hereafter we will refer to (8) as the ε-KKT conditions. An advantage of the ε-KKT is their simplicity and a low computational overhead: $O(m)$ operations since $\nabla_i(\boldsymbol{\alpha})$ is usually available during the course of the

QP solver. A disadvantage of the ε-KKT conditions is a tricky choice of ε. Provided $\varepsilon = 0$, the solution $\boldsymbol{\alpha}$ satisfying the ε-KKT conditions is guaranteed to be optimal. The practically applicable QP solvers, however, are guaranteed to halt in a finite number of iterations only for $\varepsilon > 0$. A typically used value is $\varepsilon = 0.001$, e.g., in software packages SVM^{light} ((Joachims, 1998)) or `svmlib` ((Chang & Lin, 2001)). To our knowledge, there is no theoretical result connecting $\varepsilon > 0$ to the value of $R_{LOO}[f(\cdot; \boldsymbol{\alpha}^*, b^*)]$ which is the only desired outcome of the entire computation.

We will illustrate the impact of ε when the LOO error estimator is used for a model selection. Let Θ be a given finite set of hyper-parameters $\boldsymbol{\theta} = (C, k)$. Let $R_{LOO}(\boldsymbol{\theta}, \varepsilon)$ denote the LOO error estimate computed for given $\boldsymbol{\theta}$ using Algorithm 1 with a QP solver in Algorithm 2. Thus the estimated LOO error $R_{LOO}(\theta, \varepsilon)$ is a function of both the hyper-parameters $\boldsymbol{\theta}$ and ε. For a fixed value $\varepsilon > 0$, the model selection produces the hyper-parameters

$$\boldsymbol{\theta}(\varepsilon) = \operatorname*{argmin}_{\boldsymbol{\theta} \in \Theta} R_{LOO}(\boldsymbol{\theta}, \varepsilon) \; . \tag{9}$$

Figure 1 plots the behavior of $R_{LOO}(\boldsymbol{\theta}(\varepsilon), \varepsilon)$ and $R_{LOO}(\boldsymbol{\theta}(10^{-4}), \varepsilon)$, as well as the cost of the LOO error computation as a function of ε for three datasets selected from the IDA repository (cf. Section 5).

The "golden truth" expected risk is given by the left-most plots in the graphs (using $\varepsilon = 10^{-4}$ for both model selection and risk estimation). The dashed line representing $R_{LOO}(\boldsymbol{\theta}(10^{-4}), \varepsilon)$ shows that the expected risk is slightly overestimated provided we use high accuracy for model selection and variable accuracy for risk estimation. The solid line representing $R_{LOO}(\boldsymbol{\theta}(\varepsilon), \varepsilon)$ shows that a low-accuracy LOO computation used in model selection eventually results in overfitting, as a model is selected that grossly underestimates the expected risk. Interestingly, both plots coincide until a certain breakdown point beyond which the low-accuracy LOO estimation runs aground. The breakdown point varies between 0.001 and 0.1 depending on a dataset. This suggests that a commonly used $\varepsilon = 0.001$ is a reasonable setting to obtain an accurate estimate. It is, however, clear from the timing plots that knowing the right accuracy could significantly lower the computational cost.

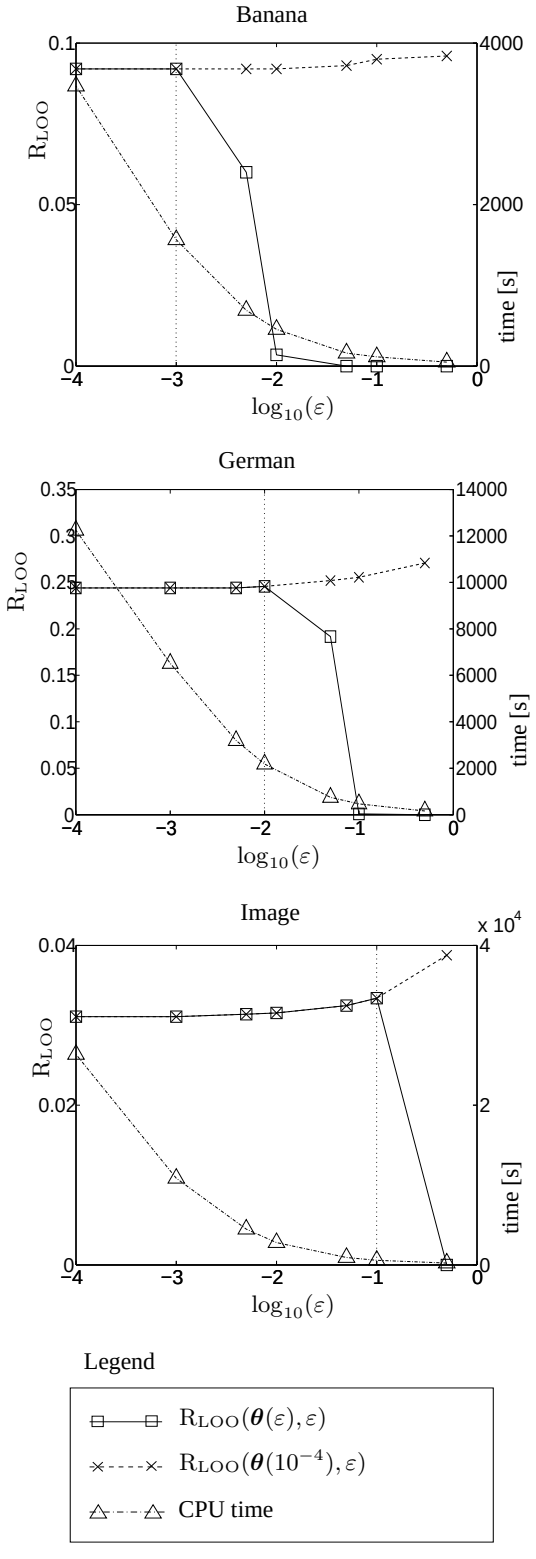


Figure 1. Influence of the parameter ε of the ε-KKT conditions on the LOO error estimate and the required computational time for three data sets (Banana, German and Image) selected from IDA repository.

3. Exact Computation of the LOO Error

In this section we show that a response of the optimal classifier $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$, required when the LOO error is being computed, can be obtained without the need to solve the QP task (4) (or its dual (7)) optimally. Let us define

three convex sets

$$\mathcal{A}_+(r) = \mathcal{A}(r) \cap \Big\{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \mid \sum_{i\in\mathcal{I}} \alpha_i y_i k(x_i, x_r) + b > 0\Big\},$$
$$\mathcal{A}_0(r) = \mathcal{A}(r) \cap \Big\{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \mid \sum_{i\in\mathcal{I}} \alpha_i y_i k(x_i, x_r) + b = 0\Big\},$$
$$\mathcal{A}_-(r) = \mathcal{A}(r) \cap \Big\{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \mid \sum_{i\in\mathcal{I}} \alpha_i y_i k(x_i, x_r) + b < 0\Big\}. \tag{10}$$

Notice, that to compute $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ we do not necessarily need to know the optimal $(\boldsymbol{\alpha}^*(r), b^*(r), \boldsymbol{\xi}^*(r))$ but it suffices to determine whether $(\boldsymbol{\alpha}^*(r), b^*(r), \boldsymbol{\xi}^*(r))$ belongs to $\mathcal{A}_+(r) \cup \mathcal{A}_0(r)$ or to $\mathcal{A}_-(r)$. Our method is based on a simple observation which can be formally stated by the following theorem:

Theorem 1 *For any $(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) \in \mathcal{A}(r)$ which satisfy*

$$F(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) < \min_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \in \mathcal{A}_0(r)} F(\boldsymbol{\alpha}, b, \boldsymbol{\xi})\,, \tag{11}$$

the equation $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ holds.

Proof 1 *We proof Theorem 1 by transposition: we show that $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) \neq f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ implies the assumption (11) is violated. Without loss of generality let $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) = +1$ and $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r)) = -1$. Let us define three vectors*

$$\hat{\boldsymbol{\theta}} = \begin{pmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{b} \\ \hat{\boldsymbol{\xi}} \end{pmatrix}, \quad \boldsymbol{\theta}^*(r) = \begin{pmatrix} \boldsymbol{\alpha}^*(r) \\ b^*(r) \\ \boldsymbol{\xi}^*(r) \end{pmatrix}, \quad \boldsymbol{\theta}_0 = \begin{pmatrix} \boldsymbol{\alpha}_0 \\ b_0 \\ \boldsymbol{\xi}_0 \end{pmatrix}.$$

With a slight abuse of notation, we will handle F as a function of a single argument $\boldsymbol{\theta} \in \mathbb{R}^{2m+1}$. Then the assumptions $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) = +1$ and $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r)) = -1$ is equivalent to $\hat{\boldsymbol{\theta}} \in \mathcal{A}_+(r) \cup \mathcal{A}_0(r)$ and $\boldsymbol{\theta}^*(r) \in \mathcal{A}_-(r)$. From (10) it follows that for any $\hat{\boldsymbol{\theta}} \in \mathcal{A}_+(r) \cup \mathcal{A}_0(r)$ and $\boldsymbol{\theta}^*(r) \in \mathcal{A}_-(r)$ there exists $\tau \in [0, 1]$ such that $\boldsymbol{\theta}_0 = ((1-\tau)\hat{\boldsymbol{\theta}} + \tau\boldsymbol{\theta}^*(r)) \in \mathcal{A}_0(r)$. Since F is convex, $\tau \in [0, 1]$ and $F(\boldsymbol{\theta}^*) \leq F(\hat{\boldsymbol{\theta}})$ we can write

$$\begin{aligned} F(\boldsymbol{\theta}_0) &\leq (1-\tau)F(\hat{\boldsymbol{\theta}}) + \tau F(\boldsymbol{\theta}^*(r)) \\ &\leq \max\big\{F(\hat{\boldsymbol{\theta}}), F(\boldsymbol{\theta}^*(r))\big\} = F(\hat{\boldsymbol{\theta}})\,, \end{aligned}$$

which shows that there exist $\boldsymbol{\theta}_0 \in \mathcal{A}_0(r)$ such that $F(\boldsymbol{\theta}_0) \leq F(\hat{\boldsymbol{\theta}})$. Using the original notation, this is equivalent to $F(\boldsymbol{\alpha}_0, b_0, \boldsymbol{\xi}_0) \leq F(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}})$. However, this contradicts the assumption (11) which was to be shown.

By Theorem 1, any triplet $(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) \in \mathcal{A}(r)$ satisfying the condition (11) determines a classifier $f(x; \hat{\boldsymbol{\alpha}}, \hat{b})$ which has the same response on the input x_r as the optimal classifier $f(x; \boldsymbol{\alpha}^*(r), b^*(r))$. From a practical point of view, this result cannot be used directly due to the unknown value of the right hand side of the inequality (11), i.e.,

$$\min_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \in \mathcal{A}_0(r)} F(\boldsymbol{\alpha}, b, \boldsymbol{\xi})\,. \tag{12}$$

The problem (12) is a convex QP task its dual reads

$$\boldsymbol{\beta}^*(r) = \max_{\boldsymbol{\beta} \in \mathcal{B}_0(r)} \left(\sum_{i\in\mathcal{I}} \beta_i - \frac{1}{2} \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{I}} \beta_i \beta_j y_i y_i k'(x_i, x_j) \right), \tag{13}$$

where $k'(x_i, x_j) = k(x_i, x_j) - k(x_r, x_i) - k(x_r, x_j) - k(x_r, x_r)$ and $\mathcal{B}_0(r)$ is a convex feasible set which contains all $\boldsymbol{\beta} \in \mathbb{R}^m$ satisfying

$$0 \leq \beta_i \leq C\,, i \in \mathcal{I} \setminus \{r\}\,, \quad \text{and} \quad \beta_r = 0\,.$$

We will use $H(\boldsymbol{\beta})$ to denote the objective function of (13). By the weak duality theorem, the inequality $F(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \geq H(\boldsymbol{\beta})$ holds for any $(\boldsymbol{\alpha}, b, \boldsymbol{\xi}) \in \mathcal{A}_0(r)$ and $\boldsymbol{\beta} \in \mathcal{B}_0(r)$. This allows us to derive the following useful corollary:

Corollary 1 *For any $(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) \in \mathcal{A}_0(r)$ and $\boldsymbol{\beta} \in \mathcal{B}_0(r)$ which satisfy*

$$F(\hat{\boldsymbol{\alpha}}, \hat{b}, \hat{\boldsymbol{\xi}}) < H(\hat{\boldsymbol{\beta}})\,, \tag{14}$$

the equation $f(x_r; \hat{\boldsymbol{\alpha}}, \hat{b}) = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ holds.

Notice, that the condition (14) is satisfiable except for a very rare degenerate cases. It is easy to show, that if the condition (14) is not satisfiable then the error estimate $V(y_r; f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ is unstable anyway since there exists an optimal classifier $f(x; \boldsymbol{\alpha}^*(r), b^*(r))$ its separating hyperplane passes through the tested point x_r.

4. Algorithm

A direct application of Corollary 1 would require solving a mixed set of one quadratic and many linear inequalities. We are not aware of any simple and efficient algorithm to solve such task. Instead, we show how to use Corollary 1 to derive a novel stopping condition for a standard iterative QP solver (cf. Algorithm 2).

Algorithm 3 Proposed QP solver

1: Initialize $t := 0$, $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}(r)$ and $\boldsymbol{\beta}^{(t)} \in \mathcal{B}_0(r)$.
2: $t := t + 1$.
3: Update $\boldsymbol{\alpha}^{(t-1)} \rightarrow \boldsymbol{\alpha}^{(t)}$, i.e., find $\boldsymbol{\alpha}^{(t)} \in \mathcal{A}(r)$ such that $G(\boldsymbol{\alpha}^{(t-1)}) < G(\boldsymbol{\alpha}^{(t)})$.
4: If $\boldsymbol{\alpha}^{(t)}$ satisfies the ε-KKT conditions then halt otherwise continue to Step 5.
5: For fixed $\boldsymbol{\alpha}^{(t)}$ compute feasible $b^{(t)}$ and $\boldsymbol{\xi}^{(t)}$ minimizing $F(\boldsymbol{\alpha}^{(t)}, b, \boldsymbol{\xi})$.
6: Update $\boldsymbol{\beta}^{(t-1)} \rightarrow \boldsymbol{\beta}^{(t)}$, i.e., find $\boldsymbol{\beta}^{(t)} \in \mathcal{B}_0(r)$ such that $H(\boldsymbol{\beta}^{(t-1)}) < H(\boldsymbol{\beta}^{(t)})$.
7: If $F(\boldsymbol{\alpha}^{(t)}, b^{(t)}, \boldsymbol{\xi}^{(t)}) < H(\boldsymbol{\beta}^{(t)})$ holds then halt otherwise go to Step 2.

The proposed method is described by Algorithm 3 which, compared to a standard QP solver, involves three additional Steps 5, 6 and 7. In Step 5, the algorithm computes the

primal variables $(\boldsymbol{\xi}^{(t)}, b^{(t)})$ minimizing $F(\boldsymbol{\alpha}^{(t)}, b^{(t)}, \boldsymbol{\xi}^{(t)})$ which amounts to a simple optimization problem since $\boldsymbol{\alpha}^{(t)}$ is known. In Step 6, the algorithm maximizes the auxiliary criterion $H(\boldsymbol{\beta}^{(t)})$ w.r.t. $\boldsymbol{\beta}^{(t)}$. Finally, in Step 7, the algorithm checks whether $H(\boldsymbol{\beta}^{(t)})$ has become greater than $F(\boldsymbol{\alpha}^{(t)}, b^{(t)}, \boldsymbol{\xi}^{(t)})$; as soon as this occurs the algorithm halts since $f(x_r; \boldsymbol{\alpha}^{(t)}, b^{(t)}) = f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$ is guaranteed according to Corollary 1. The ε-KKT conditions are retained in Step 3 of Algorithm 3 since the condition (14) need not be satisfiable in general.

The proposed Algorithm 3 is intended to be used for computation of $f(x_r; \boldsymbol{\alpha}^*(r), b^*(r))$, i.e., it is called in Step 4 of Algorithm 1 calculating the LOO error, as a replacement for the standard QP solver. In terms of accuracy of computing the LOO error, the proposed algorithm cannot perform worse than the standard one. If the ε-KKT conditions are satisfied earlier than the proposed stopping condition then both the solvers find an identical classifier. In the opposite case, however, the response of the classifier found by the proposed algorithm is guaranteed to be optimal. Albeit the proposed algorithm provides a theoretical guarantee for the found solution to be optimal, from the practical point of view both the algorithms will produce an identical LOO error estimate for a sufficiently low ε. We will empirically show, however, that the proposed algorithm is numerically more efficient though it optimizes two QP tasks simultaneously compared to the standard approach. The higher efficiency is achieved by the proposed stopping condition which is often satisfied earlier than the ε-KKT condition.

To increase the numerical performance we also implemented the following simple efficiency test. The proposed algorithm is not applied on a single example but rather on a set of examples which cannot be resolved by the sufficiency checks. We experimentally observed, that the efficiency of the proposed algorithm can be reliably estimated from a few examples. This allows us to switch to using the standard QP solver when the efficiency of the proposed algorithm is low. The efficiency test, implemented in Step 4 of Algorithm 1, works as follows: We apply the proposed Algorithm 3 on the first M examples. Let M_{Prec} denote the number of examples for which Algorithm 3 halt in Step 7 (i.e., the proposed stopping condition was applied). If $M_{Prec}/M < 0.5$ we switch from using Algorithm 3 to using the standard Algorithm 2. We empirically found $M = 10$ to be a good choice number in all our experiments.

5. Experiments

In this section, we experimentally evaluate the proposed method for computing the LOO error compared to the standard approach on the datasets from the IDA benchmark repository [1].

The standard approach computes the LOO error using the procedure described by Algorithm 1. An iterative QP solver with the ε-KKT conditions (Algorithm 2) is called whenever the solution of the QP task is required. In particular, we used the Improved SMO algorithm (Keerthi et al., 2001) to implement the QP solver. In addition, we implemented α-seeding approach (DeCoste & Wagstaff, 2000) which re-uses the solution $\boldsymbol{\alpha}^*$ (obtained in Step 1 of Algorithm 1) to efficiently set up the initial solution of the QP solver (initialization of $\boldsymbol{\alpha}^{(0)}$ in Step 1 of Algorithm 2).

The proposed approach uses the same procedure for computing the LOO error except for a different QP solver used in Step 4 of Algorithm 1. As the QP solver, we applied the proposed Algorithm 3 which involves optimization of the QP tasks $G(\boldsymbol{\alpha}^{(t)})$ w.r.t. $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}(r)$ and $H(\boldsymbol{\beta}^{(t)})$ w.r.t. $\boldsymbol{\beta}^{(t)} \in \mathcal{B}_0(r)$ required in Step 3 and Step 6, respectively. We again used the Improved SMO algorithm to optimize $G(\boldsymbol{\alpha}^{(t)})$ w.r.t. $\boldsymbol{\alpha}^{(t)} \in \mathcal{B}(r)$ and its straightforward modification to optimize $H(\boldsymbol{\beta}^{(t)})$ w.r.t. $\boldsymbol{\beta}^{(t)} \in \mathcal{B}_0(r)$ ($\mathcal{B}_0(r)$ does not contain the equality constraint thus a single variable can be updated).

The experiments were carried out in Matlab 6 environment runnig on the Linux machine with the AMD K8 2.2GHz processor. Algorithms 1,2 and 3 were implemented in C.

The IDA repository consists of 13 artificial and real-world binary classification problems collected from UCI, DELVE and STATLOG repositories (c.f. (Rätsch et al., 2001)). For each dataset, there are 100 random realizations of training and testing set (except for Image and Splice sets, where it is 20). The training parts of the first 5 realizations are used for model selection. The best hyper-parameters (C, k) were selected from a finite set Θ by minimizing an average LOO error $\hat{R}_{LOO}$. The average LOO error $\hat{R}_{LOO}$ is computed over the 5 realizations.

We considered two separate model selection problems for the linear and the RBF kernel. In the case of the liner kernel, the model was select from $\Theta = \{C \mid C = 10^i, i = -2, \ldots, 2\} \times \{k \mid k(x, x') = \langle x, x' \rangle\}$ and, in the case of the RBF kernel $\Theta = \{C \mid C = 10^{\frac{i}{7}\log(500)}, i = 0, \ldots, 7\} \times \{k \mid k(x, x') = \exp(-2^{\frac{i}{7}11}\|x - x'\|^2), i = 0, \ldots, 7\}$. Having the model selected, the classifier is trained for all 100 realizations of the training sets and the testing error is computed on the corresponding testing set. The reported testing errors $\hat{R}_{TST}$ are averages accompanied with the standard deviations computed over the 100 realizations.

Table 2 shows the average LOO errors $\hat{R}_{LOO}$ and the testing errors $\hat{R}_{TST}$ for the best selected models. We experimentally verified, that both the standard and the proposed

[1] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

approach yielded identical LOO errors since a low value of $\varepsilon = 0.001$ was used in the ε-KKT conditions. Therefore the errors listed in Table 2 apply for both the approaches. We also found that the classification errors for the RBF kernel are very similar to the errors reported in (Rätsch et al., 2001) for the SVM classifier with the RBF kernel tuned by the 5-fold cross-validation. Interestingly, the linear kernel achieves in some cases comparable performance as the more complex RBF kernel. We can also observe, that the average LOO errors $\hat{R}_{LOO}$ for the linear kernel are very good estimators of the testing errors $\hat{R}_{TST}$.

Table 3 summarizes the numerical efficiency of the proposed approach and the standard one. The efficiency was measured in terms of the computational time and the number of kernel evaluations. The reported $Time$ is the overall computational time spent by a given algorithm to calculate all the LOO errors needed for the model selection. E.g., in the case of the RBF kernel it was necessary to compute $5 \times 64 = 320$ LOO error estimates (5 stands for the number of the training set realizations and 64 is the cardinality of Θ). Similarly, the number of kernel evaluations $KerEval$ is the overall value normalized to the number of training data m, i.e., $KerEval$ is the number of columns of the kernel matrix. In the case of the standard approach, we listed the absolute values of $Time$ and $KerEval$. In the case of the proposed approach, we listed the gained speed up computed as the ratio $Standard/Proposed$. The last column of Table 3 contains the value $Prec$ being the percentage of the cases when the proposed stopping condition was satisfied earlier then the ε-KKT conditions, i.e., in $Prec$ cases the computed LOO error is theoretically guaranteed to be optimal.

It can be seen, that the proposed method was never slower (up to the rounding error in computing the speed up) than the standard algorithm both in terms of the computational time and the kernel evaluations. A higher performance was achieved for the linear kernel compared to the RBF kernel. For the linear kernel, the proposed approach was on average 4 times faster than the standard approach. The best performance was achieved for the Image dataset when the speed up was nearly 13. For RBF kernel, the average speed up was slightly higher than 2, and, in the best case the speed up was 5 for the Banana dataset. It shows that while the efficiency gained for the RBF kernel is only moderate, in the case of the linear kernel it is much appealing.

6. Conclusions

The new stopping conditions for an SVM solver proposed in this contribution allow to determine an optimal solution accuracy needed for exact computation of a LOO error. Our new algorithm allows one to significantly reduce complexity of the LOO error computation without a risk of overfitting due to imprecise optimization. Our experiments on 13 datasets from the IDA repository achieved the average speedup of 2 to 4 times and the maximal speedup of up to the factor of 13.

	Classification performance			
	Linear kernel		RBF kernel	
	$\hat{\mathrm{R}}_{\mathrm{LOO}}$	$\bar{\mathrm{R}}_{\mathrm{TST}}$	$\hat{\mathrm{R}}_{\mathrm{LOO}}$	$\bar{\mathrm{R}}_{\mathrm{TST}}$
Banana	41.40	47.80 ($\pm$4.58)	8.55	10.43 ($\pm$0.44)
Breast	27.20	29.00 ($\pm$4.83)	23.00	26.06 ($\pm$4.91)
Diabetis	22.05	23.44 ($\pm$1.70)	21.50	23.27 ($\pm$1.65)
Flare	32.85	32.33 ($\pm$1.82)	32.31	34.04 ($\pm$2.04)
German	24.91	24.06 ($\pm$2.22)	23.71	23.61 ($\pm$2.23)
Heart	14.24	15.22 ($\pm$3.22)	13.53	15.55 ($\pm$3.36)
Image	15.48	15.34 ($\pm$0.84)	2.94	3.15 ($\pm$0.63)
Ring.	23.45	24.59 ($\pm$0.67)	1.10	1.60 ($\pm$0.11)
Splice	15.36	16.20 ($\pm$0.59)	10.60	10.95 ($\pm$0.64)
Thyroid	8.43	10.16 ($\pm$2.60)	2.29	4.87 ($\pm$2.28)
Titanic	21.20	23.01 ($\pm$4.62)	15.47	23.99 ($\pm$3.47)
Twono.	2.50	2.90 ($\pm$0.27)	2.25	2.59 ($\pm$0.18)
Wave.	10.90	12.95 ($\pm$0.54)	8.85	10.50 ($\pm$0.43)

Table 2. Classification performance of the best models selected by minimizing the LOO error estimate.

These results demonstrate the importance of investigating relationships between the optimization accuracy and the expected risk estimation in machine learning, as suggessed by recent work (Bartlett & Mendelson, 2006; Bottou & Bousquet, 2008). To our knowledge, the new algorithm is the first theoretically justified constructive instrument to guide an optimization algorithm – for the particular case of the SVM QP solver and the LOO error – towards achieving the best attainable expected risk at optimal computational cost. Future work should explore more general mechanisms of relating parameters of optimization algorithms deployed in machine learning with the estimation of expected risk.

Acknowledgements

VF was supported by Marie Curie Intra-European Fellowship grant SCOLES (MEIF-CT-2006-042107). This work was also supported by *Bundesministerium für Bildung und Forschung* under the project REMIND (FKZ 01-IS07007A).

References

Bartlett, P., & Mendelson, S. (2006). Empirical minimization. *Probability Theory and Related Fields*, *135*, 311–334.

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in neural information processing systems*, vol. 20. Cambridge, MA: MIT Press. to appear.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimiza-*

Linear Kernel

	Standard approach		Proposed approach		
	Time [s]	KerEval $\times 10^6$	Time speedup	KerEval speedup	Prec [%]
Banana	728.0	109.5	9.9	12.3	56.9
Breast	128.7	34.6	2.0	2.1	64.2
Diabetis	3705.3	432.1	6.3	7.2	93.1
Flare	61.5	4.7	1.0	1.1	59.3
German	47603.8	3685.7	4.2	4.4	82.0
Heart	280.7	90.1	1.9	2.1	83.1
Image	114373.9	4793.8	12.9	14.9	98.3
Ring.	11632.8	1536.0	2.8	3.0	91.8
Splice	41072.8	2222.1	1.5	1.5	74.5
Thyroid	21.6	8.2	3.2	5.2	66.0
Titanic	0.8	0.1	1.0	1.0	17.6
Twono.	52.8	8.3	1.9	2.2	89.1
Wave.	3422.8	503.3	2.4	2.6	90.7

RBF Kernel

	Standard approach		Proposed approach		
	Time [s]	KerEval $\times 10^6$	Time speedup	KerEval speedup	Prec [%]
Banana	1565.1	222.4	5.1	6.1	81.9
Breast	265.9	74.0	1.5	1.6	55.6
Diabetis	3163.6	382.3	1.4	1.5	54.8
Flare	2263.0	189.1	2.2	2.4	50.3
German	6513.7	549.4	1.1	1.1	31.3
Heart	59.5	20.7	1.1	1.2	50.3
Image	10852.7	514.5	2.9	3.1	75.3
Ring.	277.2	47.4	1.4	1.4	39.1
Splice	6360.3	453.8	1.0	1.0	14.6
Thyroid	10.3	3.3	1.5	2.4	80.3
Titanic	25.5	7.9	5.7	2.7	50.2
Twono.	175.3	36.8	1.0	1.0	22.2
Wave.	345.6	59.4	1.1	1.1	30.9

Table 3. Efficiency of computing the LOO error for the standard and the proposed approach

tion. Cambridge University Press.

Cauwenberghs, G., & Poggio, T. (2000). Incremental and decremental support vector machine learning. *NIPS* (pp. 409–415). MIT Press.

Chang, C., & Lin, C. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

DeCoste, D., & Wagstaff, K. (2000). Alpha seeding for support vector machines. *Int. Conf. on Knowledge Disc. and Data Mining*.

Duan, K., Keerti, S., & Poo, A. (2003). Evaluation of simple performance measures for tuning SVM hyperparamaters. *Neurocomputing, 15*, 487–507.

Jaakkola, T., & Haussler, D. (1999). Probabilistic kernel regression models. *Conference on AI and Statistics*. Morgan Kaufmann.

Joachims, T. (1998). Making large-scale support vector machine learning practical. In Schölkopf et B. al. (Ed.), *Advances in kernel methods: Support vector machines*. MIT Press, Cambridge, MA.

Joachims, T. (2000). Estimating the generalization performance of a SVM efficiently. *ICML*. San Francisco, CA.

Keerthi, S., Shevade, S., Bhattacharyya, C., & Murthy, K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation, 13*, 637–649.

Laskov, P., Gehl, C., Krüger, S., & Müller, K.-R. (2007). Incremental support vector learning: analysis, implementation and applications. *Journal of Machine Learning Research, 7*, 1900–1936.

Lee, J., & Lin, C. (2000). *Automatic model selection for support vector machines* (Technical Report). Dept. of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan,.

Lunts, A., & Brailovskiy, V. (1967). Evaluation of attributes obtained in statistical decision rules. *Engineering Cybernetics, 3*, 98–109.

Martin, M., Keerthi, S., Ong, C., & DeCoste, D. (2004). An efficient method for computing leave-one-out error in support vector machines with gaussian kernels. *IEEE TNN, 15*, 750–757.

Osuna, E., Freund, R., & Girosi, F. (1997). An improved training algorithms for support vector machines. *Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop* (pp. 276–285). IEEE.

Platt, J. (1998). Fast training of SVMs using sequential minimal optimization. In Schölkopf et B. al. (Ed.), *Advances in kernel methods: Svm learning*. MIT Press.

Rätsch, G., Onoda, T., & Müller, K. (2001). Soft margins for AdaBoost. *Machine Learning, 43*, 287–320.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. MIT Press.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer.

Vapnik, V., & Chapelle, O. (2000). Bounds on error expectation for SVM. In Smola et A. al. (Ed.), *Advances in large margin classifiers*, 261–280. MIT Press.

Zhang, T. (2001). A leave-one-out cross validaton bound for kernel methods with application in learning. *COLT*. Springer.

Reinforcement Learning in the Presence of Rare Events

Jordan Frank JORDAN.FRANK@CS.MCGILL.CA
Department of Computer Science, McGill University, Montreal, Quebec, Canada

Shie Mannor SHIE.MANNOR@MCGILL.CA
Department of Electrical and Computer Engineering, McGill University, Montreal, Quebec, Canada

Doina Precup DPRECUP@CS.MCGILL.CA
Department of Computer Science, McGill University, Montreal, Quebec, Canada

Abstract

We consider the task of reinforcement learning in an environment in which rare significant events occur independently of the actions selected by the controlling agent. If these events are sampled according to their natural probability of occurring, convergence of conventional reinforcement learning algorithms is likely to be slow, and the learning algorithms may exhibit high variance. In this work, we assume that we have access to a simulator, in which the rare event probabilities can be artificially altered. Then, importance sampling can be used to learn with this simulation data. We introduce algorithms for policy evaluation, using both tabular and function approximation representations of the value function. We prove that in both cases, the reinforcement learning algorithms converge. In the tabular case, we also analyze the bias and variance of our approach compared to TD-learning. We evaluate empirically the performance of the algorithm on random Markov Decision Processes, as well as on a large network planning task.

1. Introduction

We consider a practically important class of control tasks, in which rare (potentially catastrophic) events might take place. For example, in a computer network, links and nodes can fail, causing traffic to be undelivered and large penalties to be incurred. A robot exploring a rugged terrain may be caught by a sudden gust of wind which rolls it over. An investment agent may be faced with a market that is in turmoil due to a sudden unforeseen event. In such cases, the rare events occur *independently* of the actions of the agent, with some small probability. However, such rare events can have a disproportionate effect on the agent's utility. If such events are sampled on-line, as is the case in most reinforcement learning (RL) applications, they may not occur often enough to obtain an accurate estimate of the value function.

In this paper, we formalize this problem and propose solution algorithms. We assume that learning will be done in a simulation environment in which the probability of the rare event can be set to desired levels. In most safety-critical applications, training in a simulated environment is a common approach. In this case, we can sample rare events more often, and use importance sampling corrections similar to (Precup et al. 2000, 2001) to evaluate a given policy. However, importance sampling can cause high variance in the learning updates. We propose to use an adaptive algorithm in which the sampling rate for the rare event is adjusted in such a way as to minimize variance. For the case in which the value function is represented as a table, we show that the algorithm converges and provide a bias-variance analysis, based on (Mannor et al., 2007) . For the case of linear function approximation, we prove convergence. We note that a bias-variance analysis for this case is not even available for TD-learning without importance sampling. We illustrate the performance of our approach on two domains: random Markov Decision Processes (MDPs), and a large network planning task. Our approach proves quite successful when compared to on-line TD-learning.

The literature on simulation of rare events is vast; see (Bucklew, 2004; Asmussen & Glynn, 2007) for comprehensive reviews. There are many Markov (or Markov-like) models that have been studied in the simulation community including queues, inventory control problems, call centers, communication systems, etc. The main objective of these works is to estimate the probability of a rare event by simulating the system under an alternative probability mea-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

sure, and then use importance sampling to unbias the results. The search for the optimal change of measure can be done in several ways, including the cross-entropy method (Rubinstein & Kroese, 2004) and stochastic approximation. Variance reduction has also been studied within the RL community. In particular, (Baxter & Bartlett, 2001) considered adaptive control-variates for policy gradient algorithms.

The explicit modeling of rare events in reinforcement learning-style algorithms was studied in (Bhatnagar et al., 2006). Their objective is to find an optimal control policy conditioned on the occurrence of a rare event. A model that closely resembles our approach is presented in (Ahamed et al., 2006). However, they assume that the model of the transition probabilities is known, and can be arbitrarily modified. We make a less restrictive assumption: the only parameter of the simulator that can be modified is the rate at which the rare events occur. Also, the bias-variance analysis and discussion of the function approximation case are novel.

The rest of the paper is organized as follows. In Section 2 we provide the essential background on RL and MDPs. In Section 3 we formally describe the rare events model used in this paper. We review RL algorithms that use importance sampling in Section 4. The learning algorithm we propose is described in Section 5. In Section 6 we present bias-variance results for learning in MDPs with rare events. Section 7 presents a learning algorithm with function approximation and a proof of convergence in this case. The empirical results of our approach are presented in Section 8. Finally, Section 9 presents conclusions and avenues for future work.

2. Background

We use the standard RL framework (Sutton & Barto, 1998) in which an agent interacts with its environment at discrete time steps $t = 0, 1, 2, \ldots$. At time t, the agent finds itself in a state $s_t \in S$, chooses an available action $a_t \in A_{s_t}$, and then receives a numerical reward, $r_{t+1} \in \mathbb{R}$ and observes the next state s_{t+1}. We denote $A = \bigcup_{s \in S} A_s$. If the environment is modeled as an MDP, its dynamics are characterized by the stationary transition probability distribution:

$$p(s'|s,a) = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\},$$

and a bounded, real-valued reward function $r(s,a,s')$ with $|r(s,a,s')| \leq R_{\max} < \infty, \forall s, s' \in S, a \in A$.

We are concerned with the problem of policy evaluation in discounted infinite horizon problems with discount factor $\gamma \in (0,1)$. The agent chooses its actions according to a stationary policy $\pi(s,a) = \Pr\{a_t = a | s_t = s\}$. We are interested in computing the state-value function $V^\pi : S \to \mathbb{R}$ for the given policy π. This value function is the solution to the well-known Bellman equations

$$V^\pi(s) = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} p(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')]. \quad (1)$$

In RL, this value function is often estimated on-line using the well-known TD-learning algorithm (Sutton, 1988). If the actions are chosen using the desired policy π, after observing transition (s,a,r,s'), the estimate of the value function, V, can be updated as:

$$V(s) \leftarrow V(s) + \alpha \left[r + \gamma V(s') - V(s) \right]. \quad (2)$$

In control tasks, the objective is to find the policy that maximizes $V^\pi(s)$ at all states s.

3. Rare Events

We are concerned with problems involving rare, significant events that occur as a result of environmental factors, and which are independent of the current action taken by the agent. We model this using a mixture of two separate transition probability distributions: $f(s'|s,a)$, which captures the environment dynamics during "normal" operating conditions, and $g(s'|s)$, which is the "rare event" transition distribution. We assume that at every state $s \in S$, there is a small probability, $\varepsilon(s)$, that an unusual event might occur from this state. In this case, the transition to the next state is determined exclusively by g. If such an event does not occur, the next state is drawn according to f, and depends on both the current state and the agent's action. Hence, the transition probability in the environment can be re-written as:

$$p(s'|s,a) = (1 - \varepsilon(s)) f(s'|s,a) + \varepsilon(s) g(s'|s). \quad (3)$$

Without loss of generality, we will assume that the "normal" states (reachable by f) and the "rare event" states (reachable by g) are disjoint. Hence, the transition probability distribution can be re-written as:

$$p(s'|s,a) = \begin{cases} (1 - \varepsilon(s)) f(s'|s,a) & \text{if } s' \notin T, \\ \varepsilon(s) g(s'|s) & \text{if } s' \in T, \end{cases}$$

where $T \subseteq S$ is the set of "rare event" states.

We are concerned with rare events that have a significant impact on the state-value function for a given policy. Therefore we define the rare events state set as follows.

Definition 3.1. *A subset of states $T \subseteq S$ is called a* rare events state set *if the following three properties hold:*

1. *For all $s \in S$, $a \in A$, $s' \in T$, $f(s'|s,a) = 0$ (i.e., T is not reachable from any state s using the agent's actions).*

2. *There exists $s \in S$, $s' \in T$ such that $g(s'|s) > 0$ (i.e., T can be forced by the environment)*

3. *Let V_f^π denote the value function obtained by replacing p with f in (1). Then, for the given policy π,*

$$\exists s \in S \text{ s.t. } |V_f^\pi(s) - V^\pi(s)| \gg 0.$$

The last condition means that the states in the rare event state set must (collectively) have a large impact on the state-value function. We define *rare events* to be transitions into the rare event state set. For convenience, we will refer to the states that are not in the rare event state set, $S \setminus T$, as the normal states.

We note that we use the term "rare event" loosely from the point of view of the simulation community (Bucklew, 2004), because our definition is not based solely on the probability of the event. We deviate from the typical definition due to the fact that there may be events that occur infrequently but do not have a noticeable effect on our value function estimates, and we are not concerned with these events.

4. Importance Sampling for Reinforcement Learning

The TD update (2) is based on the idea that the right-hand side of the Bellman equations (1) can be approximated using samples of the next transition, $r(s,a,s') + \gamma V(s')$, where $a \sim \pi(s,\cdot)$ and $s' \sim p(\cdot|s,a)$. However, in an environment with rare events, if $\varepsilon(s)$ is very small, a very large number of samples will be needed in order for the rare events to be averaged properly in the value function estimates. Instead, we investigate a sampling distribution which allows these events to be sampled preferentially, and then we use importance sampling corrections to account for this in the TD updates.

Importance sampling is a variance-reduction technique commonly used in statistics, as well as in the simulation community (Bucklew, 2004). The main idea is that instead of obtaining samples from the true distribution p, they will be drawn from a different distribution q, called the *proposal distribution*, in which events of interest occur more frequently. If q is devised well, then using these samples will reduce the variance of the estimator. Precup, Sutton & Singh (2000) extended this approach to TD-learning. They studied the case in which a target policy π is evaluated based on data generated by a different behavior policy. In this case, they showed that a TD-learning algorithm can still be used, in which the TD targets are adjusted by using the appropriate importance sampling weights: $w(s,a,s')(r(s,a,s') + \gamma V(s'))$, where:

$$w(s,a,s') = \frac{p(s'|s,a)}{q(s'|s,a)}.$$

In their case, the change of measure is induced by the behavior policy, and the importance sampling weights are the likelihood ratios of the probabilities of action a under the two policies.

Ahamed, Borkar & Juneja (2004) use the same idea but with the goal of changing the next-state probabilities in a discrete-time finite-state Markov chain with positive costs. They assume that the transition probabilities are known and can be modified at will, and propose an adaptive importance sampling algorithm (ASA) which finds an alternative set of transition probabilities in order to minimize the variance of the value function estimator. They provide a convergence proof (assuming a tabular representation of the value function), a discussion of convergence rates, and simulation results.

5. Learning in the Presence of Rare Events

The ASA algorithm assumes that we have full knowledge of the transition model, and can completely control the transition probabilities, so all the transition probabilities can be tilted towards the zero-variance importance sampling distribution. In this paper, we relax this assumption because it is difficult to achieve in practical applications. We assume that the true rare event probability ε is known (e.g., as the mean of a Poisson process that generates failures in a network, or the weight of the tail of a distribution in which rare events occur). We assume that the system dynamics, f and g are unknown and cannot be modified, but that the probability with which rare events are generated can be changed as the simulation proceeds. In general, with only this parameter at our disposal, we cannot achieve the zero-variance importance sampling distribution; however, we can tilt the transition probability distribution p towards the zero-variance distribution, and therefore reduce the variance of our estimates.

We define $\hat{\varepsilon} : S \to [0,1]$ to be the probability of a rare event occurring from every state during the simulation. Hence, the next states will be sampled from a proposal distribution given by:

$$q(s'|s,a) = (1 - \hat{\varepsilon}(s)) f(s'|s,a) + \hat{\varepsilon}(s) g(s'|s), \qquad (4)$$

where f and g remain unchanged. By considering that the state space S is separated into disjoint normal and rare event subsets of states, we note that the importance sampling corrections $w(s,a,s')$ can be computed by:

$$w(s,a,s') = \begin{cases} \varepsilon(s)/\hat{\varepsilon}(s) & \text{if } s \in T, \\ (1-\varepsilon(s))/(1-\hat{\varepsilon}(s)) & \text{if } s \notin T. \end{cases} \qquad (5)$$

Following a similar argument as in the development of the ASA algorithm, we can determine the following optimal

form for the rare event sampling distribution:

$$\varepsilon^*(s) = \varepsilon(s) \frac{\sum_{s' \in T} g(s'|s)[(\sum_{a \in A} \pi(s,a) r(s,a,s')) + \gamma V^\pi(s')]}{V^\pi(s)}, \tag{6}$$

Fortunately, the values $\varepsilon^*(s)$ can be estimated on-line using samples.

Algorithm 1 is our proposed approach for learning in the presence of rare events. We call this algorithm rare events adaptive stochastic approximation (REASA). It is based on the observation that we can rewrite $\varepsilon^*(s)$ as follows:

$$\varepsilon^*(s) = \frac{T^*(s)}{T^*(s) + U^*(s)}, \text{ where}$$

$$T^*(s) = \varepsilon(s) \sum_{s' \in T} g(s'|s)[(\sum_{a \in A} \pi(s,a) r(s,a,s')) + \gamma V^\pi(s')]$$

is the contribution to the value of s by the rare event state set T, and

$$U^*(s) = (1 - \varepsilon(s)) \sum_{a \in A} \pi(s,a) \sum_{s' \notin T} f(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')]$$

is the contribution to $V^\pi(s)$ from the normal states.

In the algorithm, $T(s)$ is an unbiased estimator of $T^*(s)$ and $U(s)$ is an unbiased estimator of $U^*(s)$. It follows that as $t \to \infty$, from Equation (6),

$$\hat{\varepsilon}(s) = \frac{T(s)}{T(s) + U(s)} \to \varepsilon^*(s),$$

for every state $s \in S$. Since we use importance sampling to calculate $\hat{V}^\pi(s)$, we also have that as $t \to \infty$, $\hat{V}^\pi(s) \to V^\pi(s)$ from standard stochastic approximation arguments under some mild assumptions on the MDP structure. We summarize the result in the following proposition.

Proposition 1. *Using Algorithm 1 and assuming that the MDP is unichain for* $\varepsilon = \delta$ [1] *we have that:*

$$\hat{V}^\pi(s) \to V^\pi(s) \text{ almost surely.}$$

Moreover, $\forall s$ *s.t.* $\varepsilon^*(s) \in (\delta, 1-\delta)$ *we have that* $\hat{\varepsilon}(s) \to \varepsilon^*(s)$ *almost surely.*

We note that we guarantee that we have enough persistent exploration by requiring that $\hat{\varepsilon}(s)$ is bounded from below by δ and from above by $1 - \delta$ (step 5h in Algorithm 1).

Although the treatment above is assuming positive rewards (for ease of notation), our algorithm is actually formulated for the general case in which rewards can be both positive and negative, which is an extension of the ASA algorithm.

[1] The unichain assumption is needed to invoke the stochastic approximation argument; see (Bertsekas & Tsitsiklis, 1996). Also note that if the MDP is unichain for one value of $\varepsilon \in (\delta, 1-\delta)$ it is unichain for all values.

Algorithm 1 Rare-event Adaptive Importance Sampling

Input: Rare event set $T \subset S$, true rare-event probabilities $\varepsilon(s)$, and parameter $\delta > 0$, used to keep the sampling distribution non-zero everywhere.

1. Initialize $\hat{V}^\pi$ arbitrarily.
2. Initialize the rare-event sampling distribution: $\hat{\varepsilon}(s) \leftarrow 1/2, \forall s$.
3. Initialize the variables $T(s)$, $U(s)$ (which measure the contribution of T and $S \setminus T$ to V^π) to 0.
4. Initialize eligibility traces: $e(s) = 0, \forall s$.
5. Select the initial state s_0.
6. Repeat for $t = 0, 1, \ldots$:
   (a) Update the eligibility trace of the current state
   $$e(s_t) = e(s_t) + 1.$$
   (b) Select an action $a_t \sim \pi(s_t, \cdot)$.
   (c) Select whether a rare event happens, according to $\hat{\varepsilon}(s_t)$, and sample s_{t+1} from f or g accordingly. Observe the reward r_{t+1}.
   (d) Compute the importance sampling weight w_t according to Equation (5).
   (e) Compute the importance-sampling TD-error:
   $$\Delta_t = w_t(r_{t+1} + \gamma \hat{V}^\pi(s_{t+1})) - \hat{V}^\pi(s_t).$$
   (f) Update the value estimates:
   $$\hat{V}^\pi(s) \leftarrow \hat{V}^\pi(s) + \alpha e(s) \Delta_t, \forall s,$$
   where $\alpha \in [0,1]$ is a learning rate.
   (g) If $s_{t+1} \in T$, then:
   $$T(s_t) \leftarrow (1 - \alpha_T) T(s_t) + \alpha_T \varepsilon(s_t)(r_{t+1} + \gamma \hat{V}^\pi(s_{t+1})),$$
   else
   $$U(s_t) \leftarrow (1 - \alpha_U) U(s_t) + \alpha_U (1 - \varepsilon(s_t))(r_{t+1} + \gamma \hat{V}^\pi(s_{t+1})),$$
   where $\alpha_T, \alpha_U \in (0,1)$ are learning rates. In the experiments, we use the inverse of the number of times a transition from s_t has been observed to T and $S \setminus T$ respectively.
   (h) Update the rare event probabilities:
   $$\hat{\varepsilon}(s_t) \leftarrow \min\left(\max\left(\delta, \frac{|T(s_t)|}{|T(s_t)| + |U(s_t)|}\right), 1 - \delta\right).$$
   (i) Update eligibility traces:
   $$e(s) \leftarrow \gamma \lambda w_t e(s), \forall s.$$

6. Bias and Variance of Reinforcement Learning with Rare Events

For simplicity, let us assume that $\varepsilon(s) = \varepsilon$ for all states $s \in S$ (all the analysis can be done without this assumption, but becomes more tedious). Let R^π denote the vector of immediate rewards for every state, with entries:

$$R^\pi_s = \sum_{a \in A} \sum_{s' \in S} \pi(s,a) p(s'|s,a) r(s,a,s'),$$

and P^π be an $|S| \times |S|$ transition matrix under π, with entries:

$$P^\pi_{ss'} = \sum_a \pi(s,a) p(s'|s,a).$$

From (3), we can re-write P^π as:

$$P^\pi = (1-\varepsilon) F^\pi + \varepsilon G,$$

where F^π is the transition matrix corresponding to staying in the normal states, and G is the matrix corresponding to transiting into the rare event states. Note that according to our assumptions, G does not depend on π. Similarly, the reward vector can be decomposed into two components, R^π_F and R^π_G. We use two sequences, $\{X_k\}_{k=1}^\infty$ and $\{Y_k\}_{k=1}^\infty$, of geometrically distributed random variables, with means $(1-\varepsilon)^{-1}$ and ε^{-1} respectively, to represent the amount of time between transitions from the normal states and the rare event states respectively. We also assume that the initial state is a normal state. Hence, the simulation starts in some normal state and stays in the set of normal states for X_1 time steps, at which point it transitions to a state in the rare event set, where it stays for Y_1 time steps, then transitions back to the normal set for X_2 time steps, etc. We make two further simplifications. First, we assume that after each excursion into the rare event state set, the system "jumps back" to the normal state in which it was before entering; that is, $(F^\pi)^i G^j (F^\pi)^k \approx (F^\pi)^{i+k}$. Second, we assume that the rewards for transitioning to states in the normal set are similar regardless of the origin, that is that $G R^\pi_F \approx F^\pi R^\pi_F$. The analysis can be done without these assumptions, but it becomes more tedious. These assumptions are reasonable because in general the rare events model failures in the system, such as a failed link in a network, and when the failure is no longer present, the system resumes from the state prior to the failure. We define $\tau(k) = \sum_{i=1}^{k-1} X_k$ and $\upsilon(k) = \sum_{i=1}^{k-1} Y_k$. The value function estimate, V^π, can be re-written as:

$$V^\pi \approx \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^{\tau(k)+\upsilon(k)} (F^\pi)^{\tau(k)-1} \cdot \left(\sum_{i=0}^{X_k-1} \gamma^i (F^\pi)^i R^\pi_F + \gamma^{X_k} (F^\pi)^{X_k-1} \sum_{i=0}^{Y_i-1} \gamma^i G^i R^\pi_G \right) \right].$$

When the value function is estimated from data using TD-learning, we can analyze the bias and variance of this estimate by considering that all the model component estimates are affected by noise components (Mannor et al., 2007). The estimate of the value function, $\hat{V}^\pi$ can be broken up into two components; the first component ignores rare events, and the second takes rare events into account. The first component is:

$$\mathbb{E}[\hat{V}^\pi_F] = \sum_{k=1}^{\infty} \mathbb{E}\left[\gamma^{\tau(k)+\upsilon(k)} (F^\pi + \tilde{F}^\pi)^{\tau(k)-1} \cdot \sum_{i=0}^{\infty} (1-\varepsilon)^i \varepsilon \gamma^i (F^\pi + \tilde{F}^\pi)^i (R^\pi_F + \tilde{R}^\pi_F) \right]$$

where $\tilde{R}^\pi_F$, $\tilde{F}^\pi$ represent the noise estimates in the normal part of the model. The bias and variance of this estimate can be derived directly as in (Mannor et al., 2007), noting that $\tau(k)$ and $\upsilon(k)$ are sums of independent geometrically distributed variables, and are therefore distributed according to a negative binomial distribution.

The second component is:

$$\mathbb{E}[\hat{V}^\pi_G] = \sum_{k=1}^{\infty} \mathbb{E}\left[\gamma^{\tau(k+1)+\upsilon(k)} (F^\pi + \tilde{F}^\pi)^{\tau(k+1)-1} \cdot \sum_{i=0}^{\infty} (1-\varepsilon) \varepsilon^i \gamma^i (G + \tilde{G})^i (R^\pi_G + \tilde{R}^\pi_G) \right]$$

Note that the noise components $\tilde{G}$, $\tilde{F}^\pi$, $\tilde{R}^\pi_G$ depend on the number of transitions observed in the environment. If we observe N transitions, then the expected number of transitions observed in the normal state set is $(1-\varepsilon)N$ and the expected number of transitions observed in the rare event state set is εN. Hence, we assume that the noise component $\tilde{F}^\pi$ is negligible compared to $\tilde{G}$, and $\tilde{R}^\pi_G$. Hence, to establish bias-variance estimates for $\hat{V}^\pi_G$, we need to look at $\mathbb{E}\left[(G+\tilde{G})(R^\pi_G + \tilde{R}^\pi_G)\right]$. Similarly to (Mannor et al., 2007), we assume that $\mathbb{E}[\tilde{G}] = 0$ and $\mathbb{E}[\tilde{R}^\pi_G] = 0$. Hence, the remaining term which will determine the bias and variance is $\mathbb{E}[\tilde{G}\tilde{R}^\pi_G]$, which captures the correlations between the transition and model estimates, due to the fact that they are estimated from the same samples. This expectation can be derived directly from the formulas in (Mannor et al., 2007).

We would like to point out that we could also have applied the analysis of (Mannor et al., 2007) directly to P^π. However, this would lead to very loose bounds, because their results depend on the inverse of the minimum number of samples obtained for any transition, and we expect that there will be very few transitions into the rare event set. In our analysis, only the second term depends on numbers of transitions into the rare events states, so we can focus our analysis on the effect of the rare events on the bias and variance in our estimates.

Also, note that the purpose of the algorithm is to sample rare events proportionately to their contribution to the value function for all states. Hence, intuitively, it will reduce bias and variance in the second component by oversampling the rare events, and thus decreasing the noise components $\tilde{G}$ and $\tilde{R}^{\pi}_{G}$. Given the same amount of data, the errors in $\tilde{F}$ and $\tilde{R}^{\pi}_{F}$, but not by much.

7. Learning with Rare Events and Function Approximation

If the state space is very large or continuous, function approximation must be used to estimate the value function. Here, we are concerned with the case of linear function approximation, in which the value of a state is estimated as:

$$V^{\pi}(s) \approx \theta\phi(s), \tag{7}$$

where θ is a parameter vector that needs to be estimated and $\phi(s)$ is a set of features corresponding to state s. In this case, the eligibility traces are also represented as a vector $\mathbf{e}$ of the same size as θ. We now extend the REASA algorithm to deal with this case. First, note that in this case, we may not be able to have a state-dependent probability of obtaining a rare event state, because specifying this on a state-by-state basis would be too expensive. Hence, we will assume for the moment, without loss of generality, that the true rare event probability ε is constant over the entire state space. We discuss possible extension to this in Section 9. The algorithm will estimate a parameter $\hat{\varepsilon}$ by taking the view that, at a high level, the agent switches between the normal states $S \setminus T$ and the rare-event states T. These are now treated as two states in a high-level MDP, and $\hat{\varepsilon}$ is estimated like in REASA, on this 2-state system.

Algorithm 2 presents the approach, which adapts the algorithm of Precup et al. (2001). Unlike in the tabular case, here importance sampling corrections have to be made to account for the difference in the distribution of observed features, as well as for the difference in the TD target. As explained in Precup et al. (2001), these corrections, which are collected in the trajectory weight c, can result in high variance. However, since we assume that the sets of normal and rare event states are disjoint, we can assume, without loss of generality, that they are represented by disjoint features as well. In this case, step 7i of Algorithm 2 can be eliminated, and variance will be greatly improved.

Proposition 2. *Under standard stochastic approximation conditions, Algorithm 2 converges in the limit, with probability 1, to the same estimates as the on-policy TD-learning algorithm.*

Algorithm 2 Rare-event Adaptive Importance Sampling with Function Approximation

Input: Rare event set $T \subset S$, true rare-event probability ε, and parameter $\delta > 0$, used to keep the sampling distribution non-zero everywhere.

1. Initialize parameter vector θ arbitrarily.
2. Initialize rare-event sampling parameter: $\hat{\varepsilon} \leftarrow 1/2$.
3. Initialize $\hat{T} \leftarrow 0$, $\hat{U} \leftarrow 0$.
4. Initialize eligibility vector: $\mathbf{e} \leftarrow 0$.
5. Initialize the total importance sampling trajectory weight: $c \leftarrow 1$.
6. Select the initial state s_0.
7. Repeat for $t = 0, 1, \ldots$:
   (a) Update the eligibility trace of the current state
   $$\mathbf{e} = \mathbf{e} + c\phi(s_t).$$
   (b) Select an action $a_t \sim \pi(s_t, \cdot)$.
   (c) Select whether a rare event happens, according to $\hat{\varepsilon}$, and sample s_{t+1} from f or g accordingly. Observe the reward r_{t+1}.
   (d) Compute the importance sampling weight w_t according to (5).
   (e) Compute the importance-sampling TD-error:
   $$\Delta_t = w_t(r_{t+1} + \gamma\hat{V}^{\pi}(s_{t+1})) - \hat{V}^{\pi}(s_t),$$
   where $\hat{V}^{\pi}$ is computed according to (7).
   (f) If $s_{t+1} \in T$, then:
   $$\hat{T} \leftarrow ((1-\alpha_T)\hat{T} + \alpha_T\varepsilon(r_{t+1} + \gamma\hat{V}^{\pi}(s_{t+1})),$$
   else
   $$\hat{U} \leftarrow (1-\alpha_U)\hat{U} + \alpha_U(1-\varepsilon)(r_{t+1} + \gamma\hat{V}^{\pi}(s_{t+1})).$$
   (g) Update the parameter vector: $\theta \leftarrow \theta + \alpha\mathbf{e}\Delta_t$, where $\alpha \in [0,1]$ is a learning rate.
   (h) Update the rare event probabilities:
   $$\hat{\varepsilon} \leftarrow \min\left(\max\left(\delta, \frac{|\hat{T}|}{|\hat{T}| + |\hat{U}|}\right), 1-\delta\right).$$
   (i) Update the trajectory weight: $c \leftarrow cw_t$.
   (j) Update eligibility traces:
   $$\mathbf{e} \leftarrow \gamma\lambda w_t \mathbf{e}.$$

8. Experimental Results

8.1. Random MDPs

We first compare the performance of REASA to on-line TD(λ) and to ASA on a testbed of randomly generated

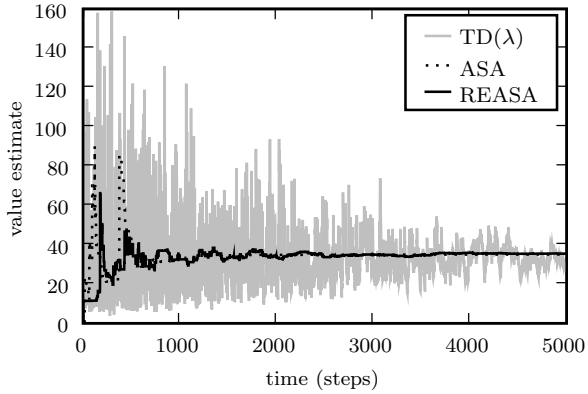


Figure 1. Value function estimate for state 0.

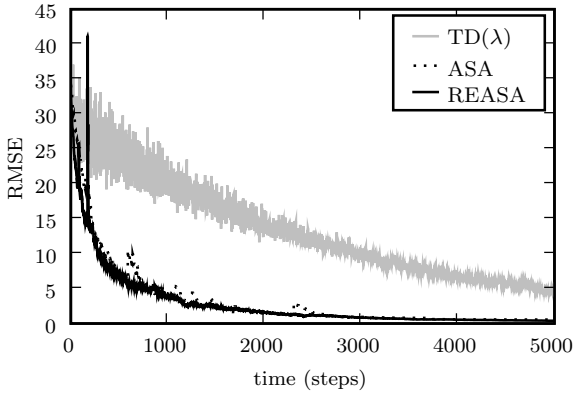


Figure 2. Root MSE for value function estimate for state 0.

Markov chains. Each environment contains 10 regular states and one rare event state. Each regular state can transition to seven other regular states (chosen randomly) with probabilities drawn from a uniform distribution, and to the rare event state with probability $\varepsilon = 0.001$. The rewards for transitioning between the regular states and from the rare event state to the regular states are drawn from a normal distribution with mean 1.0 and standard deviation 0.5, with negative values being discarded (so that we can run ASA). The rewards for transitioning to the rare event state are drawn from a normal distribution with mean $10/\varepsilon$ and standard deviation $1/\varepsilon$. The initial state is state 0, and the discount factor is $\gamma = 0.7$.

In the following results, a step is considered to be one transition for both ASA and REASA, but for TD(λ), a "step" actually consists of 2300 real time steps. We chose this number of steps so that the probability of observing at least one rare event transition in each episode is approximately 0.9. Therefore, we put TD(λ) at a significant advantage in terms of the number of samples that it is provided. In Figure 1 we plot the estimate for the value function at the initial state over time, averaged across 70 independent runs. We use a value of $\lambda = 0.7$ and the learning rates are on decreasing schedules that have been tuned separately for each of the algorithms. Figure 2 shows the root mean squared error for the value function estimate at the initial state, again averaged across 70 independent runs.

The learning and error curves for REASA and ASA are nearly indistinguishable, and both outperform TD(λ). We note that in the case of ASA, the original transition probability distribution is needed, and the algorithm has full control over the transition probabilities that are used in the simulation (an unlikely case in many practical applications). We observe that despite the fact that REASA can only know and control the rare event probability, it performs nearly as well as ASA.

8.2. Policy Evaluation for Network Planning

In order to demonstrate REASA in a practical setting with a large state space, we use a network planning task in which a reinforcement learning agent has to build and maintain a telecommunications network linking ten North American cities. Each pair of cities has a certain traffic demand, ranging from 3GBs[2] to 60GBs initially, and this demand grows stochastically at a rate of approximately 3% per year. The goal is to place links between the cities in order to deliver this data. Links consist of bundles of fiber optic cables, and each fiber can carry a specific unit of bandwidth. Building links between the cities incurs a large one-time cost of \$500k/mile. Once a link has been built, the capacity of the link can be increased by activating fibers, in units of 25GBs; this incurs a cost of \$30k/mile. The revenue from traffic is generated daily: traffic delivered generates a reward of \$1k/GBs/mile, and undelivered traffic is penalized at a rate of \$200k/GBs/mile every hour.

Link failures occur with a small probability, completely severing a link for a short period of time. Without considering link failures, a minimum spanning tree (MST) could be built, with enough activated fibers to carry the traffic. However, in such a network, any link failure would disconnect the network, which would lead to undelivered traffic and a high penalty. Hence, link failures in a network that lacks robustness are rare events according to our definition. On each day, each link goes down with probability $1/1460$, or approximately once every four years. When a link fails, it remains down for a random amount of time that is normally distributed with a mean of 12 hours and standard deviation of 2 hours. In a tree network with 9 links, this is equivalent to seeing at least one link fail with probability of approximately 0.00896 each day during the 10 year simulation period; this is our rare event probability.

[2]We use GBs to represent an average sustained traffic rate of 1 gigabyte per second; because the time interval under consideration is always roughly the same, we also use it as a unit of traffic, with an abuse of notation.

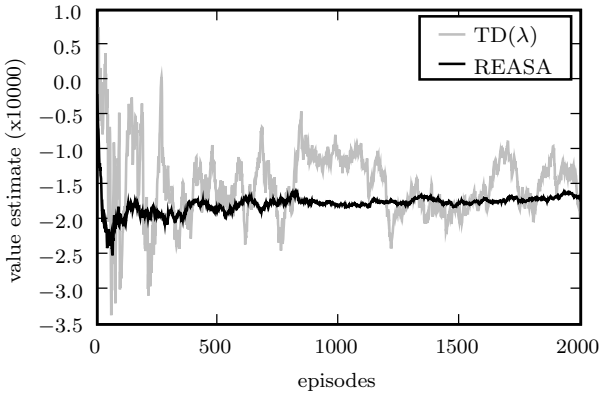


Figure 3. Value estimate for tree network.

We implemented a network planning agent with a simple heuristic policy, which first builds a tree network and then monitors the links, adding capacity when the utilization of a link reaches 90%. We use REASA and TD(λ) to estimate the value of this policy. We represent the network state as a vector of binary features, and use linear function approximation to represent the value function. In order to cope with the high dimensionality, we use a fairly coarse state representation, consisting of: indicator variables regarding whether each of the possible links have been built; indicator variables for each link, which are true if the link is currently failing; and the percentage utilization of each link, partitioned into 4 bins: $[0]$, $(0, 0.6]$, $(0.6, 0.9]$, and $(0.9, 1.0]$. For our 10 node network, this corresponds to 270 binary features plus an additional bias feature.

We use a discount factor of 0.95 and we set $\lambda = 1.0$. We use a decaying schedule for the learning rate parameter α, starting with a value of $\alpha_0 = 2^{-15}$ for $T = 100$ episodes, then using $\alpha_0/2$ for $2T$ episodes, $\alpha_0/4$ for $4T$ episodes, etc. We note that α_0 is extremely small due to the fact that the rewards often have large magnitude and can vary between -10^7 and 10^5. In the following results, an episode consists of a simulated 10-year time span.

In Figure 3, we show the value estimate for the initial tree network state. We see that REASA converges quickly, while the TD(λ) estimates have high variance and converge quite slowly. On longer runs, the TD(λ) estimates do converge to the same value as REASA. REASA estimates the optimal failure probability to be 0.155, which in a tree network with 9 links corresponds to each link going down approximately every 54 days; this is quite far from the original failure probability of once every 1460 days.

The rate of convergence is crucial for applications such as the network task. Here, each episode corresponds to a simulated 10 year period, and these simulations are computationally expensive to run, because on each day, a routing algorithm has to be run to determine the reward. Hence, the gains obtained by REASA are significant.

9. Conclusions and Future Work

We presented an approach for reinforcement learning in environments with rare events, aimed at reducing the variance of RL algorithms. Our algorithm modifies the sampling probability of the rare events, and makes minimal assumptions on the simulator available to the agent. The empirical results demonstrate the viability of our approach for solving large-scale problems. Future work will include measuring empirically the bias and variance of the algorithm. We would also like to lift the assumption that the rare event probability is constant for the function approximation case. Note that Algorithm 2 can be easily adapted to compute $\hat{T}$ and $\hat{U}$ as a function of the features available. Hence, if a representation of $\varepsilon(s)$ as a function of the available features ϕ is given, we could estimate $\hat{\varepsilon}$ as a function of features as well. It is possible also to learn the true rare event probabilities ε from data, but we anticipate that in practice this may be difficult.

Acknowledgments

The authors gratefully acknowledge the support of NSERC and the Canada Research Chairs program.

References

Ahamed, T. P. I., Borkar, V. S., & Juneja, S. (2006). Adaptive importance sampling technique for Markov chains using stochastic approximation. *Oper. Res.*, *54*, 489–504.

Asmussen, S. & Glynn, P. (2007). *Stochastic Simulation: Algorithms and Analysis*. Springer.

Baxter, J. & Bartlett, P. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15*, 319–350.

Bertsekas, D. & Tsitsiklis, J. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bhatnagar, S., Borkar, V. S., & Akarapu, M. (2006). A simulation-based algorithm for ergodic control of Markov chains conditioned on rare events. *Journal of Machine Learning Research*, *7*, 1937–1962.

Bucklew, J. (2004). *Introduction to Rare Event Simulation*. Springer.

Mannor, S., Simester, D., Sun, P., & Tsitsiklis, J. (2007). Bias and variance approximation in value function estimates. *Management Science*, *53*, 308.

Precup, D., Sutton, R., & Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. *Proc. 18th International Conf. on Machine Learning*, 417–424.

Precup, D., Sutton, R., & Singh, S. (2000). Eligibility traces for off-policy policy evaluation. *Proc. 17th International Conf. on Machine Learning*, 759–766.

Rubinstein, R. & Kroese, D. (2004). *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer-Verlag.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning*. The MIT Press.

Memory Bounded Inference in Topic Models

Ryan Gomes GOMES@VISION.CALTECH.EDU
Dept. of Computation and Neural Systems, California Institute of Technology, Pasadena, CA 91125 USA

Max Welling WELLING@ICS.UCI.EDU
Bren School of Information and Computer Science, University of California at Irvine, Irvine, CA 92697 USA

Pietro Perona PERONA@VISION.CALTECH.EDU
Dept. of Computation and Neural Systems, California Institute of Techology, Pasadena, CA 91125 USA

Abstract

What type of algorithms and statistical techniques support learning from very large datasets over long stretches of time? We address this question through a memory bounded version of a variational EM algorithm that approximates inference in a topic model. The algorithm alternates two phases: "model building" and "model compression" in order to always satisfy a given memory constraint. The model building phase expands its internal representation (the number of topics) as more data arrives through Bayesian model selection. Compression is achieved by merging data-items in clumps and only caching their sufficient statistics. Empirically, the resulting algorithm is able to handle datasets that are orders of magnitude larger than the standard batch version.

1. Introduction

Consider a collection of surveillance cameras monitoring at an airport. The cameras learn a model of their environment without supervision. Moreover, they learn for many years without significant interruption. Gradually, as more data is captured, the cameras build a joint model of visual object categories.

This problem is akin to the way children learn to understand the world through the *continuous* process of mostly unsupervised learning. As children grow up they build an increasingly sophisticated internal representation of object categories that continuously restructures itself.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

In this paper we ask ourselves: What statistical techniques are suitable for this "*lifelong learning task*"? First, we need a class of models that can naturally expand as more data arrives, i.e. it's capacity should not be bounded a priori. Second, these models should allow efficient learning algorithms, both in terms of time and space. For instance, we should not have to store every single piece of information that has been captured. Our technique must produce a sequence of model estimates that reflect new information as it arrives, and the time required to produce each model update must scale modestly as more data is acquired. Finally, we require that the sequence of learned models are sufficiently similar to those that would be produced by a batch algorithm with access to the entire history of data observed at the time of each model update.

Nonparametric Bayesian techniques such as the Dirichlet Process (DP) (Ferguson, 1973) and the Hierarchical Dirichlet Process (HDP) (Teh et al., 2006) satisfy our first desideratum, in that they naturally increase their model complexity with the available data. However, most existing Nonparametric Bayesian approaches are batch algorithms: they require every single data-point to be stored and revisited during learning. A batch algorithm could be naively applied to the continuous learning scenario, but all data would need to be cached and a new batch learning process would be run on the entire dataset to produce each model update. This would violate our second criterion in that the time and space requirements would increase unacceptably as the system ages.

Here we propose a more flexible setup, where we impose a bound on the available memory but still allow the model order to increase with more data. We *compress* the data and the internal representation of the model without losing much in terms of model accuracy. The effect is that time and space requirements scale much more gradually over the lifetime of the system. The memory bound does impose a limit on the total capacity of the model, but this trade-off

is flexible and can be adjusted online, i.e. as the model is learned. Experiments with a memory bounded variational approximation to HDP show that this technique can handle datasets many times larger than the standard implementations and results in substantially shorter run-times.

2. A Memory Bounded Variational Topic Model

At a high level the idea is to use a variational approximation related to LDA (Blei et al., 2003) and HDP (Teh et al., 2006). Memory savings are achieved by "clumping" together data-cases. That is, we constrain groups of datapoints to have equal topic assignment variational distributions: $q(z_{ij}) = q(z_{i'j'}) = q(z_c)$ when points x_{ij} and $x_{i'j'}$ are members of the clump c. This allows us to achieve memory savings, because variational optimization performed under this constraint requires only the sufficient statistics of the data-cases in a clump, and the system can forget the exact identities of the summarized data points. Similarly, we will also clump entire documents (or images) by tying their variational distributions over topics: $q(\boldsymbol{\pi}_j) = q(\boldsymbol{\pi}_{j'}) = q(\boldsymbol{\pi}_s)$ if document j and j' belong to the same document group s. This tying of variational distributions guarantees that learning optimizes a lower bound to the exact free energy objective function, where the bound is increasingly loose with more tying. This idea was also leveraged in (Verbeek et al., 2003) and (Kurihara et al., 2006) to accelerate learning of Mixtures of Gaussians and DP Mixtures of Gaussians by using KD-trees.

In the following we will talk about documents, but we note that this refers to other structured objects such as images as well.

2.1. The Variational Topic Model

The following Bayesian topic model is our starting point,

$$p(\mathbf{x}, \mathbf{z}, \boldsymbol{\eta}, \boldsymbol{\pi}, \boldsymbol{\alpha}) = \prod_{ij} p(\mathbf{x}_{ij}|z_{ij}; \boldsymbol{\eta})\, \pi_{j,z_{ij}} \left[\prod_k p(\boldsymbol{\eta}_k|\boldsymbol{\beta})\right] \left[\prod_j \mathcal{D}(\boldsymbol{\pi}_j; \boldsymbol{\alpha})\right] \left[\prod_k p(\alpha_k)\right] \tag{1}$$

where $\mathbf{x}_{ij}$ is word i in document j and z_{ij} denotes the topic that generated $\mathbf{x}_{ij}$. $\boldsymbol{\pi}_j$ denotes the mixture of topics that generated the words in document j, with $\sum_k \pi_{jk} = 1$. $\boldsymbol{\pi}_j$ are distributed according to a Dirichlet distribution with parameter $\boldsymbol{\alpha}$. Boldface symbols denote vector valued quantities. In this expression we will assume that $p(\mathbf{x}|z, \boldsymbol{\eta})$ is in the exponential family[1],

$$p(\mathbf{x}|z = k, \boldsymbol{\eta}) = \exp\left[\sum_l \eta_{kl}\, \phi_l(\mathbf{x}) - A_k(\boldsymbol{\eta}_k)\right] \tag{2}$$

and $p(\boldsymbol{\eta}|\beta)$ is conjugate to $p(\mathbf{x}|z, \boldsymbol{\eta})$,

$$p(\boldsymbol{\eta}_k|\boldsymbol{\beta}) = \exp\left[\sum_l \beta_l \eta_{kl} - \beta_0 A_k(\boldsymbol{\eta}_k) - B(\boldsymbol{\beta})\right] \tag{3}$$

The posterior distributions over $\boldsymbol{\pi}, \boldsymbol{\eta}, \mathbf{z}$ are approximated variationally as

$$q(\boldsymbol{\eta}) = \prod_k q(\boldsymbol{\eta}_k; \boldsymbol{\xi}_k) \tag{4}$$

$$q(\boldsymbol{\pi}) = \prod_j \mathcal{D}(\boldsymbol{\pi}_j; \boldsymbol{\zeta}_j) \tag{5}$$

$$q(\mathbf{z}) = \prod_{ij} q(z_{ij}) \tag{6}$$

where we have introduced variational parameters $\{\boldsymbol{\xi}_{kl}, \zeta_{kj}, q_{ijk}\}$, the latter subject to $\sum_k q_{ijk} = 1$. Furthermore, $\mathcal{D}$ denotes a Dirichlet distribution while $q(\boldsymbol{\eta}_k; \boldsymbol{\xi}_k)$ is also conjugate to $p(\mathbf{x}|z = k, \boldsymbol{\eta})$,

$$q(\boldsymbol{\eta}_k; \boldsymbol{\xi}_k) = \exp\left[\sum_l \xi_{kl}\eta_{kl} - \xi_{k0} A_k(\boldsymbol{\eta}_k) - B_k(\boldsymbol{\xi}_k)\right] \tag{7}$$

By writing down the variational free energy and minimizing it over $\boldsymbol{\xi}, \boldsymbol{\zeta}$ we find the following intuitive updates,

$$\xi_{kl} = F_{kl} + \beta_l; \qquad F_{kl} \triangleq \sum_{ij} q_{ijk}\, \phi_l(\mathbf{x}_{ij}) \tag{8}$$

$$\xi_{k0} = N_k + \beta_0; \qquad N_k \triangleq \sum_{ij} q_{ijk} \tag{9}$$

$$\zeta_{kj} = N_{kj} + \alpha_k; \qquad N_{kj} \triangleq \sum_i q_{ijk} \tag{10}$$

and

$$q_{ijk} \leftarrow \frac{1}{Z_{ij}} \frac{\exp\left[\sum_l \mathbb{E}[\eta_{kl}|\xi_{kl}]\, \phi_{kl}(\mathbf{x}_{ij})\right]}{\exp\left[\mathbb{E}[A_k(\boldsymbol{\eta}_k)|\xi_{k0}]\right]}\, \exp\left[\psi(\zeta_{kj})\right] \tag{11}$$

where Z_{ij} enforces the constraint $\sum_k q_{ijk} = 1$ and the expectations are over $q(\boldsymbol{\eta})$.

To learn the parameters $\{\alpha_k\}$ we first introduce gamma priors,

$$p(\boldsymbol{\alpha}) = \prod_k \mathcal{G}(\alpha_k; a, b) \tag{12}$$

[1] Strictly speaking, the exponential family includes additional multiplicative terms $h(\mathbf{x})$ in the expression for $p(\mathbf{x}|\boldsymbol{\eta})$ and $g(\boldsymbol{\eta})$ in the expression for $p(\boldsymbol{\eta}|\boldsymbol{\beta})$. We have left these terms out to simplify the derivation and because for most well known distributions they are simply 1. However, it is straightforward to include them.

Using the bounds in (Minka, 2000) we can derive the following updates if we first insert the updates for $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ into the free energy,

$$\alpha_k \leftarrow \frac{(a-1)+\alpha_k \sum_j \left[\psi(\zeta_{kj})-\psi(\alpha_k)\right]}{b+\sum_j \left[\psi(\zeta_j)-\psi(\alpha)\right]} \tag{13}$$

with $\zeta_j = \sum_k \zeta_{kj}$ and $N_j = \sum_k N_{kj}$.

2.2. Optimizing the Number of Topics K

Our strategy to search for a good value of K is to *truncate* the topic distributions as $q(z_{ij} > K) = 0$ (see also (Teh et al., 2008)). This will have the effect that most terms in the free energy with $k > K$ will cancel, the exception being the prior terms $p(\alpha_k),\ k > K$. For these terms we know that the value for α_k minimizing the free energy is given by the MAP value of the gamma-prior $\alpha_k = \frac{a-1}{b},\ k > K$. Inserting this back into the free energy we accumulate $K_{\max} - K$ terms

$$\Lambda = a\log b - \log\Gamma(a) + (a-1)\log\frac{a-1}{b} - (a-1) \tag{14}$$

where $K_{\max}$ is the maximum number of topics.

It is guaranteed that there exists a solution with lower free energy if we increase K. The reason is that we relax a self-imposed constraint on variational parameters (that $q(z_{ij} > K) = 0$). As K increases the relative improvement in free energy quickly attenuates. The final value for K is obtained by thresholding this relative improvement.

The *nesting* property (models with larger K are better) is the same for variational approximations to the DP in (Kurihara et al., 2006) and HDP (Teh et al., 2008). This raises the question if we can take the infinite limit for our model as well. The problem is that $(K_{\max} - K)\Lambda \rightarrow \infty$ as $K_{\max} \rightarrow \infty$. This can be traced back to the fact that we should have added a proper prior $p(K)$ which would have diminished the contribution at large K. Instead we choose an improper, constant prior to avoid the need to estimate likely values for K a priori. However, it is still possible to work with infinite free energies because we are only interested in the relative *change* in free energy after increasing K, which is a finite quantity.

In our experiments we chose $a = 1$ and $b = 0.5$, so that the MAP prior value of α_k is 0.

2.3. Clumping Data-Items and Documents

We will now tie some of the variational distributions $\{q_{ijk}\}$ across different data-items within and across documents (images) to a "clump distribution" q_{ck}. Similarly, we will tie some document specific distributions over topics $\{q(\boldsymbol{\pi}_j)\}$ into a document group $q(\boldsymbol{\pi}_s)$. Note that since we impose constraints on the variational distributions this has the effect of loosening the variational bound.

Define D_s to be the number of documents in a document group, N_c the number of data-items in a word clump, N_{cs} the number of words in document group s and word clump c and finally $\Phi^c_{kl} \triangleq \sum_{ij\in c} \phi_{kl}(\mathbf{x}_{ij})$. In terms of these we further define,

$$N_{ks} \triangleq \sum_c q_{ck} N_{cs} \tag{15}$$

$$N_k \triangleq \sum_c q_{ck} N_c \tag{16}$$

$$F_{kl} \triangleq \sum_c q_{ck}\ \Phi^c_{kl} \tag{17}$$

With these definitions we derive the following "clumped" update rules for the variational parameters ξ_{kl} and ζ_{ks},

$$\xi_{kl} = F_{kl} + \beta_l \tag{18}$$

$$\xi_{k0} = N_k + \beta_0 \tag{19}$$

$$\zeta_{ks} = \tfrac{N_{ks}}{D_s} + \alpha_k \tag{20}$$

and

$$q_{ck} \leftarrow \frac{1}{Z_c}\frac{\exp\left[\sum_l \mathbb{E}[\eta_{kl}|\xi_{kl}]\ \frac{\Phi^c_{kl}}{N_c}\right]}{\exp\left[\mathbb{E}[A_k(\boldsymbol{\eta}_k)|\xi_{k0}]\right]}\ \exp\left[\sum_s \frac{N_{sc}}{N_c}\psi(\zeta_{ks})\right] \tag{21}$$

The update for $\boldsymbol{\alpha}$ becomes

$$\alpha_k \leftarrow \frac{(a-1)+\alpha_k \sum_s D_s \left[\psi(\zeta_{ks})-\psi(\alpha_k)\right]}{b+\sum_s D_s \left[\psi(\zeta_s)-\psi(\alpha)\right]} \tag{22}$$

An expression for the free energy, after inserting expressions 18, 19 and 20, is given by eq. 29 in the appendix.

3. Incremental Learning with a Memory Constraint

Our algorithm processes data in small groups composed of E documents, which we refer to as *epochs*. After the arrival of each epoch the algorithm proceeds in two stages: a model building phase during which a new model estimate is produced, and a compression phase in which decisions are made as to which words and documents to clump. The sufficient statistics of each clump are computed and data summarized by clumps are purged from memory. The assignment distributions $q(z)$ of purged data and topic distributions of merged documents $q(\boldsymbol{\pi})$ are discarded as well. The clump sufficient statistics are retained along with the current model estimate, which serves as a starting point for the next round of learning.

Model Building Phase (Algorithm 3.1)

Input: Previous model $\{\xi_{kl}, \zeta_{ks}, \alpha_k, \Phi^c_{kl}, N_{cs}, D_s\}$, and current epoch of E documents.
Initialize $\zeta_{jk} = \alpha_k$ for $j = |S| + 1, \cdots, |S| + E$
Iterate eqs. 21, 18, 19, 20, and 22 until convergence
repeat
  Rank splits and merges according to criteria in (Ueda et al., 1999)
  for $i = 1$ **to** 10 **do**
    Split i-th ranked candidate topic along principal component
    Restricted iteration of eqs. 21, 18, 19, and 20 until convergence
    Evaluate change in eq. 29 resulting from split
  end for
  for $i = 1$ **to** 10 **do**
    Merge i-th ranked pair of topics
    Evaluate change in eq. 29 resulting from merge
  end for
  Select split or merge that yielded largest change in eq. 29
  Iterate eqs. 21, 18, 19, and 20 until convergence
until Change in eq. 29 is less than threshold

3.1. Model Building Phase

The model building phase optimizes the free energy under the parameter tying constraints induced by the choice of clumps in previous compression phases. We perform a split-merge procedure similar to (Ueda et al., 1999) to determine the number of topics, using the heuristics in that work to rank topic suitability for split or merge. In our experiments we use Gaussian topic distributions, so splits are proposed along the principal component of the topic. The split proposals are refined by restricted variational updates. That is: equations 21, 18, 19, 20, and 22 are iterated but only for data-points whose highest responsibility is to the split topic, and the points may be assigned only to the two descendent topics. Merges are carried out by instantiating a new topic with the data-points with highest responsibility to the merged topics. A total of 10 splits and 10 merges are proposed, and evaluated by the resultant change in free energy (eq. 29). The top ranked change is then used to initialize full variational updates (which involve all data points). The model building phase halts once the change in free energy divided by its previous value is below a threshold, which was chosen to be $1E-5$ in our experiments. The procedure is summarized in algorithm 3.1.

3.2. Compression Phase

The goal of the compression phase is to determine groups of data-points that are to be summarized by clumps, and to identify documents that are to be merged into document groups.

Clumps are identified using a greedy top down splitting procedure. Because datapoints summarized by clumps are ultimately discarded, the compression process is irreversible. Therefore it is of fundamental importance to predict the locations of future data when deciding which points to clump. In order to estimate this, we rank cluster splits according to a modified free energy (eq. 30) in which the data sample size is artificially increased by a factor $\frac{T_{pts}}{\sum_c N_c}$ and the number of documents is scaled by $\frac{T_{docs}}{\sum_s D_s}$, where T_{pts} and T_{docs} are the target number of data-points and documents expected during the lifetime of the system. This is equivalent to using the data empirical distribution as a predictive model of future data. If we determine clumps using the standard free energy, then the algorithm fails to split large groups of points that are likely to split once more data has arrived. Instead, it wastes memory by placing "stray" points in their own clumps.

We initialize the process by hard assigning each clump or data-point to the cluster with highest responsibility during the previous model building phase. We then proceed through each cluster and split it along the principal component, and refine this split by iterating restricted variational updates equations for the points in the cluster. The updates are modified by the data magnification factors:

$$\xi_{kl} = \left(\frac{T_{pts}}{\sum_c N_c}\right) F_{kl} + \beta_l \tag{23}$$

$$\xi_{k0} = \left(\frac{T_{pts}}{\sum_c N_c}\right) N_k + \beta_0 \tag{24}$$

$$\alpha_k \leftarrow \frac{(a-1) + \left(\frac{T_{docs}}{\sum_s D_s}\right) \alpha_k \sum_j \left[\psi(\zeta_{ks}) - \psi(\alpha_k)\right]}{b + \left(\frac{T_{docs}}{\sum_s D_s}\right) \sum_s \left[\psi(\zeta_s) - \psi(\alpha)\right]} \tag{25}$$

Updates for q_{ck} and ζ_{ks} are unchanged. After the clusters are refined, the data-points are then hard assigned to the sub-cluster with greatest responsibility, and the proposed split is ranked according to the resultant change in eq. 30. We then greedily split the cluster with highest rank. The process repeats itself, with new clusters ranked in the same way described above. We cache the results of each split evaluation to avoid redundant computation. After we have reached a given memory bound we extract the partitions resulting from this recursive splitting procedure as our new clumps.

Each clump must store sufficient statistics for full covariance Gaussian components which require $\frac{d^2+3d}{2}$ values, where d is the dimension of the feature space. In addition, $|S|$ (the number of document groups) values must be

Clump Compression (Algorithm 3.2)

Input: Output from model building phase: $\{q_{ck}, \Phi^c_{kl}, N_{cs}, D_s\}$, current epoch of E documents and memory bound M.
Hard partition clumps: $r_c = \arg\max_k q_{ck}$
while $MC < M$ (eq. 26) **do**
  for $i = 1$ **to** K **do**
    Split i-th cluster along principal component
    Iterate data magnified restricted updates until convergence
    Hard partition clumps into child clusters
    Evaluate change in eq. 30 resulting from split
  end for
  Select split that yielded largest change in eq. 30
  $K = K + 1$
end while

stored to represent the counts N_{cs} for each clump. Note that from this perspective, it only makes sense to create clumps within a cluster if it contains more than $\frac{d+3}{2} + \frac{1}{d}$ data-points. If not, then it is more efficient to store the individual data-points and we refer to them as "singlets". The total memory cost of summarizing the data is then

$$MC = \left(\frac{d^2 + 3d}{2}\right)|N_c > 1| + |S||N_c > 1| + d|N_c = 1|, \tag{26}$$

where $|N_c > 1|$ is the number of clumps with more than 1 data-item in them, and $|N_c = 1|$ is the number of singlets. The clump compression procedure is summarized in algorithm 3.2.

Document merging provides another way of controlling the memory cost, by reducing the number of image groups $|S|$. We use the following simple heuristic to rank the suitability of merging document groups s and s':

$$DM_{s,s'} = \frac{\sum_k \mathbb{E}[\pi_{sk}]\mathbb{E}[\pi_{s'k}]}{||\mathbb{E}[\boldsymbol{\pi}_s]||||\mathbb{E}[\boldsymbol{\pi}_{s'}]||} \tag{27}$$

Clumping and document merging enable a number of potential schemes for controlling space and time costs, depending on the application. We note that the time complexity per variational iteration scales as $O(K(|N_c > 1| + |N_c = 1|) + |S|K)$ and the space required to store $q(z_c)$ distributions is $O(K(|N_c > 1| + |N_c = 1|))$.

4. Experiments

We test our approach with two machine vision experiments. The first is an image segmentation task, and the second is an object recognition and retrieval task.

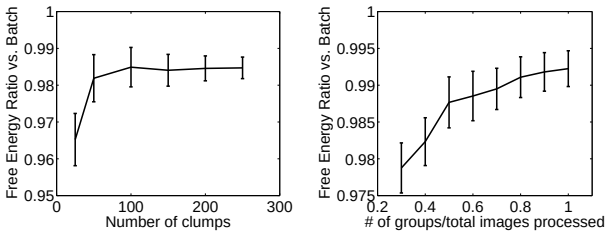


Figure 1. Image Segmentation experiment. Left: Free energy ratio as a function of the number of clumps permitted by the memory bound. Right: Free energy ratio versus the number of image groups relative to the total number of images processed.

4.1. Joint Image Segmentation

Our first experient is a joint image segmentation problem. The dataset is the Faces-Easy category of the Caltech 101 image dataset (Fei-Fei et al., 2004) consisting of 435 images. Each image contains a face centered in the image, but the lighting conditions and background vary. In terms of the vocabulary of the preceding sections, each image is a document and each pixel in the image is a word. Pixels are represented as five dimensional vectors of the following features: X and Y position relative to the center of the image, and three color coordinates in the CIELAB colorspace. The goal of our experiment is to find similar image regions across the multiple images, in an unsupervised way. We emphasize that our main objective is to study the efficiency of our algorithm, not to produce a state of the art image segmentation algorithm.

The images were scaled to be 200 by 160 pixels in size. Thus, the total size of the dataset is 32,000 pixels per image, times 435 images, times 5 features per pixel equals 69,600,000 real numbers. Each pixel requires an assignment distribution. Our baseline implementation (i.e. a batch algorithm that processes all images in memory at once and does not use pixel clumping or image merging) was only able to jointly segment 30 images simultaneously, before running out of memory. The majority of memory is used to store the assignment distributions of pixels, and this is problematic as the number of topics increases during learning, since the space requirements scale as $O(NK)$, where N is the total number of pixels and K is the number of topics.

We first compare the memory bounded approach to the baseline implementation on a joint segmentation task of 30 images in order to judge the impact of the pixel clumping approximation. We vary the upper limit on the number of clumps used to summarize the data during the compression phase, and compare the free energy bounds produced by the memory bounded algorithm to those produced by the baseline implementation. We define the free energy ratio

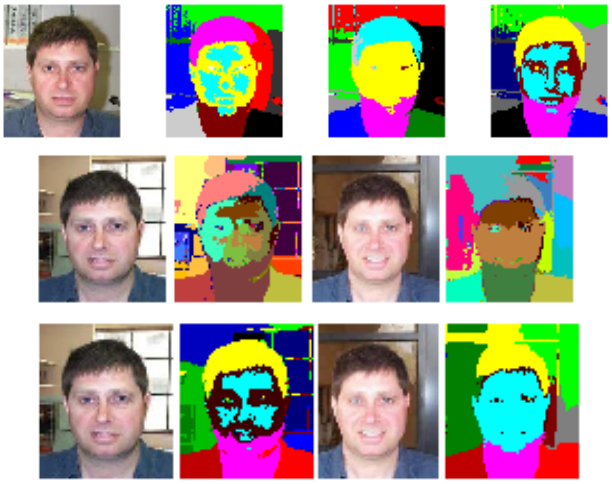


Figure 2. Top row: From left to right: an example segmentation produced by the baseline method, memory bounded algorithm with 30% of total images and 125 clumps, and the memory bounded algorithm with no images merged and 125 clumps. Row 2: Example clump distributions. Pixels of the same color are summarized in a single clump. Row 3: segmentations corresponding to clumps in row 2.

as $1 - \frac{FE_{batch} - FE_{mb}}{|FE_{batch}|}$. This process was repeated for different subsets of 30 images from the dataset. In the memory bounded approach, images were processed in epochs of five images at a time. Figure 1 summarizes the results. We find that performance tends to saturate beyond a certain number of clumps.

We also note a significant run time advantage of the memory bounded algorithm over the batch method. The average run time of the batch method was 3.09 hours versus 0.68 hours for the memory bounded approach.

Next we study the impact of image (document) merges on the relative performance of the memory bounded algorithm versus the baseline batch algorithm, while varying the maximum number of image (document) groups permitted. The results are shown in figure 1.

We find little qualitative difference between segmentations produced by the baseline and memory bounded algorithms. The possible exception is in the case when the memory bounded algorithm is run with a large number of image merges, in which case the algorithm seemed to discover fewer topics than the batch and memory bounded algorithm with only word clumping. Example image segmentations and clump distributions are shown in figure 2.

Finally, we demonstrate the memory bounded algorithm on the full dataset of 435 images, which is more than an order of magnitude larger than can be handled with the baseline algorithm. We process images in *epochs* of 10 images at a time, for a total of 44 learning rounds. The upper limit on the number of clumps was set to 1000, which was likely many more than required since there were only 85 inferred topics. Because the number of documents was relatively small, we chose not to use document merges. The total run time of the algorithm was 15 hours. Figure 3 shows the number of topics as a function of the number of images processed, and the run time required during each image round. The run time is longer during learning rounds in which more new topics are discovered, because more split-merge operations are necessary. The memory required for the memory bounded algorithm was 22 MB to store the current image epoch and clumps, less than 1MB for the current model estimate, and 235 MB for assignment distributions, for a total of 257 MB. In contrast, the baseline batch implementation would have required 531 MB to store all 435 images, 8.8155 GB to store assignment distributions for each pixel assuming 85 topics, and less than 1 MB for the model, for a total of 9.3 GB. (All memory amounts assume double precision floating point.) The memory bounded implementation therefore achieved a memory savings factor of about 38 with very little loss in accuracy.

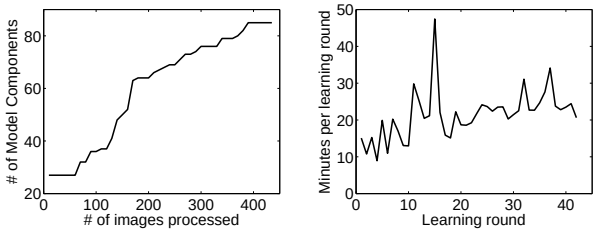


Figure 3. Joint segentation of 435 faces. The left plot shows the number of topics recovered as the system processes images. The right plot shows the run time for each learning round. This fluctuates with the number of new topics discovered during each round and tends to increase gradually with the total number of topics.

Figure 4 shows example joint segmentations produced by the memory bounded algorithm. These images were retrieved by first computing responsibilities for every image in the dataset, with respect to the final model estimate produced by the MB algorithm. Then, the images were sorted according to those that have the most pixels assigned to the largest topic. The largest topic indeed corresponds to a face, and is represented by the olive green segment in the figure. Other topics shared across images include hair and certain backgrounds.

4.2. Object Recognition and Retrieval

Our object recognition and retrieval experiment involves all 101 object categories in the Caltech 101 dataset. We randomly select 3000 training images and 1000 test images. We extract 128-dimensional SIFT (Lowe, 2004) local ap-

Figure 4. Examples of joint segmentation produced after processing all Caltech Face images. Pixels that are the same color have highest responsibility to the same topic. These images were retrieved by sorting images according to those that have the most pixels assigned to the largest topic, which is the olive green colored face segment in each image.

pearance descriptors from 500 randomly chosen locations in each image. The scale of each feature is also chosen randomly. In the language of topic models, each feature descriptor is a word, and the collection of feature descriptors in an image forms a document. This image representation is known as 'bag-of-features', because images are modeled as unordered collections of feature descriptors whose geometric positions are ignored. This dataset proved too large to compare directly to the batch algorithm

We train a single topic model on all training images, using epochs of 60 images at a time. Because hundreds of topics are discovered we use diagonal covariance Gaussians and adjust equation 26 accordingly. Given a test image $\tilde{\mathbf{x}}$, retrieval is performed by ranking each training image's similarity to the test image. To develop the similarity measure we begin with $\log \prod_i p(\tilde{\mathbf{x}}_{ij}|\mathbf{x})$, which is the log-probability that the detections in the test image were generated by training image j given the training set. Then we variationally lower bound this quantity to obtain a test free energy and drop all constant terms not involving the test image and index j. Finally we lower bound this quantity by assuming that detections in the test image are hard assigned to the topic with highest responsibility (this leads to an expression that is much faster to evaluate with neglible impact on retrieval performance.) The retrieval score is:

$$score(j) = \sum_i \max_k \Big\{ \sum_l \mathbb{E}[\eta_{kl}|\xi_{kl}]\ \phi_{kl}(\tilde{\mathbf{x}}_{ij}) - \mathbb{E}[A_k(\boldsymbol{\eta}_k)|\xi_{k0}] + \psi(\zeta_{kj}) - \psi(\sum_k \zeta_{kj}) \Big\} \quad (28)$$

where the expectations are with respect to $q(\boldsymbol{\eta})$ learned during training and ξ_{kl} and ζ_{kj} are from training as well. ζ_{kj}

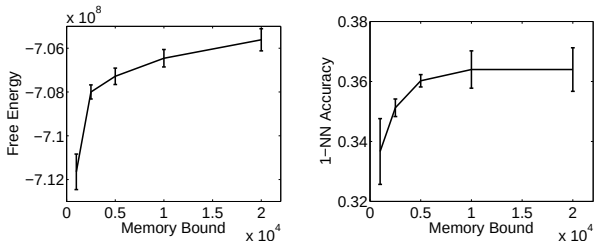


Figure 5. Object Recognition and Retrieval. Left: Training set free energy as a function of the memory bound. Right: 1-NN classification accuracy as a function of memory bound (measured as the equivalent number of data-points that could be stored in the same space).

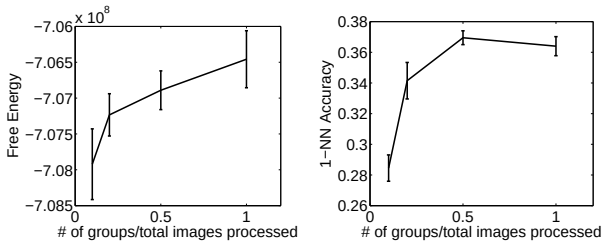


Figure 6. Object Recogniton and Retrieval. Left: Training set free energy versus the ratio of document groups to the total number of images processed. Right: 1-NN classification accuracy versus the ratio of document groups to total number of images processed.

are re-estimated for images that were merged into a document group during training. We compute nearest neighbor (1-NN) classification accuracy by classifying the test image to the class label of the highest scoring image in the training set.

Figure 5 shows the training set free energy and 1-NN classification accuracy as a function of the memory bound M (measured as the equivalent number of data points that could be stored in the same space.) Because we used diagonal covariance matrices, there were enough clumps even at low levels of memory to maintain comparable classification performance. We note that the training free energy increases with memory as expected, and that the 1-NN accuracy tends to saturate as memory increases.

Figure 6 shows the 1-NN accuracy and training free energy when the percentage of document groups relative to the number of total images processed is varied (the memory bound M is held fixed at 10000). We note that the classification performance suffers substantially when only small numbers of document groups are permitted. We use a heuristic for determining documents to merge (eq. 27). It is possible that a well motivated criterion (perhaps derived from the free energy) would give better performance.

5. Conclusion

Machine learning has largely focussed on algorithms that run for a relatively short period of time, fitting models of finite capacity on a data-set of fixed size. We believe that this scenario is unrealistic if we aim at building truly intelligent systems. We have identified nonparametric Bayesian models as promising candidates that expand their model complexity in response to new incoming data. The flip-side is that nonparametric Bayesian algorithms are "example-based" and as such require one to cache and process repeatedly every data-case ever seen. The objectives of infinite, adaptive model capacity on the one hand and efficiency, both in time and space on the other therefore seem to be fundamentally at odds with each other.

In this paper we have made a first step towards resolving this issue by introducing a class of models that can adapt their model complexity adaptively but are able to do so at a fraction of the memory requirements and processing times necessary for their batch counterparts. There is no magic of course: with a fixed memory budget there is a limit to how complex the model can be, but we have shown that one can learn much larger models reliably with much less memory than a naive implementation would allow. Moreover, our learning algorithms allow a flexible tradeoff between memory requirements and model complexity requirements that can be adapted online.

Intuitively, our method may be thought of as a two level clustering process. At the bottom level, data is clustered into clumps in order to limit time and space costs. At the top level, clumps are clustered to form topics in order to ensure good generalization performance.

Potential application areas of the techniques introduced here are manyfold. For instance, we can imagine learning topic models from very large text corpora or the world wide web to understand its structure and facilitate fast searching algorithms. Another exciting direction is to build a taxonomy of visual object categories from a continuous stream of video data captured by surveillance cameras.

5.1. Acknowledgements

We thank the anonymous reviewers for their helpful comments. This material is based on work supported by the National Science Foundation under grant numbers 0447903 and 0535278, the Office of Naval Research under grant numbers 00014-06-1-0734 and 00014-06-1-0795, and The National Institutes of Health Predoctoral Training in Integrative Neuroscience grant number T32 GM007737.

5.2. Appendix

The following expressions for the free energy are used in the main text. Note that they are only valid after the updates for $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ have been performed.

$$\begin{aligned}\mathcal{F} = & KB(\boldsymbol{\beta}) - \textstyle\sum_k B_k(\mathbf{F}_k + \boldsymbol{\beta}) \\ & + \textstyle\sum_{ks} D_s \log\Big(\Gamma(\alpha_k)/\Gamma(\alpha_k + \frac{N_{ks}}{D_s})\Big) - \sum_s D_s \log\Big(\Gamma(\alpha)/\Gamma(\alpha + \frac{N_s}{D_s})\Big) \\ & + \textstyle\sum_{ck} N_c q_{ck} \log q_{ck} \\ & - \textstyle\sum_k \Big((a-1)\sum_k \log(\alpha_k) - b\sum_k \alpha_k\Big) \\ & - (K_{\max} - K)\Big((a-1)\log\tfrac{a-1}{b} - (a-1)\Big) - K_{\max}(b\log(a) - \log\Gamma(a))\end{aligned} \tag{29}$$

$$\begin{aligned}\mathcal{F} = & KB(\boldsymbol{\beta}) - \textstyle\sum_k B_k(\Big(\frac{T_{pts}}{\sum_c N_c}\Big)\mathbf{F}_k + \boldsymbol{\beta}) \\ & + \Big(\frac{T_{docs}}{\sum_s D_s}\Big)\textstyle\sum_{ks} D_s \log\Big(\Gamma(\alpha_k)/\Gamma(\alpha_k + \frac{N_{ks}}{D_s})\Big) \\ & - \Big(\frac{T_{docs}}{\sum_s D_s}\Big)\textstyle\sum_s D_s \log\Big(\Gamma(\alpha)/\Gamma(\alpha + \frac{N_s}{D_s})\Big) \\ & + \Big(\frac{T_{pts}}{\sum_c N_c}\Big)\textstyle\sum_{ck} N_c q_{ck} \log q_{ck} \\ & - \textstyle\sum_k \Big((a-1)\sum_k \log(\alpha_k) - b\sum_k \alpha_k\Big) \\ & - (K_{\max} - K)\Big((a-1)\log\tfrac{a-1}{b} - (a-1)\Big) - K_{\max}(b\log(a) - \log\Gamma(a))\end{aligned} \tag{30}$$

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Fei-Fei, L., Fergus, R., & Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *IEEE CVPR Workshop of Generative Model Based Vision (WGMBV)*.

Ferguson, T. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, *1*, 209–230.

Kurihara, K., Welling, M., & Vlassis, N. (2006). Accelerated variational dirichlet process mixtures. *NIPS*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

Minka, T. (2000). *Estimating a dirichlet distribution* (Technical Report).

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet processes. *To appear in Journal of the American Statistical Association*.

Teh, Y. W., Kurihara, K., & Welling, M. (2008). Collapsed variational inference for HDP. *Advances in Neural Information Processing Systems*.

Ueda, N., Nakano, R., Gharamani, Z., & Hinton, G. (1999). Smem algorithm for mixture models.

Verbeek, J., Nunnink, J., & Vlassis, N. (2003). *Accelerated variants of the em algorithm for gaussian mixtures* (Technical Report). University of Amsterdam.

Localized Multiple Kernel Learning

Mehmet Gönen GONEN@BOUN.EDU.TR
Ethem Alpaydın ALPAYDIN@BOUN.EDU.TR
Department of Computer Engineering, Boğaziçi University, TR-34342, Bebek, İstanbul, Turkey

Abstract

Recently, instead of selecting a single kernel, multiple kernel learning (MKL) has been proposed which uses a convex combination of kernels, where the weight of each kernel is optimized during training. However, MKL assigns the same weight to a kernel over the whole input space. In this paper, we develop a localized multiple kernel learning (LMKL) algorithm using a gating model for selecting the appropriate kernel function locally. The localizing gating model and the kernel-based classifier are coupled and their optimization is done in a joint manner. Empirical results on ten benchmark and two bioinformatics data sets validate the applicability of our approach. LMKL achieves statistically similar accuracy results compared with MKL by storing fewer support vectors. LMKL can also combine multiple copies of the same kernel function localized in different parts. For example, LMKL with multiple linear kernels gives better accuracy results than using a single linear kernel on bioinformatics data sets.

1. Introduction

Kernel-based methods such as the support vector machine (SVM) gained much popularity due to their success. For classification tasks, the basic idea is to map the training instances from the input space to a feature space (generally a higher dimensional space than the input space) where they are linearly separable. The SVM discriminant function obtained after training is:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle + b \tag{1}$$

where $\boldsymbol{w}$ is the weight coefficients, b is the threshold, and $\Phi(\boldsymbol{x})$ is the mapping function to the corresponding feature space. We do not need to define the mapping function explicitly and if we plug $\boldsymbol{w}$ vector from dual formulation into (1), we obtain the discriminant:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \underbrace{\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_i) \rangle}_{K(\boldsymbol{x}, \boldsymbol{x}_i)} + b$$

where n is the number of training instances, $\boldsymbol{x}_i$, and $K(\boldsymbol{x}, \boldsymbol{x}_i) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_i) \rangle$ is the corresponding kernel.

Each $\Phi(\boldsymbol{x})$ function has its own characteristics and corresponds to a different kernel function and leads to a different discriminant function in the original space. Selecting the kernel function (i.e., selecting the mapping function) is an important step in SVM training and is generally performed using cross-validation.

In recent studies (Lanckriet et al., 2004a; Sonnenburg et al., 2006), it is reported that using multiple different kernels instead of a single kernel improves the classification performance. The simplest way is to use an unweighted sum of kernel functions (Pavlidis et al., 2001; Moguerza et al., 2004). Using an unweighted sum gives equal preference to all kernels and this may not be ideal. A better strategy is to learn a weighted sum (e.g., convex combination); this also allows extracting information from the weights assigned to kernels. Lanckriet et al. (2004b) formulate this as a semidefinite programming problem which allows finding the combination weights and support vector coefficients together. Bach et al. (2004) reformulate the problem and propose an efficient algorithm using sequential minimal optimization (SMO). Their discriminant function can be seen as an unweighted summation of discriminant values (but a weighted summation of kernel functions) in different feature spaces:

$$f(\boldsymbol{x}) = \sum_{m=1}^{p} \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}) \rangle + b \tag{2}$$

where m indexes kernels, $\boldsymbol{w}_m$ is the weight coefficients, $\Phi_m(\boldsymbol{x})$ is the mapping function for feature space m, and p is the number of kernels. By plugging $\boldsymbol{w}_m$ de-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

rived from duality conditions into (2), we obtain:

$$f(\boldsymbol{x}) = \sum_{m=1}^{p} \eta_m \sum_{i=1}^{n} \alpha_i y_i \underbrace{\langle \Phi_m(\boldsymbol{x}), \Phi_m(\boldsymbol{x}_i) \rangle}_{K_m(\boldsymbol{x}, \boldsymbol{x}_i)} + b \quad (3)$$

where the kernel weights satisfy $\eta_m \geq 0$ and $\sum_{m=1}^{p} \eta_m = 1$. The kernels we combine can be the same kernel with different hyperparameters (e.g., degree in polynomial kernel) or different kernels (e.g., linear, polynomial, and Gaussian kernels). We can also combine kernels over different data representations or different feature subsets.

Using a fixed combination rule (unweighted or weighted) assigns the same weight to a kernel over the whole input space. Assigning different weights to a kernel in different regions of the input space may produce a better classifier. If data has underlying localities, we should give higher weights to appropriate kernel functions (i.e., kernels which match the complexity of data distribution) for each local region. Lewis et al. (2006) propose to use a nonstationary combination method derived with a large-margin latent variable generative method. They use a log-ratio of Gaussian mixtures as the classifier. Lee et al. (2007) combine Gaussian kernels with different width parameters to capture the underlying local distributions, by forming a compositional kernel matrix from Gaussian kernels and using it to train a single classifier.

In this paper, we introduce a localized formulation of the multiple kernel learning (MKL) problem. In Section 2, we modify the discriminant function of the MKL framework proposed by Bach et al. (2004) with a localized one and describe how to optimize the parameters with a two-step optimization procedure. Section 3 explains the key properties of the proposed algorithm. We then demonstrate the performance of our localized multiple kernel learning (LMKL) method on toy, benchmark, and bioinformatics data sets in Section 4. We conclude in Section 5.

2. Localized Multiple Kernel Learning

We describe the LMKL framework for binary classification SVM but the derivations in this section can easily be extended to other kernel-based learning algorithms. We propose to rewrite the discriminant function (2) of Bach et al. (2004) as follows, in order to allow local combinations of kernels:

$$f(\boldsymbol{x}) = \sum_{m=1}^{p} \eta_m(\boldsymbol{x}) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}) \rangle + b \quad (4)$$

where $\eta_m(\boldsymbol{x})$ is the *gating function* which chooses feature space m as a function of input $\boldsymbol{x}$. $\eta_m(\boldsymbol{x})$ is defined up to a set of parameters which are also learned from data, as we will discuss below. By modifying the original SVM formulation with this new discriminant function, we get the following optimization problem:

$$\begin{aligned} &\min \frac{1}{2} \sum_{m=1}^{p} \|\boldsymbol{w}_m\|^2 + C \sum_{i=1}^{n} \xi_i \\ &\text{w.r.t. } \boldsymbol{w}_m, b, \boldsymbol{\xi}, \eta_m(\boldsymbol{x}) \\ &\text{s.t. } y_i \left(\sum_{m=1}^{p} \eta_m(\boldsymbol{x}_i) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i) \rangle + b \right) \geq 1 - \xi_i \quad \forall i \\ &\xi_i \geq 0 \quad \forall i \end{aligned} \quad (5)$$

where C is the regularization parameter and $\boldsymbol{\xi}$ is the slack variables as usual. Note that the optimization problem in (5) is not convex due to the nonlinearity introduced in the separation constraints.

Instead of trying to solve (5) directly, we can use a two-step alternate optimization algorithm inspired from Rakotomamonjy et al. (2007), to find the parameters of $\eta_m(\boldsymbol{x})$ and the discriminant function. The first step is to solve (5) with respect to $\boldsymbol{w}_m$, b, and $\boldsymbol{\xi}$ while fixing $\eta_m(\boldsymbol{x})$ and the second step is to update the parameters of $\eta_m(\boldsymbol{x})$ using a gradient-descent step calculated from the objective function in (5). The objective value obtained for a fixed $\eta_m(\boldsymbol{x})$ is an upper bound for (5) and the parameters of $\eta_m(\boldsymbol{x})$ are updated according to the current solution. The objective value obtained at the next iteration can not be greater than the current one due to the use of gradient-descent procedure and as iterations progress with a proper step size selection procedure (see Section 3.1), the objective value of (5) never increases. Note that this does not guarantee convergence to the global optimum and the initial parameters of $\eta_m(\boldsymbol{x})$ may affect the solution quality.

For a fixed $\eta_m(\boldsymbol{x})$, we obtain the Lagrangian of the primal problem in (5) as follows:

$$\begin{aligned} L_D = &\frac{1}{2} \sum_{m=1}^{p} \|\boldsymbol{w}_m\|^2 + \sum_{i=1}^{n} (C - \alpha_i - \beta_i) \xi_i + \sum_{i=1}^{n} \alpha_i \\ &- \sum_{i=1}^{n} \alpha_i y_i \left(\sum_{m=1}^{p} \eta_m(\boldsymbol{x}_i) \langle \boldsymbol{w}_m, \Phi_m(\boldsymbol{x}_i) \rangle + b \right) \end{aligned}$$

and taking the derivatives of L_D with respect to the primal variables gives:

$$\begin{aligned} &\frac{\partial L_D}{\partial \boldsymbol{w}_m} \Rightarrow \boldsymbol{w}_m = \sum_{i=1}^{n} \alpha_i y_i \eta_m(\boldsymbol{x}_i) \Phi_m(\boldsymbol{x}_i) \quad \forall m \\ &\frac{\partial L_D}{\partial b} \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \\ &\frac{\partial L_D}{\partial \xi_i} \Rightarrow C = \alpha_i + \beta_i \quad \forall i \,. \end{aligned} \quad (6)$$

From (5) and (6), the dual formulation is obtained as:

$$\begin{aligned}
&\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&\text{w.r.t. } \boldsymbol{\alpha} \\
&\text{s.t. } \sum_{i=1}^{n} \alpha_i y_i = 0 \\
&\quad C \geq \alpha_i \geq 0 \quad \forall i
\end{aligned} \tag{7}$$

where the *locally combined kernel matrix* is defined as:

$$K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{m=1}^{p} \eta_m(\boldsymbol{x}_i) \underbrace{\langle \Phi_m(\boldsymbol{x}_i), \Phi_m(\boldsymbol{x}_j) \rangle}_{K_m(\boldsymbol{x}_i, \boldsymbol{x}_j)} \eta_m(\boldsymbol{x}_j) \; .$$

This formulation corresponds to solving a canonical SVM dual problem with the kernel matrix $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$, which should be positive semidefinite. We know that multiplying a kernel function with outputs of a nonnegative function for both input instances, known as quasi-conformal transformation, gives a positive semidefinite kernel matrix (Amari & Wu, 1998). So, the locally combined kernel matrix can be viewed as applying a quasi-conformal transformation to each kernel function and summing them to construct a combined kernel matrix. The only restriction is to have nonnegative $\eta_m(\boldsymbol{x})$ to get a positive semidefinite kernel matrix.

Choosing among possible kernels can be considered as a classification problem and we assume that the regions of use of kernels are linearly separable. In this case, the gating model can be expressed as:

$$\eta_m(\boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{v}_m, \boldsymbol{x} \rangle + v_{m0})}{\sum_{k=1}^{p} \exp(\langle \boldsymbol{v}_k, \boldsymbol{x} \rangle + v_{k0})}$$

where $\boldsymbol{v}_m, v_{m0}$ are the parameters of this gating model and the softmax guarantees nonnegativity. One can use more complex gating models for $\eta_m(\boldsymbol{x})$ or equivalently implement the gating not in the original input space but in a space defined by a basis function, which can be one or some combination of the $\Phi_m(\boldsymbol{x})$ in which the SVM works (thereby also allowing the use of nonvectorial data). If we use a gating model which is constant (not a function of $\boldsymbol{x}$), our algorithm finds a fixed combination over the whole input space, similar to the original MKL formulation.

The proposed method differs from taking subsets of the training set and training a classifier in each subset then combining them. For example, Collobert et al. (2001) define such a procedure which learns an independent SVM for each subset and reassigns instances to subsets by training a gating model with a cost function. Our approach is different in that LMKL couples subset selection and combination of local classifiers in a joint optimization problem. LMKL is similar to but also different from the mixture of experts framework (Jacobs et al., 1991) in the sense that the gating model combines kernel-based experts and is learned together with experts; the difference is that in the mixture of experts, experts individually are classifiers whereas in our formulation, there is no discriminant per kernel.

For a given $\eta_m(\boldsymbol{x})$, we can say that the objective value of (7) is equal to the objective value of (5) due to strong duality. We can safely use the objective function of (7) as $J(\eta)$ function to calculate the gradients of the primal objective with respect to the parameters of $\eta_m(\boldsymbol{x})$. To train the gating model, we take derivatives of $J(\eta)$ with respect to $\boldsymbol{v}_m, v_{m0}$ and use gradient-descent:

$$\begin{aligned}
\frac{\partial J(\eta)}{\partial v_{m0}} &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{p} \alpha_i \alpha_j y_i y_j \eta_k(\boldsymbol{x}_i) K_k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&\qquad \eta_k(\boldsymbol{x}_j)\Big(\delta_m^k - \eta_m(\boldsymbol{x}_i) + \delta_m^k - \eta_m(\boldsymbol{x}_j)\Big) \\
\frac{\partial J(\eta)}{\partial \boldsymbol{v}_m} &= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{p} \alpha_i \alpha_j y_i y_j \eta_k(\boldsymbol{x}_i) K_k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&\qquad \eta_k(\boldsymbol{x}_j)\Big(\boldsymbol{x}_i\left[\delta_m^k - \eta_m(\boldsymbol{x}_i)\right] + \boldsymbol{x}_j\left[\delta_m^k - \eta_m(\boldsymbol{x}_j)\right]\Big)
\end{aligned}$$

where δ_m^k is 1 if $m = k$ and 0 otherwise. After updating the parameters of $\eta_m(\boldsymbol{x})$, we are required to solve a single kernel SVM with $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ at each step.

The complete algorithm of LMKL with the linear gating model is summarized in Algorithm 1. Convergence of the algorithm can be determined by observing the change in $\boldsymbol{\alpha}$ or the parameters of $\eta_m(\boldsymbol{x})$.

Algorithm 1 LMKL with the linear gating model

1: Initialize $\boldsymbol{v}_m$ and v_{m0} to small random numbers for $m = 1, \ldots, p$
2: **repeat**
3: Calculate $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with gating model
4: Solve canonical SVM with $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$
5: $v_{m0}^{(t+1)} \Leftarrow v_{m0}^{(t)} - \mu^{(t)} \dfrac{\partial J(\eta)}{\partial v_{m0}}$ for $m = 1, \ldots, p$
6: $\boldsymbol{v}_m^{(t+1)} \Leftarrow \boldsymbol{v}_m^{(t)} - \mu^{(t)} \dfrac{\partial J(\eta)}{\partial \boldsymbol{v}_m}$ for $m = 1, \ldots, p$
7: **until** convergence

After determining the final $\eta_m(\boldsymbol{x})$ and SVM solution, the resulting discriminant function is:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\sum_{m=1}^{p} \alpha_i y_i \eta_m(\boldsymbol{x}) K_m(\boldsymbol{x}, \boldsymbol{x}_i) \eta_m(\boldsymbol{x}_i) + b \; . \tag{8}$$

3. Discussions

We explain the key properties and possible extensions of the proposed algorithm in this section.

3.1. Computational Complexity

In each iteration, we are required to solve a canonical SVM problem with the combined kernel obtained with the current gating model and to calculate the gradients of $J(\eta)$. The gradient calculation step has ignorable time complexity compared to the SVM solver. The step size of each iteration, $\mu^{(t)}$, should be determined with a line search method which requires additional SVM optimizations for better convergence. The computational complexity of our algorithm mainly depends on the complexity of the canonical SVM solver used in the main loop, which can be reduced by using hot-start (i.e., giving previous $\boldsymbol{\alpha}$ as input). The number of iterations before convergence clearly depends on the training data and the step size selection procedure. The time complexity for testing is also reduced as a result of localizing. $K_m(\boldsymbol{x}, \boldsymbol{x}_i)$ in (8) needs to be evaluated only if both $\eta_m(\boldsymbol{x})$ and $\eta_m(\boldsymbol{x}_i)$ are nonzero.

3.2. Extensions to Other Kernel-Based Algorithms

LMKL can also be applied to kernel-based algorithms other than binary classification SVM, such as regression and one-class SVMs. We need to make two basic changes: (a) optimization problem and (b) gradient calculations from the objective value found. Otherwise, the same algorithm applies.

3.3. Knowledge Extraction

The MKL framework is used to extract knowledge about the relative contributions of kernel functions used in combination. If kernel functions are evaluated over different feature subsets or data representations, the important ones have higher combination weights. With our LMKL framework, we can deduce similar information based on different regions of the input space.

Our proposed method also allows combining multiple copies of the same kernel to obtain localized discriminants, thanks to the nonlinearity introduced by the gating model. For example, we can combine linear kernels with the gating model to obtain nearly piecewise linear boundaries.

4. Experiments

We implement the main body of our algorithm in C++ and solve the optimization problems with MOSEK optimization software (Mosek, 2008). Our experimental methodology is as follows: Given a data set, a random one-third is reserved as the test set and the remaining two-thirds is resampled using 5×2 cross-validation to generate ten training and validation sets, with stratification. The validation sets of all folds are used to optimize C by trying values 0.01, 0.1, 1, 10, and 100. The best configuration (the one that has the highest average accuracy on the validation folds) is used to train the final SVMs on the training folds and their performance is measured over the test set. So, for each data set, we have ten test set results.

We perform simulations with three commonly used kernels: linear kernel (K_L), polynomial kernel (K_P), and Gaussian kernel (K_G):

$$K_L(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$
$$K_P(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + 1)^q$$
$$K_G(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2 / s^2\right) .$$

We use the second degree ($q = 2$) polynomial kernel and estimate s in the Gaussian kernel as the average nearest neighbor distance between instances of the training set. All kernel matrices are calculated and normalized to unit trace before training. The step size of each iteration, $\mu^{(t)}$, is fixed as 0.01 without performing line search and a total of 50 iterations are performed.

4.1. Toy Data Set

In order to illustrate our proposed algorithm, we create a toy data set, named GAUSS4, which consists of 1200 data instances generated from four Gaussian components (two for each class) with the following prior probabilities, mean vectors and covariance matrices:

$$p_{11} = 0.25 \quad \mu_{11} = \begin{pmatrix} -3.0 \\ +1.0 \end{pmatrix} \quad \Sigma_{11} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$$
$$p_{12} = 0.25 \quad \mu_{12} = \begin{pmatrix} +1.0 \\ +1.0 \end{pmatrix} \quad \Sigma_{12} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 2.0 \end{pmatrix}$$
$$p_{21} = 0.25 \quad \mu_{21} = \begin{pmatrix} -1.0 \\ -2.2 \end{pmatrix} \quad \Sigma_{21} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$$
$$p_{22} = 0.25 \quad \mu_{22} = \begin{pmatrix} +3.0 \\ -2.2 \end{pmatrix} \quad \Sigma_{22} = \begin{pmatrix} 0.8 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$$

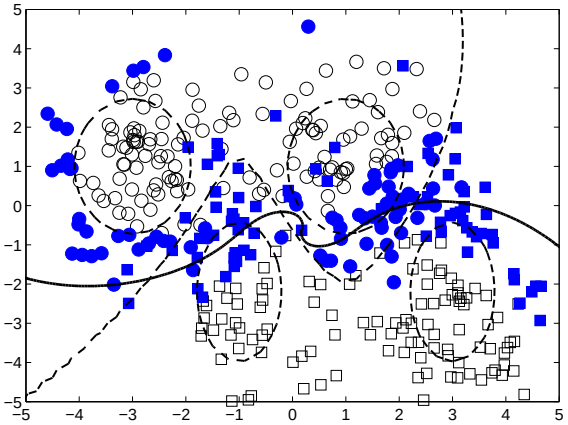
where data instances from the first two components are of class 1 (labeled as positive) and others are of class 2 (labeled as negative)[1]. We perform two sets of experiments on GAUSS4 data set: (K_L-K_P) and (K_L-K_L-K_L).

[1]MATLAB implementation of LMKL with an SMO-based canonical SVM solver and GAUSS4 dataset are available at `http://www.cmpe.boun.edu.tr/~gonen/lmkl`.

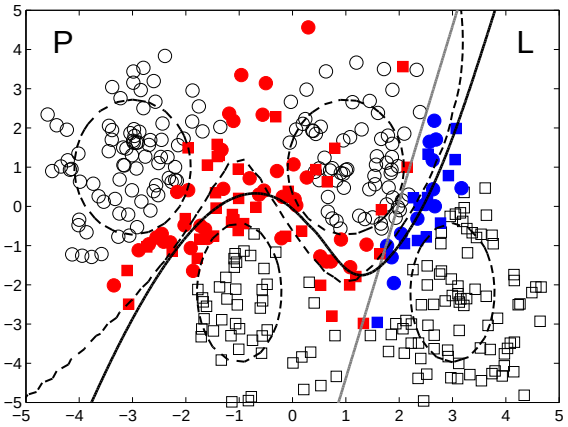
First, we train both MKL and LMKL for (K_L-K_P) combination. Figure 1(a) shows the classification boundaries calculated and the support vectors stored by MKL which assigns combination weights 0.30 and 0.70 to K_L and K_P, respectively. Using the kernel matrix obtained combining K_L and K_P with these weights, we do not achieve a good approximation to the optimal Bayes' boundary. As we see in Figure 1(b), LMKL divides the input space into two regions and uses the polynomial kernel to separate one component from two others quadratically and the linear kernel for the other component. We see that the locally combined kernel matrix obtained from K_L and K_P with the linear gating model learns a classification boundary very similar to the optimal Bayes' boundary. Note that the softmax function in the gating model achieves a smooth transition between kernels.

The effect of combining multiple copies of the same kernel can be seen in Figure 1(c) which shows the classification and gating model boundaries of LMKL with (K_L-K_L-K_L) combination. Using linear kernels in three different regions enables us to approximate the optimal Bayes' boundary in a piecewise linear manner. Instead of using complex kernels such as the Gaussian kernel, local combination of simple kernels (e.g., linear and polynomial kernels) can produce accurate classifiers and avoid overfitting. For example, the Gaussian kernel achieves 89.67 per cent average testing accuracy by storing all training instances as support vectors. However, LMKL with three linear kernels achieves 92.00 per cent average testing accuracy by storing 23.18 per cent of training instances as support vectors on the average.

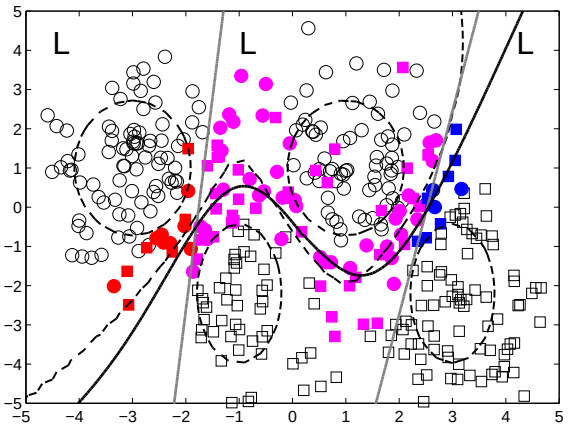
Initially, we assign small random numbers to the gating model parameters and this gives nearly equal combination weights for each kernel. This is equivalent to taking an unweighted summation of the original kernel matrices. The gating model starts to give crisp outputs as iterations progress and the locally combined kernel matrix becomes more sparse (see Figure 2). The kernel function values between data instances from different regions become 0 due to the multiplication of the gating model outputs. This localizing characteristics is also effective for the test instances. If the gating model gives crisp outputs for a test instance, the discriminant function in (8) is calculated over only the support vectors having nonzero gating model outputs for the selected kernels. Hence, discriminant function value for a data instance is mainly determined by the neighboring training instances and the active kernel function in its region.



(a) MKL with (K_L-K_P).

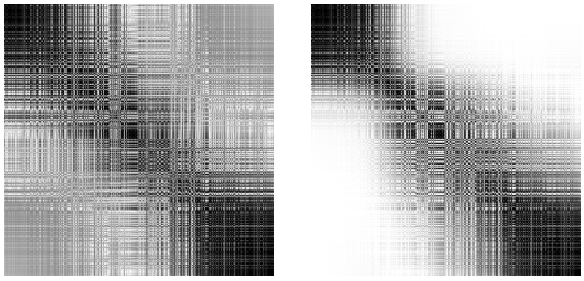


(b) LMKL with (K_L-K_P).



(c) LMKL with (K_L-K_L-K_L).

Figure 1. Separating hyperplanes (black solid lines) and support vectors (filled points) on GAUSS4 data set. Dashed lines show the Gaussians from which data are sampled and the optimal Bayes' discriminant. The gray solid lines shows the boundaries calculated from the gating models by considering them as classifiers which select a kernel function.



(a) First iteration (b) Last iteration

Figure 2. Locally combined kernel matrices, $K_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$, of LMKL with ($K_L$-$K_P$) on GAUSS4 data set.

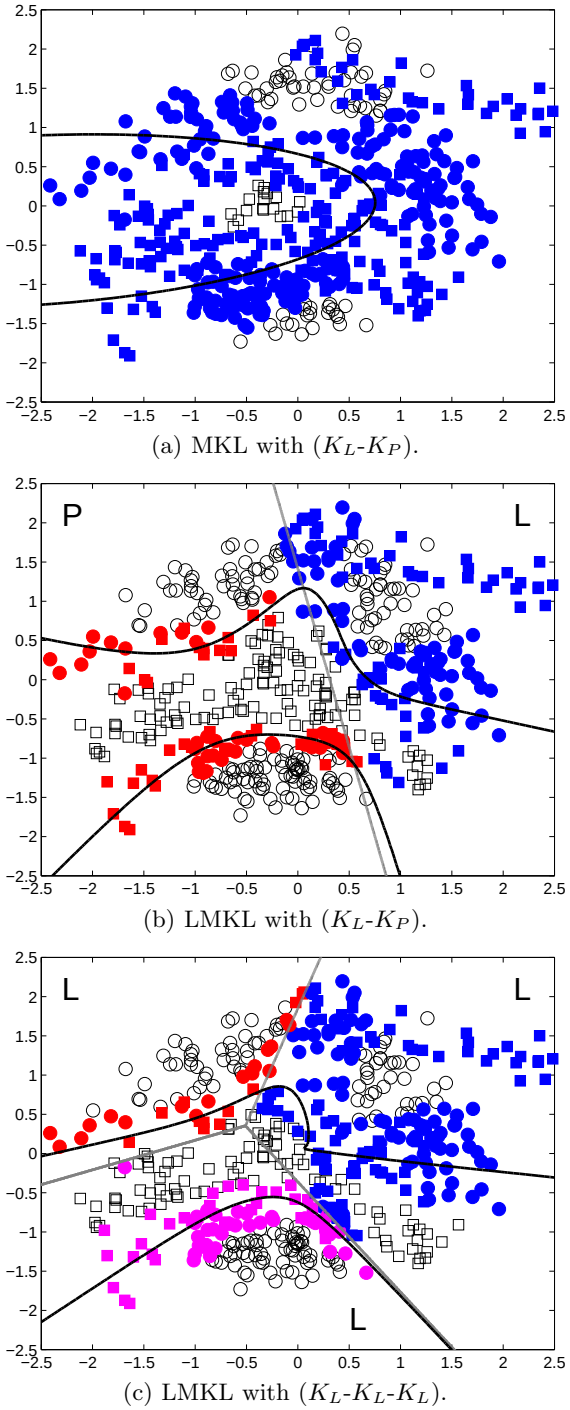
4.2. Benchmark Data Sets

We perform experiments on ten two-class benchmark data sets from the UCI machine learning repository and Statlog collection. In the result tables, we report the average testing accuracies and support vector percentages. The average accuracies and support vector percentages are made bold if the difference between the two compared classifiers is significant using the 5×2 cross-validation paired F test (Alpaydın, 1999).

Figure 3(a)-(b) illustrate the difference between MKL and LMKL on BANANA data set with (K_L-K_P) combination. We can see that MKL can not capture the localities exist in the data by combining linear and polynomial kernels with fixed combination weights (it assigns 1.00 to K_P ignoring the linear kernel). However, LMKL finds a more reasonable decision boundary using much fewer support vectors by dividing the input space into two regions using the linear gating model. The average testing accuracy increases from 70.52 to 84.46 per cent and the support vector count is halved (decreases from 82.36 to 41.28 per cent).

The classification and gating model boundaries found by LMKL with (K_L-K_L-K_L) combination on BANANA data set can be seen in Figure 3(c). The gating model divides the input space into three regions and in each region a local and (nearly) linear decision boundary is induced. Combination of these local boundaries with softmax gating gives us a more complex boundary.

The results by MKL and LMKL for (K_P-K_G) and canonical SVMs with K_L, K_P, K_G are given in Table 1. LMKL achieves statistically similar accuracies compared with MKL on all data sets. LMKL stores significantly fewer support vectors on HEART, PIMA, and WDBC data sets. With direct comparison of average values, the localized variant performs better on seven and eight out of ten data sets in terms of testing accuracy and support vector percentage, respectively. Other kernel combinations behave similarly.



(a) MKL with (K_L-K_P).

(b) LMKL with (K_L-K_P).

(c) LMKL with (K_L-K_L-K_L).

Figure 3. Separating hyperplanes (black solid lines) and support vectors (filled points) on BANANA data set. The gray solid lines shows the boundaries calculated from the gating models by considering them as classifiers which select a kernel function. Both accuracy increases and support vector count decreases.

We also combine $p = 2, \ldots, 5$ linear kernels on benchmark data sets with LMKL. Table 1 (to the right) compares the results of canonical SVM with the linear kernel and LMKL with three linear kernels. LMKL uses statistically fewer support vectors on six out of ten data sets and on three of these (BANANA, PIMA, and SPAMBASE), accuracy is significantly improved. With direct comparison of average values, LMKL performs better than canonical SVM on seven and eight out of ten data sets in terms of accuracy and support vector percentages, respectively. Using localized linear kernels also improves testing time due to evaluating linear kernels over only neighboring support vectors, instead of evaluating it over all support vectors.

Using Wilcoxon's signed rank test on ten data sets (see Table 1), when different kernels are combined, LMKL stores significantly fewer support vectors than MKL; when multiple copies of the same (linear) kernel are combined, LMKL achieves significantly higher accuracy than canonical SVM using a single kernel.

4.3. Bioinformatics Data Sets

We perform experiments on two bioinformatics data sets in order to see the applicability of LMKL to real-life problems. These translation initiation site data sets are constructed by using the same procedure described by Pedersen and Nielsen (1997). Each data instance is represented by a window of 200 nucleotides. Each nucleotide is encoded by five bits and the position of the set bit indicates whether the nucleotide is A, T, G, C, or N (for unknown).

As in benchmark data sets when combining different kernels, LMKL achieves statistically similar accuracy results compared with MKL by storing fewer support vectors for all combinations (see Table 2). For example, using (K_P-K_G), LMKL needs on the average 24.55 and 22.32 per cent fewer support vectors on ARABIDOPSIS and VERTEBRATES data sets, respectively.

We combine $p = 2, \ldots, 5$ linear kernels on bioinformatics data sets using LMKL. Table 2 shows that LMKL with three linear kernels improves the average accuracy statistically significantly. LMKL also uses significantly fewer support vectors (the decrease is almost one-third) on these data sets.

5. Conclusions

This work introduces a localized multiple kernel learning framework for kernel-based algorithms. The proposed algorithm consists of: (a) a gating model which assigns weights to kernels for a data instance, (b) a kernel-based learning algorithm with the locally combined kernel matrix. The training of these two components are coupled and the parameters of both components are optimized together by using a two-step alternate optimization procedure in a joint manner.

For binary classification tasks, the algorithm of the proposed framework with linear gating is derived and tested on ten benchmark and two bioinformatics data sets. LMKL achieves statistically similar accuracy results compared with MKL by storing fewer support vectors. Because kernels are evaluated locally (i.e., zero weighted kernels for a test instance are not calculated), the whole testing process is also much faster. This framework allows using multiple copies of the same kernel in different regions of the input space, obtaining more complex boundaries than what the underlying kernel is capable of. In order to illustrate this advantage, we combine different number of linear kernels on all data sets and learn piecewise linear boundaries. LMKL with three linear kernels gives significantly better accuracy results than canonical SVM with linear kernel on bioinformatics data sets.

Acknowledgments

This work was supported by the Turkish Academy of Sciences in the framework of the Young Scientist Award Program under EA-TÜBA-GEBİP/2001-1-1, Boğaziçi University Scientific Research Project 07HA101 and the Turkish Scientific Technical Research Council (TÜBİTAK) under Grant EEEAG 107E222. The work of M. Gönen was supported by the PhD scholarship (2211) from TÜBİTAK.

References

Alpaydın, E. (1999). Combined 5×2 cv F test for comparing supervised classification learning algorithms. *Neural Computation*, *11*, 1885–1892.

Amari, S., & Wu, S. (1998). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, *12*, 783–789.

Bach, F. R., Lanckriet, G. R. G., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. *Proceedings of the 21st International Conference on Machine Learning* (pp. 41–48).

Collobert, R., Bengio, S., & Bengio, Y. (2001). A parallel mixture of SVMs for very large scale problems. *Advances in Neural Information Processing Systems (NIPS)* (pp. 633–640).

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton,

Table 1. The average testing accuracies and support vector percentages on benchmark data sets. Comparisons are performed between MKL and LMKL for (K_P-K_G). LMKL with (K_L-K_L-K_L) is compared to canonical SVM with K_L.

	SVM				MKL		LMKL		SVM		LMKL	
	K_P		K_G		(K_P-K_G)		(K_P-K_G)		K_L		(K_L-K_L-K_L)	
Data Set	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV
Banana	56.51	75.99	83.57	92.67	81.99	93.39	83.84	83.97	59.18	93.99	**81.39**	**54.03**
Germannumeric	71.80	54.17	68.65	58.44	73.32	84.89	73.92	80.90	74.58	97.09	75.09	**57.21**
Heart	72.78	73.89	77.67	79.11	75.78	87.89	79.44	**81.44**	78.33	67.00	77.00	**58.44**
Ionosphere	91.54	38.55	94.36	61.71	93.68	64.10	93.33	53.33	86.15	36.58	87.86	49.06
Liverdisorder	60.35	69.83	64.26	74.43	63.39	93.57	64.87	92.52	64.78	85.65	64.78	78.35
Pima	66.95	24.26	71.91	74.26	72.62	80.39	72.89	**73.63**	70.04	100.00	**73.98**	**53.09**
Ringnorm	70.66	53.91	98.82	40.68	98.86	57.68	98.69	56.69	76.91	78.68	78.92	**52.53**
Sonar	65.29	67.54	72.71	73.48	80.29	89.57	79.57	90.00	73.86	68.41	77.14	60.43
Spambase	84.18	47.92	79.80	49.50	90.46	57.47	91.41	58.24	85.98	77.43	**91.18**	**34.93**
Wdbc	88.73	27.11	94.44	54.74	95.50	58.11	95.98	**42.95**	95.08	**13.11**	94.34	21.89
5×2 cv Paired F Test (W-T-L)							0-10-0	3-7-0			3-7-0	6-4-0
Direct Comparison (W-T-L)							7-0-3	8-0-2			7-1-2	8-0-2
Wilcoxon's Signed Rank Test (W/T/L)							T	W			W	T

Table 2. The average testing accuracies and support vector percentages on bioinformatics data sets.

	SVM				MKL		LMKL		SVM		LMKL	
	K_P		K_G		(K_P-K_G)		(K_P-K_G)		K_L		(K_L-K_L-K_L)	
Data Set	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV	Acc.	SV
Arabidopsis	74.30	68.08	77.41	42.36	80.10	89.96	80.82	**65.41**	74.30	99.64	**81.29**	**68.66**
Vertebrates	75.50	68.54	75.72	41.64	78.67	90.46	77.67	68.14	75.50	99.02	**78.69**	**67.41**

G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*, 79–87.

Lanckriet, G. R. G., Bie, T. D., Cristianini, N., Jordan, M. I., & Noble, W. S. (2004a). A statistical framework for genomic data fusion. *Bioinformatics*, *20*, 2626–2635.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004b). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, *5*, 27–72.

Lee, W., Verzakov, S., & Duin, R. P. W. (2007). Kernel combination versus classifier combination. *Proceedings of the 7th International Workshop on Multiple Classifier Systems* (pp. 22–31).

Lewis, D. P., Jebara, T., & Noble, W. S. (2006). Nonstationary kernel combination. *Proceedings of the 23rd International Conference on Machine Learning* (pp. 553–560).

Moguerza, J. M., Muñoz, A., & de Diego, I. M. (2004). Improving support vector classification via the combination of multiple sources of information. *Proceedings of Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops* (pp. 592–600).

Mosek (2008). *The MOSEK optimization tools manual version 5.0 (revision 79)*. MOSEK ApS, Denmark.

Pavlidis, P., Weston, J., Cai, J., & Grundy, W. N. (2001). Gene functional classification from heterogeneous data. *Proceedings of the 5th Annual International Conference on Computational Molecular Biology* (pp. 242–248).

Pedersen, A. G., & Nielsen, H. (1997). Neural network prediction of translation initiation sites in eukaryotes: Perspectives for EST and genome analysis. *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology* (pp. 226–233).

Rakotomamonjy, A., Bach, F., Canu, S., & Grandvalet, Y. (2007). More efficiency in multiple kernel learning. *Proceedings of the 24th International Conference on Machine Learning* (pp. 775–782).

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, *7*, 1531–1565.

No-Regret Learning in Convex Games

Geoffrey J. Gordon GGORDON@CS.CMU.EDU
Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15213

Amy Greenwald AMY@CS.BROWN.EDU
Casey Marks CASEY@CS.BROWN.EDU
Department of Computer Science, Brown University, Providence, RI 02912

Abstract

Quite a bit is known about minimizing different kinds of regret in experts problems, and how these regret types relate to types of equilibria in the multiagent setting of repeated matrix games. Much less is known about the possible kinds of regret in online convex programming problems (OCPs), or about equilibria in the analogous multiagent setting of repeated convex games. This gap is unfortunate, since convex games are much more expressive than matrix games, and since many important machine learning problems can be expressed as OCPs. In this paper, we work to close this gap: we analyze a spectrum of regret types which lie between *external* and *swap* regret, along with their corresponding equilibria, which lie between *coarse correlated* and *correlated* equilibrium. We also analyze algorithms for minimizing these regret types. As examples of our framework, we derive algorithms for learning correlated equilibria in polyhedral convex games and extensive-form correlated equilibria in extensive-form games. The former is exponentially more efficient than previous algorithms, and the latter is the first of its type.

1. Introduction

We wish to build agents that can learn to act effectively in multiagent decision problems. We represent such problems as general-sum games: each agent i is given a feasible region A_i from which to choose an action a_i. The payoff to agent i depends not only on i's choice, but also on the actions $a_{\neg i}$ chosen by other agents. Since we are modeling learning, we assume that each agent knows only its own feasible region and observes only its own payoff structure. So, an agent cannot simply compute an equilibrium of the game and play it (even leaving aside the complexity of such a computation and the problem of coordinating with other agents on an equilibrium). All an agent can do is *learn* a preferred course of action by playing the game repeatedly and observing its own payoffs.

What, then, is an appropriate goal for a learning agent? Unlike zero-sum games, general-sum games do not have a well-defined *value*: even if we had complete knowledge of the game and all players were completely rational, we would not be able to predict how much payoff we should receive. Instead, researchers have defined other goals for learning agents. One popular one is regret minimization. For example, a number of previous algorithms have been designed to minimize *external regret* (defined in Sec. 2) in convex games, including Generalized Gradient Descent (Gordon, 1999b), GIGA (Zinkevich, 2003), Follow the Perturbed Leader (Kalai & Vempala, 2003), Lagrangian Hedging (Gordon, 2006), and algorithms based on Fenchel duality (Shalev-Shwartz & Singer, 2006).

However, no external regret may not be a sufficient goal: a set of agents can all achieve no external regret (which guarantees that the empirical distribution of joint play converges to the set of *coarse correlated equilibria*, defined in Sec. 4) and still have an incentive to change their play. For example, a no-external-regret learner can consistently observe that its average payoff per trial would have been higher if it had chosen action a' every time that it actually played a, and yet never switch to playing action a' in these situations. To avoid such behavior, we seek algorithms that provide guarantees stronger than no external regret. In a seminal paper, Foster and Vohra (1997) present an algorithm that exhibits no *internal regret* (defined in Sec. 2) in matrix games, and further, show that if all players achieve no internal regret, the empirical distribution of joint play converges to the set of *correlated*

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

equilibria (see Sec. 4). This guarantee rules out precisely the anomalous behavior described above.

Stoltz and Lugosi (2007) generalize these results to convex games. Extending the framework of Greenwald and Jafari (2003) for matrix games, they define a continuum of regret measures called Φ-regret, as well as corresponding Φ-equilibria, for convex games. Given a feasible region A, Φ is a collection of *action transformations*; that is, each $\phi \in \Phi$ is a function from A to itself. An agent calculates its Φ-regret by comparing the losses it obtained during its past history of play to the losses it would have obtained had it transformed each action it played according to some $\phi \in \Phi$.

Different choices of Φ lead to different types of regret and corresponding equilibria. In matrix games, the only two regret types known to be of interest are the above-mentioned external and internal regret. No internal regret is equivalent to no *swap* regret, in which Φ is the set of all transformations from A to itself. In convex games, by contrast, there is a much richer variety of regret concepts. We identify and analyze two novel regret types, which we call *extensive-form* and *finite-element* regret. We also analyze *linear* regret. Each of these regret types is distinct from the others and from external and swap regret. In fact, they form a progression: no swap regret (the strongest property) implies no finite element regret, which implies no linear regret, which implies no extensive-form regret, which implies no external regret (the weakest property).

Different regret types require different regret-minimization algorithms. For convex games, until recently, most algorithms minimized only external regret. More recently, Stoltz and Lugosi (2007) proved the existence of a no-swap-regret algorithm, and Hazan and Kale (2007) derived an algorithm that exhibits no Φ-regret for any set Φ which is the convex hull of a finite set of transformations. Simultaneously and independently, we developed an algorithm similar to Hazan and Kale's: our algorithm handled more-general representations of transformation sets, but required exact fixed-point calculations (Gordon et al., 2007).

Unfortunately, constructing an algorithm according to Stoltz and Lugosi's proof would be prohibitively expensive: both the time and space requirements would grow exponentially with the number of rounds. And, Hazan and Kale's algorithm, which runs in time polynomial in the number of corners of Φ, can also be prohibitively expensive: for example, if A is the unit cube in $\mathbb{R}^d$ and Φ is the set of linear transformations that map A to itself, then Φ, which is the Cartesian product of d copies of the unit L_1 ball, has $(2d)^d$ corners.

In this work, we extend our earlier algorithms and proofs, unifying them with Hazan and Kale's. The result is an algorithm which accommodates more-efficient representations of Φ. In the example above, the natural representation of Φ is as a set of $d \times d$ matrices satisfying certain linear constraints. Using this representation, our algorithm runs in time polynomial in d—an exponential speedup. In general, we can efficiently achieve no linear regret so long as we can efficiently optimize over the set of linear mappings from A to itself.

We also instantiate our algorithm for extensive-form and finite-element regret. These regret types are important in practice: extensive-form regret corresponds to extensive-form correlated equilibrium (Forges & von Stengel, 2002), arguably the most natural notion of equilibrium in extensive-form games. And, our no-finite-element-regret algorithm, with a simple modification described below, guarantees that the empirical distribution of joint play converges to a correlated equilibrium.

For extensive-form regret, our algorithm is polynomial in the dimension of the action set A; we are not aware of any prior no-extensive-form-regret algorithms. For finite-element regret, our algorithm is polynomial in the dimension of the action set and in the size of a finite-element mesh that covers Φ. Although the necessary mesh for some choices of Φ is quite large, our algorithm is still by far the most efficient known that guarantees convergence to correlated equilibrium.

2. The General Algorithm

When playing a repeated convex game, a single agent's learning problem is called an **online convex program** (OCP): in each round t, the agent chooses an action $a_t \in A$. At the same time, forces external to the agent choose a convex loss function $l_t \in L$. (A loss is just a negative payoff.) The agent observes l_t and pays $l_t(a_t)$. The action space A is assumed to be a convex and compact subset of $\mathbb{R}^d$. The set L includes convex loss functions with bounded subgradients. The commonly studied **experts problem** is a special case of an OCP in which the feasible region is the probability simplex in $\mathbb{R}^d$.

A **learning algorithm** takes as input a sequence of loss functions l_t and produces as output a sequence of actions a_t. Action a_t may depend on $l_1 \dots l_{t-1}$, but not on l_t or later loss functions. The learner's objective is to minimize its cumulative loss, $L_t = \sum_{t=1}^{T} l_t(a_t)$.

The minimum achievable loss depends on the specific sequence l_t. To measure how well a learning algorithm

performs against a given sequence, we calculate its **regret**. The simplest type of regret is called **external** regret, and is defined as follows:

$$\rho_t^{\text{EXT}} = \sup_{a \in A} \sum_{t=1}^{T} (l_t(a_t) - l_t(a))$$

That is, the external regret is the difference between the actual loss achieved and the smallest possible loss that could have been achieved on the sequence l_t by playing a fixed $a \in A$.

We say that an algorithm $\mathcal{A}$ exhibits **no external regret** for feasible region A and set L if we can guarantee that its average external regret per trial eventually falls below any $\epsilon > 0$, regardless of the particular sequence l_t. In other words, $\mathcal{A}$ exhibits no external regret if there is a function $f(T, A, L)$ which is $o(T)$ for any fixed A and L, such that for all $a \in A$, $t \geq 1$

$$\sum_{t=1}^{T} l_t(a_t) \leq \sum_{t=1}^{T} l_t(a) + f(T, A, L) \tag{1}$$

The function f can depend on A and L in complicated ways, but usually depends on properties like the diameter of A under some norm, or the length of $\partial l(a)$ under some norm for $a \in A$ and $l \in L$.

More generally, an agent can consider replacing its sequence $a_1 \dots a_t$ with $\phi(a_1) \dots \phi(a_t)$, where ϕ is some **action transformation**, that is, a measurable function that maps A into itself. If Φ is a set of such action transformations, we define an algorithm's **Φ-regret** as

$$\rho_t^{\Phi} = \sup_{\phi \in \Phi} \sum_{t=1}^{T} (l_t(a_t) - l_t(\phi(a_t)))$$

and we say that it exhibits **no Φ-regret** if it satisfies the following analogue of Eq. 1: for all $\phi \in \Phi$, $t \geq 1$

$$\sum_{t=1}^{T} l_t(a_t) \leq \sum_{t=1}^{T} l_t(\phi(a_t)) + g(T, A, L, \Phi) \tag{2}$$

where $g(T, A, L, \Phi)$ is $o(T)$ for any fixed A, L, and Φ.

Note that external regret is just Φ-regret with Φ equal to the set of constant transformations: i.e., $\Phi_{\text{EXT}} = \{\phi_x \mid x \in A\}$, where $\phi_x(a) = x$. By setting Φ to larger, more flexible transformation sets, we can define stronger varieties of regret. However, before studying any specific regret types in detail, we next discuss how to achieve no Φ-regret for general Φ.

2.1. General Φ

In this section, we develop an algorithm $\mathcal{A}$ that exhibits no Φ-regret for any suitable $\Phi \subset A \mapsto A$. The algorithm itself is fairly simple, and embodies essentially the same idea that was proposed earlier by Gordon et al. (2007) and Hazan and Kale (2007). However, we develop the idea here so that it applies to a more general class of transformation sets Φ than considered previously, and provide a proof that it achieves no Φ-regret under more general conditions. Our extra generality is crucial for developing efficient implementations for important choices of Φ including linear, extensive-form, and finite-element transformations.[1]

Our Φ-regret minimizing algorithm $\mathcal{A}$ is described in Fig. 1. It takes as input a sequence of loss functions $l_t \in L$ and outputs a sequence of actions $a_t \in A$, which, we will show, satisfies Eq. 2.

In designing $\mathcal{A}$, we assume that we have access to subroutines $\mathcal{A}'$ and $\mathcal{A}''$. The subroutine $\mathcal{A}'$ computes approximate fixed points of transformations $\phi \in \Phi$. That is, given any $\phi \in \Phi$ and any $\epsilon > 0$, $\mathcal{A}'$ returns some $a \in A$ such that $\|a - \phi(a)\|_A \leq \epsilon$. Here, $\|\cdot\|_A$ is an arbitrary norm on $\mathbb{R}^d$. The subroutine $\mathcal{A}''$ is an external-regret minimizing algorithm whose feasible region is Φ; we assume that its regret bound is $o(T)$ whenever we can provide a bound (in an appropriate norm) on the subgradients of the loss functions it encounters.

Since algorithm $\mathcal{A}$ accesses the transformation set Φ only through the subroutines $\mathcal{A}'$ and $\mathcal{A}''$, it does not depend on any special properties of Φ beyond the existence of these subroutines. To state our theorem, though, we will embed Φ in a vector space, as follows. Since $A \subset \mathbb{R}^d$, we can write $\phi \in \Phi$ as a d-tuple of "coordinate" functions $(\psi_1, \psi_2, \dots, \psi_d)$, $\psi_i : A \to \mathbb{R}$. For all $\phi \in \Phi$ and $i = 1 \dots d$, we assume ψ_i is a member of some reproducing-kernel Hilbert space (RKHS) $\mathcal{H} \subset A \mapsto \mathbb{R}$.[2] Finally, we assume that Φ is a convex and compact subset of $\mathcal{H}^d$.

To make these assumptions concrete, suppose for example that Φ is the convex hull of a finite set of transformations $\{\phi^1, \dots, \phi^p\}$: i.e.,

$$\Phi = \left\{ \textstyle\sum_{j=1}^{p} \alpha_j \phi^j \mid \alpha_j \geq 0, \sum_{j=1}^{p} \alpha_j = 1 \right\}$$

(This is the case treated by Hazan and Kale.) If we take $\mathcal{H}$ to be the span of all of the coordinate functions ψ_i^j, then Φ is a simplex in $\mathcal{H}^d$ with corners ϕ^j, for $j = 1 \dots p$. (In general, Φ's shape may be much more

[1] Hazan and Kale's algorithm is efficient in the special case of external transformations. Indeed, this section's algorithm specializes to their algorithm in this case.

[2] A Hilbert space is a (possibly infinite-dimensional) vector space that has an inner product. A reproducing-kernel Hilbert space is a Hilbert space of real- or complex-valued functions in which evaluation at the point a is a continuous linear functional for any a.

Given feasible region A, transformation set Φ, initial transformation $\phi_1 \in \Phi$, and subroutines $\mathcal{A}'$ and $\mathcal{A}''$.

For $t = 1, \ldots, T$:

1. Send transformation ϕ_t to the fixed-point algorithm $\mathcal{A}'$, along with accuracy parameter $\epsilon_t = 1/\sqrt{t}$. Receive action a_t satisfying $\|\phi_t(a_t) - a_t\|_A \leq \epsilon_t$.
2. Play a_t; observe loss function l_t and incur loss $l_t(a_t)$.
3. Define $m_t : \Phi \mapsto \mathbb{R}$ by $m_t(\phi) = l_t(\phi(a_t))$.
4. Send m_t to the no-external-regret algorithm $\mathcal{A}''$. Receive transformation $\phi_{t+1} \in \Phi$.

Figure 1. Algorithm $\mathcal{A}$.

complicated than a simplex, as we will see for example in the definition of Φ_{FE} below.)

To bound the Φ-regret of algorithm $\mathcal{A}$, we will need bounds on the actions a and the loss-function subgradients $\partial l(a)$, for all $l \in L$ and $a \in A$. In particular, we will suppose that $\|a\|_A \leq C_1$ and $\|\partial l(a)\|_{A^*} \leq C_2$, for any $a \in A$, any $l \in L$, and some constants $C_1, C_2 > 0$. Here $\|\cdot\|_{A^*}$ is the norm that is dual to $\|\cdot\|_A$.

Theorem 1 *Fix a convex and compact feasible region A and a set of loss functions L satisfying the above norm bounds, as well as a set of transformations $\Phi \subset \mathcal{H}^d$, where $\mathcal{H} \subset A \mapsto \mathbb{R}$ is a RKHS. Assume we are given an algorithm $\mathcal{A}''$ which, for any set of possible loss functions M with bounded subgradients, achieves no external regret on Φ. Also assume we are given an algorithm $\mathcal{A}'$ which can compute an approximate fixed point of any $\phi \in \Phi$. Then algorithm $\mathcal{A}$, using subroutines $\mathcal{A}'$ and $\mathcal{A}''$, achieves no Φ-regret.*

PROOF: Define the set of functions $M \subset \Phi \mapsto \mathbb{R}$ as $M = \{l(\phi(a)) \mid l \in L, a \in A\}$. Note that each $m \in M$ is convex because each $l \in L$ is convex and $\phi(a)$ is linear in ϕ. Moreover, the norm of the subgradient of any $m \in M$ at any point $\phi \in \Phi$ is bounded by $C_1 C_2$. (A proof of this fact, as well as a definition of the appropriate norm, is given by Gordon et al. (2008).)

Because $\mathcal{A}''$ exhibits no external regret on Φ with the bounded-subgradient set of potential loss functions M,

$$\sum_{t=1}^{T} m_t(\phi_t) \leq \sum_{t=1}^{T} m_t(\phi) + f(T, \Phi, M) \quad \forall \phi \in \Phi$$

where f is sublinear in T. So, by the definition of m_t,

$$\sum_{t=1}^{T} l_t(\phi_t(a_t)) \leq \sum_{t=1}^{T} l_t(\phi(a_t)) + f(T, \Phi, M) \quad \forall \phi \in \Phi$$

But, since $\|\phi_t(a_t) - a_t\|_A \leq \epsilon_t$ and $\|\partial l_t(a_t)\|_{A^*} \leq C_2$, we have by Hölder's inequality that $l_t(a_t) \leq l_t(\phi_t(a_t)) + \epsilon_t C_2$. So,

$$\sum_{t=1}^{T} l_t(a_t) \leq \sum_{t=1}^{T} (l_t(\phi(a_t)) + \epsilon_t C_2) + f(T, \Phi, M) \quad \forall \phi \in \Phi$$

Since $C_2 \sum_{t=1}^{T} \epsilon_t = O(\sqrt{T})$, this is exactly the desired no-Φ-regret guarantee. □

Clearly, the run-time of $\mathcal{A}$ depends on the run-times of its subroutines. In particular, since $\mathcal{A}$ requires that $\mathcal{A}'$'s accuracy parameter ϵ approach 0 as T increases, it is important that $\mathcal{A}'$ run efficiently even for small ϵ. We will discuss run-times in more detail in the context of specific examples below. For now, we note the following trivial generalization of a result due to Hazan and Kale: if the fixed-point algorithm $\mathcal{A}'$ is a FPTAS, and if the no-external-regret algorithm $\mathcal{A}''$ runs in polynomial time, then $\mathcal{A}$ can process T actions and loss functions in time polynomial in T. Hazan and Kale allow run-times to be polynomial in the number of corners of Φ (among other parameters); this renders their efficiency guarantees meaningless when Φ has many corners. With our more-efficient representations of Φ, we can replace the dependence on the number of corners with parameters like the dimension of Φ and the norm bounds for $a \in A$ and ∂l for $l \in L$; since these latter parameters can be much smaller, the result will be a much faster run-time.

As described so far, the algorithm $\mathcal{A}$ is deterministic if its subroutines $\mathcal{A}'$ and $\mathcal{A}''$ are. Below, we will also define a randomized variant of $\mathcal{A}$, to strengthen the connection to game-theoretic equilibria.

2.2. Finite-dimensional Φ

We defined algorithm $\mathcal{A}$ in terms of a generic set of transformations $\Phi \subset \mathcal{H}^d$, where $\mathcal{H}$ is a RKHS, and each element of $\mathcal{H}$ is a real-valued function on A. (So, each $\phi \in \Phi$ is a d-tuple of real-valued functions on A, which we interpret as a function from A to $\mathbb{R}^d$.)

Because of the reproducing-kernel property, computing component $\psi_i(a)$ of some $\phi \in \mathcal{H}^d$ for $a \in A$ is the same as computing the inner product $\langle \psi_i, K(a) \rangle$. In other words, each ψ_i is the composition of a fixed, possibly-nonlinear function $K(\cdot)$ with a linear mapping $\langle \psi_i, \cdot \rangle$. This is the so-called "kernel trick" (Cortes & Vapnik, 1995): first, K computes a vector of features of the action a. The inner product with ψ_i then combines all of these features to produce the final output $\psi_i(a)$. To evaluate $\phi(a)$ in its entirety, we can compute $K(a)$ once, and then evaluate the d inner products $\langle \psi_1, K(a) \rangle, \ldots, \langle \psi_d, K(a) \rangle$.

In this paper, we are chiefly interested in cases where the dimension of $\mathcal{H}$ is manageable, so that we can di-

rectly write down and work with the transformations $\phi \in \mathcal{H}^d$. So, for the remainder of the paper, we will assume that $\mathcal{H}$ is isomorphic to $\mathbb{R}^p$ for some finite p. We will also restrict our interest to linear loss functions $l_t(a) = a \cdot \partial l_t$. This is without loss of generality, since we can achieve no regret for a sequence of convex loss functions l_t by working with appropriately-chosen linear lower bounds on each l_t (Gordon, 1999a).

With these additional assumptions, the steps of $\mathcal{A}$ can be simplified: each derived loss function m_t is linear, and can be described by its subgradient as follows:

$$\partial m_t(\phi) = \partial(l_t(\phi(a_t))) = \partial(\phi(a_t) \cdot \partial l_t) = \partial l_t K(a_t)^{\mathrm{T}}$$

The subgradient ∂m_t is a $d \times p$ matrix, since ∂l_t is a d-vector and $K(a_t)$ is a p-vector. Each transformation ϕ also corresponds to a $d \times p$ matrix (a d-tuple of p-vectors). To evaluate the loss function m_t on a transformation ϕ, we take the dot product $\partial m_t \cdot \phi$, which is defined to be $\mathrm{tr}(\partial {m_t}^{\mathrm{T}} \phi) = \mathrm{tr}(K(a_t) \partial {l_t}^{\mathrm{T}} \phi) = \mathrm{tr}(\partial {l_t}^{\mathrm{T}} \phi K(a_t)) = \partial {l_t}^{\mathrm{T}} \phi K(a_t)$.

As we will see in the next section, a number of interesting transformation sets can be represented as $d \times p$ matrices. Representing transformations and subgradients in this way means we can manipulate them efficiently, and, in turn, design efficient no-regret algorithms.

3. Specific Algorithms

We now instantiate our algorithm with various transformation sets Φ. We define each Φ as a set of $d \times p$ matrices ϕ, together with a kernel function $K : A \mapsto \mathbb{R}^p$, with the guarantee that $\phi K(a) \in A$ for all $a \in A$ and $\phi \in \Phi$. To minimize each ensuing regret type, we go on to identify efficient subroutines $\mathcal{A}'$ and $\mathcal{A}''$ for finding fixed points and achieving no external regret. (All other calculations in our algorithm are $O(pd)$, so these subroutines will usually be what limits our efficiency.)

For completeness, we also mention Φ_{EXT}, the set of constant transformations on A, and Φ_{SWAP}, the set of all measurable transformations on A. Φ_{EXT} is the weakest form of regret of interest here, and Φ_{SWAP} the strongest. These are the only two regret types known to be of interest in matrix games (no swap regret and no internal regret are equivalent in this setting).

In convex games, however, there is a much richer variety of interesting regret concepts. Below, we analyze linear, finite-element, and extensive-form regret, corresponding to transformation sets Φ_{LIN}, Φ_{FE}, and Φ_{EF}. As we will see, in general, $\Phi_{\mathrm{EXT}} \subset \Phi_{\mathrm{EF}} \subset \Phi_{\mathrm{LIN}} \subset \Phi_{\mathrm{FE}} \subset \Phi_{\mathrm{SWAP}}$. So, no swap regret implies no finite-element regret, which implies no linear regret, which implies no extensive-form regret, which implies no external regret. We show in the long version of this paper (Gordon et al., 2008) that these five regret varieties are in fact distinct: it is possible to have, e.g., no Φ_{LIN}-regret while still having positive Φ_{FE}-regret.

Linear Regret The set Φ_{LIN} includes all linear transformations that map A into itself. To achieve no linear regret, we can take K to be the identity. Φ will then be a set of square $d \times d$ matrices. To find a fixed point of $\phi \in \Phi$, we choose an appropriate element of the null space of $\phi - I$, which takes time polynomial in d. The more expensive task is to achieve no external regret on Φ: depending on the form of A, Φ may or may not lend itself to a description in terms of a small number of simple constraints.

If A is a probability simplex, then Φ is the set of stochastic matrices, which can be expressed with $O(d^2)$ linear constraints on the entries of ϕ (this setting yields an algorithm very similar to that of Blum and Mansour (2005)). If A is a unit Euclidean ball, then Φ consists of those matrices whose largest singular value is ≤ 1; this set can be represented using a single semidefinite constraint. For general (convex compact) A, the best choice may be to use either GIGA or lazy projection (Zinkevich, 2003): the difficult step in these algorithms is a Euclidean projection onto Φ, which can be achieved via the ellipsoid algorithm.

Finite-Element Regret The finite-element transformations only apply to polyhedral feasible regions A. For finite-element regret, we will define K as a mapping from a polyhedral feasible set A to a high-dimensional space $K(A)$ called the **barycentric coordinate space**. To construct $K(a)$, we first associate each of the p corners of A with one dimension of $\mathbb{R}^p$. We then triangulate A by dividing it into mutually exclusive and exhaustive d-simplices, so that each corner of A is a corner of one or more simplices.

Now, to calculate $K(a)$, we first determine the simplex in which a lies (or choose one arbitrarily if it is on a boundary) and calculate the weights of a with respect to the $d + 1$ corners of that simplex. That is, if $j(1) \ldots j(d+1)$ are the indices of the corners of the simplex containing a, and if $c_{j(1)} \cdots c_{j(d+1)}$ are their coordinates, we find the weights $b_1 \ldots b_{d+1}$ by solving $a = \sum_i b_i c_{j(i)}$, $\sum_i b_i = 1$. We then set entry $[K(a)]_{j(i)} = b_i$ for each corner $j(i)$, and set all other entries of $K(a)$ to 0.

For example, if $A = [0,1]^2$, we can divide A into two triangles, one with corners $(0,0)$, $(0,1)$, and $(1,1)$, and the other with corners $(0,0)$, $(1,0)$, and $(1,1)$. To calculate $K(\frac{1}{3}, \frac{2}{3})$, note that $(\frac{1}{3}, \frac{2}{3})$ is in the first

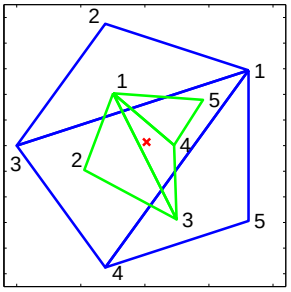


Figure 2. Illustration of barycentric coordinates and Φ_{FE}. A is the outer pentagon, triangulated into three simplices. $K(A)$ is a subset of the simplex in $\mathbb{R}^5$ (not shown). $\phi(A)$ is the distorted pentagon. The $\times$ marks a fixed point of ϕ.

triangle. If we associate corners of A with dimensions of $K(A)$ in the order $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$, then $K(\frac{1}{3},\frac{2}{3}) = (\frac{1}{3},\frac{1}{3},0,\frac{1}{3})$, since these weights express $(\frac{1}{3},\frac{2}{3})$ as a convex combination of corners 1, 2, and 4.

Given this definition of K, Φ_{FE} is the set of matrices ϕ that map $K(A)$ into A. If A is a simplex, then K will be a linear mapping and $\Phi_{\text{FE}} = \Phi_{\text{LIN}}$. (In general, $\Phi_{\text{FE}} \supset \Phi_{\text{LIN}}$.) For another example, see Fig. 2.

We note that Φ_{FE} can be factored: it is the Cartesian product of p copies of A, since it just needs to map each corner of A to a point inside A. So, to achieve no external regret in Φ, we can separately run p copies of any no-external-regret algorithm for A. A typical cost for doing so might be $O(pd^3)$.[3] To find a fixed point of ϕ, we just need to check each of its linear pieces separately. Each individual check costs $O(d^3)$, and there is one for each simplex in our mesh.

Extensive-Form Regret Let T be a player's **sequence tree**, describing all possible sequences of choices and observations in an extensive-form game (e.g., Fig. 3 (left)). Suppose that each element of the feasible region A is a **sequence weight vector** on T (Forges & von Stengel, 2002), specifying a behavior strategy for the game. Define an **extensive form** transformation as follows: fix a set D of choice nodes in T, along with pure-strategy sequence weight vectors w_b for each $b \in D$. If the original strategy is ever about to play $b \in D$, the transformed strategy deviates, and instead follows w_b. We assume that there are no $b, b' \in D$ with b' an ancestor of b (so that all $b \in D$ are reachable), and that each $b \in D$ has a sibling a with $w_b(a) = 1$. Extensive-form transformations are interesting since they correspond to the incentive constraints in extensive-form correlated equilibrium (Forges & von Stengel, 2002).

We show (Gordon et al., 2008) that each extensive form transformation can be represented by a matrix ϕ, whose rows and columns are indexed by choices, so that any action $w \in A$ is transformed into another action $\phi w \in A$. The entries of ϕ are as follows:

$$\phi_{ab} = \begin{cases} w_b(a) & \text{if } b \preceq a \text{ and } b \in D \\ 1 & \text{if } b = a \text{ and } \forall b' \in D,\ b \notin T_{b'} \\ 0 & \text{otherwise} \end{cases}$$

($T_{b'}$ is the subtree of T rooted at b', so that $b \notin T_{b'}$ means b is not a descendent of b'; $b \preceq a$ means b is an ancestor or a sibling of an ancestor of a in T.) This equation says that column b of ϕ is either: a copy of w_b with entries $w_b(a)$ replaced by 0s for $b \not\preceq a$ (if $b \in D$, cases 1, 3); a single 1 on the diagonal (if neither b nor any of its ancestors is in D, cases 2, 3); or all 0s (if $b \notin D$, but one of b's (strict) ancestors is in D, case 3).

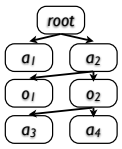
Now, if we take Φ_{EF} to be the convex hull of all such ϕs, then $\Phi_{\text{EF}} \subset \Phi_{\text{LIN}}$, and no Φ_{EF}-regret immediately implies no regret vs. any extensive form transformation. (So, no Φ_{EF}-regret is related to extensive-form correlated equilibrium; see Sec. 4).

For example, if T is as shown in Fig. 3 (left), elements of A are vectors of 4 sequence weights, one each for $a_1 \ldots a_4$. The weight for, e.g., a_3 is $P(a_2 \mid \text{root})P(a_3 \mid o_2)$, the product of the conditional probabilities of all choice nodes along the path from the root to a_3. So, strategy a_1, a_3 yields weights $w = (1,0,0,0)^{\text{T}}$, while a_2, a_3 yields $w' = (0,1,1,0)^{\text{T}}$.

The set Φ_{EF} for this game is shown in Fig. 3 (right). The parameters a, d, e, and f determine the probability that each choice node is included in D: $a \geq 0$ is $P(a_1 \in D)$, $d \geq 0$ is $P(a_2 \in D)$, $e \geq 0$ is $P(a_3 \in D)$, and $f \geq 0$ is $P(a_4 \in D)$. If $a_1 \in D$, parameters b and c specify a strategy for the subtree rooted at a_2. (If $a_1 \notin D$, the game ends right after we reach D, and so we need not specify further choices.) The inequalities listed in Fig. 3 are consistency constraints: e.g., the events $a_2 \in D$ and $a_3 \in D$ are mutually exclusive, so we must have $d + e \leq 1$.

To represent the transformation "play a_2, a_3 instead of a_1," we construct a matrix ϕ by setting $a, b = 1$ and $c, d, e, f = 0$. It is easy to verify that $\phi w = w'$ as expected. On the other hand, the transformation "play a_1 instead of a_2" corresponds to ψ with $d = 1$ and $a, b, c, e, f = 0$; again, it is easy to check $\psi w' = w$.

[3] The precise cost will depend heavily on the shape of A. For general A, most no-external-regret algorithms have a step like solving an LP with feasible region A or projecting onto A by minimum Euclidean distance. These computations cost $O(d^3)$ if we assume that an appropriate measure of the complexity of A is held constant.



$$\begin{bmatrix} 1-a & d & 0 & 0 \\ a & 1-d & 0 & 0 \\ b & 0 & 1-d-e & f \\ c & 0 & e & 1-d-f \end{bmatrix}$$

Figure 3. Φ_{EF} example. $b + c = a$, $d + e \leq 1$, $d + f \leq 1$, and $0 \leq a, b, c, d, e, f \leq 1$.

4. Regret and Equilibria

Algorithm $\mathcal{A}$ achieves no Φ-regret in an online convex program, for any suitable Φ. In this section, we relate this guarantee back to equilibria in convex games.

A game consists of a set of players N, a set of actions A_i for each player $i \in N$, and a payoff function $r_i : \otimes_i A_i \to \mathbb{R}$ for each player $i \in N$. A **matrix** game is one in which each action set is finite. A variant on a matrix game is an **experts** game in which each action set is a probability simplex. Generalizing experts games, a **convex** game is one in which each action set is a convex and compact subset of Euclidean space and each payoff function is multi-linear. In experts games and convex games, players can play interior points; but, assuming polyhedral action sets (PAS), we can generate a corresponding **corner game** by restricting each player's actions to the corners of its action sets.

Following Stoltz and Lugosi (2007), who generalize the definition for matrix games given in Greenwald and Jafari (2003), we define equilibria in convex games in terms of transformation sets.

Definition 2 *Given a game and a collection of transformation sets, $\langle \Phi_i \rangle_{i \in N}$, with each $\Phi_i \subseteq \Phi_{SWAP}$, a probability distribution q over $\otimes_i A_i$ is a $\langle \Phi_i \rangle_{i \in N}$-*equilibrium *iff the expectation over $a \sim q$ satisfies*

$$\mathbb{E}\left[r_i(\phi(a_i), a_{\neg i}) - r_i(a)\right] \leq 0 \quad \forall i \in N,\ \phi \in \Phi_i \qquad (3)$$

Intuitively, an equilibrium is a distribution from which no player prefers to deviate using any transformation in its set. Taking each Φ_i to be the set of swap transformations defines **correlated equilibria**; taking each Φ_i to be the set of external (i.e., constant) transformations defines **coarse correlated equilibria**. These definitions lead to the following propositions, proved by Marks (2008) and Gordon et al. (2007).

Proposition 3 *A correlated equilibrium of the corner game generated from a PAS convex game is also a correlated equilibrium of the convex game itself.*

Proposition 4 *For every correlated equlibrium in a PAS convex game, the corresponding corner game has a payoff-equivalent correlated equilibrium.*

4.1. Repeated Games

As described above, we assume the agents play some game repeatedly and learn by observing the relationship between their actions and their payoffs. In repeated matrix games, Greenwald and Jafari (2003) have shown that if each agent plays according to a no Φ_i-regret algorithm, then the empirical distribution of joint play converges to the set of $\langle \Phi_i \rangle_{i \in N}$-equilibria with probability 1. The **empirical distribution of joint play** at step t is the following distribution over the joint action set, where $a^t \in \otimes_i A_i$ is the joint action played at time step t: $z^t(\alpha) = \left|\{\tau \mid a^\tau = \alpha\}\right|/t$. The analogous result holds for $\langle \Phi_i \rangle_{i \in N}$-equilibrium in repeated convex games (e.g., Stoltz and Lugosi (2007)).

Because extensive-form games are one class of convex games (Forges & von Stengel, 2002), this result implies that, if the agents all play extensive-form regret-minimization algorithms, their play will converge to the set of extensive-form correlated equilibria. (Marks (2008) also provides algorithms with this property, using the less-efficient normal-form representation of extensive-form games.)

We can also say something about convergence to full-fledged correlated equilibria in repeated convex games: define a **randomized** variant of $\mathcal{A}$ as follows. On a trial where the deterministic algorithm would have played $\bar{a}_t$, the randomized algorithm samples its play a_t from any distribution D such that

$$E_D(a_t) = \bar{a}_t \qquad E_D(K(a_t)) = K(\bar{a}_t) \qquad (4)$$

(We still use $\bar{a}_t$, rather than a_t, in constructing m_t.) With such a D, if loss functions are linear, our Φ-regret on A and external regret on Φ differ by a zero-mean random variable; so, we can use standard stochastic convergence results to prove:

Corollary 5 *Under the conditions of Thm. 1, the additional assumption (4), and restricting L to include only linear loss functions, the randomized variant of $\mathcal{A}$ achieves no Φ-regret with probability 1.*

For Φ_{FE}-regret, we can always find a D that satisfies Equation (4); so (Gordon et al., 2007):

Corollary 6 *If, in a repeated PAS convex game, each agent plays only corner points and uses an algorithm that achieves no internal regret for the corner game (such as the randomized version of $\mathcal{A}$ with $\Phi = \Phi_{\text{FE}}$), then the empirical distribution of joint play converges to the set of correlated equilibria of the convex game with probability 1.*

To our knowledge, ours is the most efficient algorithm which can make this claim, by a factor which is exponential in the dimension d.

5. Discussion

We have presented several new forms of regret for online convex programs, analyzed their relationships to one another and to known regret types, and given the first efficient algorithms that directly minimize some of these forms of regret. These algorithms are by far the most efficient known for several purposes, including guaranteeing convergence to a correlated equilibrium in a repeated convex game, and to an extensive-form correlated equilibrium in an extensive-form game. By contrast, most previous OCP algorithms only guarantee convergence to coarse correlated equilibrium, an outcome which may yield much lower payoffs and may leave incentives for rational agents to deviate.

In the process of designing our algorithms, we derived efficient representations of the transformation sets for each of our regret types except Φ_{SWAP}: we wrote each as a nonlinear kernel mapping followed by a linear transformation chosen from an appropriate set of matrices. These representations may be of separate interest for designing future algorithms. In this paper, we were chiefly interested in cases where the dimension of the kernel mapping was manageable, so that we could directly work with the transformation matrices. However, it would be very interesting to try to design "kernelized" no-Φ-regret algorithms. In such algorithms we would never explicitly write down a transformation ϕ, but instead represent it in terms of observed actions and loss functions, thereby making it feasible to use very high-dimensional sets of transformations.

Important application areas for OCPs and convex games include multi-agent planning (in which the feasible region for each player is a set of plans, and interactions include contending for resources) and learning in extensive-form games such as poker. We are particularly interested in extensive-form games; this application requires further developments such as learning efficiently from bandit feedback and abstracting large games into smaller representations which we can work with in real time.

Acknowledgments

The authors would like to thank Martin Zinkevich for very helpful discussions during an early phase of this work. This work was supported in part by a grant from DARPA's Computer Science Study Panel program and in part by a grant from the Sloan Foundation.

References

Blum, A., & Mansour, Y. (2005). From external to internal regret. *Proceedings of the Conference on Computational Learning Theory (COLT)*.

Cortes, C., & Vapnik, V. N. (1995). Support-vector networks. *Machine Learning Journal*, *20*, 273–297.

Forges, F., & von Stengel, B. (2002). *Computationally efficient coordination in game trees* (Technical Report LSE-CDAM-2002-02). London School of Economics and Political Science, Centre for Discrete and Applicable Mathematics.

Foster, D., & Vohra, R. (1997). Regret in the on-line decision problem. *Games and Economic Behavior*, *21*, 40–55.

Gordon, G., Greenwald, A., & Marks, C. (2008). *No-regret learning in convex games* (Technical Report CS-08-03). Brown University, Department of Computer Science.

Gordon, G., Greenwald, A., Marks, C., & Zinkevich, M. (2007). *No-regret learning in convex games* (Technical Report CS-07-10). Brown University, Department of Computer Science.

Gordon, G. J. (1999a). *Approximate solutions to Markov decision processes*. Doctoral dissertation, Carnegie Mellon University.

Gordon, G. J. (1999b). Regret bounds for prediction problems. *Proceedings of the ACM Conference on Computational Learning Theory*.

Gordon, G. J. (2006). No-regret algorithms for online convex programs. *Advances in Neural Information Processing Systems (NIPS), 19*.

Greenwald, A., & Jafari, A. (2003). A general class of no-regret algorithms and game-theoretic equilibria. *Proceedings of the 2003 Computational Learning Theory Conference* (pp. 1–11).

Hazan, E., & Kale, S. (2007). Computational equivalence of fixed points and no regret algorithms, and convergence to equilibria. *Advances in Neural Information Processing Systems (NIPS), 20*.

Kalai, A., & Vempala, S. (2003). Efficient algorithms for online decision problems. *Proceedings of the 16th Annual Conference on Learning Theory*.

Marks, C. (2008). No-regret learning and game-theoretic equilibria. Ph.D. Dissertation, Department of Computer Science, Brown University, Providence, RI.

Shalev-Shwartz, S., & Singer, Y. (2006). Convex repeated games and Fenchel duality. *Advances in Neural Information Processing Systems (NIPS), 19*.

Stoltz, G., & Lugosi, G. (2007). Learning correlated equilibria in games with compact sets of strategies. *Games and Economic Behavior*, *59*, 187–208.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the 20th International Conference on Machine Learning*.

Boosting with Incomplete Information

Gholamreza Haffari[1*] GHAFFAR1@CS.SFU.CA
Yang Wang[1*] YWANG12@CS.SFU.CA
Shaojun Wang[2] SHAOJUN.WANG@WRIGHT.EDU
Greg Mori[1] MORI@CS.SFU.CA
Feng Jiao[3] FENGJIAO@YAHOO-INC.COM
[1]Simon Fraser University, Canada [2]Wright State University, USA [3]Yahoo! Inc. USA

Abstract

In real-world machine learning problems, it is very common that part of the input feature vector is incomplete: either not available, missing, or corrupted. In this paper, we present a boosting approach that integrates features with incomplete information and those with complete information to form a strong classifier. By introducing hidden variables to model missing information, we form loss functions that combine fully labeled data with partially labeled data to effectively learn normalized and unnormalized models. The primal problems of the proposed optimization problems with these loss functions are provided to show their close relationship and the motivations behind them. We use auxiliary functions to bound the change of the loss functions and derive explicit parameter update rules for the learning algorithms. We demonstrate encouraging results on two real-world problems — visual object recognition in computer vision and named entity recognition in natural language processing — to show the effectiveness of the proposed boosting approach.

1. Introduction

Boosting is a general supervised learning technique for incrementally building linear combinations of "weak" models to generate a "strong" predicative model. It is one of the most successful and practical methods in machine learning. Over the last decade, it has attracted much attention in the machine learning community and related areas such as statistics. It has been widely applied in many real-world problems such as text filtering and routing, ranking, learning in natural language processing, image retrieval, medical diagnosis, and customer monitoring and segmentation (Schapire, 2004).

It is very common in real-world machine learning problems that part of the input feature vector is incomplete: either not available, missing, or corrupted. In a web-page ranking problem, for example, using click-though data as part of the features, we find that a small number of valid pages have click features and most do not. In the case of object recognition in computer vision, many approaches assume a part-based model. However, certain parts of the object are hard to detect reliably due to small support in the image, occlusion or clutter, which also lead to missing information. Handling these kinds of classification problems containing incomplete information is a very important and realistic task. Excluding popular EM algorithms for generative models, some methods have been recently proposed for discriminative models (Chechik et al., 2007; Koo & Collins, 2005; Quattoni et al., 2005; Shivaswamy et al., 2006; Bi & Zhang, 2004).

In this paper, we show how to handle incomplete data under the boosting approach. We first describe the precise problem we are trying to solve, then we formulate optimization problems where the loss functions consist of two parts, one using partially labeled data and the other using fully labeled data. The primal problems of the proposed optimization problems with these loss functions are provided to show their close relationship and shed light on the rationale behind them. We derive explicit parameter update rules of the learning algorithms by introducing auxiliary functions to bound the change of loss functions. Finally, we demonstrate encouraging results on two real-world problems to show the effectiveness of the proposed boosting approach: visual object recognition in computer vision and named entity recognition in natural language processing.

[*]These authors contributed equally to this work.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Preliminaries

Let $X \in \mathcal{X}$ be a random variable over data instances to be labeled, and Y be a random variable over corresponding labels ranging over a finite label alphabet $\mathcal{Y}$. The classification task is to learn a mapping from data instances $\mathcal{X}$ to labels $\mathcal{Y}$. Assume we have a set of feature functions $\mathcal{F}_1 := \{f_k(x,y)\}$ where each feature maps $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$. Same as in (Collins et al., 2002; Lebanon & Lafferty, 2002) and without loss of generality, we assume that the range of all feature functions in this paper is $[0,1]$. These feature functions correspond to weak learners in boosting and sufficient statistics in an exponential family model.

Suppose the target predictor can be derived from a scoring function written as a linear combination of feature functions $t(x,y) = \sum_{f_k \in \mathcal{F}_1} \lambda_k f_k(x,y)$. Given a training dataset $\{(x_i, y_i)\}$, it has been shown (Lebanon & Lafferty, 2002) that Adaboost (Freund & Schapire, 1997) combines features to minimize the following exponential loss

$$\sum_{x_i} \sum_{y} q_\lambda(y|x_i) \tag{1}$$

where $q_\lambda(y|x) := \exp \sum_{f_k \in \mathcal{F}_1} \lambda_k \big[f_k(x,y) - f_k(x,\tilde{y}_x)\big]$ is called the unnormalized model, and $\tilde{y}_x$ denotes the label of instance x over the empirical data. Equivalently, it has been shown (Lebanon & Lafferty, 2002) that Logitboost (Friedman et al., 2000) minimizes the following log loss

$$-\sum_{x_i} \log p_\lambda(\tilde{y}_{x_i}|x_i) \tag{2}$$

where $p_\lambda(y|x) := q_\lambda(y|x)/Z_\lambda(x)$ is called the normalized model. Optimizing the two objective functions above can be done by either parallel or sequential updates (Collins et al., 2002; Lebanon & Lafferty, 2002).

Now assume that there is a random variable $h \in \mathcal{H}$ which is *hidden* in some part of the training data $\mathcal{D}_1 := \{(x_i, y_i)\}$ but has been observed in the rest of the training data $\mathcal{D}_2 := \{(x_j, h_j, y_j)\}$. Consider a second set of feature functions $\mathcal{F}_2 := \{f_k(x,h,y)\}$ where each feature maps $\mathcal{X} \times \mathcal{H} \times \mathcal{Y}$ to $\mathbb{R}$. In many real-world applications, the number of fully observed instances is much smaller than that of partially observed instances, that is, $|\mathcal{D}_2| \ll |\mathcal{D}_1|$, since obtaining fully observed instances is either expensive or time-consuming. To take full advantage of all available training data, we need to develop new methods, because the information cannot be fully exploited by the original boosting algorithm.

Hereafter we use subscripts i and j to range over training data in $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. For a datum (x,h,y), we denote all of its $\mathcal{F}_1$ features by the vector $\mathbf{f}_1(x,y)$ and all of its $\mathcal{F}_2$ features by the vector $\mathbf{f}_2(x,h,y)$.

3. Boosting with Hidden Variables

The challenge in this paper is, besides using the feature set $\mathcal{F}_1$ and training set $\mathcal{D}_1$, how to use the additional feature set $\mathcal{F}_2$ and training set $\mathcal{D}_2$ to obtain a better approximation for the mapping from instances to labels.

To this end, the main object of focus is a mapping from $\mathcal{X} \times \mathcal{H}$ to $\mathcal{Y}$, which is modeled by a conditional probability distribution $p_\lambda(y|x,h)$. This distribution is called the normalized model and is defined parametrically as

$$p_\lambda(y|x,h) \propto e^{\boldsymbol{\lambda}_1^T \cdot [\mathbf{f}_1(x,y) - \mathbf{f}_1(x,\tilde{y}_x)] + \boldsymbol{\lambda}_2^T \cdot [\mathbf{f}_2(x,h,y) - \mathbf{f}_2(x,h,\tilde{y}_x)]}$$

where $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ are the model's parameter vectors corresponding to features in $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively[1]. To estimate the parameters of the distribution, we can maximize the conditional likelihood of the training data:

$$\mathcal{L}(\lambda) := \sum_i \log p_\lambda(y_i|x_i) + \gamma \sum_j \log p_\lambda(y_j|x_j,h_j)$$

where γ is used to balance the influence of the two data sources on the objective function. Let $q_0(h|x)$ be a fixed distribution representing the prior belief in values of the hidden variable given an instance x, then $p_\lambda(y,h|x) = q_0(h|x)p_\lambda(y|x,h)$ and the first term in $\mathcal{L}(\lambda)$ can be computed based on $p_\lambda(y|x) = \sum_h p_\lambda(y,h|x)$.

We now turn our attention to model the mapping from $\mathcal{X} \times \mathcal{H}$ to $\mathcal{Y}$ by a linear scoring function that is the basis of our Adaboost type algorithms. When h is observed, the mapping is defined based on

$$t_\lambda(x,h,y) := \boldsymbol{\lambda}_1^T \cdot \boldsymbol{f}_1(x,y) + \boldsymbol{\lambda}_2^T \cdot \boldsymbol{f}_2(x,h,y)$$

and when h is hidden, it is defined as $t_\lambda(x,y) := \sum_h q_0(h|x) t_\lambda(x,h,y)$. As before, $q_0(h|x)$ is used to inject prior domain knowledge. To learn the parameters, we pose the minimization of the loss function $\mathcal{E}(\lambda)$ defined as

$$\mathcal{E}(\lambda) := \sum_i \sum_h q_0(h|x) \sum_y q_\lambda(y|x_i,h) + \gamma \sum_j \sum_y q_\lambda(y|x_j,h_j)$$

where $q_\lambda(y|x_i,h)$ is called the unnormalized model

$$q_\lambda(y|x,h) := e^{\boldsymbol{\lambda}_1^T \cdot [\mathbf{f}_1(x,y) - \mathbf{f}_1(x,\tilde{y}_x)] + \boldsymbol{\lambda}_2^T \cdot [\mathbf{f}_2(x,h,y) - \mathbf{f}_2(x,h,\tilde{y}_x)]}$$

The second term in $\mathcal{E}(\lambda)$ can be thought of as the loss incurred for the jth instance over all possible labels, and the first term as the *expected* loss for the ith instance. Note that if $q_0(h|x_j)$ puts a point mass γ on the observed h_j for instances in $\mathcal{D}_2$, then $\mathcal{E}(\lambda)$ can be rewritten compactly as

$$\mathcal{E}(\lambda) = \sum_{x \in \mathcal{D}_1 \cup \mathcal{D}_2} \sum_{h,y} q_0(h|x) q_\lambda(y|x,h)$$

[1] It is equivalent to the more familiar form $p_\lambda(x,h,y) \propto e^{\boldsymbol{\lambda}_1^T \cdot \mathbf{f}_1(x,y) + \boldsymbol{\lambda}_2^T \cdot \mathbf{f}_2(x,h,y)}$ by simply removing the constants $e^{\boldsymbol{\lambda}_1^T \cdot \mathbf{f}_1(x,\tilde{y}_x) + \boldsymbol{\lambda}_2^T \cdot \mathbf{f}_2(x,h,\tilde{y}_x)}$ from the numerator and denominator.

In the next section, we will show that there is a close relationship between minimizing $\mathcal{E}(\lambda)$ and maximizing the lowerbound $\ell(\lambda)$ on $\mathcal{L}(\lambda)$, which is derived based on Jensen's inequality and defined as

$$\ell(\lambda) := \sum_{i,h} q_0(h|x_i) \log p_\lambda(y_i|x_i, h) + \gamma \sum_j \log p_\lambda(y_j|x_j, h_j)$$

By extending q_0 to instances in $\mathcal{D}_2$ as before, we can write

$$\ell(\lambda) = \sum_{x \in \mathcal{D}_1 \cup \mathcal{D}_2} \sum_h q_0(h|x) \log p_\lambda(\tilde{y}_x|x, h)$$

Furthermore, we will show a close relationship between maximizing $\mathcal{L}(\lambda)$ and minimizing the following lowerbound on $\mathcal{E}(\lambda)$ derived by Jensen's inequality

$$\varepsilon(\lambda) := \sum_i \sum_y e^{t_\lambda(x_i,y) - t_\lambda(x_i,y_i)} + \gamma \sum_j \sum_y e^{t_\lambda(x_j,h_j,y) - t_\lambda(x_j,h_j,y_j)}$$

In the test time, depending on whether h is hidden or not, either $p_\lambda(y|x)$ or $p_\lambda(y|x, h)$ can be used to determine the class label of a given instance if we use the probabilistic model. Accordingly, for the linear map, either $t_\lambda(x, y)$ or $t_\lambda(x, h, y)$ can be used.

Our definitions of both normalized and unnormalized models are similar to those in (Lebanon & Lafferty, 2002). If we ignore fully labeled data in $\mathcal{L}(\lambda)$, we get the hidden conditional random field proposed in (Koo & Collins, 2005; Quattoni et al., 2005) by assuming $q_0(h|x)$ to be constant; however, the second term in $\mathcal{L}(\lambda)$ should exist to take advantage of $\mathcal{D}_2$. If we ignore the first term in $\mathcal{E}(\lambda)$, we get the standard boosting algorithm's loss function; however, the first term is needed to take advantage of the partially observed data $\mathcal{D}_1$. In the next section, we will provide the primal problems for the proposed loss functions to motivate the rationale of optimizing them and show their relationships. We then give sequential and parallel algorithms to optimize $\mathcal{E}(\lambda)$ and $\mathcal{L}(\lambda)$ in section 5.

4. Primal and Dual Programs

It is well known (Lebanon & Lafferty, 2002) that for standard boosting with no hidden information, the primal optimization problems for Adaboost and Logitboost are the same except for the additional constraints for the latter to ensure a probabilistic model. For our boosting with incomplete information, this relationship does not exist for the original optimization problems themselves, but rather between $\mathcal{E}(\lambda)$ and $\ell(\lambda)$ which is the lowerbound on $\mathcal{L}(\lambda)$.

Let the set of non-negative measures $\mathcal{M} := \{m : \mathcal{X} \times \mathcal{H} \times \mathcal{Y} \to \mathbb{R}_+\}$, and $\mathcal{F} := \mathcal{F}_1 \cup \mathcal{F}_2$. Let $\mathbf{r}$ be the reference measure $\mathbf{1}$; however, it can be any arbitrary measure that generalizes the objective functions introduced in the previous section.

Theorem 1. *The following optimization program:*

$$\max_\lambda \sum_{x \in D_1 \cup D_2} \sum_{h,y} q_0(h|x) q_\lambda(\tilde{y}_x|x, h) \tag{3}$$

is the dual of $\min_{\mathbf{p} \in \mathcal{S}(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F})} KL(\mathbf{p}||\mathbf{r})$ *where the extended* $KL(\mathbf{p}||\mathbf{r})$ *is defined as*

$$\sum_{x,h} \tilde{p}(x) q_0(h|x) \sum_y p(y|h, x) \Big[\log \frac{p(y|x, h)}{r(x, h, y)} - 1 \Big] + r(x, h, y)$$

and the set $\mathcal{S}(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F})$ *is defined as*

$$\Big\{ \mathbf{p} \in \mathcal{M} \Big| \sum_x \tilde{p}(x) \mathbb{E}_{q_0(h|x) p(y|x,h)} \Big[f - \mathbb{E}_{\tilde{p}(y|x)}[f] \Big] = 0, \forall f \in \mathcal{F} \Big\}$$

Proof sketch. The key idea in this theorem is the definition of the extended KL divergence and $\mathcal{S}(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F})$. Construct the Lagrangian of the dual, which is a constrained optimization problem, take its derivative, and set it to zero. It will give the form of the optimal solution; plug this form back into the Lagrangian, and make the data consistency assumption ($\tilde{\mathbf{p}}$ is the *empirical* probability distribution) $\sum_y \tilde{p}(y|x) f(x, y) = f(x, y_x)$ for $f \in \mathcal{F}_1$ and $\sum_y \tilde{p}(y|x) f(x, h, y) = f(x, h, y_x)$ for $f \in \mathcal{F}_2$, we will obtain the optimization problem in (3) . □

Theorem 2. *The following optimization program:*

$$\max_\lambda \sum_{x \in D_1 \cup D_2} \sum_h q_0(h|x) \log p_\lambda(\tilde{y}_x|x, h) \tag{4}$$

is the dual of $\min_{\mathbf{p} \in \mathcal{S}_\triangle(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F})} KL(\mathbf{p}||\mathbf{r})$ *where the extended* $KL(\mathbf{p}||\mathbf{r})$ *is defined as in Theorem 1, and*

$$\mathcal{S}_\triangle(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F}) := \Big\{ \mathbf{p} \in \mathcal{S}(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F}) \Big| \forall x, h : \sum_y p(y|x, h) = 1 \Big\}$$

The proof of this theorem is similar to that of Theorem 1 and is omitted because of space constraints. As can be seen from the theorems above, the primal optimization problems corresponding to the objective functions $\mathcal{E}(\lambda)$ and $\ell(\lambda)$ are the same except for the additional constraints for the later one to ensure $\sum_y p(y|x, h) = 1$. The extended KL divergence gives the expected discrepancy between $p(y|x, h)$ and the reference measure $r(x, h, y)$ where the expectation is taken with respect to the distribution $\tilde{p}(x) q_0(h|x)$. Hence minimizing the extended KL subject to the constraints forces $p(y|x, h)$ to become similar to $\mathbf{r}$, or in particular when the reference measure is $\mathbf{1}$ or constant, to have more entropy.

5. Learning Algorithms

Convergence of boosting algorithms has been studied in various ways. Much work has been done to prove the convergence in terms of an optimization method, which can be categorized into two approaches: greedy function optimization and greedy feature induction.

In the first approach, the boosting algorithm is viewed as a sequential gradient descent algorithm (Breiman, 1999;

Algorithm 1 Parallel Updates for the Normalized Model

repeat
  for $f_k \in \mathcal{F}_1$ **do**
    $A_k^+ = \sum_i \mathbb{E}^+_{p_\lambda(y|x_i)}[g_k(x_i, y)] + \gamma \sum_j \mathbb{E}^+_{p_\lambda(y|x_j,h_j)}[g_k(x_j, y)]$
    $A_k^- = \sum_i \mathbb{E}^+_{p_\lambda(y|x_i)}[-g_k(x_i, y)] + \gamma \sum_j \mathbb{E}^+_{p_\lambda(y|x_j,h_j)}[-g_k(x_j, y)]$
    $\Delta\lambda_k = \frac{\log A_k^- - \log A_k^+}{2C}$
  end for
  for $f_k \in \mathcal{F}_2$ **do**
    $A_k^+ = \sum_i \mathbb{E}^+_{p_\lambda(y,h|x_i)}[g_k(x_i, h, y)] + \gamma \sum_j \mathbb{E}^+_{p_\lambda(y|x_j,h_j)}[g_k(x_j, h_j, y)]$
    $A_k^- = \sum_i \mathbb{E}^+_{p_\lambda(y,h|x_i)}[-g_k(x_i, h, y)] + \gamma \sum_i \mathbb{E}^+_{p_\lambda(y|x_j,h_j)}[-g_k(x_j, h_j, y)]$
    $\Delta\lambda_k = \frac{\log A_k^- - \log A_k^+}{2C}$
  end for
  for $f_k \in \mathcal{F}_1 \cup \mathcal{F}_2$ **do**
    $\lambda_k \leftarrow \Delta\lambda_k + \lambda_k$
  end for
until convergence

Friedman et al., 2000; Mason et al., 2000) in function space, inspired by numerical optimization and statistical estimation. It is a forward stage-wise additive modeling that approximates the solution by sequentially adding new basis functions without adjusting the parameters and coefficients of those that have already been added. At each iteration, one solves for the optimal basis function and corresponding coefficients to add to the current expansion. This produces new expansion, and the process is repeated.

In the second approach (Collins et al., 2002; Lebanon & Lafferty, 2002), the boosting algorithm is described as a greedy feature induction algorithm to incrementally build random fields. The greediness of the algorithm arises in steps that select the most informative feature. In these steps each feature in a pool of candidate features is evaluated by estimating the reduction in the Kullback-Lieber divergence that would result from adding the feature to the field. This reduction is approximated as a function of a single parameter and is equal to the exponential loss reduction or log loss increment. This approximation is one of the key elements that make it practical to evaluate a large number of candidate features at each stage of the induction algorithm. Various parameter update rules can be derived By using an auxiliary function to bound the change of loss function from above, and thus convergence to the global optimal solution is proved.

In this paper we take the second approach to learn the discriminative model. We construct an auxiliary function to bound the change of exponential loss, $\mathcal{E}(\lambda+\Delta\lambda)-\mathcal{E}(\lambda)$ or log-loss $\mathcal{L}(\lambda)-\mathcal{L}(\lambda+\Delta\lambda)$. Similar to (Collins et al., 2002; Lebanon & Lafferty, 2002), either parallel or sequential updates can be used. By the same argument as in (Collins et al., 2002; Lebanon & Lafferty, 2002), we can show the convergence of these updates to a *local minimum* of the loss function. For simplicity in presenting the results, we introduce some notation for $\tilde{x} \in \mathcal{D}_1 \cup \mathcal{D}_2$:

$$\forall f_k \in \mathcal{F}_1, g_k(\tilde{x}, y) = f_k(\tilde{x}, y) - f_k(\tilde{x}, \tilde{y}_{\tilde{x}}) \quad (5)$$

$$\forall f_k \in \mathcal{F}_2, g_k(\tilde{x}, h, y) = f_k(\tilde{x}, h, y) - f_k(\tilde{x}, h, \tilde{y}_{\tilde{x}}) \quad (6)$$

$$C := \max_{\tilde{x},y,h} \Big(\sum_{f_k \in \mathcal{F}_1} |g_k(\tilde{x}, y)| + \sum_{f_k \in \mathcal{F}_2} |g_k(\tilde{x}, h, y)| \Big) \quad (7)$$

$$\mathbb{E}^+_{p(t)}[\psi(t)] := \sum_{t:\psi(t)>0} p(t)\psi(t) \quad (8)$$

For the normalized model, the learning algorithm with parallel updates is summarized in Algorithm 1 and with the sequential updates in Algorithm 2. For the unnormalized model, the update rules (parallel or sequential) are exactly the same; the only difference is that we will use $q_\lambda(y|x, h)$ rather than $p_\lambda(y|x, h)$ in all the algorithms' equations. For details of the derivation of updating rules in the learning algorithms, see Appendix A.

For ease of presentation, we have assumed that the potentially missing attributes are always the same. This is an interesting and nontrivial situation that occurs in many real-world applications, where the missing attribute h is the information that requires expensive human labeling (see the experiments for example applications). However, our approach can be easily extended to the cases where the data could have different missing attributes. In this more general setting, the i-th training datum has the form (x_i, y_i) with missing information $h_i \in \mathcal{H}_i$, where $\mathcal{H}_i$ can vary for different i's depending on which information is missing. The contribution of this datum to the log loss in the normalized model is simply $-\log p_\lambda(y_i|x_i)$. All the arguments in this paper will go through with some minor changes.

6. Experiments

We evaluate our approach in two real-world problems: visual object recognition in computer vision and named entity recognition in natural language processing. In both cases, we use simple and independent features, so when we calculate the values of A_j^+ and A_j^-, feature expectations can be done efficiently. For simplicity, we set γ to be 1. In practice, this parameter can be set by cross-validation. We set our prior belief in values of the hidden variable given an

Algorithm 2 Sequential Updates for the Norm. Model

repeat
  for $f_k \in \mathcal{F}_1$ **do**
    $A_k^+ = \sum_i \sum_{y \neq y_i} p_\lambda(y|x_i)(1 + g_k(x_i, y)) + \gamma \sum_j \sum_{y \neq y_j} p_\lambda(y|x_j, h_j)(1 + g_k(x_j, y))$
    $A_k^- = \sum_i \sum_{y \neq y_i} p_\lambda(y|x_i)(1 - g_k(x_i, y)) + \gamma \sum_j \sum_{y \neq y_j} p_\lambda(y|x_j, h_j)(1 - g_k(x_j, y))$
    $\lambda_k \leftarrow \frac{\log A_k^- - \log A_k^+}{2} + \lambda_k$
  end for
  for $f_k \in \mathcal{F}_2$ **do**
    $A_k^+ = \sum_i \sum_{y \neq y_i} \sum_h p_\lambda(y, h|x_i)(1 + g_k(x_i, h, y)) + \gamma \sum_j \sum_{y \neq y_j} p_\lambda(y|x_j, h_j)(1 + g_k(x_j, h_j, y))$
    $A_k^- = \sum_i \sum_{y \neq y_i} \sum_h p_\lambda(y, h|x_i)(1 - g_k(x_i, h, y)) + \gamma \sum_j \sum_{y \neq y_j} p_\lambda(y|x_j, h_j)(1 - g_k(x_j, h_j, y))$
    $\lambda_k \leftarrow \frac{\log A_k^- - \log A_k^+}{2} + \lambda_k$
  end for
until convergence

instance, $q_0(h|x)$ to be constant. In all the experiments, we use parallel updates. We have tried sequential updates and find that they are much slower. Although they can achieve higher likelihood on the training data, the results on the test data remain the same.

We compare our proposed boosting approach with three different baseline algorithms, in both normalized and unnormalized cases. The first baseline algorithm (BL1) uses both sets of features $\mathcal{F}_1$ and $\mathcal{F}_2$, but is trained only on the fully observed training data $\mathcal{D}_2$. The second baseline algorithm (BL2) is trained on all the training data $\mathcal{D}_1 \cup \mathcal{D}_2$ but uses only features $\mathcal{F}_1$, that is, it ignores features $\mathcal{F}_2$ that involve the hidden information h. Notice that the second baseline algorithm is identical to the algorithm in (Lebanon & Lafferty, 2002). The third baseline algorithm (BL3) uses all the training data $\mathcal{D}_1 \cup \mathcal{D}_2$ and both types of features $\mathcal{F}_1 \cup \mathcal{F}_2$ but ignores observed h on fully observed data; that is, it assumes all the data are in the form of $\{(x_i, y_i)\}$. Notice that the third baseline algorithm is similar to the hidden conditional random field (Quattoni et al., 2005).

6.1. Visual Object Recognition

We first consider a visual object recognition task where some of the data have missing features. In this task, we attempt to classify an image based on the existence of an object of interest in the image. We test our approach on the Caltech 4 dataset: airplanes, cars, faces, and motorbikes.

Common approaches to object recognition involve some form of supervision, which may range from manually segmenting the objects (Winn & Shotton, 2006), to specifying a bounding box of the objects (Viola & Jones, 2001), to only indicating the existence of the objects (Fergus et al., 2003). Naturally, there is a trade-off among different levels of supervisions. Manually segmenting the object of interest in an image obviously provides very accurate information for any learning algorithm, but it is very expensive and time-consuming to annotate a large number of images. On the other hand, it is relatively easy to label an image based only on the existence of an object. In our experiment, we assume we have two sets of training images. The first set of images has only class labels associated with them; we represent them as (x, y), where x refers to the image and y refers to its class label. The second set of images has both class label and the contour of the object being manually labeled; we represent them as (x, h, y), where h is the information about the contour of the object. Our learning problem is then in precisely the scenario in which our proposed method is expected to be effective.

We first run an interest-point detector (Kadir & Brady, 2001) to identity regions of interest on each image. Each interest point is represented by a SIFT descriptor (Lowe, 2004) as a 128-dimensional vector. The SIFT descriptors from all the training images are then vector quantized into K visual words (we choose $K = 200$ in our experiment) by k-means clustering. All the images are then represented by a bag-of-words representation by counting the occurrence of each visual word in an image. We denote an image as $x = (x_1, x_2, ..., x_t)$, where t is the number of interest points in x, and each x_i is an entry to a visual word. The information h about the object contour is represented as $h = (h_1, h_2, ..., h_t)$, where h_i is a binary value indicating whether x_i is on the object or not. Since we assume the "bag-of-words" model, the summation over h required for calculating A_j^+ and A_j^- can be solved efficiently by factoring out the contribution of each interest point. Although bag-of-words representation ignores a lot of positional information between features, previous work (Sivic et al., 2005; Fergus et al., 2005) has demonstrated that it to be quite effective in object recognition tasks.

We define the following three sets of features for our boosting algorithm, based on the bag-of-words representation of images. (1) feature $f_{jy'}(x, y)$ is calculated as the count of visual words j in an image x if $y = y'$, and zero otherwise; (2) feature $o_{jy'}(x, h, y)$ is the count of visual words j on the foreground of image x if $y = y'$, and zero otherwise; (3) feature $b_{jy'}(x, h, y)$ is the count of visual words j on the background of image x if $y = y'$, and zero otherwise. Notice that features $f_{jy'}$ are always observed for a training image. Features $o_{jy'}$ and $b_{jy'}$ are observed only when a training image does not have missing information (i.e., the manually labeled object contour). We normalize all the features by the total number of interest points in an image

	accuracy	log-likelihood
Our method	97.22%	-0.0916
BL1	89.26%	-1.1417
BL2	88.01%	-0.5698
BL3	90.43%	-0.4375

normalized model

	accuracy	log of loss
Our method	94.83%	-0.7412
BL1	82.57%	-1.1231
BL2	89.86%	-0.7977
BL3	87.64%	-0.8068

unnormalized model

Table 1. Results of our approach on visual object recognition, compared with three baseline algorithms

to make sure their values are between 0 and 1. During testing, we observed the image x, and we try to infer its label y based on the learned model. Although we can also infer y assuming both x and h are observed during testing, it is actually an unrealistic setting in our application. It requires a perfect figure/ground segmentation of the image x. However, since figure/ground segmentation is itself a very challenging problem in computer vision, it is not reasonable to assume we could have this information during the testing. So we do not investigate this case.

Our dataset contains more than 2000 images. We randomly split them equally into training and testing sets. We choose 30% of the training images to be fully observed and the rest to be partially observed. We compare both normalized and unnormalized models with the three baseline algorithms defined above, in terms of classification accuracy and the log-likelihood of the test data. The results are shown in Table 1. We also visualize the most discriminative patches in some sample images in Figure 1. We find that our approach is significantly superior to the three baseline algorithms, in term of both accuracy and log-likelihood on the test images.

6.2. Named Entity Recognition

Named entity extraction (NEE) is a subtask of information extraction in which we try to identify names of persons, locations, and organizations in a given set of documents. One approach to this problem is to do first named entity recognition (NER) and then named entity classification (NEC). In this section we apply our method to the NER problem and demonstrate its effectiveness compared to the baseline systems.

We consider NER as a sequence labeling problem, that is, specifying a sequence of zero and one for a sentence to classify a word as part of a named entity or not. For each word w, its surrounding words in a window of length 5, its part-of-speech tag (when available), and previous predictions represent its local context, which then used by the classifier. The part-of-speech tag is a valuable source of information and is not available in some annotations of the data sets for this task, so we treat it as the hidden variable that is not observed for some portion of the training data. We could use the *sequence* of POS tags of the words in the current window as the hidden variable. In that case, we may use a finite state automata to characterize the eligible sequence of POS tags when we want to sum over their values to speed up the training algorithms. The features that we used are summarized in Table 6.2; they are described in more details in (Carreras et al., 2003).

Feature	Explanation
Lexical	word forms and their positions in the window
Syntactic	part-of-speech tags (when available)
Orthographic	capitalized, include digits, ...
Affixes	the suffixes and prefixes (up to four characters)
Left predict	predicted labels for the two previous words

Table 2. Details of the features used for the NER task. Syntactic features belong to $\mathcal{F}_2$ and the rest of features belong to $\mathcal{F}_1$.

We use the data set of the CONLL 2003 shared task. To reduce the training time, we collapse the original 45 different POS tags into five tags as done in (McCallum et al., 2003). After training the model, we do the classification for each individual position by normalizing the prediction score of the model using the class mass normalization (CMN) procedure as introduced in (Zhu et al., 2003).

We compare our approach to the three baseline systems defined before. There are 5K sentences in $\mathcal{D}_1$, 6K sentences in D_2, and 1K sentences in the test set. The first set of experiments show the performance of our model compared to the baselines when, at the test time, only x is available (see Table 3). In the second set of experiments, (x, h) is given at the test time (see Table 4); for this setting, BL2 and BL3 cannot be used. Our method outperforms baseline systems in both sets of experiments in terms of f-measure and log-likelihood or loss function.

7. Comparison to the Related Work

Originally boosting is considered as a way to boost weak learners to strong learners by: learning weak hypotheses to classify hard examples in each round, and finally combining these weak hypotheses. Another view to boosting is through the statistical perspective which interprets it as: optimizing some objective function via parallel or sequential updates to determine the weights of all possible weak hypotheses (aka features). There is a debate between the statistic and algorithmic perspective; see (Mease & Wyner, 2008) for more information. Our work takes the statistical perspective and do not engage in that debate.

Figure 1. Visualization of the most discriminative patches in each image.

	f-measure	log-likelihood
Our method	49.45%	-0.5784
BL1	46.63%	-0.5932
BL2	48.10%	-0.5803
BL3	47.80%	-0.5880

normalized model

	f-measure	log of loss
Our method	49.04%	-2.6337
BL1	46.24%	-2.6458
BL2	47.58%	-2.6378
BL3	46.39%	-2.6434

unnormalized model

Table 3. Results of our approach on the NER task, compared with three baseline algorithms when only x is given in the test data.

	f-measure	log-likelihood
Our method	59.60%	-0.5759
BL1	56.51%	-0.5916

normalized model

	f-measure	log of loss
Our method	60.17%	-0.2586
BL1	55.46%	-0.2655

unnormalized model

Table 4. Results of our approach on the NER task, compared with the baseline algorithm BL1 when (x, h) is given in the test data. Even by having h, namely POS tags, the NER task is not easy.

A related algorithm that takes the first perspective to boosting is AdaBoost with confidence-rated predictions in which a weak learner outputs a real value representing the confidence level (Schapire & Singer, 1999). When provided with incomplete input, the weak learner's contribution is its uncertainty about its vote, which is represented by the produced real number. The details of the connection between our approach and confidence-rated AdaBoost will be an interesting topic to explore for future research.

8. Conclusions and Further Work

In this work we have presented a novel boosting approach that extends the traditional boosting framework by incorporating hidden variables such that fully labeled data can be integrated with partially labeled data to form a powerful strong classifier. Thus, compared with both the original boosting algorithms and hidden CRF, our model performs better in two real-world problems by fully exploiting relevant complete information of data resources.

We consider only simple independent features in our model. In fact, the hidden variables may have complex dependencies that respect certain cyclic graph structure; then it may be necessary to use variational methods, such as loopy belief propagation, to compute feature expectation for the values of A^+ and A^-. As future work, we would like to incorporate more complex dependent features in these two applications.

Acknowledgment

SW is supported in part by the Research Challenge Grant at Wright State University. We would like to thank Anoop Sarkar, David Blei and anonymous reviewers for their constructive comments.

References

Bi, J., & Zhang, T. (2004). Support vector classification with input data uncertainty. *NIPS*.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*.

Carreras, X., Màrquez, L., & Padró, L. (2003). A simple named entity extractor using AdaBoost. *Proceedings of CoNLL*.

Chechik, G., Heitz, G., Elidan, G., Abbeel, P., & Koller, D. (2007). Max-margin classification of incomplete data. *NIPS*.

Collins, M., Schapire, R. E., & Singer, Y. (2002). Logistic regression, adaboost and bregman distances. *Machine Learning*.

Fergus, R., Fei-Fei, L., Perona, P., & Zisserman, A. (2005). Learning object categories from google's image search. *IEEE International Conference on Computer Vision*.

Fergus, R., Perona, P., & Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. *CVPR*.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55(1):119-139.*

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics, 28*, 337–407.

Kadir, T., & Brady, M. (2001). Scale, saliency and image description. *International Journal of Computer Vision, 45*, 83–105.

Koo, T., & Collins, M. (2005). Hidden-variable models for discriminative reranking. *Proceedings of EMNLP*.

Lebanon, G., & Lafferty, J. D. (2002). Boosting and maximum likelihood for exponential models. *NIPS.*

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Functional gradient techniques for combining hypotheses. In *Advances in large margin classifiers*, 221–246. MIT Press.

McCallum, A., Rohanimanesh, K., & Sutton, C. (2003). Dynamic conditional random fields for jointly labeling multiple sequences. *NIPS Workshop on Syntax, Semantics, Statistics*.

Mease, D., & Wyner, A. (2008). Evidence contrary to the statistical view of boosting. *JMLR, 9.*

Quattoni, A., Collins, M., & Darrell, T. (2005). Conditional random fields for object recognition. *NIPS.*

Schapire, R. (2004). The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*. Springer.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning, 37*, 297–336.

Shivaswamy, P. K., Bhattacharyya, C., & Smola, A. J. (2006). Second order cone programming approaches for handling missing and uncertain data. *JMLR, 7*.

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., & Freeman, W. T. (2005). Discovering objects and their location in images. *IEEE International Conference on Computer Vision*.

Viola, P., & Jones, M. (2001). Robust real-time object detection. *Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling*.

Winn, J., & Shotton, J. (2006). The layout consistent random field for recognizing and segmenting partially occluded objects. *CVPR.*

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *ICML*.

Appendix A. Deriving Learning Algorithms

Exponential loss

We derive parallel updates for exponential loss. Let $\lambda+\Delta\lambda$ be the new parameters value. We find an upper-bound to the change of objective function $\mathcal{E}(\lambda+\Delta\lambda)-\mathcal{E}(\lambda)$ by an *auxiliary function*, and then minimize the bound.

$$\begin{aligned}
&\mathcal{E}(\lambda+\Delta\lambda)-\mathcal{E}(\lambda)= \\
&\sum_{i,h,y} q_0(h|x_i)e^{\lambda.G(x_i,h,y)}\Big(e^{\Delta\lambda.G(x_i,h,y)}-1\Big)+ \\
&\gamma\sum_{j,y} e^{\lambda.G(x_j,h_j,y)}\Big(e^{\Delta\lambda.G(x_j,h_j,y)}-1\Big)\leq \\
&\sum_{i,h,y} q_\lambda(h,y|x_i)\Big(\frac{\sum_k g_k e^{\Delta\lambda s_k(x_i,h,y)C}}{C}+w_{x_i,h,y}-1\Big)+ \\
&\gamma\sum_{j,y} q_\lambda(y|x_j,h_j)\Big(\frac{\sum_k g_k e^{\Delta\lambda s_k(x_j,h_j,y)C}}{C}+w_{x_j,h_j,y}-1\Big) \\
&:=\mathcal{A}(\lambda,\Delta\lambda)
\end{aligned}$$

where $w_{x,y,h}=1-\sum_k\frac{|g_k(x,h,y)|}{C}$, $G(x,h,y)$ is a vector built from $g_k(x,h,y)$, and $s_k(x,h,y)$ is the sign of $g_k(x,h,y)$. We find the stationary point of the auxiliary function $\mathcal{A}(\lambda,\Delta\lambda)$ with respect to $\Delta\lambda_k$ by taking the derivative and setting it to zero, which gives us the updating rules. The sequential updates can also be derived similarly.

Log loss

The objective is to minimize the objective function $-\mathcal{L}(\lambda+\Delta\lambda)+\mathcal{L}(\lambda)$. First we find an upper-bound on the objective function:

$$\begin{aligned}
&\mathcal{L}(\lambda)-\mathcal{L}(\lambda+\Delta\lambda)= \\
&\sum_i\log\sum_{y,h}e^{\lambda+\Delta\lambda\cdot G(x_i,h,y)}-\log\sum_{y,h}e^{\lambda\cdot G(x_i,h,y)}- \\
&\gamma\sum_j\log\sum_y e^{\lambda+\Delta\lambda\cdot G(x_j,h_j,y)}-\log\sum_y e^{\lambda\cdot G(x_j,h_j,y)}= \\
&\sum_i\log\sum_{y,h}p_\lambda(h,y|x)e^{\Delta\lambda\cdot G(x_i,h,y)}+ \\
&\gamma\sum_j\log\sum_y p_\lambda(y|x_j,h_j)e^{\Delta\lambda\cdot G(x_j,h_j,y)}\leq \\
&\sum_i\sum_{y,h}p_\lambda(h,y|x)e^{\Delta\lambda\cdot G(x_i,h,y)} \\
&+\gamma\sum_j\sum_y p_\lambda(y|x_j,h_j)e^{\Delta\lambda\cdot G(x_j,h_j,y)}
\end{aligned}$$

The inequality holds because $\log x\leq x$. The last expression can be upper-bounded again (using a similar technique used for exponential loss), and the resultant upper-bound will be the auxiliary function. It can be shown that the update rules are the same to unnormalized model, but the difference is to use $p_\lambda(\cdot)$ rather than $q_\lambda(\cdot)$. The update rules for sequential updates can be derived similarly.

Grassmann Discriminant Analysis: a Unifying View on Subspace-Based Learning

Jihun Hamm JHHAM@SEAS.UPENN.EDU
Daniel D. Lee DDLEE@SEAS.UPENN.EDU
GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104 USA

Abstract

In this paper we propose a discriminant learning framework for problems in which data consist of linear subspaces instead of vectors. By treating subspaces as basic elements, we can make learning algorithms adapt naturally to the problems with linear invariant structures. We propose a unifying view on the subspace-based learning method by formulating the problems on the Grassmann manifold, which is the set of fixed-dimensional linear subspaces of a Euclidean space. Previous methods on the problem typically adopt an inconsistent strategy: feature extraction is performed in the *Euclidean* space while *non-Euclidean* distances are used. In our approach, we treat each subspace as a point in the Grassmann space, and perform feature extraction and classification in the same space. We show feasibility of the approach by using the Grassmann kernel functions such as the Projection kernel and the Binet-Cauchy kernel. Experiments with real image databases show that the proposed method performs well compared with state-of-the-art algorithms.

1. Introduction

We often encounter learning problems in which the basic elements of the data are *sets of vectors* instead of vectors. Suppose we want to recognize a person from multiple pictures of the individual, taken from different angles, under different illumination or at different places. When comparing such sets of image vectors, we are free to define the similarity between sets based on the similarity between image vectors (Shakhnarovich et al., 2002; Kondor & Jebara, 2003; Zhou & Chellappa, 2006).

In this paper, we specifically focus on those data that can be modeled as a collection of linear subspaces. In the example above, let's assume that the set of images of a single person is well approximated by a low dimensional subspace (Turk & Pentland, 1991), and the whole data is the collection of such subspaces. The benefits of using subspaces are two-fold: 1) comparing two subspaces is cheaper than comparing two sets directly when those sets are very large, and 2) it is more robust to missing data since the subspace can 'fill-in' the missing pictures. However the advantages come with the challenge of representing and handling the subspaces appropriately.

We approach the subspace-based learning problems by formulating the problems on the Grassmann manifold, the set of fixed-dimensional linear subspaces of a Euclidean space. With this unifying framework we can make analytic comparisons of the various distances of subspaces. In particular, we single out those distances that are induced from the *Grassmann kernels*, which are positive definite kernel functions on the Grassmann space. The Grassmann kernels allow us to use the usual kernel-based algorithms on this unconventional space and to avoid ad hoc approaches to the problem.

We demonstrate the proposed framework by using the Projection metric and the Binet-Cauchy metric and by applying kernel Linear Discriminant Analysis to classification problems with real image databases.

1.1. Contributions of the Paper

Although the Projection metric and the Binet-Cauchy metric were previously used (Chang et al., 2006; Wolf & Shashua, 2003), their potential for subspace-based learning has not been fully explored. In this work, we provide an analytic exposition of the two metrics as examples of the Grassmann kernels, and contrast the

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

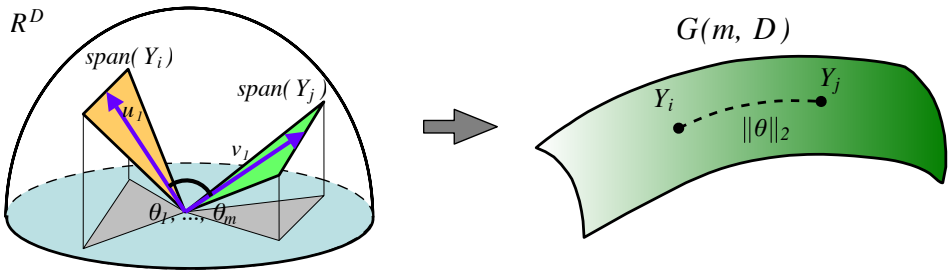


Figure 1. Principal angles and Grassmann distances. Let span(Y_i) and span(Y_j) be two subspaces in the Euclidean space $\mathbb{R}^D$ on the left. The distance between two subspaces span(Y_i) and span(Y_j) can be measured by the principal angles $\boldsymbol{\theta} = [\theta_1, \dots, \theta_m]'$ using the usual innerproduct of vectors. In the Grassmann manifold viewpoint, the subspaces span(Y_i) and span(Y_j) are considered as two points on the manifold $\mathcal{G}(m, D)$, whose Riemannian distance is related to the principal angles by $d(Y_i, Y_j) = \|\boldsymbol{\theta}\|_2$. Various distances can be defined based on the principal angles.

two metrics with other metrics used in the literature.

Several subspace-based classification methods have been previously proposed (Yamaguchi et al., 1998; Sakano, 2000; Fukui & Yamaguchi, 2003; Kim et al., 2007). However, these methods adopt an inconsistent strategy: feature extraction is performed in the *Euclidean* space when *non-Euclidean* distances are used. This inconsistency can result in complications and weak guarantees. In our approach, the feature extraction and the distance measurement are integrated around the Grassmann kernel, resulting in a simpler and better-understood formulation.

The rest of the paper is organized as follows. In Sec. 2 and 3 we introduce the Grassmann manifolds and derive various distances on the space. In Sec. 4 we present a kernel view of the problem and emphasize the advantages of using positive definite metrics. In Sec. 5 we propose the Grassmann Discriminant Analysis and compare it with other subspace-based discrimination methods. In Sec. 6 we test the proposed algorithm for face recognition and object categorization tasks. We conclude in Sec. 7 with a discussion.

2. Grassmann Manifold and Principal Angles

In this section we briefly review the Grassmann manifold and the principal angles.

Definition 1 *The Grassmann manifold $\mathcal{G}(m, D)$ is the set of m-dimensional linear subspaces of the $\mathbb{R}^D$.*

The $\mathcal{G}(m, D)$ is a $m(D-m)$-dimensional compact Riemannian manifold.[1] An element of $\mathcal{G}(m, D)$ can be represented by an orthonormal matrix Y of size D by m such that $Y'Y = I_m$, where I_m is the m by m identity matrix. For example, Y can be the m basis vectors of a set of pictures in $\mathbb{R}^D$. However, the matrix representation of a point in $\mathcal{G}(m, D)$ is not unique: two matrices Y_1 and Y_2 are considered the same if and only if span(Y_1) = span(Y_2), where span(Y) denotes the subspace spanned by the column vectors of Y. Equivalently, span(Y_1) = span(Y_2) if and only if $Y_1 R_1 = Y_2 R_2$ for some $R_1, R_2 \in \mathcal{O}(m)$. With this understanding, we will often use the notation Y when we actually mean its equivalence class span(Y), and use $Y_1 = Y_2$ when we mean span(Y_1) = span(Y_2), for simplicity.

Formally, the Riemannian distance between two subspaces is the length of the shortest geodesic connecting the two points on the Grassmann manifold. However, there is a more intuitive and computationally efficient way of defining the distances using the *principal angles* (Golub & Loan, 1996).

Definition 2 *Let Y_1 and Y_2 be two orthonormal matrices of size D by m. The principal angles $0 \le \theta_1 \le \cdots \le \theta_m \le \pi/2$ between two subspaces* span(Y_1) *and* span(Y_2)*, are defined recursively by*

$$\cos\theta_k = \max_{\boldsymbol{u_k} \in \text{span}(Y_1)} \max_{\boldsymbol{v_k} \in \text{span}(Y_2)} \boldsymbol{u_k}'\boldsymbol{v_k}, \quad \textit{subject to}$$

$$\boldsymbol{u_k}'\boldsymbol{u_k} = 1,\ \boldsymbol{v_k}'\boldsymbol{v_k} = 1,$$

$$\boldsymbol{u_k}'\boldsymbol{u_i} = 0,\ \boldsymbol{v_k}'\boldsymbol{v_i} = 0,\ (i = 1, \dots, k-1).$$

In other words, the first principal angle θ_1 is the smallest angle between all pairs of unit vectors in the first and the second subspaces. The rest of the principal

[1] $\mathcal{G}(m, D)$ can be derived as a quotient space of orthogonal groups $\mathcal{G}(m, D) = \mathcal{O}(D)/\mathcal{O}(m) \times \mathcal{O}(D-m)$, where $\mathcal{O}(m)$ is the group of m by m orthonormal matrices. We refer the readers to (Wong, 1967; Absil et al., 2004) for details on the Riemannian geometry of the space.

angles are similarly defined. It is known (Wong, 1967; Edelman et al., 1999) that the principal angles are related to the geodesic distance by $d_G^2(Y_1, Y_2) = \sum_i \theta_i^2$ (refer to Fig. 1.)

The principal angles can be computed from the Singular Value Decomposition (SVD) of $Y_1'Y_2$,

$$Y_1'Y_2 = U(\cos\Theta)V', \tag{1}$$

where $U = [\boldsymbol{u_1}\ ...\ \boldsymbol{u_m}]$, $V = [\boldsymbol{v_1}\ ...\ \boldsymbol{v_m}]$, and $\cos\Theta$ is the diagonal matrix $\cos\Theta = \mathrm{diag}(\cos\theta_1\ ...\ \cos\theta_m)$. The cosines of the principal angles $\cos\theta_1,\ ...\ , \cos\theta_m$ are also known as *canonical correlations*.

Although the definition can be extended to the cases where Y_1 and Y_2 have different number of columns, we will assume Y_1 and Y_2 have the same size D by m throughout this paper. Also, we will occasionally use $\mathcal{G}$ instead of $\mathcal{G}(m, D)$ for simplicity.

3. Distances for Subspaces

In this paper we use the term *distance* as any assignment of nonnegative values for each pair of points in a space $\mathcal{X}$. A valid *metric* is, however, a distance that satisfies the additional axioms:

Definition 3 *A real-valued function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a metric if*

1. $d(x_1, x_2) \geq 0$,
2. $d(x_1, x_2) = 0$ *if and only if* $x_1 = x_2$,
3. $d(x_1, x_2) = d(x_2, x_1)$,
4. $d(x_1, x_2) + d(x_2, x_3) \leq d(x_1, x_3)$,

for all $x_1, x_2, x_3 \in \mathcal{X}$.

A distance (or a metric) between subspaces $d(Y_1, Y_2)$ has to be invariant under different representations $d(Y_1, Y_2) = d(Y_1R_1, Y_2R_2),\ \ \forall R_1, R_2 \in \mathcal{O}(m)$.

In this section we introduce various distances for subspaces derivable from the principal angles.

3.1. Projection Metric and Binet-Cauchy Metric

We first underline two main distances of this paper.

1. *Projection metric*

$$d_P(Y_1, Y_2) = \left(\sum_{i=1}^{m} \sin^2\theta_i\right)^{1/2} = \left(m - \sum_{i=1}^{m} \cos^2\theta_i\right)^{1/2}. \tag{2}$$

The Projection metric is the 2-norm of the sine of principal angles (Edelman et al., 1999; Wang et al., 2006).

2. *Binet-Cauchy metric*

$$d_{BC}(Y_1, Y_2) = \left(1 - \prod_i \cos^2\theta_i\right)^{1/2}. \tag{3}$$

The Binet-Cauchy metric is defined with the product of canonical correlations (Wolf & Shashua, 2003; Vishwanathan & Smola, 2004).

As the names hint, these two distances are in fact valid metrics satisfying Def. 3. The proofs are deferred until Sec. 4.

3.2. Other Distances in the Literature

We describe a few other distances used in the literature. The principal angles are the keys that relate these distances.

1. *Max Correlation*

$$d_{\mathrm{Max}}(Y_1, Y_2) = \left(1 - \cos^2\theta_1\right)^{1/2} = \sin\theta_1. \tag{4}$$

The max correlation is a distance based on only the largest canonical correlation $\cos\theta_1$ (or the smallest principal angle θ_1). This max correlation was used in previous works (Yamaguchi et al., 1998; Sakano, 2000; Fukui & Yamaguchi, 2003).

2. *Min Correlation*

$$d_{\mathrm{Min}}(Y_1, Y_2) = \left(1 - \cos^2\theta_m\right)^{1/2} = \sin\theta_m. \tag{5}$$

The min correlation is defined similarly to the max correlation. However, the min correlation is more closely related to the Projection metric: we can rewrite the Projection metric as $d_P = 2^{-1/2}\ \|Y_1Y_1' - Y_2Y_2'\|_F$ and the min correlation as $d_{\mathrm{Min}} = \|Y_1Y_1' - Y_2Y_2'\|_2$.

3. *Procrustes metric*

$$d_{CF}(Y_1, Y_2) = 2\left(\sum_{i=1}^{m} \sin^2(\theta_i/2)\right)^{1/2}. \tag{6}$$

The Procrustes metric is the minimum distance between different representations of two subspaces $\mathrm{span}(Y_1)$ and $\mathrm{span}(Y_2)$: (Chikuse, 2003)

$$d_{CF} = \min_{R_1, R_2 \in \mathcal{O}(m)} \|Y_1R_1 - Y_2R_2\|_F = \|Y_1U - Y_2V\|_F,$$

where U and V are from (1). By definition, the distance is invariant of the choice of the

bases of $\text{span}(Y_1)$ and $\text{span}(Y_2)$. The Procrustes metric is also called chordal distance (Edelman et al., 1999). We can similarly define the minimum distance using other matrix norms such as $d_{C2}(Y_1, Y_2) = \|Y_1U - Y_2V\|_2 = 2\sin(\theta_m/2)$.

3.3. Which Distance to Use?

The choice of the best distance for a classification task depends on a few factors. The first factor is the distribution of data. Since the distances are defined with particular combinations of the principal angles, the best distance depends highly on the probability distribution of the principal angles of the given data. For example, d_{Max} uses the smallest principal angle θ_1 only, and may be robust when the data are noisy. On the other hand, when all subspaces are sharply concentrated on one point, d_{Max} will be close to zero for most of the data. In this case, d_{Min} may be more discriminative. The Projection metric d_P, which uses all the principal angles, will show intermediate characteristics between the two distances. Similar arguments can be made for the Procrustes metrics d_{CF} and d_{C2}, which use all angles and the largest angle only, respectively.

The second criterion for choosing the distance, is the degree of structure in the distance. Without any structure a distance can be used only with a simple K-Nearest Neighbor (K-NN) algorithm for classification. When a distance have an extra structure such as triangle inequality, for example, we can speed up the nearest neighbor searches by estimating lower and upper limits of unknown distances (Faragó et al., 1993). From this point of view, the max correlation is not a metric and may not be used with more sophisticated algorithms. On the other hand, the Min Correlation and the Procrustes metrics are valid metrics[2].

The most structured metrics are those which are induced from a positive definite kernel. Among the metrics mentioned so far, only the Projection metric and the Binet-Cauchy metric belong to this class. The proof and the consequences of positive definiteness are the main topics of the next section.

4. Kernel Functions for Subspaces

We have defined a valid metric on Grassmann manifolds. The next question is whether we can define a kernel function compatible with the metric. For this purpose let's recall a few definitions. Let $\mathcal{X}$ be any set, and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric real-valued function $k(x_i, x_j) = k(x_j, x_i)$ for all $x_i, x_j \in \mathcal{X}$.

Definition 4 *A real symmetric function is a (resp. conditionally) positive definite kernel function, if* $\sum_{i,j} c_i c_j k(x_i, x_j) \geq 0$, *for all* $x_1, ..., x_n (x_i \in \mathcal{X})$ *and* $c_1, ..., c_n (c_i \in \mathbb{R})$ *for any* $n \in \mathbb{N}$. *(resp. for all* $c_1, ..., c_n (c_i \in \mathbb{R})$ *such that* $\sum_{i=1}^n c_i = 0$.*)*

In this paper we are interested in the kernel functions on the Grassmann space.

Definition 5 *A Grassmann kernel function is a positive definite kernel function on* $\mathcal{G}$.

In the following we show that the Projection metric and the Binet-Cauchy are induced from the Grassmann kernels.

4.1. Projection Metric

The Projection metric can be understood by associating a point $\text{span}(Y) \in \mathcal{G}$ with its projection matrix YY' by an embedding:

$$\Psi_P : \mathcal{G}(m, D) \to \mathbb{R}^{D \times D}, \quad \text{span}(Y) \mapsto YY'. \tag{7}$$

The image $\Psi_P(\mathcal{G}(m, D))$ is the set of rank-m orthogonal projection matrices. This map is in fact an isometric embedding (Chikuse, 2003) and the projection metric is simply a Euclidean distance in $\mathbb{R}^{D \times D}$. The corresponding innerproduct of the space is $\text{tr}\left[(Y_1Y_1')(Y_2Y_2')\right] = \|Y_1'Y_2\|_F^2$, and therefore

Proposition 1 *The Projection kernel*

$$k_P(Y_1, Y_2) = \|Y_1'Y_2\|_F^2 \tag{8}$$

is a Grassmann kernel.

Proof The kernel is well-defined because $k_P(Y_1, Y_2) = k_P(Y_1R_1, Y_2R_2)$ for any $R_1, R_2 \in \mathcal{O}(m)$. The positive definiteness follows from the properties of the Frobenius norm. For all $Y_1, ..., Y_n (Y_i \in \mathcal{G})$ and $c_1, ..., c_n (c_i \in \mathbb{R})$ for any $n \in \mathbb{N}$, we have

$$\begin{aligned} \sum_{ij} c_i c_j \|Y_i'Y_j\|_F^2 &= \sum_{ij} c_i c_j \text{tr}(Y_iY_i'Y_jY_j') \\ = \text{tr}(\sum_i c_i Y_iY_i')^2 &= \|\sum_i c_i Y_iY_i'\|_F^2 \geq 0. \quad \blacksquare \end{aligned}$$

We can generate a family of kernels from the Projection kernel. For example, the square-root $\|Y_i'Y_j\|_F$ is also a positive definite kernel.

[2]The metric properties follow from the properties of matrix 2-norm and F-norm. To check the conditions in Def. 3 for Procrustes we use the equality $\min_{R_1,R_2} \|Y_1R_1 - Y_2R_2\|_{2,F} = \min_{R_3} \|Y_1 - Y_2R_3\|_{2,F}$ for $R_1, R_2, R_3 \in \mathcal{O}(m)$.

4.2. Binet-Cauchy Metric

The Binet-Cauchy metric can also be understood from an embedding. Let s be a subset of $\{1, ..., D\}$ with m elements $s = \{r_1, ..., r_m\}$, and $Y^{(s)}$ be the $m \times m$ matrix whose rows are the $r_1, \ ... \ , r_m$-th rows of Y. If $s_1, s_2, ..., s_n$ are all such choices of the subset s ordered lexicographically, then the Binet-Cauchy embedding is defined as

$$\Psi_{BC} : \mathcal{G}(m, D) \to \mathbb{R}^n, \quad Y \mapsto \left(\det Y^{(s_1)}, ..., \det Y^{(s_n)} \right), \tag{9}$$

where $n = {}_D C_m$ is the number of choosing m rows out of D rows. The natural innerproduct in this case is $\sum_{r=1}^{n} \det Y_1^{(s_i)} \det Y_2^{(s_i)}$.

Proposition 2 *The Binet-Cauchy kernel*

$$k_{BC}(Y_1, Y_2) = (\det Y_1' Y_2)^2 = \det Y_1' Y_2 Y_2' Y_1 \tag{10}$$

is a Grassmann kernel.

Proof First, the kernel is well-defined because $k_{BC}(Y_1, Y_2) = k_{BC}(Y_1 R_1, Y_2 R_2)$ for any $R_1, R_2 \in \mathcal{O}(m)$. To show that k_{BC} is positive definite it suffices to show that $k(Y_1, Y_2) = \det Y_1' Y_2$ is positive definite. From the Binet-Cauchy identity, we have

$$\det Y_1' Y_2 = \sum_s \det Y_1^{(s)} \det Y_2^{(s)}.$$

Therefore, for all $Y_1, ..., Y_n (Y_i \in \mathcal{G})$ and $c_1, ..., c_n (c_i \in \mathbb{R})$ for any $n \in \mathbb{N}$, we have

$$\begin{aligned} \sum_{ij} c_i c_j \det Y_i' Y_j &= \sum_{ij} c_i c_j \sum_s \det Y_i^{(s)} \det Y_j^{(s)} \\ &= \sum_s \left(\sum_i c_i \det Y_i^{(s)} \right)^2 \geq 0. \end{aligned}$$ ∎

We can also generate another family of kernels from the Binet-Cauchy kernel. Note that although $\det Y_1' Y_2$ is a Grassmann kernel we prefer using $k_{BC}(Y_1, Y_2) = \det(Y_1' Y_2)^2$, since it is directly related to principal angles $\det(Y_1' Y_2)^2 = \prod \cos^2 \theta_i$, whereas $\det Y_1' Y_2 \neq \prod \cos \theta_i$ in general.[3] Another variant $\arcsin k_{BC}(Y_1, Y_2)$ is also a positive definite kernel[4] and its induced metric $d = (\arccos(\det Y_1' Y_2))^{1/2}$ is a conditionally positive definite metric.

4.3. Indefinite Kernels from Other Metrics

Since the Projection metric and the Binet-Cauchy metric are derived from positive definite kernels, all the kernel-based algorithms for Hilbert spaces are at our disposal. In contrast, other metrics in the previous sections are not associated with any Grassmann kernel. To show this we can use the following result (Schoenberg, 1938; Hein et al., 2005):

Proposition 3 *A metric d is induced from a positive definite kernel if and only if*

$$\hat{k}(x_1, x_2) = -d^2(x_1, x_2)/2, \quad x_1, x_2 \in \mathcal{X} \tag{11}$$

is conditionally positive definite.

The proposition allows us to show a metric's non-positive definiteness by constructing an indefinite kernel matrix from (11) as a counterexample.

There have been efforts to use indefinite kernels for learning (Ong et al., 2004; Haasdonk, 2005), and several heuristics have been proposed to make an indefinite kernel matrix to a positive definite matrix (Pekalska et al., 2002). However, we do not advocate the use of the heuristics since they change the geometry of the original data.

5. Grassmann Discriminant Analysis

In this section we give an example of the Discriminant Analysis on Grassmann space by using kernel LDA with the Grassmann kernels.

5.1. Linear Discriminant Analysis

The Linear Discriminant Analysis (LDA) (Fukunaga, 1990), followed by a K-NN classifier, has been successfully used for classification.

Let $\{\boldsymbol{x_1}, ..., \boldsymbol{x_N}\}$ be the data vectors and $\{y_1, ..., y_N\}$ be the class labels $y_i \in \{1, ..., C\}$. Without loss of generality we assume the data are ordered according to the class labels: $1 = y_1 \leq y_2 \leq ... \leq y_N = C$. Each class c has N_c number of samples.

Let $\boldsymbol{\mu_c} = 1/N_c \sum_{\{i|y_i=c\}} \boldsymbol{x_i}$ be the mean of class c, and $\boldsymbol{\mu} = 1/N \sum_i \boldsymbol{x_i}$ be the overall mean. LDA searches for the discriminant direction $\boldsymbol{w}$ which maximizes the Rayleigh quotient $L(\boldsymbol{w}) = \boldsymbol{w}' S_b \boldsymbol{w} / \boldsymbol{w}' S_w \boldsymbol{w}$ where S_b and S_w are the between-class and within-class covariance matrices respectively:

$$\begin{aligned} S_b &= \frac{1}{N} \sum_{c=1}^{C} N_c (\boldsymbol{\mu_c} - \boldsymbol{\mu})(\boldsymbol{\mu_c} - \boldsymbol{\mu})' \\ S_w &= \frac{1}{N} \sum_{c=1}^{C} \sum_{\{i|y_i=c\}} (\boldsymbol{x_i} - \boldsymbol{\mu_c})(\boldsymbol{x_i} - \boldsymbol{\mu_c})' \end{aligned}$$

The optimal $\boldsymbol{w}$ is obtained from the largest eigenvector of $S_w^{-1} S_b$. Since $S_w^{-1} S_b$ has rank $C - 1$, there are

[3] $\det Y_1' Y_2$ can be negative whereas $\prod \cos \theta_i$, the product of singular values, is nonnegative by definition.

[4] Theorem 4.18 and 4.19 (Schölkopf & Smola, 2001).

$C-1$-number of local optima $W = \{\boldsymbol{w_1}, ..., \boldsymbol{w_{C-1}}\}$. By projecting data onto the space spanned by W, we achieve dimensionality reduction and feature extraction of data onto the most discriminant subspace.

5.2. Kernel LDA with Grassmann Kernels

Kernel LDA can be formulated by using the kernel trick as follows. Let $\boldsymbol{\phi} : \mathcal{G} \to \mathcal{H}$ be the feature map, and $\Phi = [\boldsymbol{\phi_1}...\boldsymbol{\phi_N}]$ be the feature matrix of the training points. Assuming w is a linear combination of the those feature vectors, $\boldsymbol{w} = \Phi\boldsymbol{\alpha}$, we can rewrite the Rayleigh quotient in terms of $\boldsymbol{\alpha}$ as

$$L(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha}'\Phi' S_B \Phi\boldsymbol{\alpha}}{\boldsymbol{\alpha}'\Phi' S_W \Phi\boldsymbol{\alpha}} = \frac{\boldsymbol{\alpha}' K(V - \mathbf{1}_N\mathbf{1}_N'/N)K\boldsymbol{\alpha}}{\boldsymbol{\alpha}'(K(I_N - V)K + \sigma^2 I_N)\boldsymbol{\alpha}}, \tag{12}$$

where K is the kernel matrix, $\mathbf{1}_N$ is a uniform vector $[1\ ...\ 1]'$ of length N, V is a block-diagonal matrix whose c-th block is the uniform matrix $\mathbf{1}_{N_c}\mathbf{1}_{N_c}'/N_c$, and $\sigma^2 I_N$ is a regularizer for making the computation stable. Similarly to LDA, the set of optimal $\boldsymbol{\alpha}$'s are computed from the eigenvectors.

The procedures for using kernel LDA with the Grassmann kernels are summarized below:

Assume the D by m orthonormal bases $\{Y_i\}$ are already computed from the SVD of sets in the data.

Training:

1. *Compute the matrix $[K_{\text{train}}]_{ij} = k_P(Y_i, Y_j)$ or $k_{BC}(Y_i, Y_j)$ for all Y_i, Y_j in the training set.*
2. *Solve $\max_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha})$ by eigen-decomposition.*
3. *Compute the $(C-1)$-dimensional coefficients $F_{\text{train}} = \boldsymbol{\alpha}' K_{\text{train}}$.*

Testing:

1. *Compute the matrix $[K_{\text{test}}]_{ij} = k_P(Y_i, Y_j)$ or $k_{BC}(Y_i, Y_j)$ for all Y_i in training set and Y_j in the test set.*
2. *Compute the $(C-1)$-dim coefficients $F_{\text{test}} = \boldsymbol{\alpha}' K_{\text{test}}$.*
3. *Perform 1-NN classification from the Euclidean distance between F_{train} and F_{test}.*

Another way of applying LDA to subspaces is to use the Projection embedding Ψ_P (7) or the Binet-Cauchy embedding Ψ_{BC} (9) directly. A subspace is represented by a D by D matrix in the former, or by a vector of length $n = {}_D C_m$ in the latter. However, using these embeddings to compute S_b or S_w is a waste of computation and storage resources when D is large.

5.3. Other Subspace-Based Algorithms

5.3.1. Mutual Subspace Method (MSM)

The original MSM (Yamaguchi et al., 1998) performs simple 1-NN classification with d_{Max} with no feature extraction. The method can be extended to any distance described in the paper. There are attempts to use kernels for MSM (Sakano, 2000). However, the kernel is used only to represent data in the original space, and the algorithm is still a 1-NN classification.

5.3.2. Constrained MSM

Constrained MSM (Fukui & Yamaguchi, 2003) is a technique that applies dimensionality reduction to bases of the subspaces in the original space. Let $G = \sum_i Y_i Y_i'$ be the sum of the projection matrices and $\{\boldsymbol{v_1}, ..., \boldsymbol{v_D}\}$ be the eigenvectors corresponding to the eigenvalues $\{\lambda_1 \leq ... \leq \lambda_D\}$ of G. The authors claim that the first few eigenvectors $\boldsymbol{v_1}, ..., \boldsymbol{v_d}$ of G are more discriminative than the later eigenvectors, and they suggest projecting the basis vectors of each subspace Y_1 onto the span$(\boldsymbol{v_1}, ..., \boldsymbol{v_l})$, followed by normalization and orthonormalization. However these procedure lack justifications, as well as a clear criterion for choosing the dimension d, on which the result crucially depends from our experience.

5.3.3. Discriminant Analysis of Canonical Correlations (DCC)

DCC (Kim et al., 2007) can be understood as a nonparametric version of linear discrimination analysis using the Procrustes metric (6). The algorithm finds the discriminating direction $\boldsymbol{w}$ which maximize the ratio $L(\boldsymbol{w}) = \boldsymbol{w}' S_B \boldsymbol{w} / \boldsymbol{w}' S_w \boldsymbol{w}$, where S_b and S_w are the nonparametric between-class and within-class 'covariance' matrices:

$$\begin{aligned} S_b &= \sum_i \sum_{j \in B_i} (Y_i U - Y_j V)(Y_i U - Y_j V)' \\ S_w &= \sum_i \sum_{j \in W_i} (Y_i U - Y_j V)(Y_i U - Y_j V)', \end{aligned}$$

where U and V are from (1). Recall that $\text{tr}(Y_i U - Y_j V)(Y_i U - Y_j V)' = \|Y_i U - Y_j V\|_F^2$ is the squared Procrustes metric. However, unlike our method, S_b and S_w do not admit a geometric interpretation as true covariance matrices, and cannot be kernelized either. A main disadvantage of the DCC is that the algorithm iterates the two stages of 1) maximizing the ratio $L(\mathbf{w})$ and of 2) computing S_b and S_w, which results in computational overheads and more parame-

ters to be determined. This reflects the complication of treating the problem in a Euclidean space with a non-Euclidean distance.

6. Experiments

In this section we test the Grassmann Discriminant Analysis for 1) a face recognition task and 2) an object categorization task with real image databases.

6.1. Algorithms

We use the following six methods for feature extraction together with an 1-NN classifier.

1) GDA1 (with Projection kernel), 2) GDA2 (with Binet-Cauchy kernel), 3) Min dist , 4) MSM, 5) cMSM, and 6) DCC.

For GDA1 and GDA2, the optimal values of σ are found by scanning through a range of values. The results do not seem to vary much as long as σ is small enough. The Min dist is a simple pairwise distance which is not subspace-based. If Y_i and Y_j are two sets of basis vectors: $Y_i = \{\boldsymbol{y_{i1}}, ..., \boldsymbol{y_{im_i}}\}$ and $Y_j = \{\boldsymbol{y_{j1}}, ..., \boldsymbol{y_{jm_j}}\}$, then $d_{\mathrm{Mindist}}(Y_i, Y_j) = \min_{k,l} \|\boldsymbol{y_{ik}} - \boldsymbol{y_{jl}}\|_2$. For cMSM and DCC, the optimal dimension l is found by exhaustive searching. For DCC, we have used two nearest-neighbors for B_i and W_i in Sec. 5.3.3. Since the S_w and S_b are likely to be rank deficient, we first reduced the dimension of the data to $N - C$ using PCA as recommended. Each optimization is iterated 5 times.

6.2. Testing Illumination-Invariance with Yale Face Database

The Yale face database and the Extended Yale face database (Georghiades et al., 2001) together consist of pictures of 38 subjects with 9 different poses and 45 different lighting conditions. Face regions were cropped from the original pictures, resized to 24×21 pixels ($D = 504$), and normalized to have the same variance. For each subject and each pose, we model the illumination variations by a subspace of the size $m = 1, ..., 5$, spanned by the 1 to 5 largest eigenvectors from SVD. We evaluate the recognition rate of subjects with nine-fold cross validation, holding out one pose of all subjects from the training set and using it for test.

The recognition rates are shown in Fig. 2. The GDA1 outperforms the other methods consistently. The GDA2 also performs well for small m, but performs worse as m becomes large. The rates of the others also seem to decrease as m increases. An interpretation of the observation is that the first few eigenvectors from the data already have enough information and the smaller eigenvectors are spurious for discriminating the subjects.

6.3. Testing Pose-Invariance with ETH-80 Database

The ETH-80 (Leibe & Schiele, 2003) database consists of pictures of 8 object categories ('apple', 'pear', 'tomato', 'cow', 'dog', 'horse', 'cup', 'car'). Each category has 10 objects that belong to the category, and each object is recorded under 41 different poses. Images were resized to 32×32 pixels ($D = 1024$) and normalized to have the same variance. For each category and each object, we model the pose variations by a subspace of the size $m = 1, ..., 5$, spanned by the 1 to 5 largest eigenvectors from SVD. We evaluate the classification rate of the categories with ten-fold cross validation, holding out one object instance of each category from the training set and using it for test.

The recognition rates are also summarized in Fig. 2. The GDA1 also outperforms the other methods most of the time, but the cMSM performs better than GDA2 as m increases. The rates seem to peak around $m = 4$ and then decrease as m increases. This results is consistent with the observation that the eigenvalues from this database decrease more gradually than the eigenvalues from the Yale face database.

7. Conclusion

In this paper we have proposed a Grassmann framework for problem in which data consist of subspaces. By using the Projection metric and the Binet-Cauchy metric, which are derived from the Grassmann kernels, we were able to apply kernel methods such as kernel LDA to subspace data. In addition to having theoretically sound grounds, the proposed method also outperformed state-of-the-art methods in two experiments with real data. As a future work, we are pursuing a better understanding of probabilistic distributions on the Grassmann manifold.

References

Absil, P., Mahony, R., & Sepulchre, R. (2004). Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. *Acta Appl. Math.*, *80*, 199–220.

Chang, J.-M., Beveridge, J. R., Draper, B. A., Kirby, M., Kley, H., & Peterson, C. (2006). Illumination face spaces are idiosyncratic. *IPCV* (pp. 390–396).

Chikuse, Y. (2003). *Statistics on special manifolds, lecture notes in statistics, vol. 174.* New York: Springer.

Edelman, A., Arias, T. A., & Smith, S. T. (1999). The

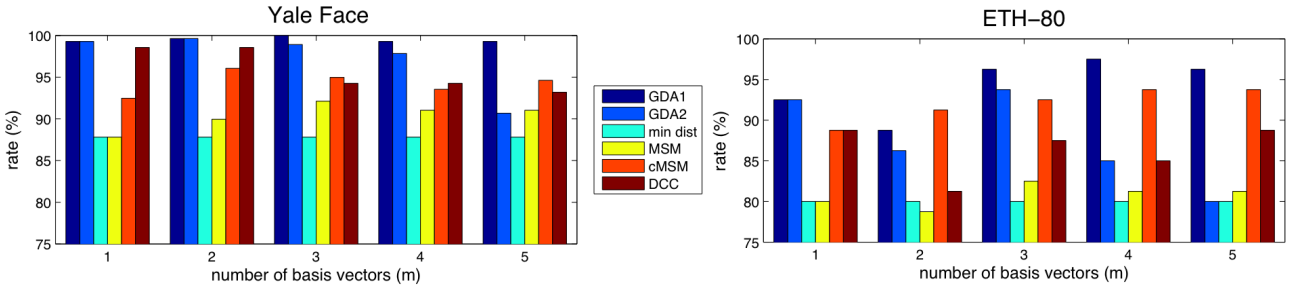


Figure 2. Recognition rates of subjects from Yale face database (Left), and classification rates of categories in ETH-80 database (Right). The bars represent the rates of six algorithms (GDA1, GDA2, Min Dist, MSM, cMSM, DCC) evaluated for $m = 1, ..., 5$ where m is the number of basis vectors for subspaces. The GDA1 achieves the best rates consistently, and the GDA2 also performs competitively for small m.

geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, *20*, 303–353.

Faragó, A., Linder, T., & Lugosi, G. (1993). Fast nearest-neighbor search in dissimilarity spaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, *15*, 957–962.

Fukui, K., & Yamaguchi, O. (2003). Face recognition using multi-viewpoint patterns for robot vision. *Int. Symp. of Robotics Res.* (pp. 192–201).

Fukunaga, K. (1990). *Introduction to statistical pattern recognition (2nd ed.)*. San Diego, CA, USA: Academic Press Professional, Inc.

Georghiades, A. S., Belhumeur, P. N., & Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.*, *23*, 643–660.

Golub, G. H., & Loan, C. F. V. (1996). *Matrix computations (3rd ed.)*. Baltimore, MD, USA: Johns Hopkins University Press.

Haasdonk, B. (2005). Feature space interpretation of svms with indefinite kernels. *IEEE Trans. Pattern Anal. Mach. Intell.*, *27*, 482–492.

Hein, M., Bousquet, O., & Schölkopf, B. (2005). Maximal margin classification for metric spaces. *J. Comput. Syst. Sci.*, *71*, 333–359.

Kim, T.-K., Kittler, J., & Cipolla, R. (2007). Discriminative learning and recognition of image set classes using canonical correlations. *IEEE Trans. Pattern Anal. Mach. Intell.*, *29*, 1005–1018.

Kondor, R. I., & Jebara, T. (2003). A kernel between sets of vectors. *Proc. of the 20th Int. Conf. on Mach. Learn.* (pp. 361–368).

Leibe, B., & Schiele, B. (2003). Analyzing appearance and contour based methods for object categorization. *CVPR*, *02*, 409.

Ong, C. S., Mary, X., Canu, S., & Smola, A. J. (2004). Learning with non-positive kernels. *Proc. of 21st Int. Conf. on Mach. Learn.* (p. 81). New York, NY, USA: ACM.

Pekalska, E., Paclik, P., & Duin, R. P. W. (2002). A generalized kernel approach to dissimilarity-based classification. *J. Mach. Learn. Res.*, *2*, 175–211.

Sakano, H.; Mukawa, N. (2000). Kernel mutual subspace method for robust facial image recognition. *Proc. of Int. Conf. on Knowledge-Based Intell. Eng. Sys. and App. Tech.* (pp. 245–248).

Schoenberg, I. J. (1938). Metric spaces and positive definite functions. *Trans. Amer. Math. Soc.*, *44*, 522–536.

Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge, MA, USA: MIT Press.

Shakhnarovich, G., John W. Fisher, I., & Darrell, T. (2002). Face recognition from long-term observations. *Proc. of the 7th Euro. Conf. on Computer Vision* (pp. 851–868). London, UK.

Turk, M., & Pentland, A. P. (1991). Eigenfaces for recognition. *J. Cog. Neurosc.*, *3*, 71–86.

Vishwanathan, S., & Smola, A. J. (2004). Binet-cauchy kernels. *Proc. of Neural Info. Proc. Sys.*.

Wang, L., Wang, X., & Feng, J. (2006). Subspace distance analysis with application to adaptive bayesian algorithm for face recognition. *Pattern Recogn.*, *39*, 456–464.

Wolf, L., & Shashua, A. (2003). Learning over sets using kernel principal angles. *J. Mach. Learn. Res.*, *4*, 913–931.

Wong, Y.-C. (1967). Differential geometry of Grassmann manifolds. *Proc. of the Nat. Acad. of Sci., Vol. 57*, 589–594.

Yamaguchi, O., Fukui, K., & Maeda, K. (1998). Face recognition using temporal image sequence. *Proc. of the 3rd. Int. Conf. on Face & Gesture Recognition* (p. 318). Washington, DC, USA: IEEE Computer Society.

Zhou, S. K., & Chellappa, R. (2006). From sample similarity to ensemble similarity: Probabilistic distance measures in reproducing kernel hilbert space. *IEEE Trans. Pattern Anal. Mach. Intell.*, *28*, 917–929.

Modified MMI/MPE: A Direct Evaluation of the Margin in Speech Recognition

Georg Heigold HEIGOLD@CS.RWTH-AACHEN.DE
Thomas Deselaers DESELAERS@CS.RWTH-AACHEN.DE
Ralf Schlüter SCHLUETER@CS.RWTH-AACHEN.DE
Hermann Ney NEY@CS.RWTH-AACHEN.DE
RWTH Aachen University Chair of Computer Science 6 - Computer Science Department D-52056 Aachen, Germany

Abstract

In this paper we show how common speech recognition training criteria such as the Minimum Phone Error criterion or the Maximum Mutual Information criterion can be extended to incorporate a margin term. Different margin-based training algorithms have been proposed to refine existing training algorithms for general machine learning problems. However, for speech recognition, some special problems have to be addressed and all approaches proposed either lack practical applicability or the inclusion of a margin term enforces significant changes to the underlying model, e.g. the optimization algorithm, the loss function, or the parameterization of the model. In our approach, the conventional training criteria are modified to incorporate a margin term. This allows us to do large-margin training in speech recognition using the same efficient algorithms for accumulation and optimization and to use the same software as for conventional discriminative training. We show that the proposed criteria are equivalent to Support Vector Machines with suitable smooth loss functions, approximating the non-smooth hinge loss function or the hard error (e.g. phone error). Experimental results are given for two different tasks: the rather simple digit string recognition task Sietill which severely suffers from overfitting and the large vocabulary European Parliament Plenary Sessions English task which is supposed to be dominated by the risk and the generalization does not seem to be such an issue.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

1. Introduction

A central issue in machine learning is the robust estimation of the model parameters Λ with good generalization ability, based on a finite number of observations. An interesting result from information theory is the PAC bound on the expected risk (Vapnik, 1995). The VC dimension plays an important role in this inequality and is a direct measure for the generalization ability. This bound is general in the sense that it does neither depend on the underlying probability distribution nor on the specific risk function. Furthermore, the bound implies that in general, the consideration of the empirical risk alone is suboptimal (Vapnik, 1995), see Tab. 1. Assuming that the features are in a sphere, the VC dimension of gap-tolerant classifiers is bounded above by an expression which is inversely proportional to the margin, leading to large-margin classifiers (Jebara, 2002).

These theoretical results are the main motivation for Support Vector Machines (SVMs) (Vapnik, 1995), M-SVMs (Weston & Watkins, 1999), or Hidden Markov SVMs (Altun et al., 2003) which have been successfully used for many applications in pattern recognition. The direct application of SVMs in Automatic Speech Recognition (ASR) has not been successful so far. This might be because they are not sufficiently flexible regarding: 1) the choice of the loss function, conventional criteria in ASR are Maximum Mutual Information (MMI), Minimum Classification Error (MCE), or Minimum Phone Error (MPE) which is probably the criterion of choice in ASR; 2) they are unable to cope with the immense amount of data used to train state-of-the-art ASR systems, which are commonly trained on more than 100 hours of speech (>30,000,000 observation vectors). Another problem might be the combinatorial number of classes (number of possible word sequences). Stimulated by the success of SVMs, different margin-based training algorithms have been proposed for ASR, e.g. (Yu et al., 2007; Yin & Jiang, 2007; Sha & Saul, 2007; Li et al., 2007). Although the reported results for these approaches are very promising, the approaches have some shortcomings in particular for large-scale appli-

Table 1. Relative importance of loss and margin terms under different conditions.

Loss	vs.	Margin
infinite data	↔	sparse data
many training errors	↔	few training errors

cations. The approach proposed in (Yu et al., 2007) comes closest to ours but uses MCE on N-best lists without regularization. In most state-of-the-art large-scale ASR systems, however, MPE in combination with strong regularization, i.e., i-smoothing has been established to be the criterion of choice (Povey & Woodland, 2002). In (Yin & Jiang, 2007; Sha & Saul, 2007; Li et al., 2007) not only the margin term is introduced, but the approaches use different optimization algorithms, different loss functions, or different model parameterizations which makes it difficult to evaluate the effect of the margin term in these approaches. Furthermore, none of these papers reports experimental results for competitive large vocabulary systems whose behavior in terms of generalization ability and relative improvements of performance often is different to systems using suboptimal models or for "simple" small vocabulary tasks (e.g. TIDIGITS and TIMIT). A large amount of training data and a relatively large number of training errors are typical of such large vocabulary systems. From this observation, we expect that the margin term has only little impact on the performance of such systems, cf. Tab. 1. In this work, we pursue a similar approach as in (Zhang et al., 2003) where the standard M-SVM with the hinge loss function is approximated by modified logistic regression. To the best of our knowledge, this approach, is computationally unfeasible in ASR because of the pairwise treatment of the correct and all the competing word sequences. To avoid the exponential complexity, our approximations are based on the Hidden Markov SVM proposed in (Altun et al., 2003). Formally similar results can be found in (Jebara, 2002), which are derived from probabilistic reasoning. Using the smoothed segment error of MCE in combination with $N-$best lists and without regularization, the margin-based MCE criterion proposed in (Yu et al., 2007) is recovered as a special instance of our approach.

The remainder of this paper is organized as follows: Sec. 2 reviews SVMs in a notation suitable for our discussion. Approximations to the SVMs with different loss functions, resembling the MMI and MPE criterion are proposed in Sec. 3 and extended to ASR in Sec. 4. Experimental results using these modified criteria are presented for the Sietill and the European Parliament Plenary Sessions (EPPS) English ASR tasks, cf. Sec.6. The results of the latter task give an idea of the importance of the margin in a state-of-the-art large vocabulary system. Finally, Sec. 5 shows that the transducer-based implementation of MMI and MPE differs merely in the choice of the semiring. This section may be skipped at the first reading.

2. Support Vector Machines (SVMs)

According to (Altun et al., 2003), the optimization problem of SVMs for C classes, N observation pairs (x_n, c_n), and feature functions $f_i(x, c)$ can be formulated as follows

$$\hat{\Lambda} = \arg\min_{\Lambda} \left\{ \frac{1}{2}\|\Lambda\|^2 + \frac{J}{N}\sum_{n=1}^{N} l(c_n; d_{n1}, \ldots, d_{nC}) \right\} \quad (1)$$

with $d_{nc} = \sum_i \lambda_i (f_i(x_n, c_n) - f_i(x_n, c))$, or more compactly in vector notation $d_{nc} = \lambda^\top (f(x_n, c_n) - f(x_n, c))$. The empirical constant $J > 0$ is used to balance the margin and the loss terms. The typical loss function of SVMs is the hinge loss function

$$l^{(hinge)}(c_n; d_{n1}, \ldots, d_{nC}) = \max_{c \neq c_n} \{\max\{-d_{nc} + 1, 0\}\}. \quad (2)$$

This effectively reduces the multiclass problem to a two-class problem ("correct" vs. "recognized"). Ideally, the loss function is the margin error

$$l^{(error)}(c_n; d_{n1}, \ldots, d_{nC}) = E[\hat{c}_n | c_n], \quad (3)$$

which in the simplest case counts the errors of the observations, $1 - \delta(\hat{c}_n, c_n)$. For ASR, however, we choose string-based error measures like the phone error. In this loss function, $\hat{c}_n$ is in fact a function of $(c_n; d_{n1}, \ldots, d_{nC})$ and denotes the recognized class (with margin)

$$\hat{c}_n = \begin{cases} \arg\min_{c \neq c_n} \{d_{nc}\} & \text{if } \exists c \neq c_n : d_{nc} < 1 \\ c_n & \text{otherwise.} \end{cases} \quad (4)$$

Due to the definition of the loss function and in contrast to (Altun et al., 2003), this formulation of SVM does not require the introduction of slack variables ξ_n^c subject to $d_{nc} \geq \xi_n^c + 1$ and $\xi_n^c \geq 0$ for all $c \neq c_n$ and n. The resulting optimization problem is non-smooth, but it is only used for theoretical purposes whereas the experiments are carried out with smoothed loss functions as it is common in ASR. In contrast to the multiclass SVM proposed by (Weston & Watkins, 1999), this definition allows for efficient calculation of the sum over the classes in ASR (cf. Sec. 5).

In (Taskar et al., 2003), the size of the margin is set to be proportional to the length of the sequence, e.g. the number of correct symbols. For ASR, due to the additional alignment problem, this is extended such that the margin between two sequences is set to the associated sequence/string accuracy. Note that this extension is reasonable because it guarantees consistency with the above SVM in case of i.i.d. sequences, see Sec. 4 for further details.

Finally, the task of testing consists of finding the class with the highest score

$$\hat{c}(x) = \arg\max_{c} \left\{ \lambda^\top f(x, c) \right\}, \quad (5)$$

which should not be confused with $\hat{c}_n$ in Eq. (4).

3. SVMs with Smooth Loss Functions

This section provides smooth approximations to the SVM in Eq. (1) for different loss functions. More precisely, the loss function is replaced with a smoothed loss function without breaking the large margin nature of the original SVM. These approximations are identical to modified formulations of the well-known training criteria MMI and MPE for Hidden Conditional Random Fields (HCRFs), which are introduced in the next two subsections. Analogously, a similar result can be derived for (lattice-based) MCE. In contrast to (most) other margin-based approaches, these approximations have the advantage that the effect of the margin can be evaluated directly without changing the parameterization of the model, the loss function, or the optimization algorithm.

Keep in mind that the modifications concern only the training, i.e., the calculation of the probabilities in the search remains unchanged:

$$p_\Lambda(c|x) = \frac{\exp(\lambda^\top f(x,c))}{\sum_{c'} \exp(\lambda^\top f(x,c'))}.$$

The resulting decision rule is equivalent to the decision rule in Eq. (5) for SVMs because monotone transformations of the discriminant function do not change the decision rule.

In the next two subsections, we define modified criteria based on the conventional MMI and MPE criteria and show the relationship with SVMs.

3.1. Modified Maximum Mutual Information (MMI)

In ASR, MMI commonly refers to the maximum likelihood (ML) for the class posteriors. We define a modified MMI criterion for log-linear HCRFs [1]

$$\mathcal{F}_\gamma^{(MMI)}(\Lambda) = \frac{1}{2}\|\Lambda\|^2 - \frac{J}{N}\sum_{n=1}^{N}\frac{1}{\gamma}\log\left(\frac{\exp(\gamma\,(\lambda^\top f(x_n,c_n)-1))}{\sum_c \exp(\gamma\,(\lambda^\top f(x_n,c)-\delta(c,c_n)))}\right). \quad (6)$$

See Fig. 1 for a comparison of the hinge loss function, MMI, and modified MMI. The approximation level γ is an additional parameter to control the smoothness of the criterion. The regularization constant is proportional to $\frac{1}{J}$. The major difference to the standard MMI formulation (including L_2-norm regularization) is the additional margin parameter which is non-zero only for the correct class c_n. This margin term can be interpreted as an additional observation dependent prior, weakening the true prior (Jebara, 2002).

It can be shown that the objective function $\mathcal{F}_\gamma^{(MMI)}(\Lambda)$ converges pointwise to the SVM optimization problem using the hinge loss function in Eq. (2) for $\gamma \to \infty$, similar to (Zhang et al., 2003). In other words, $\mathcal{F}_\gamma^{(MMI)}(\Lambda)$ is a smooth approximation to an SVM with hinge loss function, which can be optimized with standard gradient-based optimization techniques. The proof mainly consists of building the limit of the logarithm in Eq. (6):

$$\begin{aligned}
&-\frac{1}{\gamma}\log\left(\frac{\exp(\gamma\,(\lambda^\top f(x_n,c_n)-1))}{\sum_c \exp(\gamma\,(\lambda^\top f(x_n,c)-\delta(c,c_n)))}\right)\\
&=\frac{1}{\gamma}\log\left(1+\sum_{c\neq c_n}\exp(\gamma\,(-d_{nc}+1))\right)\\
&\overset{\gamma\to\infty}{\to}\begin{cases}\max_{c\neq c_n}\{-d_{nc}+1\} & \text{if } \exists c\neq c_n : d_{nc}<1\\ 0 & \text{otherwise.}\end{cases}
\end{aligned}$$

This function can be identified with the hinge loss function in Eq. (2).

We feel that the weak point about the hinge loss in pattern recognition is that it is not the measure used to evaluate the recognition systems eventually. This means that there is some guarantee regarding the generalization for the hinge loss, but *not* the recognition error. Furthermore, it is often unclear how these two quantities are related.

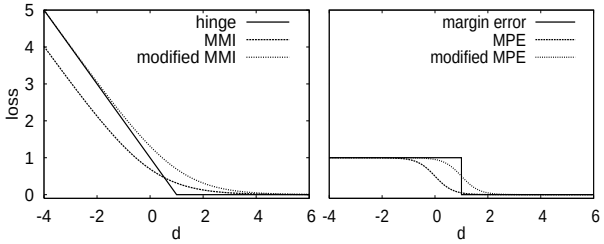


Figure 1. Left: comparison of hinge loss, MMI, and modified MMI, $\gamma = 1$. Right: comparison of margin error loss, MPE, and modified MPE, $\gamma = 3$. In either case $C = 2$, and $d = d_{nc_n}$.

3.2. Modified Minimum Phone Error (MPE)

In contrast to the hinge loss, the recognition error is bounded as illustrated in Fig. 1. Hence, a single observation cannot dominate the objective function. In particular, do not mix up a weighted margin with a weighted error.

We shall show that the modified MPE-like objective function representing a smoothed margin error with L_2-norm regularization,

$$\mathcal{F}_\gamma^{(MPE)}(\Lambda) = \frac{1}{2}\|\Lambda\|^2 + \frac{J}{N}\sum_{n=1}^{N}\sum_c E[c|c_n]\frac{\exp(\gamma\,(\lambda^\top f(x_n,c)-\delta(c,c_n)))}{\sum_{c'}\exp(\gamma\,(\lambda^\top f(x_n,c')-\delta(c',c_n)))}$$

converges to the above SVM optimization problem with a hard and weighted loss function $E[\cdot|\cdot]$ as in Eq. (3), e.g. the phone error. The proof is analogous to the proof for MMI.

[1] The first order features in (Zhang et al., 2003) are a special case of the more general feature functions used here.

The main step is to show that the "posterior probabilities" in $\mathcal{F}_\gamma^{(MPE)}(\Lambda)$ converge to a Kronecker delta such that only a single term contributes to the sum of the empirical risk

$$\begin{aligned}
&\frac{\exp(\gamma\,(\lambda^\top f(x_n, c) - \delta(c, c_n)))}{\sum_{c'} \exp(\gamma\,(\lambda^\top f(x_n, c') - \delta(c', c_n)))} \\
&= \begin{cases} \frac{1}{1+\sum_{c' \neq c_n} \exp(\gamma(-d_{nc'}+1))} & \text{if } c = c_n, \\ \frac{\exp(\gamma(-d_{nc}+1))}{1+\sum_{c' \neq c_n} \exp(\gamma(-d_{nc'}+1))} & \text{otherwise} \end{cases} \\
&\overset{\gamma\to\infty}{\to} \begin{cases} \delta(c, \arg\min_{c \neq c_n} \{d_{nc}\}) & \text{if } \exists c \neq c_n : d_{nc} < 1 \\ \delta(c, c_n) & \text{if } d_{nc_n} > 1 \end{cases} \\
&= \delta(c, \hat{c}_n).
\end{aligned}$$

Note that now we have pointwise convergence almost surely (i.e., everywhere except for points on the decision boundary $d_{nc_n} = 1$ where the loss function is not continuous). As before, $\hat{c}_n$ denotes the recognized class with margin defined in Eq. (4). In summary, we have $\mathcal{F}_\gamma^{(MPE)}(\Lambda) \overset{\gamma\to\infty}{\to} \frac{1}{2}\|\Lambda\|^2 + \frac{J}{N}\sum_{n=1}^N E[\hat{c}_n|c_n]$ which is identical to the SVM optimization problem using the loss function in Eq. (3).

3.3. Optimization

In general, the resulting optimization problems are no longer convex and thus, the optimization might get stuck in local optima. We believe that this problem is inherent in ASR, e.g. due to the time alignment from HMMs. Although it is possible to make the objective function convex by keeping the alignment fixed, the best results on large-scale tasks that are reported in the literature have been obtained by using non-convex objective functions. Finally, the problem of local optima is alleviated by combining the suggested approach with stochastic annealing techniques where the approximation level acts as the temperature.

In fact, the optimization strategy suggested in (Zhang et al., 2003) can be adopted, i.e., find the optimum for a given approximation level and carry out this step iteratively for increasingly finer levels. The optimization can be done with general optimization algorithms, e.g. RProp. The idea of incrementally regulated discriminative margins suggested by (Yu et al., 2007) is along the same lines.

In this work, the approximation level and the margin are chosen beforehand and then kept fixed during the complete optimization. This single step optimization scheme has the advantage that the loss function remains unchanged and that thus, the criterion differs only in the margin term. This approach is reasonable as long as the changes in the initial model are small, e.g. if the discriminative training is initialized with a good ML baseline. This is the typical situation in ASR. Further details and specifics of ASR are discussed in the next section.

4. Automatic Speech Recognition (ASR)

The smooth variants of SVMs introduced in Sec. 3.1 and 3.2 can directly be incorporated into the ASR framework. In this case, the HMM state sequences s_1^T correspond to the classes c. Similar to (Taskar et al., 2003) and (Sha & Saul, 2007), we would like the margin to scale with the length of the speech segments (cf. discussion in Sec. 2). In ASR, a reasonable choice is to set the margin of a sentence to the number of correct phones. More precisely, the simple accuracy $\delta(c, c_n)$ used to represent the margin so far is replaced with the phone accuracy. These approximations directly combine learning theory, HCRFs, and risk-based training of HMMs. Note that Gaussian HMMs (GHMMs) are HCRFs (possibly) with parameter constraints (Heigold et al., 2007).

Typically, MPE is used in combination with the more refined Gaussian regularization centered around Λ_0' (e.g. the maximum likelihood estimate of the generative model), which is comparable with the i-smoothing for GHMMs (Povey & Woodland, 2002). This regularization is combined with the L_2-norm regularization from the SVM

$$J_0^{-1}\|\Lambda\|^2 + J_1^{-1}\|\Lambda - \Lambda_0'\|^2 = J^{-1}\|\Lambda - \Lambda_0\|^2 + \text{const}(\Lambda)$$

with $J^{-1} = J_0^{-1} + J_1^{-1}$ and $\Lambda_0 = \frac{1}{1+\frac{J_1}{J_0}}\Lambda_0'$. Thus, the Gaussian regularization with a properly scaled center Λ_0 (scaling does not change the classification in the maximum approximation) covers the weaker L_2-norm regularization.

Similar to (Heigold et al., 2007), we use n-th order features, e.g. first order features are defined to be $f_{tsd}^{(1st)}(x_1^T, s_1^T) = \delta(s, s_t)x_{td}$. Zeroth and higher order features are defined in a similar fashion. This choice of feature function has the advantage that HCRFs and GHMMs are directly related.

The relationship between SVMs and common training criteria like MMI and MPE allows us to justify some important heuristics typically employed in discriminatively trained ASR systems to achieve good performance: the approximation level γ corresponds to the scaling of the probabilities, i-smoothing is the (refined) regularization term, and the weak unigram language model might be considered an approximation of the margin concept as explained in Sec. 3.1 ("weak prior"). We believe that the frame-based approach proposed to improve the generalization ability is also an attempt to approximate the margin by replacing the context priors (Heigold et al., 2007) by the global relative frequencies.

To apply the existing efficient algorithms, it is important that the margins of the different competing hypotheses can be represented as a weighted transducer sharing the topology with the common lattices, and thus can be integrated into most state-of-the-art systems. This is not always possible in an efficient way for the exact accuracy. Therefore,

approximate accuracies are used. For MPE, an intuitive margin is the approximate phone accuracy (Povey & Woodland, 2002), which is basically the same quantity also used for the loss function[2]. In this case, no additional quantities have to be calculated. The combined acoustic and language model scores are then augmented with these margins by composition. The subsequent steps of the accumulation and estimation remain unchanged. Thus, it is not necessary to modify our transducer-based implementation of the (discriminative) training because the margin can be incorporated by simply configuring an additional composition. The transducer-based implementation also has the advantage that the quantities used for the MMI and MPE accumulation can be represented in terms of *generalized* FB probabilities calculated in *different* semirings. This approach results in the same recursion formulae as used in (Povey & Woodland, 2002), but leads to a unified implementation of the different training criteria. The details on this issue are worked out in the next section.

5. Covariance & Expectation Semiring

In this section, we present an abstraction and generalization of the recursion formulae used for MMI and MPE (Povey & Woodland, 2002). The efficient calculation of the gradient of the objective function is an issue in ASR (and for HCRFs as well) because of the combinatorial number of possible word sequences. The proposed approach unifies these two recursion formulae and extends the speech-specific recursion formula for MPE to HCRFs. As mentioned above, this abstraction is not essential for this work. However, this formalism might be a nice feature of any (probabilistic) transducer library. As an example, it might facilitate the development of more refined training algorithms, e.g. it provides an efficient solution to the unified criterion in (He et al., 2008). The calculation of the gradient under consideration (as probably several other problems in pattern recognition) can be reduced to the calculation of the covariance of two suitably defined random variables, as discussed at the end of this section.

The expectation of the random variable $\mathcal{X}$ w.r.t. the probabilistic transducer $\mathcal{P}$ is defined to be

$$E_{\mathcal{P}}[\mathcal{X}] := \sum_{\pi \in \mathcal{P}} w_{\mathcal{P}}[\pi] w_{\mathcal{X}}[\pi]$$

where $w_{\cdot}[\pi]$ denotes the weight of path π in the respective transducer. The covariance of two (additive) random variables $\mathcal{X}$ and $\mathcal{Y}$ w.r.t. $\mathcal{P}$ is defined to be (with $E_{\mathcal{P}}[\cdot] \equiv E[\cdot]$)

$$\mathrm{Cov}_{\mathcal{P}}(\mathcal{X}, \mathcal{Y}) := \sum_{\pi \in \mathcal{Y}} w_{\mathcal{P}}[\pi] \left(w_{\mathcal{X}}[\pi] - E[\mathcal{X}]\right) \left(w_{\mathcal{Y}}[\pi] - E[\mathcal{Y}]\right).$$

[2] Assume the distance $E[w_1^N, v_1^M]$ between strings w_1^N and v_1^M. Then, the accuracy of string v_1^M given string w_1^N is $A[v_1^M | w_1^N] = N - E[w_1^N, v_1^M]$.

Here, we assume that $\mathcal{P}$, $\mathcal{X}$, and $\mathcal{Y}$ can be represented by acyclic transducers which share the topology, i.e., differ only in the weights. Using these assumptions, we shall show that the covariance can be efficiently calculated by simply exchanging the probability semiring by the expectation semiring in the standard FB algorithm. So, the probability semiring can be used to compute the first order statistics whereas the expectation semiring can be used to compute the second order statistics. It is rather straightforward to define a covariance semiring to calculate third order statistics etc.

We start with introducing the expectation semiring and the abstract definitions which are needed to formulate the propositions.

Expectation semiring. The *expectation semiring* (Eisner, 2001) is a multiplex semiring with weights $(p, v) \in \mathbb{R}^+ \times \mathbb{R}$, and

- $(p_1, v_1) \oplus (p_2, v_2) = (p_1 + p_2, v_1 + v_2)$;
- $(p_1, v_1) \otimes (p_2, v_2) = (p_1 p_2, p_1 v_2 + v_1 p_2)$;
- $\overline{1} = (1, 0)$, $\overline{0} = (0, 0)$.

In addition, the inverse is defined to be $\mathsf{inv}(p, v) = (p^{-1}, -p^{-2}v)$. Observe that the first component corresponds to the probability semiring whereas the second component accounts for the additivity of the random variable. The (partial) path weight of path π is the "product" of the corresponding arc weights $w_{\mathcal{P}}[a]$, $w_{\mathcal{P}}[\pi] = \bigotimes_{a \in \pi} w_{\mathcal{P}}[a]$.

FB potentials. The *forward potential* α_q at the state q of the transducer $\mathcal{P}$ is the sum of the weights of all partial paths π going from the initial state *init* to the state q

$$\alpha_q := \bigoplus_{\pi = (init, q) \in \mathcal{P}} w_{\mathcal{P}}[\pi].$$

These quantities are efficiently calculated by recursion

$$\alpha_{init} = \overline{1} \qquad \alpha_q = \bigoplus_{a = (p, q) \in \mathcal{P}} \alpha_p \otimes w_{\mathcal{P}}[a].$$

The "sum" is over all arcs a of the transducer $\mathcal{P}$ connecting the state p with q. The backward potentials β_q are defined similarly on the transposed $\mathcal{P}$.

Posteriors. The *posterior* transducer $\mathcal{Q}(\mathcal{P})$ associated with the transducer $\mathcal{P}$ has the arc weights

$$w_{\mathcal{Q}(\mathcal{P})}[a] := \left(\bigoplus_{\pi \in \mathcal{P} : a \in \pi} w_{\mathcal{P}}[\pi] \right) \otimes \mathsf{inv}\left(\bigoplus_{\pi \in \mathcal{P}} w_{\mathcal{P}}[\pi] \right).$$

The weight of arc $a = (p, q)$ can be expressed in terms of the above defined forward and backward potentials

$$w_{\mathcal{Q}(\mathcal{P})}[a] = \left(\alpha_p \otimes w_{\mathcal{P}}[a] \otimes \beta_q \right) \otimes \mathsf{inv}(\beta_{init}).$$

Here, we used the fact that β_{init} equals the "normalization constant" in the case of a unique initial state *init*. To make the analogy of the calculation of the expectation and the covariance more clear, we first state the well-known proposition based on the probability semiring.

Proposition 1. *Assume an acyclic transducer $\mathcal{P}$ with probability semiring, and a weighted transducer $\mathcal{X}$ with log semiring. $\mathcal{P}$ and $\mathcal{X}$ share the topology. Then,*

$$E_{\mathcal{P}}[\mathcal{X}] = \sum_{a \in \mathcal{P}} w_{\mathcal{X}}[a] w_{Q(\mathcal{P})}[a].$$

This proposition is then extended to the expectation semiring. Note that for the p-component, we recover the previous proposition.

Proposition 2. *Assume an acyclic transducer $\mathcal{P}$ with probability semiring, and transducers $\mathcal{X}$ and $\mathcal{Y}$ with log semiring. $\mathcal{P}$, $\mathcal{X}$, $\mathcal{Y}$ share the topology. Define the transducer $\mathcal{Z}$ with* expectation semiring *and assign the weights $w_{\mathcal{Z}}[a] = (w_{\mathcal{P}}[a], w_{\mathcal{P}}[a] w_{\mathcal{X}}[a])$ to the arcs. Then,*

$$Cov_{\mathcal{P}}(\mathcal{X}, \mathcal{Y}) = \sum_{a \in \mathcal{Y}} w_{\mathcal{Y}}[a] w_{Q(\mathcal{Z})}[a][v].$$

We conclude this section by showing how the calculation of the gradient of the objective function fits into this framework.

Gradient of objective function. To simplify the discussion, we restrict our consideration to objective functions of the type $\mathcal{F}(\Lambda) = f(E_{\mathcal{P}}[\mathcal{A}])$ rather than using the unified objective function in (He et al., 2008). Here, $\mathcal{P}$ stands for the word lattice with the joint probabilities $p_\Lambda(s_1^T, v_1^M | x_1^T)$ and $\mathcal{A}$ denotes some additive risk (e.g. phone error). In addition, a non-linearity f can be applied to the expectation. Then, building the derivative of this objective function leads to $\nabla \mathcal{F}(\Lambda) = \mathrm{Cov}_{\mathcal{P}}(\mathcal{L}, \nabla \log \mathcal{P})$ with $\mathcal{L} := f'(E_{\mathcal{P}}[\mathcal{A}])\mathcal{A}$. Examples: $\mathcal{A}$ = phone accuracy, $f(x) = x$ (MPE); $\mathcal{A} = \chi_{\mathrm{spk}}$ (characteristic function of spoken sequence, i.e., one for the spoken sequence and zero otherwise), $f(x) = \log x$ (MMI); or $\mathcal{A} = \chi_{\mathrm{spk}}$, $f(x)$ =sigmoid function (MCE).

6. Experimental Results

The presented approaches were evaluated on two different tasks. First, we tested the proposed criterion on the German digit string recognition task Sietill (Heigold et al., 2007), which due to its small size allows for a thorough experimental evaluation. Second, experiments were carried out on the large vocabulary EPPS English task, which represents a realistic ASR task. The baseline MPE result was part of our 2007 TC-STAR evaluation system, which performed best in the restricted and public evaluation conditions for both English and Spanish (Lööf et al., 2007). For completeness, we provide some description of the speech recognition systems. Non-experts, however, can skip these technical parts, keeping in mind that highly competitive systems are used for the discriminative training.

Table 2. Corpus statistics.

Task	Corpus	Data [h]	#run. words [k]	#frames [k]
Sietill	Train	5.5	43	1,980
	Test	5.5	43	1,980
EPPS En	Train	92.0	661	33,120
	Dev06	3.2	27	1,152
	Eval06	3.2	30	1,152
	Eval07	2.9	27	1,044

Our modified MMI criterion is identical with the recently proposed boosted MMI (Povey et al., 2008). These results, however, should be interpreted with some care because in most experiments, the boosting factor is not the only change. Probably, there is a single experiment which is directly comparable with our results on the EPPS task, i.e., which modifies only the boosting factor and which is set upon a state-of-the-art baseline. Very much like our results on the EPPS task, this result supports the hypothesis that the effect of the margin on such systems is marginal.

6.1. Sietill

The recognition system is based on gender-dependent whole-word HMMs. For each gender, 214 distinct states plus one for silence are used. The vocabulary consists of the 11 German digits (including the pronunciation variant 'zwo'). The observation vectors consist of 12 cepstral features without derivatives. The gender-independent Linear Discriminant Analysis (LDA) is applied to 5 consecutive frames and projects the resulting feature vector to 25 dimensions (Heigold et al., 2007). The corpus statistics is summarized in Tab. 2. The ML baseline system uses Gaussian mixtures with globally pooled variances and serves as initialization of the log-linear HMMs. The margin is represented by the approximate word accuracy and has been chosen to be the point where the word error rate (WER) on the training corpus begins to increase rapidly. The final performance turned out to be rather insensitive to the exact value. The optimization was carried out using RProp. Fig. 2 shows the progress of the word error rate (WER) vs. the iteration index on the test corpus. Margin-based MMI was validated on log-linear mixture models of different complexity (16 and 64 densities per HMM state with first order features only) and on a purely log-linear model with second and third order features (instead of using only first order features). The discriminative training was initialized with the respective ML baseline model except for the experiments including third order features. These were initialized with the model from frame-based training (Heigold et al., 2007). The discriminative results were all obtained

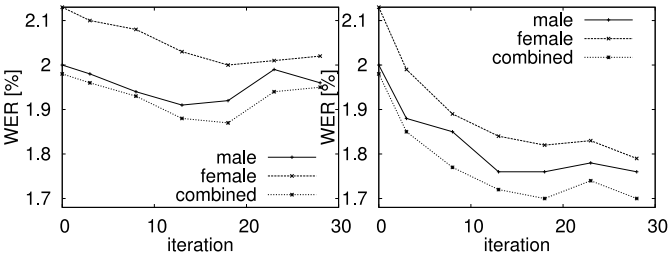


Figure 2. Effect of margin: progress of word error rate (WER) on Sietill test corpus, MMI (left) vs. modified MMI (right) (16 densities/mixture).

Table 3. Word error rates (WER) for Sietill test corpus.

Dns/Mix	Criterion	Margin	WER [%]
16	ML	-	1.98
	MMI	-	1.88
		word	1.72
64	ML	-	1.81
	MMI	-	1.77
		word	1.59
1+f2+3	Frame	word	1.75
	MMI	-	1.68
		word	1.53

using a regularization term. Tab. 3 summarizes the results. The results clearly benefit from the additional margin term, both regarding the performance and the robustness. This might be because the training data are separable for the given configurations. For the experiments using second and third order features ('1+f2+3') the training was initialized with the models from frame-based MMI training which benefits from the margin only slightly (cf. Sec. 4).

6.2. EPPS English

This task contains recordings from the European Parliament Plenary Sessions (EPPS). The corpus statistics of the different EPPS corpora can be found in Tab. 2. The acoustic front end comprises MFCC features augmented by a voicing feature. 9 consecutive frames are concatenated and the resulting vector is projected to 45 dimensions by means of LDA. The MFCC features are warped using a fast variant of the Vocal Tract Length Normalization (VTLN). On top of this, Speaker Adaptive Training (SAT) is applied. The triphones are clustered using CART, resulting in 4,501 generalized triphone states. The HMM states are modeled by Gaussian mixtures with globally pooled variances. The ML baseline system is made up of approximately 900,000 densities. For recognition, a lexicon with 50,000 entries in combination with a 4-gram language model was used (Lööf et al., 2007). The development (Dev06) and evaluation (Eval06) data from the evaluation campaign 2006 as described in Tab. 2 were used to tune the different parameters (e.g. language model scale or the number of MPE iterations). The evaluation data from the evaluation campaign 2007 (Eval07) were used only for testing.

Table 4. Word error rates (WER) for EPPS English corpus, MPE with different margins.

LM (train)	Margin	WER [%] Dev06	Eval06	Eval07
1g	-	13.4	10.1	11.5
	word	13.4	10.2	11.3
	phone	13.3	10.2	11.3
2g	-	13.3	10.3	11.6
	word	13.2	10.2	11.3
	phone	13.2	10.2	11.3

Table 5. Word error rates (WER) for EPPS English corpus, interdependence of weak language model and phone margin.

Crit.	Margin	LM (train)	WER [%] Dev06	Eval06	Eval07
ML	-	-	14.4	10.8	12.0
MPE	no	1g	13.4	10.1	11.5
		2g	13.3	10.3	11.6
	yes	1g	13.3	10.2	11.3
		2g	13.2	10.2	11.3

The word-conditioned lattices used in MPE training were generated with the VTLN/voicedness system in combination with a bigram language model. Since the lattices are dominated by silence and noise arcs, the lattices were filtered. The idea behind this filtering is to correct the posteriors for accumulation of discriminative statistics. For the acoustic rescoring during discriminative training, the exact match approach is used, i.e., the word boundary times are kept fixed.

The margins are tuned on a small fraction of the training corpus such that the margin-based approach in combination with a bigram language model and the standard MPE setup with a unigram language model have the same WER. Independent control experiments imply that no further tuning of the margin parameter is required. In the first experiment we have tested the impact of different margins on the performance, more specifically we have tested the approximate word and phone accuracies according to (Povey & Woodland, 2002). Tab. 4 shows that the differences are marginal. For convenience we decided to use the approximate phone accuracy-based margin for the remaining experiments. In Tab. 5 the interdependence of the weak unigram language model and the margin was investigated. There is ongoing work to clarify the interdependence of the language model used for the optimization and the margin. Using the acoustic model from the standard MPE training, the same 4-gram language model and only each tenth segment, the relative improvement of WER is 5.6% on the training data. This probably indicates that the generalization performance on the test data (Eval07) is not optimal with a relative improvement of 4.2% (and does not appear to be an issue on the development data, i.e., Dev06 and Eval06). The experimental results show the expected tendency, see Tab. 1.

7. Conclusions

We proposed modified formulations of MMI and MPE to include a margin term into the discriminative training of models for ASR. Furthermore, we showed that these modified criteria can directly be used in existing state-of-the-art ASR frameworks, since they can be represented as an additional transducer composition. The modified criteria are directly related to SVMs using a suitable loss function, which allows us to justify some important heuristics used in the discriminative training of acoustic models. The experimental results are consistent with our expectations. For the German digit string recognition task Sietill, where overfitting is achieved after a few iterations, the margin is essential for the robust estimation of the model parameters and allows to achieve significant improvements over the ML baseline. In contrast, on the large vocabulary EPPS English task the observed improvements are transferred well to the test data and the effect under consideration is marginal. So far, we have investigated the effect of the margin for the discriminative re-estimation based on generatively estimated and strongly tuned acoustic models. The benefits due to the margin might be better visible, when the discriminative, margin-based training builds on top of a suboptimal ML baseline. However, models building on top of better baseline models might still have a better absolute performance.

Acknowledgments

This material is partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001-06-C-0023, and was partly funded by the European Union under the integrated project TC-STAR (FP6-506738). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA.

References

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden markov support vector machines. *Int. Conf. on Machine Learning (ICML)*.

Eisner, J. (2001). Expectation semirings: Flexible EM for finite-state transducers. *Finite-State Methods and Natural Language Processing (FSMNLP)*. Helsinki, Finland.

He, X., Deng, L., & Chou, W. (2008). Discriminative learning in sequential pattern recognition – A unifying review for optimization-oriented speech recognition. *IEEE Signal Processing Magazine*.

Heigold, G., Schlüter, R., & Ney, H. (2007). On the equivalence of Gaussian HMM and Gaussian HMM-like hidden conditional random fields. *Proc. of the Int. Conf. on Spoken Language Processing (Interspeech)*. Antwerp, Belgium.

Jebara, T. (2002). *Discriminative, generative, and imitative learning*. Doctoral dissertation, Massachusetts Institute of Technology.

Li, J., Yan, Z., Lee, C., & Wang, R. (2007). A study on soft margin estimation for LVCSR. *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. Kyoto, Japan.

Lööf, J., Gollan, C., Hahn, S., Heigold, G., Hoffmeister, B., Plahl, C., Rybach, D., Schlüter, R., & Ney, H. (2007). The RWTH 2007 TC-STAR evaluation system for European English and Spanish. *Proc. of the Int. Conf. on Spoken Language Processing (Interspeech)*. Antwerp, Belgium.

Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B., Saon, G., & Visweswariah, K. (2008). Boosted MMI for model and feature-space discriminative training. *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*. Las Vegas, NV.

Povey, D., & Woodland, P. C. (2002). Minimum phone error and I-smoothing for improved discriminative training. *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*. Orlando, FL.

Sha, F., & Saul, L. (2007). Comparison of large margin training to other discriminative methods for phonetic recognition by hidden Markov models. *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*. Honolulu, Hawaii.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. *Neural Information Processing Systems Conference (NIPS)*.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer.

Weston, J., & Watkins, C. (1999). Support vector machines for multi-class pattern classification. *Proc. of the Seventh European Symposium on Artifi cial Neural Networks*.

Yin, Y., & Jiang, H. (2007). A fast optimization method for large margin estimation of HMMs based on second order cone programming. *Proc. of the Int. Conf. on Spoken Language Processing (Interspeech)*. Antwerp, Belgium.

Yu, D., Deng, L., He, X., & Acero, A. (2007). Large-margin minimum classification error training for large-scale speech recognition tasks. *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*. Honolulu, Hawaii, USA.

Zhang, J., Jin, R., Yang, Y., & Hauptmann, A. (2003). Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization. *Int. Conference on Machine Learning (ICML)*.

Statistical Models for Partial Membership

Katherine A. Heller HELLER@GATSBY.UCL.AC.UK
Sinead Williamson SAW56@CAM.AC.UK
Zoubin Ghahramani ZOUBIN@ENG.CAM.AC.UK
Engineering Department, University of Cambridge, Cambridge, UK

Abstract

We present a principled Bayesian framework for modeling *partial memberships* of data points to clusters. Unlike a standard mixture model which assumes that each data point belongs to one and only one mixture component, or cluster, a partial membership model allows data points to have fractional membership in multiple clusters. Algorithms which assign data points partial memberships to clusters can be useful for tasks such as clustering genes based on microarray data (Gasch & Eisen, 2002). Our Bayesian Partial Membership Model (BPM) uses exponential family distributions to model each cluster, and a product of these distibtutions, with weighted parameters, to model each datapoint. Here the weights correspond to the degree to which the datapoint belongs to each cluster. All parameters in the BPM are continuous, so we can use Hybrid Monte Carlo to perform inference and learning. We discuss relationships between the BPM and Latent Dirichlet Allocation, Mixed Membership models, Exponential Family PCA, and fuzzy clustering. Lastly, we show some experimental results and discuss nonparametric extensions to our model.

1. Introduction

The idea of *partial membership* is quite intuitive and practically useful. Consider, for example, an individual with a mixed ethnic background, say, partly Asian and partly European. It seems sensible to represent that individual as partly belonging to two different classes or sets. Such a partial membership representation may be relevant to predicting that individual's phenotype, or their food preferences. We clearly need models that can coherently represent partial membership.

Note that partial membership is conceptually very different from uncertain membership. Being certain that a person is partly Asian and partly European, is very different than being uncertain about a person's ethnic background. More information about the person, such as DNA tests, could resolve uncertainty, but cannot make the person change his ethnic membership.

Partial membership is also the cornerstone of fuzzy set theory. While in traditional set theory, items either belong to a set or they don't, fuzzy set theory equips sets with a membership function $\mu_k(x)$ where $0 \leq \mu_k(x) \leq 1$ denotes the degree to which x partially belongs to set k.

In this paper we describe a fully probabilistic approach to data modelling with partial membership. Our approach makes use of a simple way of representing partial membership using continuous latent variables. We define a model which can cluster data but which fundamentally assumes that data points can have partial membership in the clusters. Each cluster is represented by an exponential family distribution with conjugate priors (reviewed in section 3). Our model can be seen as a continuous latent variable relaxation of clustering with finite mixture models, and reduces to mixture modelling under certain settings of the hyperparameters. Unlike Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and Mixed Membership models (Erosheva et al., 2004), which also capture partial membership in the form of attribute-specific mixtures, our model does not assume a factorization over attributes and provides a general way of combining exponential family distributions with partial membership. The complete specification of our model is provided in section 4. Learning and inference are carried out using Markov chain Monte Carlo (MCMC) methods. We show in particular that because all the parameters in

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

our model are continuous, it is possible to employ a full hybrid Monte Carlo (HMC) algorithm, which uses gradients of the log probability, for inference (section 5).

Our Bayesian Partial Membership (BPM) model bears interesting relationships to several well-known models in machine learning and statistics, including LDA (Blei et al., 2003), mixed membership models (Erosheva et al., 2004), exponential family PCA (Collins et al., 2002), and Discrete Components Analysis (Buntine & Jakulin, 2006). We discuss these relations in section 6, where we also relate our model to fuzzy k-means. In section 7, we present both synthetic and real-world experimental results using image data and voting patterns of US senators. We conclude with future work in section 8.

2. A Partial Membership Model

We can derive our method for modeling partial memberships from a standard finite mixture model. In a finite mixture model the probability of a data point, $\boldsymbol{x}_n$ given Θ, which contains the parameters for each of the K mixture components (clusters) is:

$$p(\boldsymbol{x}_n|\Theta) = \sum_{k=1}^{K} \rho_k p_k(\boldsymbol{x}_n|\boldsymbol{\theta}_k) \tag{1}$$

where p_k is the probability distribution of mixture component k, and ρ_k is the mixing proportion (fraction of data points belonging to) for component k[1].

Equation 1 can be rewritten using indicator variables $\boldsymbol{\pi}_n = [\pi_{n1}\pi_{n2}\ldots\pi_{nK}]$ as follows:

$$p(\boldsymbol{x}_n|\Theta) = \sum_{\boldsymbol{\pi}_n} p(\boldsymbol{\pi}_n) \prod_{k=1}^{K} p_k(\boldsymbol{x}_n|\boldsymbol{\theta}_k)^{\pi_{nk}} \tag{2}$$

where $\pi_{nk} \in \{0,1\}$ and $\sum_k \pi_{nk} = 1$. Here we can notice that if $\pi_{nk} = 1$ this means that data point n belongs to cluster k (also $p(\pi_{nk} = 1) = \rho_k$). Therefore the π_{nk} denote *memberships* of data points to clusters.

In order to obtain a model for *partial memberships* we can relax the constraint $\pi_{nk} \in \{0,1\}$ to now allow π_{nk} to take any continuous value in the range $[0,1]$. However, in order to compute the probability of the data under this continuous relaxation of a finite mixture model, we need to modify equation 2 as follows:

$$p(\boldsymbol{x}_n|\Theta) = \int_{\boldsymbol{\pi}_n} p(\boldsymbol{\pi}_n) \frac{1}{c} \prod_{k=1}^{K} p_k(\boldsymbol{x}_n|\boldsymbol{\theta}_k)^{\pi_{nk}} d\boldsymbol{\pi}_n \tag{3}$$

[1]This notation differs slightly from standard notation for mixture models.

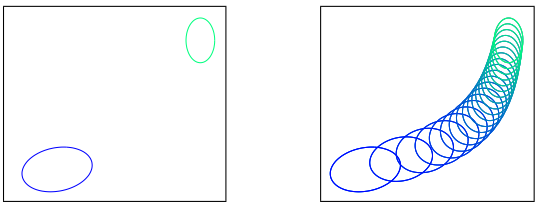


Figure 1. Left: A mixture model with two Gaussian mixture components, or clusters, can generate data from the two distributions shown. Right: Partial membership model with the same two clusters can generate data from all the distributions shown (there are actually infinitely many), which lie between the two original clusters.

The modifications include integrating over all values of $\boldsymbol{\pi}_n$ instead of summing, and since the product over clusters K from equation 2 no longer normalizes we put in a normalizing constant c, which is a function of $\boldsymbol{\pi}_n$ and Θ. Equation 3 now gives us a model for partial membership.

We illustrate the difference between our partial membership model and a standard mixture model in figure 1. Here we can see contours of the Gaussian distributions which can generate data in the mixture model (left) and the partial membership model (right), where both models are using the same two Gaussian clusters. As an example, if one of these clusters represents the ethnicity "White British" and the other cluster represents the ethnicity "Pakistani", then the figure illustrates that the partial membership model will be able to capture someone of mixed ethnicity, whose features may lie in between those of either ethnic group (for example skin color or nose size), better than the mixture model.

3. Conjugate-Exponential Models

In the previous section we derived a partial membership model, given by equation 3. However we have not yet discussed the form of the distribution for each cluster, $p_k(\boldsymbol{x}_n|\boldsymbol{\theta}_k)$, and we will now focus on the case when these distributions are in the exponential family.

An *exponential family distribution* can be written in the form:

$$p_k(\boldsymbol{x}_n|\boldsymbol{\theta}_k) = \exp\{\boldsymbol{s}(\boldsymbol{x}_n)^\top \boldsymbol{\theta}_k + h(\boldsymbol{x}_n) + g(\boldsymbol{\theta}_k)\} \tag{4}$$

where $\boldsymbol{s}(\boldsymbol{x}_n)$ is a vector depending on the data known as the *sufficient statistics*, $\boldsymbol{\theta}_k$ is a vector of *natural parameters*, $h(\boldsymbol{x}_n)$ is a function of the data, and $g(\boldsymbol{\theta}_k)$ is a function of the parameters which ensures that the probability normalizes to one when integrating or summing over $\boldsymbol{x}_n$. We will use the short-hand $\boldsymbol{x}_n \sim \text{Expon}(\boldsymbol{\theta}_k)$ to denote that $\boldsymbol{x}_n$ is drawn from an

exponential family distribution with natural parameters $\boldsymbol{\theta}_k$.

If we plug the exponential family distribution (equation 4) into our partial membership model (equation 3) it follows that:

$$\boldsymbol{x}_n|\boldsymbol{\pi}_n, \Theta \sim \text{Expon}(\sum_k \pi_{nk}\boldsymbol{\theta}_k) \tag{5}$$

where $\boldsymbol{x}_n$ comes from the *same* exponential family distribution as the original clusters p_k, but with *new* natural parameters which are a convex combination of the natural parameters of the original clusters, $\boldsymbol{\theta}_k$, weighted by π_{nk}, the partial membership values for data point $\boldsymbol{x}_n$. Computation of the normalizing constant c is therefore always tractable when p_k is in the exponential family.

A probability distribution $p(\boldsymbol{\theta}_k)$ is said to be *conjugate* to the exponential family distribution $p(\boldsymbol{x}_n|\boldsymbol{\theta}_k)$ if $p(\boldsymbol{\theta}_k|\boldsymbol{x}_n)$ has the same functional form as $p(\boldsymbol{\theta}_k)$. In particular, the conjugate prior to the above exponential family distribution can be written in the form:

$$p(\boldsymbol{\theta}) \propto \exp\{\boldsymbol{\lambda}^\top\boldsymbol{\theta} + \nu g(\boldsymbol{\theta})\} \tag{6}$$

where $\boldsymbol{\lambda}$ and ν are *hyperparameters* of the prior. We will use the short-hand, $\boldsymbol{\theta} \sim \text{Conj}(\boldsymbol{\lambda}, \nu)$. We now have the tools to define our Bayesian partial membership model.

4. Bayesian Partial Membership Models

Consider a model with K clusters, and a data set $\mathcal{D} = \{\boldsymbol{x}_n : n = 1 \dots N\}$. Let $\boldsymbol{\alpha}$ be a K-dimensional vector of positive hyperparameters. We start by drawing mixture weights from a Dirichlet distribution:

$$\boldsymbol{\rho} \sim \text{Dir}(\boldsymbol{\alpha}) \tag{7}$$

Here $\boldsymbol{\rho} \sim \text{Dir}(\boldsymbol{\alpha})$ is shorthand for $p(\boldsymbol{\rho}|\boldsymbol{\alpha}) = c\prod_{k=1}^K \rho_k^{\alpha_k - 1}$ where $c = \Gamma(\sum_k \alpha_k)/\prod_k \Gamma(\alpha_k)$ is a normalization constant which can be expressed in terms of the Gamma function[2]. For each data point, n, we draw a partial membership vector $\boldsymbol{\pi}_n$ which represents how much that data point belongs to each of the K clusters:

$$\boldsymbol{\pi}_n \sim \text{Dir}(a\boldsymbol{\rho}). \tag{8}$$

The parameter a is a positive scaling constant drawn, for example, from an exponential distribution $p(a) = be^{-ba}$, where $b > 0$ is a constant. We assume that each cluster k is characterized by an exponential family distribution with natural parameters $\boldsymbol{\theta}_k$ and that

$$\boldsymbol{\theta}_k \sim \text{Conj}(\boldsymbol{\lambda}, \nu). \tag{9}$$

Given all these latent variables, each data point is drawn from

$$\boldsymbol{x}_n \sim \text{Expon}(\sum_k \pi_{nk}\boldsymbol{\theta}_k) \tag{10}$$

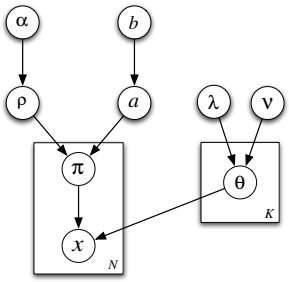


Figure 2. Graphical model for the BPM

In order to get an intuition for what the functions of the parameters we have just defined are, we return to the ethnicity example. Here, each cluster k is an ethnicity (for example, "White British" and "Pakistani") and the parameters $\boldsymbol{\theta}_k$ define a distribution over features for each of the k ethnic groups (for example, how likely it is that someone from that ethnic group likes pizza or marmite or bindi bhaji). The parameter $\boldsymbol{\rho}$ gives the ethnic composition of the population (for example, 75% "White British" and 25% "Pakistani"), while a controls how similar to the population an individual is expected to be (Are 100% of the people themselves 75% "White British" and 25% "Pakistani"? Or are 75% of the people 100% "White British" and the rest are 100% "Pakistani"? Or somewhere in between?). For each person n, $\boldsymbol{\pi}_n$ gives their individual ethnic composition, and finally $\boldsymbol{x}_n$ gives their individual feature values (e.g. how much they like marmite). The graphical model representing this generative process is drawn in Figure 2.

Since the Bayesian Partial Membership Model is a generative model, we tried generating data from it using full-covariance Gaussian clusters. Figure 3 shows the results of generating 3000 data points from our model with $K = 3$ clusters as the value of parameter a changes. We can see that as the value of a increases data points tend to have partial membership in more clusters. In fact we can prove the following lemmas:

Lemma 1 *In the limit that $a \to 0$ the exponential family BPM is a standard mixture model with K components and mixing proportions $\boldsymbol{\rho}$.*

[2]The Gamma function generalizes the factorial to positive reals: $\Gamma(x) = (x-1)\Gamma(x-1)$, $\Gamma(n) = (n-1)!$ for integer n

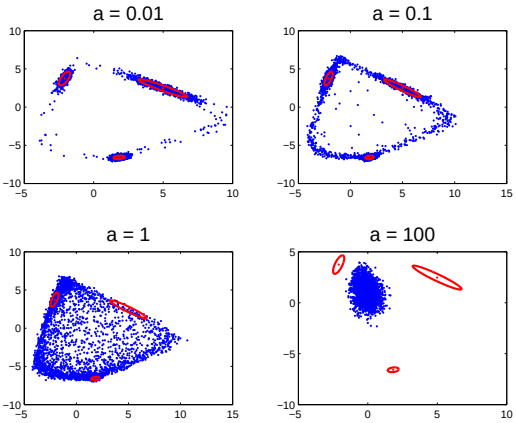


Figure 3. 3000 BPM generated data points with partial assignments to 3 Gaussian clusters shown in red, as parameter a varies.

Lemma 2 *In the limit that $a \to \infty$ the exponential family BPM model has a single component with natural parameters $\sum_k \rho_k \boldsymbol{\theta}_k$.*

Proofs of these lemmas follow simply from taking the limits of equation 8 as a goes to 0 and ∞ respectively.

5. BPM Learning

We can represent the observed data set $\mathcal{D}$ as an $N \times D$ matrix $\mathbf{X}$ with rows corresponding to $\boldsymbol{x}_n$, where D is the number of input features.[3] Let $\boldsymbol{\Theta}$ be a $K \times D$ matrix with rows $\boldsymbol{\theta}_k$ and $\boldsymbol{\Pi}$ be an $N \times K$ matrix with rows $\boldsymbol{\pi}_n$. Learning in the BPM consists of inferring all unknown variables, $\boldsymbol{\Omega} = \{\boldsymbol{\Pi}, \boldsymbol{\Theta}, \boldsymbol{\rho}, a\}$ given $\mathbf{X}$. We treat the top level variables in the graphical model in Figure 2, $\boldsymbol{\Psi} = \{\boldsymbol{\alpha}, \boldsymbol{\lambda}, \nu, b\}$ as fixed hyperparameters, although these could also be learned from data. Our goal is to infer $p(\boldsymbol{\Omega}|\mathbf{X}, \boldsymbol{\Psi})$, for which we decide to employ Markov chain Monte Carlo (MCMC).

Our key observation for MCMC is that even though BPMs contain discrete mixture models as a special case, *all* of the unknown variables $\boldsymbol{\Omega}$ of the BPM are continuous. Moreover, it is possible to take derivatives of the log of the joint probability of all variables with respect to $\boldsymbol{\Omega}$. This makes it possible to do inference using a full Hybrid Monte Carlo (HMC) algorithm on all parameters. Hybrid (or Hamiltonian) Monte Carlo is an MCMC procedure which overcomes the random walk behaviour of more traditional Metropolis or Gibbs sampling algorithms by making use of the derivatives of the log probability (Neal, 1993; MacKay, 2003). In high dimensions, this derivative information can lead to a dramatically faster mixing of the Markov chain, analogous to how optimization using derivatives is often much faster than using greedy random search.

We start by writing the probability of all parameters and variables[4] in our model:

$$p(\mathbf{X}, \boldsymbol{\Omega}|\boldsymbol{\Psi}) = p(\mathbf{X}|\boldsymbol{\Pi}, \boldsymbol{\Theta}) p(\boldsymbol{\Theta}|\boldsymbol{\lambda}, \nu) p(\boldsymbol{\Pi}|a, \boldsymbol{\rho}) p(a|b) p(\boldsymbol{\rho}|\boldsymbol{\alpha}) \tag{11}$$

We assume that the hyperparameter $\nu = 1$, and omit it from our derivation. Since the forms of all distributions on the right side of equation (11) are given in section 4, we can simply plug these in and see that:

$$\begin{aligned}
&\log p(\mathbf{X}, \boldsymbol{\Omega}|\boldsymbol{\Psi}) = \\
&\quad \log\Gamma\left(\textstyle\sum_k \alpha_k\right) - \textstyle\sum_k \log\Gamma(\alpha_k) + \textstyle\sum_k (\alpha_k - 1)\log\rho_k \\
&\quad + \log b - ba + N \log\Gamma\left(\textstyle\sum_k a\rho_k\right) - N \textstyle\sum_k \log\Gamma(a\rho_k) \\
&\quad + \textstyle\sum_n \sum_k (a\rho_k - 1)\log\pi_{nk} + \textstyle\sum_k \left[\boldsymbol{\theta}_k^\top \boldsymbol{\lambda} + \mathrm{g}(\boldsymbol{\theta}_k) + \mathrm{f}(\boldsymbol{\lambda})\right] \\
&\quad + \textstyle\sum_n \left[\left(\sum_k \pi_{nk}\boldsymbol{\theta}_k\right)^\top \boldsymbol{x}_n + \mathrm{h}(\boldsymbol{x}_n) + \mathrm{g}\left(\sum_k \pi_{nk}\boldsymbol{\theta}_k\right)\right]
\end{aligned}$$

The Hybrid Monte Carlo algorithm simulates dynamics of a system with continuous state $\boldsymbol{\Omega}$ on an energy function $\mathcal{E}(\boldsymbol{\Omega}) = -\log p(\mathbf{X}, \boldsymbol{\Omega}|\boldsymbol{\Psi})$. The derivatives of the energy function $\frac{\partial \mathcal{E}(\boldsymbol{\Omega})}{\partial \boldsymbol{\Omega})}$ provide forces on the state variables which encourage the system to find high probability regions, while maintaining detailed balance to ensure that the correct equilibrium distribution over states is achieved (Neal, 1993). Since $\boldsymbol{\Omega}$ has constraints, e.g. $a > 0$ and $\sum_k \rho_k = 1$, we use a tranformation of variables so that the new state variables are unconstrained, and we perform dynamics in this unconstrained space. Specifically, we use $a = e^\eta$, $\rho_k = \frac{e^{r_k}}{\sum_{k'} e^{r_{k'}}}$, and $\pi_{nk} = \frac{e^{p_{nk}}}{\sum_{k'} e^{p_{nk'}}}$. For HMC to be valid in this new space, the chain rule needs to be applied to the derivatives of $\mathcal{E}$, and the prior needs to be transformed through the Jacobian of the change of variables. For example, $p(a)da = p(\eta)d\eta$ implies $p(\eta) = p(a)(da/d\eta) = ap(a)$. We also extended the HMC procedure to handle missing inputs in a principled manner, by analytically integrating them out, as this was required for some of our applications. More details and general pseudocode for HMC can be found in (MacKay, 2003).

6. Related Work

The BPM model has interesting relations to several models that have been proposed in machine learning, statistics and pattern recognition. We describe these relationships here.

[3] We assume that the data is represented in its natural representation for the exponential family likelihood, so that $\boldsymbol{s}(\boldsymbol{x}_n) = \boldsymbol{x}_n$.

[4] A formal distinction between hidden variables, e.g. the $\{\boldsymbol{\pi}_n\}$, and unknown parameters is not necessary as they are both unknowns.

Latent Dirichlet Allocation: Using the notation introduced above, the BPM model and LDA (Blei et al., 2003) both incorporate a K-dimensional Dirichlet distributed $\boldsymbol{\pi}$ variable. In LDA, $\boldsymbol{\pi}_n$ are the mixing proportions of the topic mixture for each document n. Each word in document n can then be seen as having been generated by topic k, with probability π_{nk}, where the word distribution for topic k is given by a multinomial distribution with some parameters, $\boldsymbol{\theta}_k$. The BPM also combines π_{nk} with some exponential family parameters $\boldsymbol{\theta}_k$, but here the way in which they are combined does not result in a mixture model from which another variable (e.g. a word) is assumed to be generated. In contrast, the data points are indexed by n directly, and therefore exist at the document level of LDA. Each data point is assumed to have come from an exponential family distribution parameterized by a weighted sum of natural parameters $\boldsymbol{\theta}$, where the weights are given by π_n for data point n. In LDA, data is organized at two levels (e.g. documents and words). More generally, mixed membership (MM) models (Erosheva et al., 2004), or admixture models, assume that each data attribute (e.g. words) of the data point (e.g. document) is drawn independently from a mixture distribution given the membership vector for the data point, $x_{nd} \sim \sum_k \pi_{nk} P(x|\theta_{kd})$. LDA and mixed membership models do not average natural parameters of exponential family distributions like the BPM. LDA or MM models could not generate the continuous densities in figure 3 from full-covariance Gaussians. The analagous generative process for MM models is given in figure 4. Since data attributes are drawn independently, the original clusters (not explicity shown) are one dimensional and have means at 0, 10 and 20 for both attribute dimensions. We can notice from the plot that this model always generates a mixture of 9 Gaussians, which is a very different behavior than the BPM, and clearly not as suitable for the general modeling of partial memberships. LDA only makes sense when the objects (e.g. documents) being modelled constitute bags of exchangeable sub-objects (e.g. words). Our model makes no such assumption. Moreover, in LDA and MM models there is a discrete latent variable for every sub-object corresponding to which mixture component that sub-object was drawn from. This large number of discrete latent variables makes MCMC sampling in LDA potentially much more expensive than in BPM models.

Exponential Family PCA: Our model bears an interesting relationship to Exponential Family PCA (Collins et al., 2002). EPCA was originally formulated as the solution to an optimization problem based on Bregman divergences, while our model is a fully probabilistic model in which all parameters can be integrated out via MCMC. However, it is possible to think of EPCA as the likelihood function of a probabilistic model, which coupled with a prior on the parameters, would make it possible to do Bayesian inference in EPCA and would render it closer to our model. However, our model was entirely motivated by the idea of partial membership in clusters, which is enforced by forming convex combinations of the natural parameters of exponential family models, while EPCA is based on *linear* combinations of the parameters. Therefore: EPCA does not naturally reduce to clustering, none of the variables can be interpreted as partial memberships, and the coefficients define a plane rather than a convex region in parameter space.

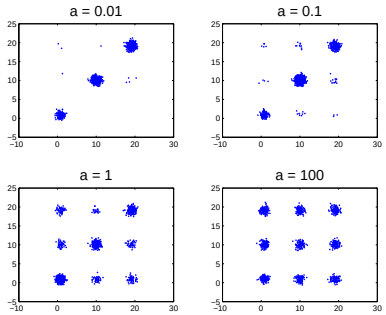


Figure 4. Generative plot for MM model with 3 Gaussian clusters

The recent work of Buntine and Jakulin (Buntine & Jakulin, 2006) focusing on the analysis of discrete data is also closely related to the BPM model. The framework of (Buntine & Jakulin, 2006) section III B expresses a model for discrete data in terms of linear mixtures of dual exponential family parameters where MAP inference is performed. Section V B also provides insights on differences between using dual and natural parameters.

Fuzzy Clustering: The notion that probabilistic models are unable to handle partial membership has been used to argue that probability is a subtheory of or different in character from fuzzy logic (Zadeh, 1965; Kosko, 1992). In this paper we described a probabilistic model for partial membership which may be of use in the many application domains where fuzzy clustering has been used.

Fuzzy K-means clustering (Bezdek, 1981) iteratively minimizes the following objective: $J = \sum_{n=1}^{N}\sum_{k=1}^{K} \pi_{nk}^{\gamma} d^2(\boldsymbol{x}_n, \boldsymbol{c}_k)$, where $\gamma > 1$ is an exponent parameter, π_{nk} represents the degree of membership of

data point n in cluster k ($\sum_k \pi_{nk} = 1$), and $d^2(\boldsymbol{x}_n, \boldsymbol{c}_k)$ is a measure of squared distance between data point $\boldsymbol{x}_n$ and cluster center $\boldsymbol{c}_k$. By varying γ it is possible to attain different amounts of partial membership, where the limiting case $\gamma = 1$ is K-means with no partial membership. Although the π parameters represent partial membership, none of the variables have probabilistic interpretations.

IOMM: Lastly, this work is related to the Infinite Overlapping Mixture Model (IOMM) (Heller & Ghahramani, 2007) in which overlapping clustering is performed, also by taking products of exponential family distributions, much like products of experts (Hinton, 1999). However in the IOMM the memberships of data points to clusters are restricted to be binary, which means that it can not model partial membership.

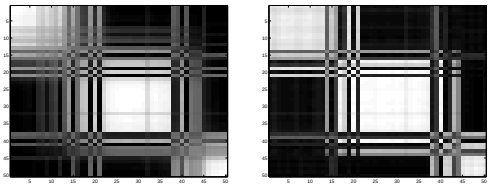
7. Experiments

We generated a synthetic binary data set from the BPM, and used this to test BPM learning. The synthetic data set had 50 data points which each have 32 dimensions and can hold partial memberships in 3 clusters. We ran our Hybrid Monte Carlo sampler for 4000 iterations, burning in the first half. In order to compare our learned partial membership assignments for data points (Π_L) to the true ones (Π_T) for this synthetic data set, we compute ($\hat{U} = \Pi_L \Pi_L^\top$) and ($U^* = \Pi_T \Pi_T^\top$), which basically give the total amount of cluster membership shared between each pair of data points, and is invariant to permutations of cluster labels. Both of these matrices can be seen in figure 5. One can see that the structure of these two matrices is quite similar, and that the BPM is learning the synthetic data reasonably. For a more quantitative measure table 5c gives statistics on the number of pairs of data points whose learned shared membership differs from the true shared membership by more than a given threshold (the range of this statistic is [0,1]).

We also used the BPM to model two "real-world" data sets. The first is senate roll call data from the 107th US congress (2001-2002) (Jakulin, 2004), and the second is a data set of images of sunsets and towers.

The senate roll call data is a matrix of 99 senators (one senator died in 2002 and neither he nor his replacement is included) by 633 votes. It also includes the outcome of each vote, which is treated as an additional data point (like a senator who always voted the actual outcome). The matrix contained binary features for yea and nay votes, and we used the BPM to cluster this data set using $K = 2$ clusters. There are missing values in this dataset but this can easily be dealt with in the HMC log probability calculations by explicitly representing both 0 and 1 binary values and leaving out missing values. The results are given in figure 6. The line in figure 6 represents the amount of membership of each senator in one of the clusters (we used the "Democrat" cluster, where senators on the far left have partial memberships very close to 0, and those on the far right have partial memberships extremely close to 1). Since there are two clusters, and the amount of membership always sums to 1 across clusters, the figure looks the same regardless of whether we are looking at the "Democrat" or "Republican" cluster. We can see that most Republicans and Democrats are tightly clustered at the ends of the line (and have partial memberships very close to 0 and 1), but that there is a fraction of senators (around 20%) which lies somewhere reasonably in between the extreme partial memberships of 0 or 1. Interesting properties of this figure include the location of Senator Jeffords who left the Republican party in 2001 to become an independent who caucused with the Democrats. Also Senator Chafee who is known as a moderate Republican and who often voted with the Democrats (for example, he was the only Republican to vote against authorizing the use of force in Iraq), and Senator Miller a conservative Democrat who supported George Bush over John Kerry in the 2004 US Presidential elections. Lastly, it is interesting to note the location of the Outcome data point, which is very much in the middle. This makes sense since the 107th congress was split 50-50 (with Republican Dick Cheney breaking ties), until Senator Jeffords became an Independent at which point the Democrats had a one seat majority.

We also tried running both fuzzy k-means clustering and Dirichlet Process Mixture models (DPMs) on this data set. While fuzzy k-means found roughly similar rankings of the senators in terms of membership to the "Democrat" cluster, the exact ranking and, in particular, the amount of partial membership ($\pi_{\mathbf{n}}$) each senator had in the cluster was *very* sensitive to the fuzzy exponent parameter, which is typically set by hand. Figure 7a plots the amount of membership for the Outcome data point in black, as well as the most extreme Republican, Senator Ensign, in red, and the most extreme Democrat, Senator Schumer, in blue, as a function of the fuzzy exponent parameter. We can see in this plot that as the assignment of the Outcome data point begins to reach a value even reasonably close to 0.5, the most extreme Republican already has 20% membership in the "Democrat" cluster. This reduction in range does not make sense semantically, and presents a trade-off between finding reasonable values



Statistic	Percent
$\lvert(\hat{U} - U^*)\rvert \leq 0.1$	60.40
$\lvert(\hat{U} - U^*)\rvert \leq 0.2$	84.28
$\lvert(\hat{U} - U^*)\rvert \leq 0.3$	95.48
$\lvert(\hat{U} - U^*)\rvert \leq 0.4$	99.68
$\lvert(\hat{U} - U^*)\rvert \leq 0.5$	100.00

Figure 5. a) left - matrix U^* showing the true shared partial memberships for pairs of data points. b) right - matrix $\hat{U}$ showing the learned shared partial memberships. c) Summary statistics for learned $\hat{U}$. Reports the percentage of pairs in $\hat{U}$ whose difference from U^* in terms of the amount of shared partial memberships is at most the given threshold (0.1 - 0.5).

for π_n in the middle of the range, versus at the extremes. This kind of sensitivity to parameters does not exist in our BPM model, which models both extreme and middle range values well.

We tried using a DPM to model this data set where we ran the DPM for 1000 iterations of Gibbs sampling, sampling both assignments and concentration parameter. The DPM confidently finds 4 clusters: one cluster consists solely of Democrats, one consists solely of Republicans, the third cluster has 9 of the most moderate Democrats and Republicans plus the "vote outcome" variable, and the last cluster has just one member, Hollings (D-SC). Figure 7b is a 100x100 matrix showing the overlap of cluster assignments for pairs of senators, averaged over 500 samples (there are no changes in relative assignments, the DPM is completely confident). The interpretation of the data provided by the DPM is very different from the BPM model's. The DPM does *not* use uncertainty in cluster membership to model Senators with intermediate views. Rather, it creates an entirely new cluster to model these Senators. This makes sense for the data as viewed by the DPM: there is ample data in the roll calls that these Senators are moderate — it is not the case that there is uncertainty about whether they fall in line with hard-core Democrats or Republicans. This highlights the fact that the responsibilities in a mixture model (such as the DPM) cannot and should not be interpreted as partial membership, they are representations of *uncertainty* in full membership. The BPM model, however, explicitly models the partial membership, and can, for example, represent the fact that a Senator might be best characterized as moderate (and quantify how moderate they are). In order to quantify this comparison we calculated the negative log predictive probability (in bits) across senators for the BPM and the DPM (Table 1). We look at a number of different measures: the mean, median, minimum and maximum number of bits required to encode a senator's votes. We also look at the number of bits needed to encode the "Outcome" in particular. On all of these measures

	Mean	Median	Min	Max	"Outcome"
BPM	**187**	**168**	**93**	422	**224**
DPM	196	178	112	**412**	245

Table 1. Comparison between the BPM and a DPM in terms of negative log predictive probability (in bits) across senators.

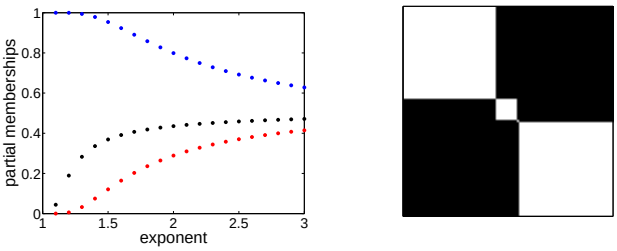


Figure 7. a) left - fuzzy k-means: plot of the partial membership values for the Outcome data point (in black) and the most extreme Republican (in red) and Democrat (in blue) as a function of the fuzzy exponent parameter. b) right - DPMs: an ordered 100x100 matrix showing the fraction of times each pair of senators was assigned to the same cluster, averaged over 500 Gibbs sampling iterations.

except for maximum, the BPM performs better than the DPM, showing that the BPM is a superior model for this data set.

Lastly, we used the BPM to model images of sunsets and towers. The dataset consisted of 329 images of sunsets or towers, each of which was represented by 240 binary simple texture and color features. Partial assignments to $K = 2$ clusters were learned, and figure 8 provides the result. The top row of the figure is the three images with the most membership in the "sunset" cluster, the bottom row contains the three images with the most membership in the "tower" cluster, and the middle row shows the 3 images which have closest to 50/50 membership in each cluster ($\pi_{nk} \approx 0.5$). In this dataset, as well as all the datasets described in this section, our HMC sampler was very fast, giving reasonable results within tens of seconds.

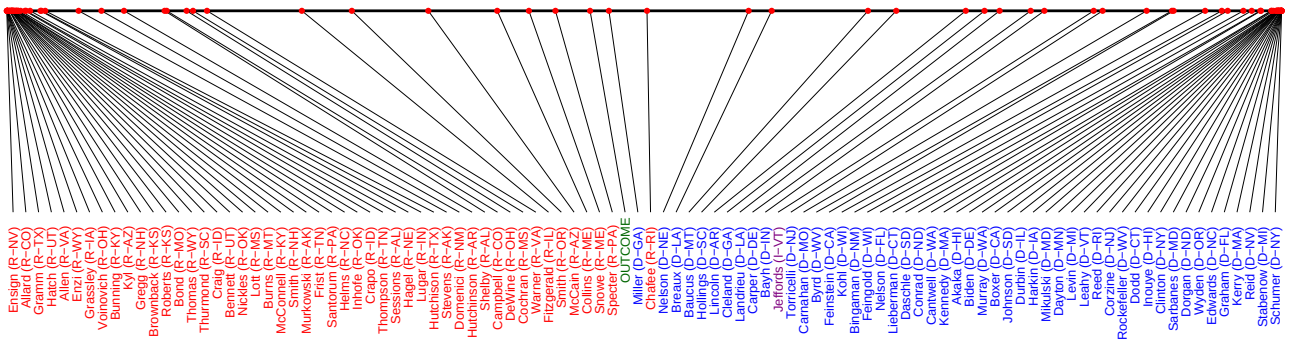


Figure 6. Analysis of the partial membership results on the Senate roll call data from 2001-2002. The line shows amount of membership in the "Democrat" cluster with the left of the line being the lowest and the right the highest.

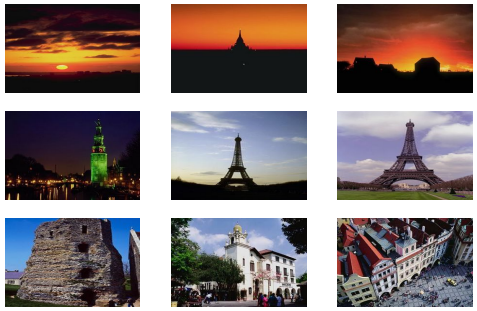


Figure 8. Tower and Sunset images. The top row are the images found to have largest membership in the "sunset" cluster, the bottom row are images found to have largest membership in the "tower" cluster, and the middle row are the images which have the most even membership in both clusters.

8. Conclusions and Future Work

In summary, we have described a fully probabilistic approach to data modelling with partial membership using continuous latent variables, which can be seen as a relaxation of clustering with finite mixture models. We employed a full Hybrid Monte Carlo algorithm for inference, and our experience with HMC has been very positive. Despite the general reputation of MCMC methods for being slow, our model using HMC seems to discover sensible partial membership structure after surprisingly few samples.

In the future we would like to develop a nonparametric version of this model. The most obvious way to try to generalize this model would be with a Hierarchical Dirichlet Process (Teh et al., 2006). However, this would involve averaging over infinitely many potential clusters, which is both computationally infeasible, and also undesirable from the point of view that each data point should have non-zero partial membership in only a few (certainly finite) number of clusters. A more promising alternative is to use an Indian Buffet Process (Griffiths & Ghahramani, 2005), where each 1 in a row in an IBP sample matrix would represent a cluster in which the data point corresponding to that row has non-zero partial membership, and then draw the continuous values for those partial memberships conditioned on that IBP matrix.

References

Bezdek, J. (1981). *Pattern recognition with fuzzy objective function algorithms*. Kluwer.

Blei, D., Ng, A., & Jordan, M. (2003). Latent dirichlet allocation. *JMLR*.

Buntine, W., & Jakulin, A. (2006). *LNCS*, vol. 3940, chapter Discrete Component Analysis. Springer.

Collins, M., Dasgupta, S., & Schapire, R. (2002). A generalization of principal components analysis to the exponential family. *NIPS*.

Erosheva, E., Fienberg, S., & Lafferty, J. (2004). Mixed membership models of scientific publications. *PNAS*.

Gasch, A., & Eisen, M. (2002). Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biol.*, *3*.

Griffiths, T., & Ghahramani, Z. (2005). *Infinite latent feature models and the indian buffet process* (Technical Report). Gatsby Computational Neuroscience Unit.

Heller, K., & Ghahramani, Z. (2007). A nonparametric bayesian approach to modeling overlapping clusters. *AISTATS*.

Hinton, G. (1999). Products of experts. *ICANN*.

Jakulin, A. (2004). http://www.ailab.si/aleks/politics/.

Kosko, B. (1992). *Neural networks and fuzzy systems*. Prentice Hall.

MacKay, D. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press.

Neal, R. (1993). *Probabilistic inference using markov chain monte carlo methods* (Technical Report). University of Toronto.

Teh, Y., Jordan, M., Beal, M., & Blei, D. (2006). Hierarchical dirichlet processes. *JASA*, *101*.

Zadeh, L. (1965). Fuzzy sets. *Info. and Control*, *8*.

Active Kernel Learning

Steven C.H. Hoi CHHOI@NTU.EDU.SG
School of Computer Engineering, Nanyang Technological University, Singapore
Rong Jin RONGJIN@CSE.MSU.EDU
Department of Computer Science and Engineering, Michigan State University

Abstract

Identifying the appropriate kernel function/matrix for a given dataset is essential to all kernel-based learning techniques. A number of kernel learning algorithms have been proposed to learn kernel functions or matrices from side information (e.g., either labeled examples or pairwise constraints). However, most previous studies are limited to "passive" kernel learning in which side information is provided beforehand. In this paper we present a framework of *Active Kernel Learning* (**AKL**) that actively identifies the most informative pairwise constraints for kernel learning. The key challenge of active kernel learning is how to measure the informativeness of an example pair given its class label is unknown. To this end, we propose a **min-max** approach for active kernel learning that selects the example pair that results in a large classification margin regardless of its assigned class label. We furthermore approximate the related optimization problem into a convex programming problem. We evaluate the effectiveness of the proposed algorithm by comparing it to two other implementations of active kernel learning. Empirical study with nine datasets on semi-supervised data clustering shows that the proposed algorithm is more effective than its competitors.

1. Introduction

Kernel methods have attracted more and more attention of researchers in computer science and engineering due to their superior performance in data clustering, classification, and dimensionality reduction (Scholkopf & Smola, 2002; Vapnik, 1998). Kernel methods have been applied to many fields, such as data mining, pattern recognition, information retrieval, computer vision, and bioinformatics, etc. Since the choice of kernel functions or matrices is often critical to the performance of many kernel-based learning techniques, it becomes a more and more important research problem for how to automatically learn a kernel function/matrix for a given dataset. Recently, a number of kernel learning algorithms (Chapelle et al., 2003; Cristianini et al., 2002; Hoi et al., 2007; Kondor & Lafferty, 2002; Kulis et al., 2006; Lanckriet et al., 2004; Zhu et al., 2005) have been proposed to learn kernel functions or matrices from side information. The side information can be provided in two different forms: either labeled examples or pairwise constraints. In the latter case, two types of pairwise constraints are examined in the previous studies: a must-link pair where two examples should belong to the same class, and a cannot-link pair where two examples should belong to different classes. In this study, we focus on kernel learning with pairwise constraints.

Most kernel learning methods, termed as "passive kernel learning", assume that labeled data is provided beforehand. However, given the labeled data may be expensive to acquire, it is more cost effective if we are able to identify the most informative example pairs such that the kernel can be learned efficiently with only a small number of pairwise constraints. To this end, we focus on **active kernel learning** (AKL) whose goal is to identify the example pairs that are informative to the target kernels. We extends our previous work on non-parametric kernel learning (Hoi et al., 2007) to active kernel learning. As shown in (Hoi et al., 2007), the parametric approaches for kernel learning are often limited by their capacity in fitting diverse patterns of real-world data, and therefore are not as effective as the non-parametric approach for kernel learning.

The simplest approach toward active kernel learning is to measure the informativeness of an example pair by its kernel similarity. Given a pair of examples $(\mathbf{x}_i, \mathbf{x}_j)$, we assume that $K_{i,j}$, the kernel similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, is a large positive number when $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same class, and a large negative number

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

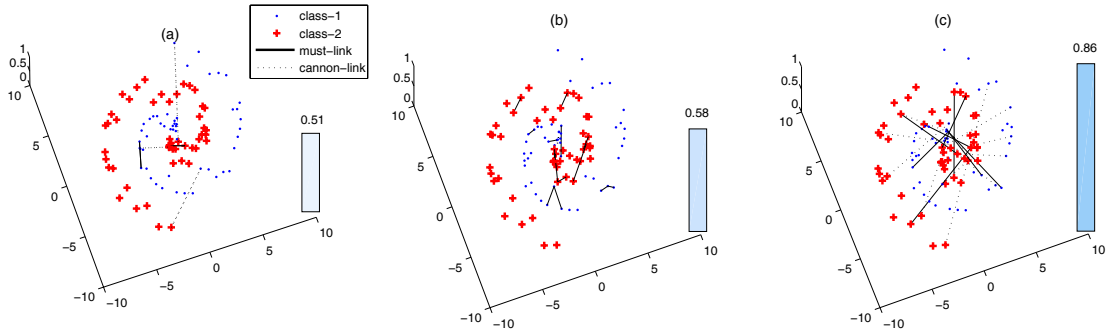


Figure 1. Examples of active kernel learning: (a) double-spiral artificial data with some given pairwise constraints, (b) AKL with the least $|K_{i,j}|$, (c) the proposed AKL method. The right bars show the resulting clustering accuracies using kernel k-means clustering methods.

when they are in different classes. Thus, by following the uncertainty principle of active learning (Tong & Koller, 2000; Hoi et al., 2006), the most informative example pairs should be the ones whose kernel similarities are closest to zero. In other words, the criterion is to select the example pair with the least absolute kernel similarity (i.e., $|K_{i,j}|$). Unfortunately, this simple approach may not always be effective in obtaining informative pairwise constraints for kernel learning. Figure 1 illustrates an example of active kernel learning for data clustering. In this example, Figure 1(a) shows an artificial dataset of two classes together with a few pairwise constraints. Figure 1(b) shows the pairwise constraints with the least $|K_{i,j}|$. We observe that most of them are must-link pairs with two data points separated by a modest distance. Since must-link constraints are not informative to the clustering boundary, a relatively small improvement is observed in clustering accuracy (from 51% to 58%) when using the kernel learned by this simple approach. In contrast, as shown in Figure 1(c), the proposed approach for active kernel learning is able to identify a pool of diverse pairwise constraints, including both must-links and cannot-links. The clustering accuracy is increased significantly, from 51% to 86%, by using the proposed active kernel learning.

The rest of this paper is organized as follows. Section 2 presents the min-max framework for our active kernel learning method, in which the problem is formulated into a convex optimization problem. Section 3 describes the results of the experimental evaluation. Section 4 concludes this work.

2. Active Kernel Learning

Our work extends the previous work on non-parametric kernel learning (Hoi et al., 2007) by introducing the component of actively identifying the example pairs that are most informative to the target kernel. In this section, we will first briefly review the non-parametric approach for kernel learning in (Hoi et al., 2007), followed by the description of the min-max framework for active kernel learning.

2.1. Non-parametric Kernel Learning

Let the entire data collection be denoted by $\mathcal{U} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ where each data point $\mathbf{x}_i \in \mathbb{R}^d$ is a vector of d elements. Let $S \in \mathbb{R}^{N \times N}$ be a symmetric matrix where each $S_{i,j} \geq 0$ represents the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. Unlike the kernel similarity matrix, S does not have to be positive semi-definite. For the convenience of presentation, we set $S_{i,i} = 0$ for all the examples. Then, according to (Hoi et al., 2007), a normalized graph Laplacian L is constructed using the similarity matrix S as follows:

$$L = (1+\delta)I - D^{-1/2} S D^{-1/2}$$

where $D = \text{diag}(d_1, d_2, \ldots, d_N)$ is a diagonal matrix with $d_i = \sum_{j=1}^{N} f(\mathbf{x}_i, \mathbf{x}_j)$. A small $\delta > 0$ is introduced to prevent L from being singular. Let's denote by $\mathcal{T}$ the set of pairwise constraints. We construct a matrix $T \in \mathbb{R}^{N \times N}$ to represent the pairwise constraints in $\mathcal{T}$, i.e.,

$$T_{i,j} = \begin{cases} +1 & (\mathbf{x}_i, \mathbf{x}_j) \text{ is a must-link pair in } \mathcal{T} \\ -1 & (\mathbf{x}_i, \mathbf{x}_j) \text{ is a cannot-link pair in } \mathcal{T} \\ 0 & \text{otherwise} \end{cases}$$

Given the similarity matrix S and the pairwise constraints in $\mathcal{T}$, the goal of kernel learning is to identify a kernel matrix $Z \in \mathbb{R}^{N \times N}$ that is consistent with both $\mathcal{T}$ and S. Following (Hoi et al., 2007), we formulate it

into the following convex optimization problem:

$$\begin{aligned} \arg\min_{Z,\varepsilon} \quad & \mathrm{tr}(LZ) + \frac{c}{2}\sum_{(i,j)\in\mathcal{T}} \varepsilon_{i,j}^2 \qquad (1) \\ \text{s. t.} \quad & \forall (i,j)\in\mathcal{T},\ Z_{i,j}T_{i,j} \geq 1-\varepsilon_{i,j},\ \varepsilon_{i,j}\geq 0 \\ & Z \succeq 0 \end{aligned}$$

The first term in the above objective function plays a similar role as the manifold regularization (Belkin & andd P. Niyogi, 2004), where the graph Laplacian is used to regularize the classification results. The second term in the above measures the inconsistency between the learned kernel matrix Z and the given pairwise constraints. Note that unlike the formulation in (Hoi et al., 2007), we change $\varepsilon_{i,j}$ in the loss function to $\varepsilon_{i,j}^2$. This modification is specifically designed for active kernel learning, and the reason will be clear later. It is not difficult to see that the problem in (1) is a semi-definite programming problem, and therefore can be solved by the standard software package, such as SeDuMi (Sturm, 1999).

2.2. Min-max Framework for Active Kernel Learning

The simplest approach toward active kernel learning is to follow the uncertainty principle of active learning, and to select the example pair $(\mathbf{x}_i, \mathbf{x}_j)$ with the least $|Z_{i,j}|$ [1]. However, as already discussed in the introduction section, the key problem with this simple approach is that the example pairs with the least $|Z_{i,j}|$ may not necessarily be the the most informative ones, and therefore may not result in an efficient learning of the kernel matrix. To address this problem, we propose a min-max framework for active kernel learning that measures the informativeness of an example pair by how significantly the selected example pair will affect the target kernel matrix.

Consider an unlabeled example pair $(\mathbf{x}_k, \mathbf{x}_l) \notin \mathcal{T}$. To measure how this example will affect the kernel matrix, we consider the kernel learning problem with the additional example pair $(\mathbf{x}_k, \mathbf{x}_l)$ labeled by $y \in \{-1, +1\}$, i.e.,

$$\begin{aligned} \min_{Z,\varepsilon} \quad & \mathrm{tr}(LZ) + \frac{c}{2}\sum_{(i,j)\in\mathcal{T}} \varepsilon_{i,j}^2 + \frac{c}{2}\varepsilon_{k,l}^2 \qquad (2) \\ \text{s. t.} \quad & T_{i,j}Z_{i,j} \geq 1-\varepsilon_{i,j},\ \forall (i,j)\in\mathcal{T} \\ & yZ_{k,l} \geq 1-\varepsilon_{k,l},\ Z\succeq 0 \end{aligned}$$

Let us denote by $\omega((k,l),y)$ the optimal value of the above optimization problem. Intuitively, $\omega((k,l),y)$ measures the overall classification accuracy with the additional example pair $(\mathbf{x}_k, \mathbf{x}_l)$ labeled by y. To further measure the informativeness of example pair $(\mathbf{x}_k, \mathbf{x}_l)$, we introduce the quantity $\kappa(k,l)$ as follows

$$\kappa(k,l) = \max_{y\in\{-1,+1\}} \omega((k,l),y) \qquad (3)$$

Clearly, $\kappa(k,l)$ measures the worst classification error with the addition of example pair $(\mathbf{x}_k, \mathbf{x}_l)$. Overall, $\kappa(k,l)$ measures how the example pair $(\mathbf{x}_k, \mathbf{x}_l)$ will affect the overall objective function, which indirectly measures the impact of the example pair on the target kernel matrix. To see this, consider an example pair $(\mathbf{x}_k, \mathbf{x}_l)$ that is highly consistent with the current kernel Z with label y (i.e., $Z_{k,l}y \geq 1$). According to the definition $\kappa(k,l)$, we would expect a large $\kappa(k,l)$ for pair $(\mathbf{x}_k, \mathbf{x}_l)$. This is because by assigning a label $-y$ to example pair $(\mathbf{x}_k, \mathbf{x}_l)$, we expect a large classification error and therefore large $\kappa(k,l)$. Hence, we use $\kappa(k,l)$ to measure the *uninformativeness* of example pairs, i.e., the smaller $\kappa(k,l)$, the less informative the example pair is. Therefore, the most informative example pair is found by minimizing $\kappa(k,l)$, i.e.,

$$(k,l)^* = \arg\min_{(k,l)\notin\mathcal{T}} \max_{t\in\{-1,+1\}} \omega((k,l),t) \qquad (4)$$

Directly solving the min-max optimization problem in (4) is challenging because function $\omega((k,l),t)$ is defined implicitly by the optimization problem in (2). The following theorem allows us to significantly simplify the optimization problem in (4)

Theorem 1. *The optimization problem in (4) is equivalent to the following optimization problem*

$$\begin{aligned} \min_{Z,\varepsilon,(k,l)\notin\mathcal{T}} \quad & \mathrm{tr}(LZ) + \frac{c}{2}\sum_{(i,j)\in\mathcal{T}} \varepsilon_{i,j}^2 + \frac{c}{2}\varepsilon_{k,l}^2 \qquad (5) \\ s.\ t. \quad & T_{i,j}Z_{i,j} \geq 1-\varepsilon_{i,j}, \varepsilon_{i,j}\geq 0,\ \forall (i,j)\in\mathcal{T} \\ & \varepsilon_{k,l} \geq 1+|Z_{k,l}|,\ Z\succeq 0 \end{aligned}$$

Proof. The above theorem follows the fact that the solution $y^* \in \{-1,+1\}$ maximizing $\omega((k,l),y)$ is $y^* = -\mathrm{sign}(Z_{k,l})$. This fact allows us to remove the maximization within (4) and obtain the result in the theorem. □

The following corollary shows that the approach of selecting the example pair with the least $|Z_{k,l}|$ indeed corresponds to a special solution for the problem in (5).

Corollary 2. *The optimal solution to (5) with fixed kernel matrix Z is the example pair with the least $|Z_{k,l}|$, i.e.,*

$$(k,l)^* = \arg\min_{(k,l)\notin\mathcal{T}} |Z_{k,l}|$$

[1] Here we assume that $Z_{i,j} > 0$ when $\mathbf{x}_i$ and $\mathbf{x}_j$ are likely to share the same class, and $Z_{i,j} < 0$ when $\mathbf{x}_i$ and $\mathbf{x}_j$ are likely to be assigned to different classes

Proof. By fixing Z, the problem in (5) is simplified as

$$\min_{(k,l)\notin\mathcal{T}} \quad \varepsilon_{k,l}^2 \quad \text{s. t.} \quad \varepsilon_{k,l} \geq 1 + |Z_{k,l}|$$

It is easy to see that the optimal solution to the above problem is the example pair with the least $|Z_{k,l}|$. □

Note that a similar observation is described in the study (Chen & Jin, 2007) for standard active learning.

2.3. Algorithm

The straightforward approach toward the optimization problem in (5) is to try out every example pair $(\mathbf{x}_k, \mathbf{x}_l) \notin \mathcal{T}$. Evidently, this approach will not scale well when the number of example pairs is large.

Our first attempt toward solving the problem (5) is to turn it into a continuous optimization problem. To this purpose, we introduce variable $p_{k,l} \geq 0$ to represent the probability of selecting the example pair $(k,l) \notin \mathcal{T}$. Using this notation, we have the optimization problem in (5) rewritten as

$$\begin{aligned} \min_{Z\succeq 0, p, \varepsilon} \quad & \mathrm{tr}(LZ) + \frac{c}{2}\sum_{(i,j)\in\mathcal{T}} \varepsilon_{i,j}^2 + \frac{c}{2}\sum_{(k,l)\notin\mathcal{T}} p_{k,l}\varepsilon_{k,l}^2 \quad (6) \\ \text{s. t.} \quad & T_{i,j}Z_{i,j} \geq 1 - \varepsilon_{i,j},\ \forall (i,j) \in \mathcal{T} \\ & \varepsilon_{k,l} - 1 \geq Z_{k,l} \geq 1 - \varepsilon_{k,l}, \forall (k,l) \notin \mathcal{T} \\ & \sum_{(k,l)\notin\mathcal{T}} p_{k,l} \geq 1,\ p_{k,l} \geq 0, \forall (k,l) \notin \mathcal{T} \end{aligned}$$

The following theorem shows the relationship between (6) and (5).

Theorem 3. *Any global optimal solution to (5) is also a global optimal solution to (6).*

The proof of the above theorem can be found in Appendix A.

Unfortunately, the optimization problem in (6) is non-convex because of the term $p_{k,l}\varepsilon_{k,l}^2$. It is therefore difficult to find the global optimal solution for (6). In order to turn (6) into a convex optimization problem, we view the constraint $\sum_{(k,l)\notin\mathcal{T}} p_{k,l} \geq 1$ as a bound for the arithmetic mean of $p_{k,l}$, i.e.,

$$\frac{1}{m}\sum_{(k,l)\notin\mathcal{T}} p_{k,l} \geq \frac{1}{m}$$

where $m = |\{(k,l)|(k,l) \notin \mathcal{T}\}|$. We then relax this constraint by the harmonic mean of $p_{k,l}$, i.e.,

$$\frac{m}{\sum_{(k,l)\notin\mathcal{T}} p_{k,l}^{-1}} \geq \frac{1}{m}, \text{ or } \sum_{(k,l)\notin\mathcal{T}} p_{k,l}^{-1} \leq m^2$$

The above relaxation is based on the property that a harmonic mean is no larger than an arithmetic mean. By replacing the constraint $\sum_{(k,l)\notin\mathcal{T}} p_{k,l} \leq 1$ with (7), we have (6) relaxed into the following optimization problem

$$\begin{aligned} \min_{Z\succeq 0, p, \varepsilon} \quad & \mathrm{tr}(LZ) + \frac{c}{2}\sum_{(i,j)\in\mathcal{T}} \varepsilon_{i,j}^2 + \frac{c}{2}\sum_{(k,l)\notin\mathcal{T}} p_{k,l}\varepsilon_{k,l}^2 \quad (7) \\ \text{s. t.} \quad & T_{i,j}Z_{i,j} \geq 1 - \varepsilon_{i,j}, \varepsilon_{i,j} \geq 0,\ \forall (i,j) \in \mathcal{T} \\ & \varepsilon_{k,l} - 1 \geq Z_{k,l} \geq 1 - \varepsilon_{k,l}, \forall (k,l) \notin \mathcal{T} \\ & \sum_{(k,l)\notin\mathcal{T}} p_{k,l}^{-1} \leq m^2,\ 0 \leq p_{k,l} \leq 1, \forall (k,l) \notin \mathcal{T} \end{aligned}$$

By defining variable $h_{k,l} = p_{k,l}^{-1}$, we have

$$\begin{aligned} \min_{Z\succeq 0, h, \varepsilon} \quad & \mathrm{tr}(LZ) + \frac{c}{2}\sum_{(i,j)\in\mathcal{T}} \varepsilon_{i,j}^2 + \frac{c}{2}\sum_{(k,l)\notin\mathcal{T}} \frac{\varepsilon_{k,l}^2}{h_{k,l}} \quad (8) \\ \text{s. t.} \quad & T_{i,j}Z_{i,j} \geq 1 - \varepsilon_{i,j}, \varepsilon_{i,j} \geq 0,\ \forall (i,j) \in \mathcal{T} \\ & \varepsilon_{k,l} - 1 \geq Z_{k,l} \geq 1 - \varepsilon_{k,l}, \forall (k,l) \notin \mathcal{T} \\ & \sum_{(k,l)\notin\mathcal{T}} h_{k,l} \leq m^2,\ h_{k,l} \geq 1, \forall (k,l) \notin \mathcal{T} \end{aligned}$$

Notice that constraint $0 \leq p_{k,l} \leq 1$ is transferred into $h_{k,l} \geq 1$. The following theorem shows the property of the formulation in (8)

Theorem 4. *We have the following properties for (8)*

- *(8) is a semi-definite programming (SDP) problem.*
- *Any feasible solution to (8) is also a feasible solution to (5) with $p_{k,l} = h_{k,l}^{-1}$, and the optimal value for (6) is upper bounded by that for (8).*

The proof is provided in Appendix B. Note that using $\varepsilon_{i,j}^2$ instead of $\varepsilon_{i,j}$ for the loss function is key to turning (6) into a convex optimization problem. The second property stated in Theorem 4 indicates that by minimizing (8), we guarantee a small value for the objective function in (6).

The following theorem shows the dual problem of (8), which is the key to the efficient computation.

Theorem 5. *The dual problem of (8) is*

$$\begin{aligned} \max_{Q,W} \quad & \sum_{(i,j)\in\mathcal{T}} \left(Q_{i,j} - \frac{Q_{i,j}^2}{2c}\right) + \sum_{(k,l)\notin\mathcal{T}} \left(|W_{k,l}| - \frac{W_{k,l}^2}{2c}\right) \\ & - \frac{2(m^2-m)}{c}\lambda \quad (9) \\ s.\ t \quad & L \succeq Q \otimes T + W \otimes \bar{T} \\ & \forall (i,j) \in \mathcal{T}, Q_{i,j} \geq 0, \quad \lambda \geq W_{k,l}^2, \forall (k,l) \notin \mathcal{T} \end{aligned}$$

where matrix $\bar{T}$ is defined as

$$\bar{T}_{i,j} = \begin{cases} 0 & (i,j) \in \mathcal{T} \\ 1 & otherwise \end{cases},$$

and $\otimes$ stands for the element wise product of matrices.

The proof can be found in Appendix C. In the dual problem, variables $Q_{i,j}$ and $W_{i,j}$ are the dual variables that indicate the importance of labeled example pairs and unlabeled examples, respectively. We thus will select the unlabeled example pair with the largest $|W_{i,j}|$. To speed up the computation, in our experiment, we first select a subset of example pairs (fixed 200) with smallest $|Z_{i,j}|$ using the current kernel matrix Z. We then set all $W_{k,l}$ to be zero if the corresponding pair is not selected. In this way, we significantly reduce the number of variables in the dual problem in (9), thus simplifying the computation.

3. Experimental Results

In our experiments, we follow the work (Hoi et al., 2007), and evaluate the proposed algorithm for active kernel learning by the experiments of data clustering. More specifically, we first apply the active kernel learning algorithm to identify the most informative example pairs, and then solicit the class labels for the selected example pairs. A kernel matrix will be learned from the labeled example pairs, and the learned kernel matrix will be used by the clustering algorithm to find the right cluster structure.

3.1. Experimental Setup

We use the same datasets as the ones described in (Hoi et al., 2007). Table 1 summarizes the information about the nine datasets used in our study. We adopt the clustering accuracy defined in (Xing et al., 2002) as the evaluation metric. It is defined as follows

$$Accuracy = \sum_{i>j} \frac{\mathbf{1}\{\mathbf{1}\{c_i = c_j\} = \mathbf{1}\{\hat{c}_i = \hat{c}_j\}\}}{0.5n(n-1)}, \quad (10)$$

where $\mathbf{1}\{\cdot\}$ is the indicator function that outputs 1 when the input argument is true and 0 otherwise. c_i and $\hat{c}_i$ denote the true cluster membership and the predicted cluster membership of the ith data point, respectively. n is the number of examples in the dataset. For the graph Laplacian L used by the nonparametric kernel learning, we apply the standard method for all experiments, i.e., by calculating the distance matrix by Euclidean distance, then constructing the adjacency matrix with five nearest neighbors, and finally normalizing the graph to achieve the final Laplacian matrix.

3.2. Performance Evaluation

To evaluate the quality of the learned kernels, we extend the proposed kernel learning algorithm to solve clustering problems with pairwise constraints. In the experiments, we employ the kernel k-means as the clustering method, in which the kernel is learned by the proposed non-parametric kernel learning method. In addition to the proposed active kernel learning method, two baseline approaches are implemented to select informative example pairs for kernel learning. Totally we have:

Table 1. The nine datasets used in our experiments. The first two are the artificial datasets from (Hoi et al., 2007) and the others are from the UCI machine learning repository.

Dataset	#Classes	#Instances	#Features
Chessboard	2	100	2
Double-Spiral	2	100	3
Glass	6	214	9
Heart	2	270	13
Iris	3	150	4
Protein	6	116	20
Sonar	2	208	60
Soybean	4	47	35
Wine	3	178	12

- **Random**: This baseline method randomly samples example pairs from the pool of unlabeled pairs.
- **AKL-min-|Z|**: This baseline method chooses the pair examples with the least $|Z_{k,l}|$, where matrix Z is learned by the non-parametric kernel learning method. As already discussed in the introduction section, this approach may not find the most informative example pairs.
- **AKL-min-H**: This is the proposed AKL algorithm. It selects the example pairs with least $H_{k,l}$ that corresponds to the maximal selection probability $P_{k,l}$.

To examine the performance of the proposed AKL algorithm in a full spectrum, we evaluate the clustering results with respect to different sampling sizes. Specifically, for each experiment, we first randomly sample N_c pairwise constraints as the initially labeled pair examples. We then employ the nonparametric kernel learning method to learn a kernel from the given pairwise constraints. This learned kernel is engaged by the kernel k-means method for data clustering. Next, we apply the AKL method to sample 20 pair examples (i.e. 20 pairwise constraints) for labeling in an iteration, and then examine the clustering results based on the kernel that is learned from the augmented set of example pairs in each iteration.

Each experiment is repeated 50 times with multiple restarts for clustering. Fig. 2 shows the experimental results on the nine datasets with five active kernel learning iterations. First of all, we observe that AKL-min-$|Z|$, i.e., the naive AKL approach that samples the example pairs with the least $|Z|$, does not always outperform the random sampling approach. In fact, it only outperforms the random sampling approach on five out of the nine datasets. It performs noticeably worse than the random approach on dataset "sonar" and "heart". Compared with the two baseline approaches, the proposed AKL algorithm (i.e., AKL-min-H) achieves considerably better performance for most datasets. For example, for the "Double-Spiral" dataset, after 3 active kernel learning iterations, the proposed algorithm is able to achieve the clustering accuracy of 99.6%, but the clustering accuracies of the other two methods are less than 98.8%. These experimental results show the effectiveness of the proposed algorithm as a promising approach for active kernel learning.

4. Conclusion

In this paper we proposed a min-max framework for active kernel learning that specifically addresses the problem of how to identify the informative pair examples for efficient kernel learning. A promising algorithm is presented that approximates the original min-max optimization problem into a convex programming problem. Empirical evaluation based on the performance of data clustering showed that our proposed algorithm for active kernel learning is effective in identifying informative example pairs for the learning of kernel matrix.

Acknowledgments

The work was supported in part by the National Science Foundation (IIS-0643494), National Institute of Health (1R01-GM079688-01), and Singapore NTU AcRF Tier-1 Research Grant (RG67/07). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and NIH.

Appendix A: Proof of Theorem 3

Proof. First, for any global optimal solution to (6), we have $\sum_{(k,l)\notin\mathcal{T}} p_{k,l} = 1$ though the constraint in (6) is $\sum_{(k,l)\notin\mathcal{T}} p_{k,l} \geq 1$. This is because we can always scale down $p_{k,l}$ if $\sum_{(k,l)\notin\mathcal{T}} p_{k,l} > 1$, which guarantees to reduce the objective function. Second, any extreme point solution (i.e., $p_{k,l} = 1$ for one example pair and zero for other pairs) to (6) is a global optimal solution to (5). This is because (6) is a relaxed version of (5). Third, one of the global optimal solutions to (6) is an extreme point. This is because the first order condition of optimality requires $p^*_{k,l}$ to be a solution to the following problem:

$$\begin{aligned} \min_{p} \quad & \frac{c}{2} \sum_{(k,l)\notin\mathcal{T}} p_{k,l}[\varepsilon^*_{k,l}]^2 \qquad (11) \\ \text{s. t.} \quad & \sum_{(k,l)\notin\mathcal{T}} p_{k,l} \geq 1,\ p_{k,l} \geq 0, \forall (k,l) \notin \mathcal{T} \end{aligned}$$

where $\varepsilon^*_{k,l}$ is the optimal solution for $\varepsilon_{k,l}$. Since (11) is a linear optimization problem, it is well known that one of its global optimal solutions is an extreme point. Combining the above arguments together, we prove there exists a global solution to (5), denoted by $((k,l)^*, Z^*, \varepsilon^*_{i,j})$ that is also a global solution to (6) with $p_{(k,l)^*} = 1$. We extend this conclusion to any other global solution $((k,l)', Z', \varepsilon'_{i,j})$ to (5) because $((k,l)', Z', \varepsilon'_{i,j})$ results in the same value for the problem in (6) as solution $((k,l)^*, Z^*, \varepsilon^*_{i,j})$. This completes our proof. □

Appendix B: Proof of Theorem 4

Proof. To show (8) is a SDP problem, we introduce slack variables for both labeled and unlabeled example pairs, i.e., $\eta_{i,j} \geq \varepsilon^2_{i,j}$ and $\eta_{k,l} \geq \varepsilon^2_{k,l}/h_{k,l}$. We can turn these two nonlinear constraints into LMI constraints, i.e.,

$$\begin{pmatrix} \eta_{i,j} & \varepsilon_{i,j} \\ \varepsilon_{i,j} & 1 \end{pmatrix} \succeq 0, \quad \begin{pmatrix} \eta_{k,l} & \varepsilon_{k,l} \\ \varepsilon_{k,l} & h_{k,l} \end{pmatrix} \succeq 0$$

Using the slack variables, we rewrite (8) as

$$\begin{aligned} \min_{Z\succeq 0, h, \varepsilon} \quad & \mathrm{tr}(LZ) + \frac{c}{2} \sum_{(i,j)\in\mathcal{T}} \eta_{i,j} + \frac{c}{2} \sum_{(k,l)\notin\mathcal{T}} \eta_{k,l} \qquad (12) \\ \text{s. t.} \quad & T_{i,j} Z_{i,j} \geq 1 - \varepsilon_{i,j}, \varepsilon_{i,j} \geq 0,\ \forall (i,j) \in \mathcal{T} \\ & \varepsilon_{k,l} - 1 \geq Z_{k,l} \geq 1 - \varepsilon_{k,l}, \forall (k,l) \notin \mathcal{T} \\ & \sum_{(k,l)\notin\mathcal{T}} h_{k,l} \leq m^2,\ h_{k,l} \geq 1, \forall (k,l) \notin \mathcal{T} \\ & \begin{pmatrix} \eta_{i,j} & \varepsilon_{i,j} \\ \varepsilon_{i,j} & 1 \end{pmatrix} \succeq 0, \forall (i,j) \in \mathcal{T} \\ & \begin{pmatrix} \eta_{k,l} & \varepsilon_{k,l} \\ \varepsilon_{k,l} & h_{k,l} \end{pmatrix} \succeq 0, \forall (k,l) \notin \mathcal{T}, \end{aligned}$$

which is clearly a SDP problem.

To show the second part of theorem, we follow the inequality that a harmonic mean is upper bounded by an arithmetic mean, i.e.,

$$\frac{1}{m} \sum_{(k,l)\notin\mathcal{T}} p_{k,l} \geq \frac{m}{\sum_{(k,l)\notin\mathcal{T}} p^{-1}_{k,l}} = \frac{m}{\sum_{(k,l)\notin\mathcal{T}} h_{k,l}} \geq \frac{1}{m}$$

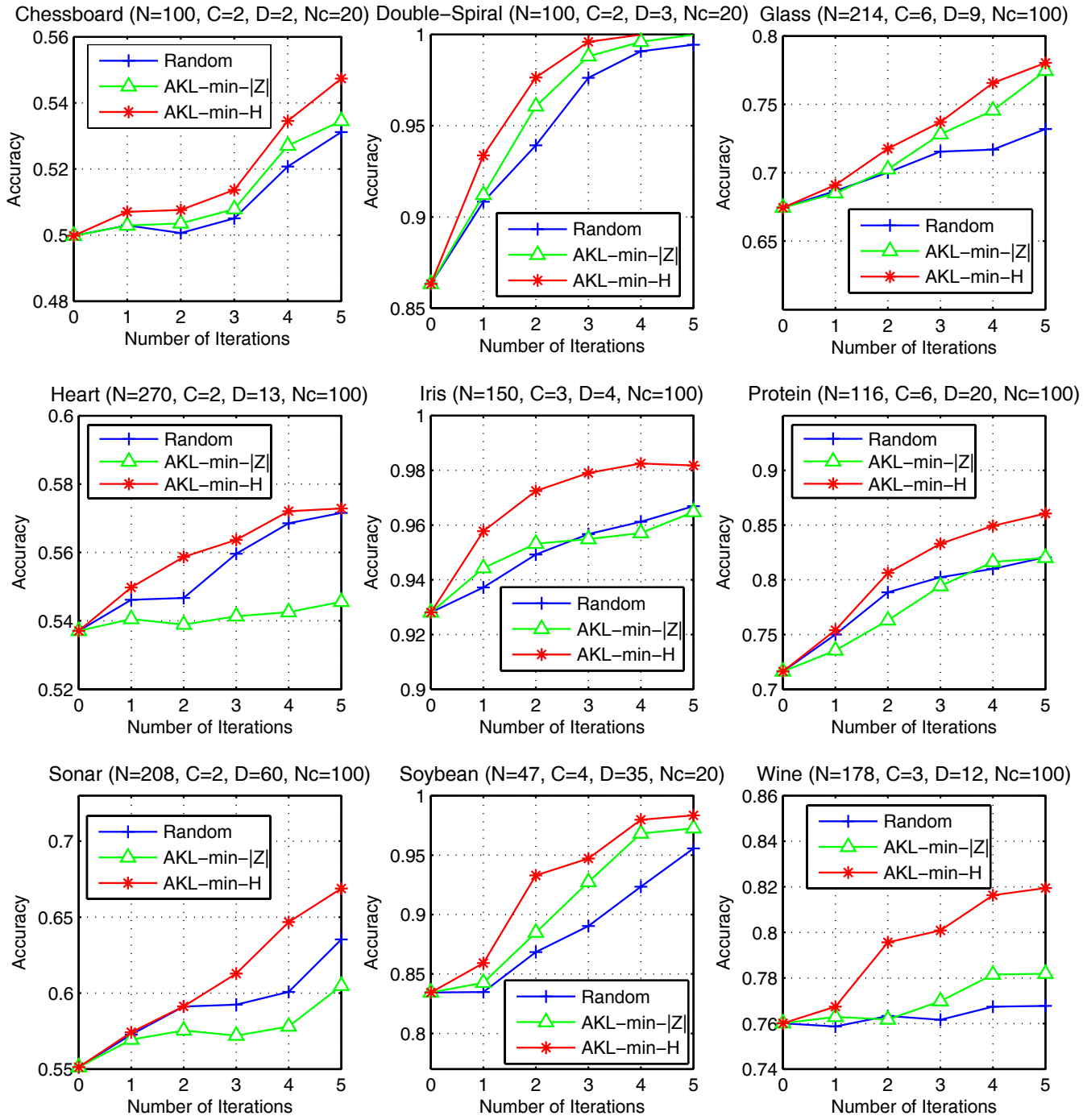


Figure 2. The clustering accuracy of different AKL methods for kernel k-means algorithms with nonparametric kernels learned from pairwise constraints. In each individual diagram, the three curves are respectively the random sampling method, the active kernel learning method for selecting pair examples with the least $|Z_{k,l}|$ (AKL-min-$|Z|$), and the active kernel learning method with minimal H values learned from our proposed algorithm (AKL-min-H). The details of the datasets are also shown in each diagram. In particular, N, C, D, and Nc respectively denote the dataset size, the number of classes, the number of features, and the number of initially sampling pairwise constraints. In each of the five iterations, 20 pair examples are sampled for labeling by the compared algorithms.

Hence, any feasible solution to (8) is also a feasible solution to (6), and (8) is a restricted version of (8), which leads to the conclusion that the optimal output value for (8) provides the upper bound for that of (6). □

Appendix C: Proof of Theorem 5

Proof. We first constructe the Lagrangian function for the above problem

$$\begin{aligned}\mathcal{L} = \mathrm{tr}(L^\top Z) &+ \frac{c}{2}\sum_{(i,j)\in\mathcal{T}}\eta_{i,j} + \frac{c}{2}\sum_{(k,l)\notin\mathcal{T}}\eta_{k,l} \\ &- \sum_{(i,j)\in\mathcal{T}} Q_{i,j}(T_{i,j}Z_{i,j} + \varepsilon_{i,j} - 1) \\ &- \sum_{(i,j)\in\mathcal{T}} (\alpha_{i,j}\eta_{i,j} + \tau_{i,j}/2 - 2\beta_{i,j}\varepsilon_{i,j}) - \mathrm{tr}(MZ) \\ &- \sum_{(k,l)\notin\mathcal{T}} s_{k,l}(h_{k,l} - 1) - \lambda\left(m^2 - \sum_{(k,l)\notin\mathcal{T}} h_{k,l}\right) \\ &- \sum_{(k,l)\notin\mathcal{T}} (\alpha_{k,l}\eta_{k,l} + \tau_{k,l}h_{k,l}/2 - 2\beta_{k,l}\varepsilon_{k,l}) \\ &- \sum_{(k,l)\notin\mathcal{T}} W_{k,l}Z_{k,l} + (\varepsilon_{k,l} - 1)|W_{k,l}|\end{aligned}$$

In the above, we introduce Lagrangian multiplier

$$\begin{pmatrix} \alpha_{i,j} & -\beta_{i,j} \\ -\beta_{i,j} & \tau_{i,j}/2 \end{pmatrix}$$

for constraints

$$\begin{pmatrix} \eta_{i,j} & \varepsilon_{i,j} \\ \varepsilon_{i,j} & 1 \end{pmatrix} \succeq 0 \text{ and } \begin{pmatrix} \eta_{k,l} & \varepsilon_{k,l} \\ \varepsilon_{k,l} & h_{k,l} \end{pmatrix} \succeq 0$$

By setting the derivative to be zero, we have

$$\begin{aligned}\max \quad & \sum_{(i,j)\in\mathcal{T}} \left(Q_{i,j} - \frac{\tau_{i,j}}{2}\right) + \sum_{(k,l)\notin\mathcal{T}} \left(|W_{k,l}| - \frac{\tau_{k,l}}{2}\right) \quad (13) \\ & -(m^2-1)\lambda \\ \text{s. t} \quad & L \succeq Q\otimes T + W\otimes\bar{T} \\ & \begin{pmatrix} c & -Q_{i,j} \\ -Q_{i,j} & \tau_{i,j} \end{pmatrix} \succeq 0, Q_{i,j} \geq 0,\ \forall (i,j)\in\mathcal{T} \\ & 0 \leq \tau_{k,l} \leq 2\lambda,\ \forall (k,l)\notin\mathcal{T} \\ & \begin{pmatrix} c & -|W_{k,l}| \\ -|W_{k,l}| & \tau_{k,l} \end{pmatrix} \succeq 0,\ \forall (k,l)\notin\mathcal{T}\end{aligned}$$

The two LMI constraints can be simplified as

$$\tau_{i,j} \geq 2Q_{i,j}^2/c, \quad \tau_{k,l} \geq 2Q_{k,l}^2/c$$

Substituting the above constraints into (13), we have (9). □

References

Belkin, M., & andd P. Niyogi, I. M. (2004). Regularization and semi-supervised learning on large graphs. *Intl. Conf. on Learning Theory (COLT)*.

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. .

Chen, F., & Jin, R. (2007). Active algorithm selection. *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI)*.

Cristianini, N., Shawe-Taylor, J., & Elisseeff, A. (2002). On kernel-target alignment. *JMLR*.

Hoi, S. C. H., Jin, R., & Lyu, M. R. (2007). Learning nonparametric kernel matrices from pairwise constraints. *ICML2007*. Corvallis, OR, US.

Hoi, S. C. H., Jin, R., Zhu, J., & Lyu, M. R. (2006). Batch mode active learning and its application to medical image classification. *ICML2006* (pp. 417–424). Pittsburgh, Pennsylvania.

Kondor, R., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. *ICML'2002*.

Kulis, B., Sustik, M., & Dhillon, I. S. (2006). Learning low-rank kernel matrices. *ICML2006* (pp. 505–512).

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. (2004). Learning the kernel matrix with semi-definite programming. *JMLR*, *5*, 27–72.

Scholkopf, B., & Smola, A. (2002). *Learning with kernels*. MIT Press.

Sturm, J. (1999). Using sedumi: a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, *11–12*, 625–653.

Tong, S., & Koller, D. (2000). Support vector machine active learning with applications to text classification. *ICML2000* (pp. 999–1006). Stanford, US.

Vapnik, V. N. (1998). *Statistical learning theory*. John Wiley & Sons.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2002). Distance metric learning with application to clustering with side-information. *NIPS2002*.

Zhu, X., Kandola, J., Ghahramani, Z., & Lafferty, J. (2005). Nonparametric transforms of graph kernels for semi-supervised learning. *NIPS2005*.

A Dual Coordinate Descent Method for Large-scale Linear SVM

Cho-Jui Hsieh B92085@CSIE.NTU.EDU.TW
Kai-Wei Chang B92084@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science, National Taiwan University, Taipei 106, Taiwan
S. Sathiya Keerthi SELVARAK@YAHOO-INC.COM
Yahoo! Research, Santa Clara, USA
S. Sundararajan SSRAJAN@YAHOO-INC.COM
Yahoo! Labs, Bangalore, India

Abstract

In many applications, data appear with a huge number of instances as well as features. Linear Support Vector Machines (SVM) is one of the most popular tools to deal with such large-scale sparse data. This paper presents a novel dual coordinate descent method for linear SVM with L1- and L2-loss functions. The proposed method is simple and reaches an ϵ-accurate solution in $O(\log(1/\epsilon))$ iterations. Experiments indicate that our method is much faster than state of the art solvers such as Pegasos, TRON, SVM$^{\text{perf}}$, and a recent primal coordinate descent implementation.

1. Introduction

Support vector machines (SVM) (Boser et al., 1992) are useful for data classification. Given a set of instance-label pairs $(\boldsymbol{x}_i, y_i), i = 1, \ldots, l,\ \boldsymbol{x}_i \in R^n,\ y_i \in \{-1, +1\}$, SVM requires the solution of the following unconstrained optimization problem:

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i), \tag{1}$$

where $\xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i)$ is a loss function, and $C \geq 0$ is a penalty parameter. Two common loss functions are:

$$\max(1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i, 0) \text{ and } \max(1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i, 0)^2. \tag{2}$$

The former is called L1-SVM, while the latter is L2-SVM. In some applications, an SVM problem appears with a bias term b. One often deal with this term by appending each instance with an additional dimension:

$$\boldsymbol{x}_i^T \leftarrow [\boldsymbol{x}_i^T, 1] \qquad \boldsymbol{w}^T \leftarrow [\boldsymbol{w}^T, b]. \tag{3}$$

Problem (1) is often referred to as the primal form of SVM. One may instead solve its dual problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T\bar{Q}\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq U, \forall i, \end{aligned} \tag{4}$$

where $\bar{Q} = Q + D$, D is a diagonal matrix, and $Q_{ij} = y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j$. For L1-SVM, $U = C$ and $D_{ii} = 0,\ \forall i$. For L2-SVM, $U = \infty$ and $D_{ii} = 1/(2C),\ \forall i$.

An SVM usually maps training vectors into a high-dimensional space via a nonlinear function $\phi(\boldsymbol{x})$. Due to the high dimensionality of the vector variable $\boldsymbol{w}$, one solves the dual problem (4) by the kernel trick (i.e., using a closed form of $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$). We call such a problem as a nonlinear SVM. In some applications, data appear in a rich dimensional feature space, the performances are similar with/without nonlinear mapping. If data are not mapped, we can often train much larger data sets. We indicate such cases as linear SVM; these are often encountered in applications such as document classification. In this paper, we aim at solving very large linear SVM problems.

Recently, many methods have been proposed for linear SVM in large-scale scenarios. For L1-SVM, Zhang (2004), Shalev-Shwartz et al. (2007), Bottou (2007) propose various stochastic gradient descent methods. Collins et al. (2008) apply an exponentiated gradient method. SVM$^{\text{perf}}$ (Joachims, 2006) uses a cutting plane technique. Smola et al. (2008) apply bundle methods, and view SVM$^{\text{perf}}$ as a special case. For L2-SVM, Keerthi and DeCoste (2005) propose modified Newton methods. A trust region Newton method (TRON) (Lin et al., 2008) is proposed for logistic re-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

gression and L2-SVM. These algorithms focus on different aspects of the training speed. Some aim at quickly obtaining a usable model, but some achieve fast final convergence of solving the optimization problem in (1) or (4). Moreover, among these methods, Joachims (2006), Smola et al. (2008) and Collins et al. (2008) solve SVM via the dual (4). Others consider the primal form (1). The decision of using primal or dual is of course related to the algorithm design.

Very recently, Chang et al. (2007) propose using coordinate descent methods for solving primal L2-SVM. Experiments show that their approach more quickly obtains a useful model than some of the above methods. Coordinate descent, a popular optimization technique, updates one variable at a time by minimizing a single-variable sub-problem. If one can efficiently solve this sub-problem, then it can be a competitive optimization method. Due to the non-differentiability of the primal L1-SVM, Chang et al's work is restricted to L2-SVM. Moreover, as primal L2-SVM is differentiable but not twice differentiable, certain considerations are needed in solving the single-variable sub-problem.

While the dual form (4) involves bound constraints $0 \le \alpha_i \le U$, its objective function is twice differentiable for both L1- and L2-SVM. In this paper, we investigate coordinate descent methods for the dual problem (4). We prove that an ϵ-optimal solution is obtained in $O(\log(1/\epsilon))$ iterations. We propose an implementation using a random order of sub-problems at each iteration, which leads to very fast training. Experiments indicate that our method is more efficient than the primal coordinate descent method. As Chang et al. (2007) solve the primal, they require the easy access of a feature's corresponding data values. However, in practice one often has an easier access of values per instance. Solving the dual takes this advantage, so our implementation is simpler than Chang et al. (2007).

Early SVM papers (Mangasarian & Musicant, 1999; Friess et al., 1998) have discussed coordinate descent methods for the SVM dual form. However, they do not focus on large data using the linear kernel. Crammer and Singer (2003) proposed an online setting for multi-class SVM without considering large sparse data. Recently, Bordes et al. (2007) applied a coordinate descent method to multi-class SVM, but they focus on nonlinear kernels. In this paper, we point out that *dual coordinate descent methods make crucial advantage of the linear kernel and outperform other solvers when the numbers of data and features are both large.*

Coordinate descent methods for (4) are related to the popular decomposition methods for training nonlinear SVM. In this paper, we show their key differences and explain why earlier studies on decomposition methods failed to modify their algorithms in an efficient way like ours for large-scale linear SVM. We also discuss the connection to other linear SVM works such as (Crammer & Singer, 2003; Collins et al., 2008; Shalev-Shwartz et al., 2007).

This paper is organized as follows. In Section 2, we describe our proposed algorithm. Implementation issues are investigated in Section 3. Section 4 discusses the connection to other methods. In Section 5, we compare our method with state of the art implementations for large linear SVM. Results show that the new method is more efficient. Proofs can be found at `http://www.csie.ntu.edu.tw/~cjlin/papers/cddual.pdf`.

2. A Dual Coordinate Descent Method

In this section, we describe our coordinate descent method for L1- and L2-SVM. The optimization process starts from an initial point $\boldsymbol{\alpha}^0 \in R^l$ and generates a sequence of vectors $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$. We refer to the process from $\boldsymbol{\alpha}^k$ to $\boldsymbol{\alpha}^{k+1}$ as an outer iteration. In each outer iteration we have l inner iterations, so that sequentially $\alpha_1, \alpha_2, \ldots, \alpha_l$ are updated. Each outer iteration thus generates vectors $\boldsymbol{\alpha}^{k,i} \in R^l$, $i = 1, \ldots, l+1$, such that $\boldsymbol{\alpha}^{k,1} = \boldsymbol{\alpha}^k$, $\boldsymbol{\alpha}^{k,l+1} = \boldsymbol{\alpha}^{k+1}$, and

$$\boldsymbol{\alpha}^{k,i} = [\alpha_1^{k+1}, \ldots, \alpha_{i-1}^{k+1}, \alpha_i^k, \ldots, \alpha_l^k]^T, \ \forall i = 2, \ldots, l.$$

For updating $\boldsymbol{\alpha}^{k,i}$ to $\boldsymbol{\alpha}^{k,i+1}$, we solve the following one-variable sub-problem:

$$\min_d \ f(\boldsymbol{\alpha}^{k,i} + d\boldsymbol{e}_i) \quad \text{subject to} \quad 0 \le \alpha_i^k + d \le U, \quad (5)$$

where $\boldsymbol{e}_i = [0, \ldots, 0, 1, 0, \ldots, 0]^T$. The objective function of (5) is a simple quadratic function of d:

$$f(\boldsymbol{\alpha}^{k,i} + d\boldsymbol{e}_i) = \frac{1}{2}\bar{Q}_{ii}d^2 + \nabla_i f(\boldsymbol{\alpha}^{k,i})d + \text{constant}, \quad (6)$$

where $\nabla_i f$ is the ith component of the gradient ∇f. One can easily see that (5) has an optimum at $d = 0$ (i.e., no need to update α_i) if and only if

$$\nabla_i^P f(\boldsymbol{\alpha}^{k,i}) = 0, \quad (7)$$

where $\nabla^P f(\boldsymbol{\alpha})$ means the projected gradient

$$\nabla_i^P f(\boldsymbol{\alpha}) = \begin{cases} \nabla_i f(\boldsymbol{\alpha}) & \text{if } 0 < \alpha_i < U, \\ \min(0, \nabla_i f(\boldsymbol{\alpha})) & \text{if } \alpha_i = 0, \\ \max(0, \nabla_i f(\boldsymbol{\alpha})) & \text{if } \alpha_i = U. \end{cases} \quad (8)$$

If (7) holds, we move to the index $i+1$ without updating $\alpha_i^{k,i}$. Otherwise, we must find the optimal solution of (5). If $\bar{Q}_{ii} > 0$, easily the solution is:

$$\alpha_i^{k,i+1} = \min\left(\max\left(\alpha_i^{k,i} - \frac{\nabla_i f(\boldsymbol{\alpha}^{k,i})}{\bar{Q}_{ii}}, 0\right), U\right). \quad (9)$$

Algorithm 1 A dual coordinate descent method for Linear SVM

- Given $\boldsymbol{\alpha}$ and the corresponding $\boldsymbol{w} = \sum_i y_i \alpha_i \boldsymbol{x}_i$.
- While $\boldsymbol{\alpha}$ is not optimal

  For $i = 1, \ldots, l$

  (a) $\bar{\alpha}_i \leftarrow \alpha_i$

  (b) $G = y_i \boldsymbol{w}^T \boldsymbol{x}_i - 1 + D_{ii}\alpha_i$

  (c)
  $$PG = \begin{cases} \min(G, 0) & \text{if } \alpha_i = 0, \\ \max(G, 0) & \text{if } \alpha_i = U, \\ G & \text{if } 0 < \alpha_i < U \end{cases}$$

  (d) If $|PG| \neq 0$,
  $\alpha_i \leftarrow \min(\max(\alpha_i - G/\bar{Q}_{ii}, 0), U)$
  $\boldsymbol{w} \leftarrow \boldsymbol{w} + (\alpha_i - \bar{\alpha}_i) y_i \boldsymbol{x}_i$

We thus need to calculate $\bar{Q}_{ii}$ and $\nabla_i f(\boldsymbol{\alpha}^{k,i})$. First, $\bar{Q}_{ii} = \boldsymbol{x}_i^T \boldsymbol{x}_i + D_{ii}$ can be precomputed and stored in the memory. Second, to evaluate $\nabla_i f(\boldsymbol{\alpha}^{k,i})$, we use

$$\nabla_i f(\boldsymbol{\alpha}) = (\bar{Q}\boldsymbol{\alpha})_i - 1 = \sum_{j=1}^{l} \bar{Q}_{ij}\alpha_j - 1. \qquad (10)$$

$\bar{Q}$ may be too large to be stored, so one calculates $\bar{Q}$'s ith row when doing (10). If $\bar{n}$ is the average number of nonzero elements per instance, and $O(\bar{n})$ is needed for each kernel evaluation, then calculating the ith row of the kernel matrix takes $O(l\bar{n})$. Such operations are expensive. However, for a linear SVM, we can define

$$\boldsymbol{w} = \sum_{j=1}^{l} y_j \alpha_j \boldsymbol{x}_j, \qquad (11)$$

so (10) becomes

$$\nabla_i f(\boldsymbol{\alpha}) = y_i \boldsymbol{w}^T \boldsymbol{x}_i - 1 + D_{ii}\alpha_i. \qquad (12)$$

To evaluate (12), the main cost is $O(\bar{n})$ for calculating $\boldsymbol{w}^T \boldsymbol{x}_i$. This is much smaller than $O(l\bar{n})$. To apply (12), $\boldsymbol{w}$ must be maintained throughout the coordinate descent procedure. Calculating $\boldsymbol{w}$ by (11) takes $O(l\bar{n})$ operations, which are too expensive. Fortunately, if $\bar{\alpha}_i$ is the current value and α_i is the value after the updating, we can maintain $\boldsymbol{w}$ by

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + (\alpha_i - \bar{\alpha}_i) y_i \boldsymbol{x}_i. \qquad (13)$$

The number of operations is only $O(\bar{n})$. To have the first $\boldsymbol{w}$, one can use $\boldsymbol{\alpha}^0 = \boldsymbol{0}$ so $\boldsymbol{w} = \boldsymbol{0}$. In the end, we obtain the optimal $\boldsymbol{w}$ of the primal problem (1) as the primal-dual relationship implies (11).

If $\bar{Q}_{ii} = 0$, we have $D_{ii} = 0$, $Q_{ii} = \boldsymbol{x}_i^T \boldsymbol{x}_i = 0$, and hence $\boldsymbol{x}_i = \boldsymbol{0}$. This occurs only in L1-SVM without the bias term by (3). From (12), if $\boldsymbol{x}_i = \boldsymbol{0}$, then $\nabla_i f(\boldsymbol{\alpha}^{k,i}) = -1$. As $U = C < \infty$ for L1-SVM, the solution of (5) makes the new $\alpha_i^{k,i+1} = U$. We can easily include this case in (9) by setting $1/\bar{Q}_{ii} = \infty$.

Briefly, our algorithm uses (12) to compute $\nabla_i f(\boldsymbol{\alpha}^{k,i})$, checks the optimality of the sub-problem (5) by (7), updates α_i by (9), and then maintains $\boldsymbol{w}$ by (13). A description is in Algorithm 1. The cost per iteration (i.e., from $\boldsymbol{\alpha}^k$ to $\boldsymbol{\alpha}^{k+1}$) is $O(l\bar{n})$. The main memory requirement is on storing $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l$. For the convergence, we prove the following theorem using techniques in (Luo & Tseng, 1992):

Theorem 1 *For L1-SVM and L2-SVM, $\{\boldsymbol{\alpha}^{k,i}\}$ generated by Algorithm 1 globally converges to an optimal solution $\boldsymbol{\alpha}^*$. The convergence rate is at least linear: there are $0 < \mu < 1$ and an iteration k_0 such that*

$$f(\boldsymbol{\alpha}^{k+1}) - f(\boldsymbol{\alpha}^*) \leq \mu(f(\boldsymbol{\alpha}^k) - f(\boldsymbol{\alpha}^*)), \forall k \geq k_0. \quad (14)$$

The global convergence result is quite remarkable. Usually for a convex but not strictly convex problem (e.g., L1-SVM), one can only obtain that any limit point is optimal. We define an ϵ-accurate solution $\boldsymbol{\alpha}$ if $f(\boldsymbol{\alpha}) \leq f(\boldsymbol{\alpha}^*) + \epsilon$. By (14), our algorithm obtains an ϵ-accurate solution in $O(\log(1/\epsilon))$ iterations.

3. Implementation Issues

3.1. Random Permutation of Sub-problems

In Algorithm 1, the coordinate descent algorithm solves the one-variable sub-problems in the order of $\alpha_1, \ldots, \alpha_l$. Past results such as (Chang et al., 2007) show that solving sub-problems in an arbitrary order may give faster convergence. This inspires us to randomly permute the sub-problems at each outer iteration. Formally, at the kth outer iteration, we permute $\{1, \ldots, l\}$ to $\{\pi(1), \ldots, \pi(l)\}$, and solve sub-problems in the order of $\alpha_{\pi(1)}, \alpha_{\pi(2)}, \ldots, \alpha_{\pi(l)}$. Similar to Algorithm 1, the algorithm generates a sequence $\{\boldsymbol{\alpha}^{k,i}\}$ such that $\boldsymbol{\alpha}^{k,1} = \boldsymbol{\alpha}^k$, $\boldsymbol{\alpha}^{k,l+1} = \boldsymbol{\alpha}^{k+1,1}$ and

$$\alpha_t^{k,i} = \begin{cases} \alpha_t^{k+1} & \text{if } \pi_k^{-1}(t) < i, \\ \alpha_t^k & \text{if } \pi_k^{-1}(t) \geq i. \end{cases}$$

The update from $\boldsymbol{\alpha}^{k,i}$ to $\boldsymbol{\alpha}^{k,i+1}$ is by

$$\alpha_t^{k,i+1} = \alpha_t^{k,i} + \arg\min_{0 \leq \alpha_t^{k,i} + d \leq U} f(\boldsymbol{\alpha}^{k,i} + d\boldsymbol{e}_t) \text{ if } \pi_k^{-1}(t) = i.$$

We prove that Theorem 1 is still valid. Hence, the new setting obtains an ϵ-accurate solution in $O(\log(1/\epsilon))$ iterations. A simple experiment reveals that this setting of permuting sub-problems is much faster than Algorithm 1. The improvement is also bigger than that observed in (Chang et al., 2007) for primal coordinate descent methods.

Algorithm 2 Coordinate descent algorithm with randomly selecting one instance at a time

- Given $\boldsymbol{\alpha}$ and the corresponding $\boldsymbol{w} = \sum_i y_i \alpha_i \boldsymbol{x}_i$.
- While $\boldsymbol{\alpha}$ is not optimal
  - Randomly choose $i \in \{1, \ldots, l\}$.
  - Do steps (a)-(d) of Algorithm 1 to update α_i.

3.2. Shrinking

Eq. (4) contains constraints $0 \le \alpha_i \le U$. If an α_i is 0 or U for many iterations, it may remain the same. To speed up decomposition methods for nonlinear SVM (discussed in Section 4.1), the shrinking technique (Joachims, 1998) reduces the size of the optimization problem without considering some bounded variables. Below we show it is much easier to apply this technique to linear SVM than the nonlinear case.

If A is the subset after removing some elements and $\bar{A} = \{1, \ldots, l\} \setminus A$, then the new problem is

$$\begin{aligned} \min_{\boldsymbol{\alpha}_A} \quad & \frac{1}{2}\boldsymbol{\alpha}_A^T \bar{Q}_{AA}\boldsymbol{\alpha}_A + (\bar{Q}_{A\bar{A}}\boldsymbol{\alpha}_{\bar{A}} - \boldsymbol{e}_A)^T\boldsymbol{\alpha}_A \\ \text{subject to} \quad & 0 \le \alpha_i \le U, i \in A, \end{aligned} \tag{15}$$

where $\bar{Q}_{AA}, \bar{Q}_{A\bar{A}}$ are sub-matrices of $\bar{Q}$, and $\boldsymbol{\alpha}_{\bar{A}}$ is considered as a constant vector. Solving this smaller problem consumes less time and memory. Once (15) is solved, we must check if the vector $\boldsymbol{\alpha}$ is optimal for (4). This check needs the whole gradient $\nabla f(\boldsymbol{\alpha})$. Since

$$\nabla_i f(\boldsymbol{\alpha}) = \bar{Q}_{i,A}\boldsymbol{\alpha}_A + \bar{Q}_{i,\bar{A}}\boldsymbol{\alpha}_{\bar{A}} - 1,$$

if $i \in A$, and one stores $\bar{Q}_{i,\bar{A}}\boldsymbol{\alpha}_{\bar{A}}$ before solving (15), we already have $\nabla_i f(\boldsymbol{\alpha})$. However, for all $i \notin A$, we must calculate the corresponding rows of $\bar{Q}$. This step, referred to as the reconstruction of gradients in training nonlinear SVM, is very time consuming. It may cost up to $O(l^2\bar{n})$ if each kernel evaluation is $O(\bar{n})$.

For linear SVM, in solving the smaller problem (15), we still have the vector

$$\boldsymbol{w} = \sum\nolimits_{i\in A} y_i\alpha_i\boldsymbol{x}_i + \sum\nolimits_{i\in\bar{A}} y_i\alpha_i\boldsymbol{x}_i$$

though only the first part $\sum_{i\in A} y_i\alpha_i\boldsymbol{x}_i$ is updated. Therefore, using (12), $\nabla f(\boldsymbol{\alpha})$ is easily available. Below we demonstrate a shrinking implementation so that reconstructing the whole $\nabla f(\boldsymbol{\alpha})$ is never needed.

Our method is related to what LIBSVM (Chang & Lin, 2001) uses. From the optimality condition of bound-constrained problems, $\boldsymbol{\alpha}$ is optimal for (4) if and only if $\nabla^P f(\boldsymbol{\alpha}) = \boldsymbol{0}$, where $\nabla^P f(\boldsymbol{\alpha})$ is the projected gradient defined in (8). We then prove the following result:

Theorem 2 *Let $\boldsymbol{\alpha}^*$ be the convergent point of $\{\boldsymbol{\alpha}^{k,i}\}$.*

1. *If $\alpha_i^* = 0$ and $\nabla_i f(\boldsymbol{\alpha}^*) > 0$, then $\exists k_i$ such that $\forall k \ge k_i,\ \forall s, \alpha_i^{k,s} = 0$.*
2. *If $\alpha_i^* = U$ and $\nabla_i f(\boldsymbol{\alpha}^*) < 0$, then $\exists k_i$ such that $\forall k \ge k_i,\ \forall s,\ \alpha_i^{k,s} = U$.*
3. $\lim_{k\to\infty} \max_j \nabla_j^P f(\boldsymbol{\alpha}^{k,j}) = \lim_{k\to\infty} \min_j \nabla_j^P f(\boldsymbol{\alpha}^{k,j}) = 0$.

During the optimization procedure, $\nabla^P f(\boldsymbol{\alpha}^k) \ne \boldsymbol{0}$, so in general $\max_j \nabla_j^P f(\boldsymbol{\alpha}^k) > 0$ and $\min_j \nabla_j^P f(\boldsymbol{\alpha}^k) < 0$. These two values measure how the current solution violates the optimality condition. In our iterative procedure, what we have are $\nabla_i f(\boldsymbol{\alpha}^{k,i}),\ i = 1, \ldots, l$. Hence, at the $(k-1)$st iteration, we obtain

$$M^{k-1} \equiv \max_j \nabla_j^P f(\boldsymbol{\alpha}^{k-1,j}), m^{k-1} \equiv \min_j \nabla_j^P f(\boldsymbol{\alpha}^{k-1,j}).$$

Then at each inner step of the kth iteration, before updating $\alpha_i^{k,i}$ to $\alpha_i^{k,i+1}$, this element is shrunken if one of the following two conditions holds:

$$\begin{aligned} &\alpha_i^{k,i} = 0 \text{ and } \nabla_i f(\boldsymbol{\alpha}^{k,i}) > \bar{M}^{k-1}, \\ &\alpha_i^{k,i} = U \text{ and } \nabla_i f(\boldsymbol{\alpha}^{k,i}) < \bar{m}^{k-1}, \end{aligned} \tag{16}$$

where

$$\bar{M}^{k-1} = \begin{cases} M^{k-1} & \text{if } M^{k-1} > 0, \\ \infty & \text{otherwise,} \end{cases}$$

$$\bar{m}^{k-1} = \begin{cases} m^{k-1} & \text{if } m^{k-1} < 0 \\ -\infty & \text{otherwise.} \end{cases}$$

In (16), $\bar{M}^{k-1}$ must be strictly positive, so we set it be ∞ if $M^{k-1} = 0$. From Theorem 2, elements satisfying the "if condition" of properties 1 and 2 meet (16) after certain iterations, and are then correctly removed for optimization. To have a more aggressive shrinking, one may multiply both $\bar{M}^{k-1}$ and $\bar{m}^{k-1}$ in (16) by a threshold smaller than one.

Property 3 of Theorem 2 indicates that with a tolerance ϵ,

$$M^k - m^k < \epsilon \tag{17}$$

is satisfied after a finite number of iterations. Hence (17) is a valid stopping condition. We also use it for smaller problems (15). If at the kth iteration, (17) for (15) is reached, we enlarge A to $\{1, \ldots, l\}$, set $\bar{M}^k = \infty, \bar{m}^k = -\infty$ (so no shrinking at the $(k+1)$st iteration), and continue regular iterations. Thus, we do shrinking without reconstructing gradients.

3.3. An Online Setting

In some applications, the number of instances is huge, so going over all $\alpha_1, \ldots, \alpha_l$ causes an expensive outer iteration. Instead, one can randomly choose an index i_k at a time, and update only α_{i_k} at the kth outer iteration. A description is in Algorithm 2. The setting is related to (Crammer & Singer, 2003; Collins et al., 2008). See also the discussion in Section 4.2.

Table 1. A comparison between decomposition methods (Decomp.) and dual coordinate descent (DCD). For both methods, we consider that one α_i is updated at a time. We assume Decomp. maintains gradients, but DCD does not. The average number of nonzeros per instance is $\bar{n}$.

	Nonlinear SVM		Linear SVM	
	Decomp.	DCD	Decomp.	DCD
Update α_i	$O(1)$	$O(l\bar{n})$	$O(1)$	$O(\bar{n})$
Maintain $\nabla f(\boldsymbol{\alpha})$	$O(l\bar{n})$	NA	$O(l\bar{n})$	NA

4. Relations with Other Methods

4.1. Decomposition Methods for Nonlinear SVM

Decomposition methods are one of the most popular approaches for training nonlinear SVM. As the kernel matrix is dense and cannot be stored in the computer memory, decomposition methods solve a sub-problem of few variables at each iteration. Only a small number of corresponding kernel columns are needed, so the memory problem is resolved. If the number of variables is restricted to one, a decomposition method is like the online coordinate descent in Section 3.3, but it differs in the way it selects variables for updating. It has been shown (Keerthi & DeCoste, 2005) that, for linear SVM decomposition methods are inefficient. On the other hand, here we are pointing out that dual coordinate descent is efficient for linear SVM. Therefore, it is important to discuss the relationship between decomposition methods and our method.

In early decomposition methods that were first proposed (Osuna et al., 1997; Platt, 1998), variables minimized at an iteration are selected by certain heuristics. However, subsequent developments (Joachims, 1998; Chang & Lin, 2001; Keerthi et al., 2001) all use gradient information to conduct the selection. The main reason is that maintaining the whole gradient does not introduce extra cost. Here we explain the detail by assuming that one variable of $\boldsymbol{\alpha}$ is chosen and updated at a time[1]. To set-up and solve the sub-problem (6), one uses (10) to calculate $\nabla_i f(\boldsymbol{\alpha})$. If $O(\bar{n})$ effort is needed for each kernel evaluation, obtaining the ith row of the kernel matrix takes $O(l\bar{n})$ effort. If instead one maintains the whole gradient, then $\nabla_i f(\boldsymbol{\alpha})$ is directly available. After updating $\alpha_i^{k,i}$ to $\alpha_i^{k,i+1}$, we obtain $\bar{Q}$'s ith column (same as the ith row due to the symmetry of $\bar{Q}$), and calculate the new whole gradient:

$$\nabla f(\boldsymbol{\alpha}^{k,i+1}) = \nabla f(\boldsymbol{\alpha}^{k,i}) + \bar{Q}_{:,i}(\alpha_i^{k,i+1} - \alpha_i^{k,i}), \quad (18)$$

where $\bar{Q}_{:,i}$ is the ith column of $\bar{Q}$. The cost is $O(l\bar{n})$ for $\bar{Q}_{:,i}$ and $O(l)$ for (18). Therefore, maintaining the whole gradient does not cost more. As using the whole gradient implies fewer iterations (i.e., faster convergence due to the ability to choose for updating the variable that violates optimality most), one should take this advantage. However, the situation for linear SVM is very different. With the different way (12) to calculate $\nabla_i f(\boldsymbol{\alpha})$, the cost to update one α_i is only $O(\bar{n})$. If we still maintain the whole gradient, evaluating (12) l times takes $O(l\bar{n})$ effort. We gather this comparison of different situations in Table 1. Clearly, for nonlinear SVM, one should use decomposition methods by maintaining the whole gradient. However, for linear SVM, if l is large, the cost per iteration without maintaining gradients is much smaller than that with. Hence, the coordinate descent method can be faster than the decomposition method by using many cheap iterations.

An earlier attempt to speed up decomposition methods for linear SVM is (Kao et al., 2004). However, it failed to derive our method here because it does not give up maintaining gradients.

[1]Solvers like LIBSVM update at least two variables due to a linear constraint in their dual problems. Here (4) has no such a constraint, so selecting one variable is possible.

4.2. Existing Linear SVM Methods

We discussed in Section 1 and other places the difference between our method and a primal coordinate descent method (Chang et al., 2007). Below we describe the relations with other linear SVM methods.

We mentioned in Section 3.3 that our Algorithm 2 is related to the online mode in (Collins et al., 2008). They aim at solving multi-class and structured problems. At each iteration an instance is used; then a sub-problem of several variables is solved. They approximately minimize the sub-problem, but for two-class case, one can exactly solve it by (9). For the batch setting, our approach is different from theirs. The algorithm for multi-class problems in (Crammer & Singer, 2003) is also similar to our online setting. For the two-class case, it solves (1) with the loss function $\max(-y_i \boldsymbol{w}^T \boldsymbol{x}_i, 0)$, which is different from (2). They do not study data with a large number of features.

Next, we discuss the connection to stochastic gradient descent (Shalev-Shwartz et al., 2007; Bottou, 2007). The most important step of this method is the following update of $\boldsymbol{w}$:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}}(y_i, \boldsymbol{x}_i), \quad (19)$$

where $\nabla_{\boldsymbol{w}}(y_i, \boldsymbol{x}_i)$ is the sub-gradient of the approximate objective function:

$$\boldsymbol{w}^T \boldsymbol{w}/2 + C \max(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i, 0),$$

and η is the learning rate (or the step size). While our method is dual-based, throughout the iterations we

Table 2. On the right training time for a solver to reduce the primal objective value to within 1% of the optimal value; see (20). Time is in seconds. The method with the shortest running time is boldfaced. Listed on the left are the statistics of data sets: l is the number of instances and n is the number of features.

Data set	Data statistics			Linear L1-SVM			Linear L2-SVM		
	l	n	# nonzeros	DCDL1	Pegasos	SVM$^{\text{perf}}$	DCDL2	PCD	TRON
a9a	32,561	123	451,592	**0.2**	1.1	6.0	0.4	**0.1**	0.1
astro-physic	62,369	99,757	4,834,550	**0.2**	2.8	2.6	**0.2**	0.5	1.2
real-sim	72,309	20,958	3,709,083	**0.2**	2.4	2.4	**0.1**	0.2	0.9
news20	19,996	1,355,191	9,097,916	**0.5**	10.3	20.0	**0.2**	2.4	5.2
yahoo-japan	176,203	832,026	23,506,415	**1.1**	12.7	69.4	**1.0**	2.9	38.2
rcv1	677,399	47,236	49,556,258	**2.6**	21.9	72.0	**2.7**	5.1	18.6
yahoo-korea	460,554	3,052,939	156,436,656	**8.3**	79.7	656.8	**7.1**	18.4	286.1

maintain $\boldsymbol{w}$ by (13). Both (13) and (19) use one single instance $\boldsymbol{x}_i$, but they take different directions $y_i\boldsymbol{x}_i$ and $\nabla_{\boldsymbol{w}}(y_i, \boldsymbol{x}_i)$. The selection of the learning rate η may be the subtlest thing in stochastic gradient descent, but for our method this is never a concern. The step size $(\alpha_i - \bar{\alpha}_i)$ in (13) is governed by solving a sub-problem from the dual.

5. Experiments

In this section, we analyze the performance of our dual coordinate descent algorithm for L1- and L2-SVM. We compare our implementation with state of the art linear SVM solvers. We also investigate how the shrinking technique improves our algorithm.

Table 2 lists the statistics of data sets. Four of them (a9a, real-sim, news20, rcv1) are at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`. The set astro-physic is available upon request from Thorsten Joachims. Except a9a, all others are from document classification. Past results show that linear SVM performs as well as kernelized ones for such data. To estimate the testing accuracy, we use a stratified selection to split each set to 4/5 training and 1/5 testing. We briefly describe each set below. Details can be found in (Joachims, 2006) (astro-physic) and (Lin et al., 2008) (others). a9a is from the UCI "adult" data set. real-sim includes Usenet articles. astro-physic includes documents from Physics Arxiv. news20 is a collection of news documents. yahoo-japan and yahoo-korea are obtained from Yahoo!. rcv1 is an archive of manually categorized newswire stories from Reuters.

We compare six implementations of linear SVM. Three solve L1-SVM, and three solve L2-SVM.

DCDL1 and DCDL2: the dual coordinate descent method with sub-problems permuted at each outer iteration (see Section 3.1). DCDL1 solves L1-SVM while DCDL2 solves L2-SVM. We *omit the shrinking setting.*

Pegasos: the primal estimated sub-gradient solver (Shalev-Shwartz et al., 2007) for L1-SVM. The source is at `http://ttic.uchicago.edu/~shai/code`.

SVM$^{\text{perf}}$ (Joachims, 2006): a cutting plane method for L1-SVM. We use the latest version 2.1. The source is at `http://svmlight.joachims.org/svm_perf.html`.

TRON: a trust region Newton method (Lin et al., 2008) for L2-SVM. We use the software LIBLINEAR version 1.22 with option `-s 2` (`http://www.csie.ntu.edu.tw/~cjlin/liblinear`).

PCD: a primal coordinate descent method for L2-SVM (Chang et al., 2007).

Since (Bottou, 2007) is related to Pegasos, we do not present its results. We do not compare with another online method Vowpal Wabbit (Langford et al., 2007) either as it has been made available only very recently. Though a code for the bundle method (Smola et al., 2008) is available, we do not include it for comparison due to its closeness to SVM$^{\text{perf}}$. All sources used for our comparisons are available at `http://csie.ntu.edu.tw/~cjlin/liblinear/exp.html`.

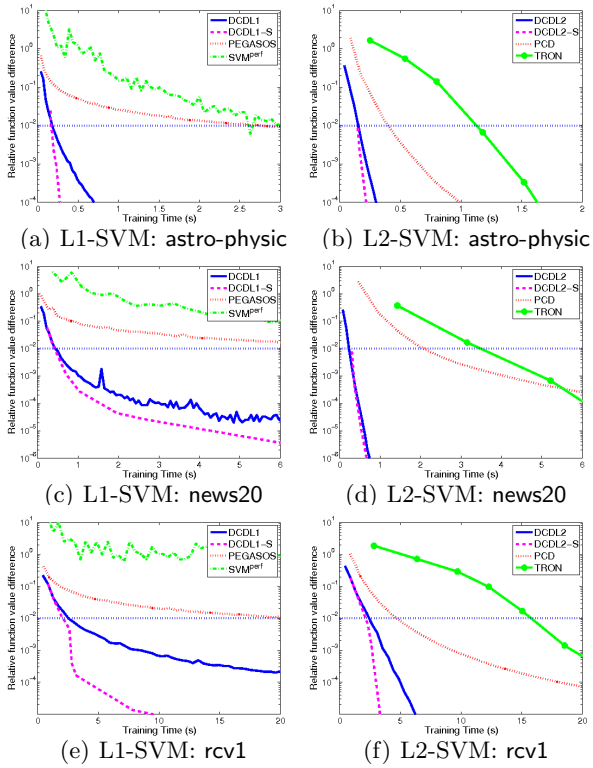
We set the penalty parameter $C = 1$ for comparison[2]. For all data sets, the testing accuracy does not increase after $C \geq 4$. All the above methods are implemented in C/C++ with double precision. Some implementations such as (Bottou, 2007) use single precision to reduce training time, but numerical inaccuracy may occur. We do not include the bias term by (3).

To compare these solvers, we consider the CPU time of reducing the relative difference between the primal objective value and the optimum to within 0.01:

$$|f^P(\boldsymbol{w}) - f^P(\boldsymbol{w}^*)|/|f^P(\boldsymbol{w}^*)| \leq 0.01, \qquad (20)$$

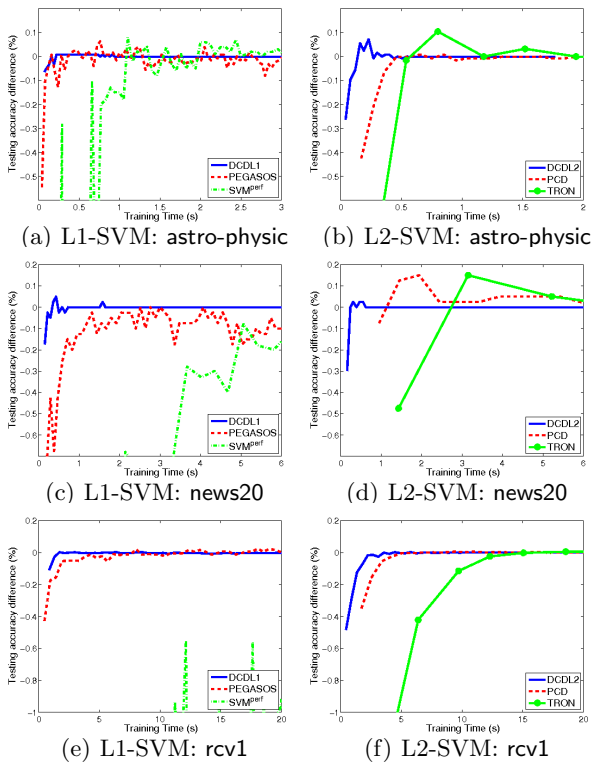
where f^P is the objective function of (1), and $f^P(\boldsymbol{w}^*)$ is the optimal value. Note that for consistency, we use primal objective values even for dual solvers. The reference solutions of L1- and L2-SVM are respectively obtained by solving DCDL1 and DCDL2 until the duality gaps are less than 10^{-6}. Table 2 lists the results. Clearly, our dual coordinate descent method

[2]The equivalent setting for Pegasos is $\lambda = 1/(Cl)$. For SVM$^{\text{perf}}$, its penalty parameter is $C_{perf} = 0.01Cl$.



(a) L1-SVM: astro-physic (b) L2-SVM: astro-physic
(c) L1-SVM: news20 (d) L2-SVM: news20
(e) L1-SVM: rcv1 (f) L2-SVM: rcv1

Figure 1. Time versus the relative error (20). DCDL1-S, DCDL2-S are DCDL1, DCDL2 with shrinking. The dotted line indicates the relative error 0.01. Time is in seconds.



(a) L1-SVM: astro-physic (b) L2-SVM: astro-physic
(c) L1-SVM: news20 (d) L2-SVM: news20
(e) L1-SVM: rcv1 (f) L2-SVM: rcv1

Figure 2. Time versus the difference of testing accuracy between the current model and the reference model (obtained using strict stopping conditions). Time is in seconds.

for both L1- and L2-SVM is significantly faster than other solvers. To check details, we choose astro-physic, news20, rcv1, and show the relative error along time in Figure 1. In Section 3.2, we pointed out that the shrinking technique is very suitable for DCD. In Figure 1, we also include them (DCDL1-S and DCDL2-S) for comparison. Like in Table 2, our solvers are efficient for both L1- and L2-SVM. With shrinking, its performance is even better.

Another evaluation is to consider how fast a solver obtains a model with reasonable testing accuracy. Using the optimal solutions from the above experiment, we generate the reference models for L1- and L2-SVM. We evaluate the testing accuracy difference between the current model and the reference model along the training time. Figure 2 shows the results. Overall, DCDL1 and DCDL2 are more efficient than other solvers. Note that we omit DCDL1-S and DCDL2-S in Figure 2, as the performances with/without shrinking are similar.

Among L1-SVM solvers, SVM$^{\text{perf}}$ is competitive with Pegasos for small data. But in the case of a huge number of instances, Pegasos outperforms SVM$^{\text{perf}}$. However, Pegasos has slower convergence than DCDL1. As discussed in Section 4.2, the learning rate of stochastic gradient descent may be the cause, but for DCDL1 we exactly solve sub-problems to obtain the step size in updating $\boldsymbol{w}$. Also, Pegasos has a jumpy test set performance while DCDL1 gives a stable behavior.

In the comparison of L2-SVM solvers, DCDL2 and PCD are both coordinate descent methods. The former one is applied to the dual, but the latter one to the primal. DCDL2 has a closed form solution for each sub-problem, but PCD has not. The cost per PCD outer iteration is thus higher than that of DCDL2. Therefore, while PCD is very competitive (only second to DCDL1/DCDL2 in Table 2), DCDL2 is even better. Regarding TRON, as a Newton method, it possesses fast final convergence. However, since it takes significant effort at each iteration, it hardly generates a reasonable model quickly. From the experiment results, DCDL2 converges as fast as TRON, but also performs well in early iterations.

Due to the space limitation, we give the following observations without details. First, Figure 1 indicates that our coordinate descent method converges faster for L2-SVM than L1-SVM. As L2-SVM has the diagonal matrix D with $D_{ii} = 1/(2C)$, we suspect that its $\bar{Q}$ is better conditioned, and hence leads to faster convergence. Second, all methods have slower convergence when C is large. However, small C's are usually enough as the accuracy is stable after a threshold. In practice, one thus should try from a small C. More-

over, if $n \ll l$ and C is too large, then our DCDL2 is slower than TRON or PCD (see problem a9a in Table 2, where the accuracy does not change after $C \geq 0.25$). If $n \ll l$, clearly one should solve the primal, whose number of variables is just n. Such data are not our focus. Indeed, with a small number of features, one usually maps data to a higher space and train a nonlinear SVM. Third, we have checked the online Algorithm 2. Its performance is similar to DCDL1 and DCDL2 (i.e., batch setting without shrinking). Fourth, we have investigated real document classification which involves many two-class problems. Using the proposed method as the solver is more efficient than using others.

6. Discussion and Conclusions

We can apply the proposed method to solve regularized least square problems, which have the loss function $(1-y_i \boldsymbol{w}^T \boldsymbol{x}_i)^2$ in (1). The dual is simply (4) without constraints, so the implementation is simpler.

In summary, we present and analyze an efficient dual coordinate decent method for large linear SVM. It is very simple to implement, and possesses sound optimization properties. Experiments show that our method is faster than state of the art implementations.

References

Bordes, A., Bottou, L., Gallinari, P., & Weston, J. (2007). Solving multiclass support vector machines with LaRank. *ICML*.

Boser, B. E., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *COLT*.

Bottou, L. (2007). Stochastic gradient descent examples. `http://leon.bottou.org/projects/sgd`.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2007). *Coordinate descent method for large-scale L2-loss linear SVM* (Technical Report). `http://www.csie.ntu.edu.tw/~cjlin/papers/cdl2.pdf`.

Collins, M., Globerson, A., Koo, T., Carreras, X., & Bartlett, P. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *JMLR*. To appear.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *JMLR*, *3*, 951–991.

Friess, T.-T., Cristianini, N., & Campbell, C. (1998). The kernel adatron algorithm: a fast and simple learning procedure for support vector machines. *ICML*.

Joachims, T. (1998). Making large-scale SVM learning practical. *Advances in Kernel Methods - Support Vector Learning*. Cambridge, MA: MIT Press.

Joachims, T. (2006). Training linear SVMs in linear time. *ACM KDD*.

Kao, W.-C., Chung, K.-M., Sun, C.-L., & Lin, C.-J. (2004). Decomposition methods for linear support vector machines. *Neural Comput.*, *16*, 1689–1704.

Keerthi, S. S., & DeCoste, D. (2005). A modified finite Newton method for fast solution of large scale linear SVMs. *JMLR*, *6*, 341–361.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Comput.*, *13*, 637–649.

Langford, J., Li, L., & Strehl, A. (2007). Vowpal Wabbit. `http://hunch.net/~vw`.

Lin, C.-J., Weng, R. C., & Keerthi, S. S. (2008). Trust region Newton method for large-scale logistic regression. *JMLR*, *9*, 623–646.

Luo, Z.-Q., & Tseng, P. (1992). On the convergence of coordinate descent method for convex differentiable minimization. *J. Optim. Theory Appl.*, *72*, 7–35.

Mangasarian, O. L., & Musicant, D. R. (1999). Successive overrelaxation for support vector machines. *IEEE Trans. Neural Networks*, *10*, 1032–1037.

Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: An application to face detection. *CVPR*.

Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*. Cambridge, MA: MIT Press.

Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: primal estimated sub-gradient solver for SVM. *ICML*.

Smola, A. J., Vishwanathan, S. V. N., & Le, Q. (2008). Bundle methods for machine learning. *NIPS*.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. *ICML*.

Discriminative Structure and Parameter Learning for Markov Logic Networks

Tuyen N. Huynh HNTUYEN@CS.UTEXAS.EDU
Raymond J. Mooney MOONEY@CS.UTEXAS.EDU
Department of Computer Sciences, The University of Texas at Austin,1 University Station C0500, Austin, TX 78712, USA

Abstract

Markov logic networks (MLNs) are an expressive representation for statistical relational learning that generalizes both first-order logic and graphical models. Existing methods for learning the logical structure of an MLN are not discriminative; however, many relational learning problems involve specific target predicates that must be inferred from given background information. We found that existing MLN methods perform very poorly on several such ILP benchmark problems, and we present improved discriminative methods for learning MLN clauses and weights that outperform existing MLN and traditional ILP methods.

1. Introduction

Statistical relational learning (SRL) concerns the induction of probabilistic knowledge that supports accurate prediction for multi-relational structured data (Getoor & Taskar, 2007). Markov Logic Networks (MLNs) are a recently developed SRL model that generalizes both full first-order logic and Markov networks (Richardson & Domingos, 2006). An MLN is represented as a set of weighted clauses in first-order logic, and learning an MLN decomposes into *structure learning*, learning the logical clauses, and *parameter learning*, setting the weight of each clause. Existing structure-learning algorithms for MLNs (Kok & Domingos, 2005; Mihalkova & Mooney, 2007) are non-discriminative and attempt to learn a set of clauses that is equally capable of predicting the truth value of all predicates given an arbitrary set of evidence. However, in many learning problems, there is a specific *target predicate* that must be inferred given evidence data about other *background predicates* used to describe the input data. Most traditional *Inductive Logic Programming* (ILP) methods focus on discriminative relational learning (Dzeroski, 2007); however, they do not address the issue of uncertainty. Discriminative methods have been developed for parameter learning in MLNs (Singla & Domingos, 2005; Lowd & Domingos, 2007); however, they do not address structure learning.

We have found that existing MLN structure learning methods perform very poorly when tested on several benchmark ILP problems on relating the activity of chemical compounds to their structure (King et al., 1995). This led us to develop new discriminative structure and parameter learning algorithms for MLNs whose performance on these problems surpasses that of traditional ILP methods. The overall approach is to use traditional ILP methods to construct a large number of potentially useful clauses, and then use discriminative MLN parameter learning methods to properly weight them, preferring to assign zero weights to clauses that do not contribute significantly to overall predictive accuracy, thereby eliminating them. Our structure learning component utilizes many of the clauses considered during the search conducted by a specific configuration of ALEPH (Srinivasan, 2001). Our parameter learning component utilizes an exact probabilistic inference algorithm for MLNs with only non-recursive definite clauses. Our parameter learner also uses L_1-regularization (Lee et al., 2006) instead of the normal L_2 in order to encourage assigning zero weights to clauses, thereby simplifying the theory. We present experimental results that demonstrate that all three of these enhancements contribute significantly to improving the performance of our system over existing MLN and ILP methods.

The remainder of the paper is organized as follows. Section 2 provides some background on MLNs, ALCHEMY (Kok et al., 2005), and ALEPH. Section 3 presents our new structure and parameter learning algorithms. Section 4 presents our experimental evaluation of these methods. Section 5 discusses related work, and section 6 presents our conclusions.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Background

2.1. ILP and Aleph

Traditional ILP systems discriminatively learn logical Horn-clause rules (logic programs) for inferring a given target predicate given information provided by a set of background predicates. These purely logical definitions are induced from Horn-clause background knowledge and a set of positive and negative tuples of the target predicate.

ALEPH is a popular and effective ILP system primarily based on PROGOL (Muggleton, 1995). The basic ALEPH algorithm consists of four steps. First, it selects a positive example to serve as the "seed" example. Then, it constructs the most specific clause, the "bottom clause", that entails that selected example. The bottom clause is formed by conjoining all known facts about the seed example. Next, ALEPH finds generalizations of this bottom clause by performing a general to specific search. These generalized clauses are scored using a chosen evaluation metric, and the clause with the best score is added to the final theory. This process is repeated

until it finds a set of clauses that covers all the positive examples. ALEPH allows users to customize each of these steps, and thereby supports a variety of specific algorithms.

2.2. MLNs and Alchemy

An MLN consists of a set of weighted first-order formulae (also called clauses or rules). It provides a way of softening first-order logic by making situations in which not all formulae are satisfied less likely but not impossible (Richardson & Domingos, 2006). More formally, let X be the set of all propositions describing a world (i.e. the set of all the ground atoms[1]), $\mathcal{F}$ be the set of all clauses in the MLN, w_i be the weight associated with clause $f_i \in \mathcal{F}$, $\mathcal{G}_{f_i}$ be the set of all possible groundings of clause f_i, and $\mathcal{Z}$ be the normalizing constant. Then the probability of a particular truth assignment x to the variables in X is given by the formula (Richardson & Domingos, 2006):

$$P(X = x) = \frac{1}{\mathcal{Z}} \exp\left(\sum_{f_i \in \mathcal{F}} w_i \sum_{g \in \mathcal{G}_{f_i}} g(x)\right) = \frac{1}{\mathcal{Z}} \exp\left(\sum_{f_i \in \mathcal{F}} w_i n_i(x)\right) \quad (1)$$

where $g(x)$ is 1 if g is satisfied and 0 otherwise, and $n_i(x) = \sum_{g \in \mathcal{G}_{f_i}} g(x)$ is the number of groundings of f_i that are satisfied given the current truth assignments to the variables in X.

[1]Ground atoms are predicates without variables, which are formed by replacing all the variables in predicates by constants.

In order to perform inference for an MLN, L, one needs to produce its corresponding ground Markov network. As described by Richardson and Domingos (2006), this is done by including a node for every possible grounding of the predicates in L and an edge between two nodes if they appear together in a grounding of a clause in L. The nodes appearing together in a ground clause form a clique. In general, exact inference in MLNs is intractable, so to calculate the probability that a ground atom or a set of them has a particular truth assignment given some evidence, one needs to run an approximate inference algorithm such as MCMC on this ground Markov network. MC-SAT (Poon & Domingos, 2006) is currently the best inference method for MLNs.

As previously mentioned, learning an MLN consists of two tasks: structure learning and weight learning. The weight learning component is independent of the structure learning one. It can learn weights for clauses produced by structure learning or written by a human expert. There are two approaches to weight learning: generative and discriminative. The former is used when there is no specific target predicate, and the latter when we know which predicate is to be queried. The current state-of-the-art discriminative weight learner is *preconditioner scaled conjugate gradient* (PSCG) (Lowd & Domingos, 2007). This algorithm uses samples from MC-SAT to approximate the intractable expected counts of satisfied clauses w.r.t. the current model. These counts are needed to compute the gradient and Hessian of the conditional log-likelihood (CLL) of an MLN. The inverse diagonal Hessian is used as the preconditioner in this method.

Regarding structure learning, there are currently two algorithms for learning clauses for MLNs. The first algorithm was proposed by Kok and Domingos (2005). This algorithm uses a top-down approach and can perform either beam-search or shortest-first search over the space of clauses. The candidate clauses are scored using *weighted pseudo log-likelihood* (WPLL) (Kok & Domingos, 2005). In contrast, the second algorithm, BUSL (Mihalkova & Mooney, 2007) follows a more bottom-up approach. It first constructs Markov network templates from the data and then generates candidate clauses from these network templates. All candidate clauses are also evaluated using WPLL, and added to the final MLN in a greedy manner. Both of these algorithms can be constrained to only learn clauses that contain a given target predicate by setting their "ne" (non-evidence) parameter to that predicate. However, they are not designed for discriminative learning since they try to find a set of clauses which maximizes WPLL, a non-discriminative measure.

ALCHEMY (Kok et al., 2005) is an open source software system for MLNs. It has implementations of all the ex-

isting algorithms for structure learning, generative weight learning, discriminative weight learning, and inference for MLNs. It is also a framework for developing new algorithms for MLNs. Our proposed algorithm is implemented in this framework.

3. Discriminative MLN Algorithms

In this section, we describe our two-step process for discriminatively learning both the structure and parameters of an MLN. The first step uses ALEPH to learn a large set of potential clauses. The second step learns the weights for these clauses, preferring to eliminate useless clauses by giving them zero weight.

3.1. Discriminative Structure Learning

Ideally, the search for discriminative MLN clauses would be directly guided by the goal of maximizing their contribution to the predictive accuracy of a complete MLN. However, this would require evaluating every proposed refinement to the existing set of learned clauses by relearning weights for all of the clauses and performing full probabilistic inference to determine the CLL of each of the query atoms. This process is computationally expensive and would have to be repeated for each of the combinatorially large number of potential clause refinements. Evaluating clauses in standard ILP is quicker since each clause can be evaluated in isolation based on the accuracy of its logical inferences about the target predicate. Consequently, we take the heuristic approach of using a standard ILP method to generate clauses; however, since the logical accuracy of a clause is only a rough approximation of its value in a final MLN, we generate a large number of candidates whose accuracy is at least markedly greater than random guessing and allow subsequent weight learning to determine their value to an overall MLN.

In order to find a set of potentially good clauses for an MLN, we use a particular configuration of ALEPH. Specifically, we use the **induce_cover** command and **m-estimate** evaluation function. The **induce_cover** command implements a variant of PROGOL's MDIE greedy covering algorithm (Muggleton, 1995) which does *not* remove previously covered examples when scoring a new clause. The normal ALEPH **induce** command scores a clause based only on its coverage of currently uncovered positive examples. However, this scoring is not reflective of its use in a final MLN, and we found that the **induce_cover** approach produces a larger set of more useful clauses that significantly increases the accuracy of our final learned MLN. The *m*-estimate (Džeroski, 1991) is a Bayesian estimation of the accuracy of a clause (Cussens, 2007). The m parameter defining the underlying prior distribution is automatically set to the maximum likelihood estimate of its best value. The output of **induce_cover** is a theory, a set of high-scoring clauses that cover all the positive examples. However, these clauses were selected based on an *m*-estimate of their accuracy under a purely logical interpretation, and may not be the best ones for an MLN. Therefore, in addition to these clauses, we also save all generated clauses whose *m*-estimate is greater than a predefined threshold (set to 0.6 in our experiments). This provides a large set of clauses of potential utility for an MLN. We use the name ALEPH++ to refer to this version of ALEPH.

3.2. Discriminative Weight Learning

Compared to ALCHEMY's current discriminative weight learning method (Lowd & Domingos, 2007), our method embodies two important modifications: *exact inference* and L_1*-regularization*. This section describes these two modifications.

First, given the restricted nature of the clauses constructed by ALEPH, we can use an efficient exact probabilistic inference method when learning their weights instead of the approximate inference algorithm that ALCHEMY uses to handle the general case. Since these clauses are non-recursive definite clauses in which the target predicate only appears once, multiple query atoms will not appear together in any grounding of any clause. For MLNs, this means that the Markov blanket of a query atom only contains evidence atoms. Consequently, the query atoms are independent given the evidence. Let Y be the set of query atoms and X be the set of evidence atoms, the conditional log likelihood of Y given X in this case is:

$$\log P(Y = y|X = x) = \log \prod_{j=1}^{n} P(Y_j = y_j|X = x)$$
$$= \sum_{j=1}^{n} \log P(Y_j = y_j|X = x)$$

and,

$$P(Y_j = y_j|X = x) = \frac{exp(\sum_{i \in \mathcal{F}_{Y_j}} w_i n_i(x, y_{[Y_j=y_j]}))}{exp(\sum_{i \in \mathcal{F}_{Y_j}} w_i n_i(x, y_{[Y_j=0]})) + exp(\sum_{i \in \mathcal{F}_{Y_j}} w_i n_i(x, y_{[Y_j=1]}))} \quad (2)$$

where $\mathcal{F}_{Y_j}$ is the set of all MLN clauses with at least one grounding containing the query atom Y_j, $n_i(x, y_{[Y_j=y_j]})$ is the number groundings of the ith clause that evaluate to true when all the evidence atoms in X and the query atom Y_j are set to their truth values, and similarly for $n_i(x, y_{[Y_j=0]})$ and $n_i(x, y_{[Y_j=1]})$ when Y_j is set to 0 and 1 respectively. Then the gradient of the CLL is:

$\frac{\partial}{\partial w_i} \log P(Y = y|X = x) =$

$$\sum_{j=1}^{n} [n_i(x, y_{[Y_j=y_j]}) - P(Y_j = 0|X = x) n_i(x, y_{[Y_j=0]}) - P(Y_j = 1|X = x) n_i(x, y_{[Y_j=1]})]$$

Notice that the sum of the last two terms in the gradient is the expected count of the number of true grounding of the i'th formula. In general, computing this expected count requires performing approximate inference under the model. For example, Singla and Domingos (2005) ran MAP inference and used the counts in the MAP state to approximate the expected counts. However, in our case, using the standard closed world assumption for evidence predicates, all the n_i's can be computed without approximate inference since there is no ground atom whose truth value is unknown. This is a result of restricting the structure learner to non-recursive definite clauses. In fact, this result still holds even when the clauses are not Horn clauses. The only restriction is that the target predicate appears only once in every clause. Note that given a set of weights, computing the conditional probability $P(y|x)$, the CLL, and its gradient requires only the n_i counts. So, in our case, the conditional probability $P(Y_j = y_j|X = x)$, the CLL, and its gradient can be computed exactly. In addition, these counts only need to be computed once, and ALCHEMY provides an efficient method for computing them. ALCHEMY also provides an efficient way to construct the Markov blanket of a query atom, in particular it ignores all ground formulae whose truth values are unaffected by the value of the query atom. In our case, this helps reduce the size of the Markov blanket of a query atom significantly since many ground clauses are satisfied by the evidence. As a result, our exact inference is very fast even when the MLN contains thousands of clauses.

Given a procedure for computing the CLL and its gradient, standard gradient-based optimization methods can be used to find a set of weights that optimizes the CLL. However, to prevent overfitting and select only the best clauses, we follow the approach suggested by Lee et al. (2006) and introduce a $Laplacian$ prior with zero mean, $P(w_i) = (\beta/2) \cdot exp(-\beta|w_i|)$, on each weight, and then optimize the posterior conditional log likehood instead of the CLL. The final objective function is:

$$\begin{aligned} \log P(Y|X)P(w) &= \log P(Y|X) + \log P(w) \\ &= \log P(Y|X) + \log(\prod_i P(w_i)) \\ &= CLL + \sum_i \log(\frac{\beta}{2} \cdot exp(-\beta|w_i|)) \\ &= CLL - \beta \sum_i |w_i| + constant \end{aligned}$$

There is now an additional term $\beta \sum_i |w_i|$ in the objective function, which penalizes each non-zero weight w_i by $\beta|w_i|$. So, the larger β is (corresponding to a smaller variance of the prior distribution), the more we penalize non-zero weights. Therefore, placing a $Laplacian$ prior with zero mean on each weight is equivalent to performing an L_1-regularization of the parameters. An important property of L_1-regularization is its tendency to force parameters to zero by strongly penalizing small terms (Lee et al., 2006). In order to learn weights that optimize the L_1-regularized CLL, we use the OWL-QN package which implements the Orthant-Wise Limited-memory Quasi-Newton algorithm (Andrew & Gao, 2007).

This approach to preventing over-fitting contrasts with the standard L_2-regularization used in previous work on learning weights for MLNs, which is equivalent to assuming a Guassian prior with zero mean on each weight and does not penalize non-zero weights as severely. Since ALEPH++ generates a very large number of potential clauses, L_1-regularization encourages eliminating the less useful ones by setting their weights to zero. In agreement with prior results on L_1-regularization (Ng, 2004; Dudík et al., 2007), our experiments confirm that it results in simpler and more accurate learned models compared to L_2-regularization.

4. Experimental Evaluation

In this section, we present experiments that were designed to answer the following questions:

1. How does our method compare to existing methods, specifically:
   (a) Extant discriminative learning for MLNs, viz. ALCHEMY.
   (b) Traditional ILP methods, viz. ALEPH.
   (c) "Advanced" ILP methods, viz. kFOIL (Landwehr et al., 2006), TFOIL (Landwehr et al., 2007), and RUMBLE (Rückert & Kramer, 2008).
2. How does each of our system's major novel components below contribute to its performance:
   (a) Generation of a larger set of potential clauses by using ALEPH++ instead of ALEPH.
   (b) Exact MLN inference for non-recursive definite clauses instead of general approximate inference.
   (c) L_1-regularization instead of L_2.

4.1. Data

We employed four benchmark data sets previously used to evaluate a variety of ILP and relational learning algorithms. They concern predicting the relative biochemical

activity of variants of Tacrine, a drug for Alzheimer's disease (King et al., 1995).[2] The data contain background knowledge about the physical and chemical properties of substituents such as their hydrophobicity and polarity, the relations between various physical and chemical constants, and other relevant information. The goal is to compare various drugs on four important biochemical properties: low **toxicity**, high **acetyl** cholinesterase inhibition, good reversal of scopolamine-induced **memory** impairment, and inhibition of **amine** re-uptake. For each property, the positive and negative examples are pairwise comparisons of drugs. For example, $less\_toxic(d_1, d_2)$ means that drug d_1's toxicity is less than d_2's. These ordering relations are transitive but not complete (i.e. for some pairs of drugs it is unknown which one is better). Therefore, this is a structured (a.k.a. collective) prediction problem since the output labels should form a partial order. However, previous work has ignored this structure and just predicted the examples separately as distinct binary classification problems. In this work, in addition to treating the problem as independent classification, we also use an MLN to perform structured prediction by explicitly imposing the transitive constraint on the target predicate. Table 1 shows some background facts and examples from one of the datasets, and Table 2 summarizes information about all four datasets.

Table 2. Summary statistics for Alzheimer's data sets.

Data set	#Examples	% Positive	# Predicates
Alzheimer acetyl	1326	50%	30
Alzheimer amine	686	50%	30
Alzheimer memory	642	50%	30
Alzheimer toxic	886	50%	30

4.2. Methodology

To answer the above questions, we ran experiments with the following systems:

ALCHEMY**:** Uses the structure learning (Kok & Domingos, 2005) in ALCHEMY and the most accurate existing discriminative weight learning PSCG (Lowd & Domingos, 2007) with the "ne" (non-evidence) parameter set to the target predicate.

BUSL**:** Uses BUSL (Mihalkova & Mooney, 2007) and PSCG discriminative weight learning with the "ne" (non-evidence) parameter set to the target predicate.

ALEPH**:** Uses ALEPH's standard settings with a few modifications. The maximum number of literals in an acceptable clause was set to 5. The minimum number of positive examples covered by an acceptable clause was set to 2. The upper bound on the number of negative examples covered by an acceptable clause was set to 300. The evaluation function was set to **auto_m**, and the minimum score of an acceptable clause was set to 0.6. The **induce_cover** command was used to learn the clauses. We found that this configuration gave somewhat better overall accuracy compared to those reported in previous work.

ALEPH**PSCG:** Uses the discriminative weight learner PSCG to learn MLN weights for the clauses in the final theory returned by ALEPH. Note that PSCG also uses L_2-regularization.

ALEPH**ExactL2** : Uses the limited-memory BFGS algorithm (Liu & Nocedal, 1989) implemented in ALCHEMY to learn discriminative MLN weights for the clauses in the final theory returned by ALEPH. The objective function is CLL with L_2 regularization. The CLL is computed exactly as described in Section 3.2.

ALEPH**++PSCG:** Like ALEPHPSCG, but learns weights for the larger set of clauses returned by ALEPH++.

ALEPH**++ExactL2:** Like ALEPHExactL2, but learns weights for the larger set of clauses returned by ALEPH++.

ALEPH**++ExactL1:** Our full proposed approach using exact inference and L_1-regularization to learn weights on the clauses returned by ALEPH++.

To force the predictions for the target predicate to properly constitute a partial ordering, we also tried adding to the learned MLNs a hard constraint (i.e. a clause with infinite weight) stating the transitive property of the target predicate, and used the MC-SAT algorithm to perform prediction on the test data. This exploits the ability of MLNs to perform collective classification (structured prediction) for the complete set of test examples.

In testing, only the background facts are provided as evidence to ensure that all predictions are based on the chemical structure of a drug. For all systems except ALEPH, a threshold of 0.5 was used to convert predicted probabilities into boolean values. The predictive accuracy of these algorithms for the target predicate were compared using 10-fold cross-validation. The significance of the results were evaluated using a two-tailed paired t-test test with a 95% confidence level. To compare the quality of the predicted probabilities, we also report the average area under the ROC curve (AUC-ROC) for all probabilistic systems by using the AUCCalculator package (Davis & Goadrich, 2006).

[2]Since the current ALCHEMY does not support real valued variables, we could not test our approach on the other standard ILP benchmark data sets in molecular biology.

Table 1. Some background evidence and examples from the Alzheimer toxic dataset.

Background evidence	Examples
r_subst_1(A1,H), r_subst_1(B1,H), r_subst_1(D1,H), x_subst(B1,7,CL), x_subst(D1,6,OCH3), polar(CL,POLAR3), polar(OCH3,POLAR2), great_polar(POLAR3,POLAR2), size(CL,SIZE1), size(OCH3,SIZE2), alk_groups(A1,0), alk_groups(B1,0), alk_groups(D1,0), great_size(SIZE2,SIZE1), flex(CL,FLEX0), flex(OCH3,FLEX1)	less_toxic(B1,A1) less_toxic(A1,D1) less_toxic(B1,D1)

Table 3. Average predictive accuracies and standard deviations for all systems. Bold numbers indicate the best result on a data set.

Data set	ALCHEMY	BUSL	ALEPH	ALEPH **PSCG**	ALEPH **ExactL2**	ALEPH++ **PSCG**	ALEPH++ **ExactL2**	ALEPH++ **ExactL1**
Alzheimer amine	50.1 ± 0.5	51.3 ± 2.5	81.6 ± 5.1	64.6 ± 4.6	83.5 ± 4.7	72.0 ± 5.2	86.8 ± 4.4	$\mathbf{89.4 \pm 2.7}$
Alzheimer toxic	54.7 ± 7.4	51.7 ± 5.3	81.7 ± 4.2	74.7 ± 1.9	87.5 ± 4.8	69.9 ± 1.2	89.5 ± 3.0	$\mathbf{91.3 \pm 2.8}$
Alzheimer acetyl	48.2 ± 2.9	55.9 ± 8.7	79.6 ± 2.2	78.0 ± 3.2	79.5 ± 2.0	76.5 ± 3.7	82.1 ± 2.1	$\mathbf{85.1 \pm 2.4}$
Alzheimer memory	50 ± 0.0	49.8 ± 1.6	76.0 ± 4.9	60.3 ± 2.1	72.6 ± 3.4	65.6 ± 5.4	72.9 ± 5.2	$\mathbf{77.6 \pm 4.9}$

Table 6. Average number of clauses learned

Data set	ALEPH++	ALEPH++ **ExactL2**	ALEPH++ **ExactL1**
Alzheimer amine	7061	5070	3477
Alzheimer toxic	2034	1194	747
Alzheimer acetyl	8662	5427	2433
Alzheimer memory	6524	4250	2471

4.3. Results and Discussion

Tables 3 and 4 show the average accuracy and AUC-ROC with standard deviation for each system running on each data set. Our complete system (ALEPH++**ExactL1**) achieves significantly higher accuracy than both ALCHEMY and BUSL on all 4 data sets and significantly higher than ALEPH on all except the **memory** data set, answering questions 1(a) and 1(b). In turn, ALEPH has been shown to give higher accuracy on these data sets than other standard ILP systems like FOIL (Landwehr et al., 2007). ALCHEMY's existing non-discriminative structure learners find only a few (3–5) simple clauses. Two of them are unit clauses for the target predicate, such as *great_ne(a1,a1)* and *great_ne(a1,a2)*; the others capture the transitive nature of the target relation. Therefore, even after they are discriminatively weighted, their predictions are not significantly better than random guessing.

The ablations that remove components from our overall system demonstrate the important contribution of each component. Regarding question 2(b), the systems using general approximate inference (ALEPH**PSCG** and ALEPH++**PSCG**) perform much worse than the corresponding versions that use exact inference (ALEPH**ExactL2** and ALEPH++**ExactL2**). Therefore, when there is a target predicate that can be accurately inferred using non-recursive definite clauses, exploiting this restriction to perform exact inference is a clear win.

Regarding question 2(a), ALEPH++**ExactL2** performs significantly better than ALEPH**ExactL2**, demonstrating the advantage of learning a large set of potential clauses and combining them with learned weights in an overall MLN. Across the four datasets, ALEPH++ returns an average of $6,070$ clauses compared to only 10 for ALEPH.

Table 5 presents average accuracies with standard deviations for the MLN systems when we include a transitivity clause for the target predicate. This constraint improves the accuracies of ALEPH**ExactL2**, ALEPH++**ExactL2**, and ALEPH++**ExactL1**, but sometimes decreases the accuracy of other systems, such as ALEPH**PSCG**. This can be explained as follows. Since most of the predictions of ALEPH++**ExactL1** are correct, enforcing transitivity can correct some of the wrong ones. However, ALEPH**PSCG** produces many wrong predictions, so forcing them to obey transitivity can produce additional incorrect predictions. Due to space constraints, we do not report the corresponding AUC-ROC results, which are qualitatively similar.

Regarding question 2(c), using L_1-regularization gives significantly higher accuracy and AUC-ROC than using standard L_2-regularization. This comparison was only performed for ALEPH++ since this is when the weight-learner must choose from a large set of candidate clauses by encouraging zero weights. Table 6 compares the average number of clauses learned (after zero-weight clauses are removed) for L_1 and L_2 regularization. As expected, the final learned MLNs are much simpler when using L_1-regularization. On average, L_1-regularization reduces the size of the final set of clauses by 26% compared to L_2-regularization.

Regarding question 1(c), several researchers have tested "advanced" ILP systems on our datasets. Table 7 compares our best results to those reported for TFOIL (a combination of FOIL and tree augmented naive Bayes), kFOIL (a kernelized version of FOIL), and RUMBLE (a max-margin approach to learning a weighted rule set). Our results are competitive with these recent systems. Additionally, unlike MLNs, these methods do not create "declarative" theories

Table 4. Average AUC-ROC and standard deviations for all systems. Bold numbers indicate the best result on a data set.

Data set	ALCHEMY	BUSL	ALEPH **PSCG**	ALEPH **ExactL2**	ALEPH++ **PSCG**	ALEPH++ **ExactL2**	ALEPH++ **ExactL1**
Alzheimer amine	.483 ± .115	.641 ± .110	.846 ± .041	.904 ± .027	.777 ± .052	.935 ± .032	**.954 ± .019**
Alzheimer toxic	.622 ± .079	.511 ± .079	.904 ± .034	.930 ± .035	.874 ± .041	.937 ± .029	**.939 ± .035**
Alzheimer acetyl	.473 ± .037	.588 ± .108	.850 ± .018	.850 ± .020	.810 ± .040	.899 ± .015	**.916 ± .013**
Alzheimer memory	.452± .088	.426 ± .065	.744 ± .040	.768 ± .032	.737 ± .059	.813 ± .059	**.844 ± .052**

Table 5. Average predictive accuracies and standard deviations for MLN systems with transitive clause added.

Data set	ALCHEMY	BUSL	ALEPH **PSCG**	ALEPH **ExactL2**	ALEPH++ **PSCG**	ALEPH++ **ExactL2**	ALEPH++ **ExactL1**
Alzheimer amine	50.0 ± 0.0	52.2 ± 5.3	61.4 ± 3.6	87.0 ± 3.3	72.9± 3.5	**91.7± 3.5**	90.5 ± 3.6
Alzheimer toxic	50.0 ± 0.0	50.1 ± 0.8	73.3 ± 1.8	88.8 ± 4.8	68.4± 1.5	91.4± 3.6	**91.9 ± 4.1**
Alzheimer acetyl	53.0 ± 6.2	54.1 ± 4.9	80.4 ± 2.7	84.1 ± 3.1	83.3± 2.5	**88.7± 2.1**	87.6 ± 2.7
Alzheimer memory	50.0 ± 0.0	50.1 ± 0.5	58.9 ± 2.3	76.5 ± 3.5	70.1± 5.2	81.3± 4.8	**81.3 ± 4.1**

that have a well-defined possible worlds semantics.

5. Related Work

Using an off-the-shelf ILP system to learn clauses for MLNs is not a new idea. Richardson and Domingos (2006) used CLAUDIEN, an non-descriminative ILP system that can learn arbitrary first-order clauses, to learn MLN structure and to refine the clauses from a knowledge base. Kok and Domingos (2005) reported experimental results comparing their MLN structure learner to learning clauses using CLAUDIEN, FOIL, and ALEPH. However, since this previous work used the relatively small set of clauses produced by these unaltered ILP systems, the performance was not very good. ILP systems have also been used to learn structures for other SRL models. The SAYU system (Davis et al., 2005) used ALEPH to propose candidate features for a Bayesian network classifier. Muggleton(2000) used PROGOL, another popular ILP system, to learn clauses for Stochastic Logic Programs (SLPs).

When restricted to learning non-recursive clauses for classification, our approach is equivalent to using ALEPH to construct features for use by L_1-regularized logistic regression. Under this view, our approach is closely related to MACCENT (Dehaspe, 1997), which uses a greedy approach to induce clausal constraints that are used as features for maximum-entropy classification. One difference between our approach and MACCENT is that we use a two-step process instead of greedily adding one feature at a time. In addition, our clauses are induced in a bottom-up manner while MACCENT uses top-down search; and our weight learner employs L_1-regularization which makes it less prone to overfitting. Unfortunately, we could not compare experimentally to MACCENT since "only an implementation of a propositional version of MACCENT is available, which only handles data in attribute-value (vector) format" (Landwehr et al., 2007). Additionally, MLNs are a more expressive formalism that also allows for structured prediction, as demonstrated by our results that include a transitivity constraint on the target relation.

6. Conclusions

We have found that existing methods for learning Markov Logic Networks perform very poorly when tested on several benchmark ILP problems in drug design. We have presented a new approach to constructing MLNs that discriminatively learns both their structure and parameters to optimize predictive accuracy for a stated target predicate when given evidence specified with a defined set of background predicates. It uses a variant of an existing ILP system (ALEPH) to construct a large number of potential clauses and then effectively learns their parameters by altering existing discriminative MLN weight-learning methods to utilize exact inference and L_1 regularization. Experimental results show that the resulting system outperforms existing MLN and ILP methods and gives state-of-the-art results for the Alzheimer's-drug benchmarks.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. We also thank Niels Landwehr for helping us set up the experiment with ALEPH. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and managed by the Air Force Research Laboratory (AFRL) under contract FA8750-05-2-0283. Most of the experiments were run on the Mastodon Cluster, provided by NSF Grant EIA-0303609. The first author also thanks the Vietnam Education Foundation (VEF) for its sponsorship. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of DARPA, AFRL, VEF, or the United States Government.

Table 7. Average predictive accuracies and standard deviations of our best results and other "advanced" ILP systems.

Data set	Our best results	TFOIL	kFOIL	RUMBLE
Alzheimer amine	**91.7± 3.5**	87.5 ± 4.4	88.8 ± 5.0	91.1
Alzheimer toxic	91.9 ± 4.1	**92.1 ± 2.6**	89.3 ± 3.5	91.2
Alzheimer acetyl	**88.7± 2.1**	82.8 ± 3.8	87.8 ± 4.2	88.4
Alzheimer memory	81.3 ± 4.1	80.4 ± 5.3	80.2 ± 4.0	**83.2**

References

Andrew, G., & Gao, J. (2007). Scalable training of L_1-regularized log-linear models. *Proc. of 24th Intl. Conf. on Machine Learning (ICML-2007)* (pp. 33–40). Corvallis, OR: Omnipress.

Cussens, J. (2007). Logic-based formalisms for statistical relational learning. In L. Getoor and B. Taskar (Eds.), *Introduction to statistical relational learning*, 269–290. Cambridge, MA: MIT Press.

Davis, J., Burnside, E. S., de Castro Dutra, I., Page, D., & Costa, V. S. (2005). An integrated approach to learning Bayesian networks of rules. *Proc. of the 16th European Conf. on Machine Learning (ECML-05)* (pp. 84–95).

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. *Proc. of 23rd Intl. Conf. on Machine Learning (ICML-2006)* (pp. 233–240).

Dehaspe, L. (1997). Maximum entropy modeling with clausal constraints. *Proc. of the 7th Intl. Workshop on Inductive Logic Programming* (pp. 109–124). Springer-Verlag.

Dudík, M., Phillips, S. J., & Schapire, R. E. (2007). Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, *8*, 1217–1260.

Džeroski, S. (1991). Handling noise in inductive logic programming. Master's thesis, Faculty of Electrical Engineering and Computer Science, University of Ljubljana.

Dzeroski, S. (2007). Inductive logic programming in a nutshell. In L. Getoor and B. Taskar (Eds.), *Introduction to statistical relational learning*, 57–92. Cambridge, MA: MIT Press.

Getoor, L., & Taskar, B. (Eds.). (2007). *Statistical relational learning*. Cambridge, MA: MIT Press.

King, R. D., Sternberg, M. J. E., & Srinivasan, A. (1995). Relating chemical activity to structure: An examination of ILP successes. *New Generation Computing*, *13*, 411–433.

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. *Proc. of 22nd Intl. Conf. on Machine Learning (ICML-2005)*. Bonn,Germany.

Kok, S., Singla, P., Richardson, M., & Domingos, P. (2005). *The Alchemy system for statistical relational AI* (Technical Report). Department of Computer Science and Engineering, University of Washington. http://www.cs.washington.edu/ai/alchemy.

Landwehr, N., Kersting, K., & Raedt, L. D. (2007). Integrating Naive Bayes and FOIL. *Journal of Machine Learning Research*, *8*, 481–507.

Landwehr, N., Passerini, A., Raedt, L. D., & Frasconi, P. (2006). kFOIL: Learning simple relational kernels. *Proc. of the Twenty-First Natl. Conf. on Artificial Intelligence (AAAI-2006)*. Boston, MA: AAAI Press.

Lee, S., Ganapathi, V., & Koller, D. (2006). Efficient structure learning of Markov networks using L_1-regularization. *Advances in Neural Information Processing Systems 18*.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematic Programming*, *45*, 503–528.

Lowd, D., & Domingos, P. (2007). Efficient weight learning for Markov logic networks. *Proc. of 7th European Conf. of Principles and Practice of Knowledge Discovery in Databases (PKDD-2007)* (pp. 200–211).

Mihalkova, L., & Mooney, R. J. (2007). Bottom-up learning of Markov logic network structure. *Proc. of 24th Intl. Conf. on Machine Learning (ICML-2007)*. Corvallis, OR.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*, 245–286.

Muggleton, S. (2000). Learning stochastic logic programs. *Proceedings of the AAAI2000 Workshop on Learning Statistical Models from Relational Data*.

Ng, A. Y. (2004). Feature selection, L_1 vs. L_2 regularization, and rotational invariance. *Proc. of 21st Intl. Conf. on Machine Learning (ICML-2004)* (pp. 78–85). Banff, Alberta, Canada: ACM.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. *Proc. of the Twenty-First Natl. Conf. on Artificial Intelligence (AAAI-2006)*. Boston, MA.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Rückert, U., & Kramer, S. (2008). Margin-based first-order rule learning. *Machine Learning*, *70*, 189–206.

Singla, P., & Domingos, P. (2005). Discriminative training of Markov logic networks. *Proc. of 20th Natl. Conf. on Artificial Intelligence (AAAI-2005)* (pp. 868–873).

Srinivasan, A. (2001). *The Aleph manual*. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/.

Causal Modelling Combining Instantaneous and Lagged Effects: an Identifiable Model Based on Non-Gaussianity

Aapo Hyvärinen AAPO.HYVARINEN@HELSINKI.FI
Dept of Computer Science and HIIT, University of Helsinki, Finland

Shohei Shimizu SSHIMIZU@AR.SANKEN.OSAKA-U.AC.JP
The Institute of Scientific and Industrial Research, Osaka University

Patrik O. Hoyer PATRIK.HOYER@HELSINKI.FI
Dept of Computer Science and HIIT, University of Helsinki, Finland

Abstract

Causal analysis of continuous-valued variables typically uses either autoregressive models or linear Gaussian Bayesian networks with instantaneous effects. Estimation of Gaussian Bayesian networks poses serious identifiability problems, which is why it was recently proposed to use non-Gaussian models. Here, we show how to combine the non-Gaussian instantaneous model with autoregressive models. We show that such a non-Gaussian model is identifiable without prior knowledge of network structure, and we propose an estimation method shown to be consistent. This approach also points out how neglecting instantaneous effects can lead to completely wrong estimates of the autoregressive coefficients.

1. Introduction

Analysis of causal influences or effects has become an important topic in machine learning (Pearl, 2000; Spirtes et al., 1993), and has numerous applications in, for example, neuroinformatics (Roebroeck et al., 2005; Kim et al., 2007) and bioinformatics (Opgen-Rhein & Strimmer, 2007). For continuous-valued variables, such an analysis can basically be performed in two different ways. First, if the time-resolution of the measurements is higher than the time-scale of causal influences, one can estimate a classic autoregressive model with time-lagged variables and interpret the autoregressive coefficients as causal effects. Second, if the measurements have a lower time resolution than the causal influences, or if the data has no temporal structure at all, one can use a model in which the causal influences are instantaneous, leading to Bayesian networks or structural equation models (Bollen, 1989).

While estimation of autoregressive methods can be solved by classic regression methods, the case of instantaneous effects is much more difficult. Most methods suffer from lack of identifiability,[1] because covariance information alone is not sufficient to uniquely characterize the model parameters. Prior knowledge of the structure (fixing some of the connections to zero) of the Bayesian network is then necessary for most practical applications. However, a method was recently proposed which uses the non-Gaussian structure of the data to overcome the identifiability problem (Shimizu et al., 2006): If the disturbance variables (external influences) are non-Gaussian, no prior knowledge on the network structure (other than the ubiquitous assumption of a directed acyclic graph (DAG)) is needed to estimate the model.

Here, we consider the general case where causal influences can occur either instantaneously or with considerable time lags. Such a model is called the structural vector autoregressive (SVAR) model in econometric theory, in which numerous attempts have been made for its estimation, see e.g. (Swanson & Granger, 1997; Demiralp & Hoover, 2003; Moneta & Spirtes, 2006). We propose to use non-Gaussianity to estimate the model. We show that this variant of the model is iden-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] Identifiability is here used in the classic statistical sense: a model is identifiable if no two different values of the parameter vector give the same distribution for the observed data.

tifiable without any other restrictions than acyclicity. To our knowledge, no model proposed for this problem has been shown to be fully identifiable without prior knowledge of network structure. We further propose a computational method for estimating the model based on the theory of independent component analysis or ICA (Hyvärinen et al., 2001).

The proposed non-Gaussian model not only allows estimation of both instantaneous and lagged effects; it also shows that taking instantaneous influences into account can change the values of the time-lagged coefficients quite drastically. Thus, we see that neglecting instantaneous influences can lead to misleading interpretations of causal effects. The framework further leads to a generalization of the well-known Granger causality measure.

The paper is structured as follows. We first define the model and discuss its relation to other models in Section 2. In Section 3 we propose an estimation method, show its consistency, and discuss an intuitive interpretation of the method. Section 4 contains some theoretical examples and a theorem on how including instantaneous effects in the model changes the resulting interpretations. The resulting generalization of Granger causality is discussed in Section 5. The validity of the estimation method is demonstrated by simulations on artificial data in Section 6, and experiments on financial and neuroscientific data in Section 7. Section 8 concludes the paper.

2. Model Combining Lagged and Instantaneous Effects

2.1. Definition and Assumptions

Let us denote the observed time series by $x_i(t), i = 1, \ldots, n, t = 1, \ldots, T$ where i is the index of the variables (time series) and t is the time index. All the variables are collected into a single vector $\mathbf{x}(t)$. Denote by k the number of time-delays used, i.e. the order of the autoregressive model. Denote by $\mathbf{B}_\tau$ the $n \times n$ matrix of the causal effects between the variables x_i with time lag $\tau, \tau = 0 \ldots k$.

The causal dynamics in our model are a combination of autoregressive and structural-equation models. The model is defined as

$$\mathbf{x}(t) = \sum_{\tau=0}^{k} \mathbf{B}_\tau \mathbf{x}(t-\tau) + \mathbf{e}(t) \qquad (1)$$

where the $e_i(t)$ are random processes modelling the external influences or "disturbances". We make the following assumptions on the external influences $e_i(t)$. First, they are mutually *independent*, and *temporally uncorrelated*, which are typical assumptions in autoregressive models. Second, they are assumed to be *non-Gaussian*, which is an important assumption which distinguishes our model from classic models, whether autoregressive models, structural-equation models, or Bayesian networks.

Further, we assume that the matrix modelling instantaneous effects, $\mathbf{B}_0$, corresponds to an *acyclic* graph, as is typical in causal analysis, but this may not be strictly necessary as will be discussed below. The acyclicity is equivalent to the existence of a permutation matrix $\mathbf{P}$, which corresponds to an ordering of the variables x_i, such that the matrix $\mathbf{P}\mathbf{B}_0\mathbf{P}^T$ is lower-triangular (i.e. entries above the diagonal are zero). Acyclicity also implies that the entries on the diagonal are zero, even before such a permutation.

2.2. Relation to Other Models

This model is a generalization of the linear non-Gaussian acyclic model (LiNGAM) proposed in (Shimizu et al., 2006). If the order of the autoregressive part is zero, i.e. $k = 0$, the model is nothing else than the LiNGAM model, modelling instantaneous effects only. As shown in (Shimizu et al., 2006), the assumption of non-Gaussianity of the e_i enables estimation of the model. This is because the non-Gaussian structure of the data provides information not contained in the covariance matrix which is the only source of information in most methods. In this sense the model is similar to independent component analysis, which solves the unidentifiability of factor analytic models using the assumption of non-Gaussianity of the factors (Comon, 1994; Hyvärinen et al., 2001). In fact, the estimation method in (Shimizu et al., 2006) uses an ICA algorithm as an essential part.

On the other hand, if the matrix $\mathbf{B}_0$ has all zero entries, the model in Equation (1) is a classic vector autoregressive model in which future observations are linearly predicted from preceding ones. If we knew in advance that $\mathbf{B}_0$ is zero, the model could thus be estimated by classic regression techniques since we do not have the same variables on the left and right-hand sides of Equation (1).

We emphasize that our model is different from classic autoregressive models two important ways: First, the external influences $e_i(t)$ are non-Gaussian. Second, the lag variable τ takes the value 0 as well, which brings instantaneous effects into the model in the form of the matrix $\mathbf{B}_0$. A coefficient $\mathbf{B}_0(i, j)$ models the instantaneous effect of $x_j(t)$ on $x_i(t)$ as in a linear Bayesian network, or a structural equation model.

2.3. Causality vs. Prediction

An autoregressive model can serve two different goals: prediction and analysis of causality. Our goal here is the latter: We estimate the parameter matrices $\mathbf{B}_\tau$ in order to interpret them as causal effects between the variables. This goal is distinct from simply predicting future outcomes when passively observing the time series, as has been extensively discussed in the literature on causality (Pearl, 2000; Spirtes et al., 1993). Thus, we emphasize that our model is not intended to reduce prediction errors if we want to predict $x_i(t)$ using (passively) observed values of the past $\mathbf{x}(t-1), \mathbf{x}(t-2), \ldots$; for such prediction, an ordinary autoregressive model is likely to be just as good.

Our model is intended to be superior in causal modelling. Causality has an obvious intuitive interpretation, which is typically formalized as the ability to predict the effect of possible new *interventions* on the system (Pearl, 2000). Thus, our model should be better in predicting effects of interventions, which is different from conventional time series prediction.

3. Estimation of the Model

3.1. Combining Least-Squares Estimation and LiNGAM

We propose the following method for estimating our model defined in Section 2.1. The method combines classic least-squares estimation of an autoregressive (AR) model with LiNGAM estimation:

1. Estimate a classic autoregressive model for the data
$$\mathbf{x}(t) = \sum_{\tau=1}^{k} \mathbf{M}_\tau \mathbf{x}(t-\tau) + \mathbf{n}(t) \qquad (2)$$
using any conventional implementation of a least-squares method. Note that here $\tau > 0$, so it is really a classic AR model. Denote the least-squares estimates of the autoregressive matrices by $\hat{\mathbf{M}}_\tau$.

2. Compute the residuals, i.e. estimates of innovations $\mathbf{n}(t)$
$$\hat{\mathbf{n}}(t) = \mathbf{x}(t) - \sum_{\tau=1}^{k} \hat{\mathbf{M}}_\tau \mathbf{x}(t-\tau) \qquad (3)$$

3. Perform the LiNGAM analysis (Shimizu et al., 2006) on the residuals. This gives the estimate of the matrix $\mathbf{B}_0$ as the solution of the instantaneous causal model
$$\hat{\mathbf{n}}(t) = \mathbf{B}_0 \hat{\mathbf{n}}(t) + \tilde{\mathbf{e}}(t) \qquad (4)$$

4. Finally, compute the estimates of the causal effect matrices $\mathbf{B}_\tau$ for $\tau > 0$ as
$$\hat{\mathbf{B}}_\tau = (\mathbf{I} - \hat{\mathbf{B}}_0)\hat{\mathbf{M}}_\tau \text{ for } \tau > 0 \qquad (5)$$

This estimation method is consistent,[2] as will be shown in Section 3.3. First, however, we show the derivation of Equation (5) and discuss its deep meaning.

3.2. Why Autoregressive Matrices Change due to Instantaneous Influences

Equation (5) shows a remarkable fact already mentioned in the Introduction: Consistent estimates of the $\mathbf{B}_\tau$ are not obtained by a simple AR model fit even for $\tau > 0$. Taking instantaneous effects into account changes the estimation procedure for all the autoregressive matrices, if we want consistent estimators as we usually do. Of course, this is only the case if there are instantaneous effects, i.e. $\mathbf{B}_0 \neq 0$; otherwise, the estimates are not changed.

Why do we have (5)? This is because from (1) we have
$$(\mathbf{I} - \mathbf{B}_0)\mathbf{x}(t) = \sum_{\tau=1}^{k} \mathbf{B}_\tau \mathbf{x}(t-\tau) + \mathbf{e}(t) \qquad (6)$$
and thus
$$\mathbf{x}(t) = \sum_{\tau=1}^{k} (\mathbf{I} - \mathbf{B}_0)^{-1} \mathbf{B}_\tau \mathbf{x}(t-\tau) + (\mathbf{I} - \mathbf{B}_0)^{-1} \mathbf{e}(t) \qquad (7)$$
Comparing this with (2), we can equate the autoregressive matrices, which gives $(\mathbf{I} - \mathbf{B}_0)^{-1}\mathbf{B}_\tau = \mathbf{M}_\tau$ for $\tau \geq 1$, and thus (5) is justified.

While this phenomenon is, in principle, well-known in econometric literature (Swanson & Granger, 1997; Demiralp & Hoover, 2003; Moneta & Spirtes, 2006), Equation (5) is seldom applied because estimation methods for $\mathbf{B}_0$ have not been well developed. To our knowledge, no estimation method for $\mathbf{B}_0$ has been proposed which is consistent without strong prior assumptions on $\mathbf{B}_0$.

3.3. Consistency and Identifiability

The consistency of our method relies on two facts. First, in the estimation of an AR model as in (2), it is not necessary that the innovation vector $\mathbf{n}(t)$ has independent or even uncorrelated elements (for fixed

[2] Consistency means classic statistical consistency, i.e. the estimator converges in probability to the right parameter values when the data follows the model and sample size grows infinite.

t); least-squares estimation will still be consistent, as is well known. Thus, least-squares estimation of (2), combined with (5), gives consistent estimators of $\mathbf{B}_\tau$ for $\tau \geq 1$, provided we have a consistent estimator of $\mathbf{B}_0$. Second, comparison of (7) with (2) shows that the residuals $\hat{\mathbf{n}}(t)$ are, asymptotically, of the form $(\mathbf{I} - \mathbf{B}_0)^{-1}\mathbf{e}(t)$. This means

$$\hat{\mathbf{n}}(t) = (\mathbf{I} - \mathbf{B}_0)^{-1}\mathbf{e}(t) \Leftrightarrow (\mathbf{I} - \mathbf{B}_0)\hat{\mathbf{n}}(t) = \mathbf{e}(t) \Leftrightarrow \hat{\mathbf{n}}(t) = \mathbf{B}_0\hat{\mathbf{n}}(t) + \mathbf{e}(t) \quad (8)$$

which is the LiNGAM model for $\hat{\mathbf{n}}(t)$. This shows that $\mathbf{B}_0$ is obtained as the LiNGAM analysis of the residuals, and the consistency of our estimator of $\mathbf{B}_0$ follows from the consistency of LiNGAM estimation (Shimizu et al., 2006). Thus, our method is consistent for all the $\mathbf{B}_\tau$. This obviously proves, by construction, the identifiability of the model as well.

We have here assumed that $\mathbf{B}_0$ is acyclic, as is typical in causal analysis. However, this assumption is only made because we do not know very well how to estimate a linear non-Gaussian Bayesian network in the cyclic case. Future work may produce methods which estimate cyclic models, and then we do not need the assumption of acyclicity in our combined model either. We could just use such a new method in Step 3 of the method instead of LiNGAM, and nothing else would be changed. Recent work in that direction is in (Lacerda et al., 2008); see also (Richardson & Spirtes, 1999) for older methods on Gaussian data.

3.4. Interpretation as ICA of Residuals

Another viewpoint on our model is analysis of the correlations of the innovations after estimating a classic AR model. Suppose we just estimate an AR model as in (2), and interpret the coefficients as causal effects. Such an interpretation more or less presupposes that the innovations n_i are independent of each other, because otherwise there is some structure in the model which has not been modelled by the AR model. If the innovations are not independent, the causal interpretation may not be justified. Thus, it seems necessary to further analyze the dependencies in the innovations in cases where they are strongly dependent.

Analysis of the dependency structure in the residuals (which are, by definition, estimates of innovations) is precisely what leads to the present model. As a first approach, one could consider application of something like principal component analysis or independent component analysis on the residuals. The problem with such an approach is that the interpretation of the obtained results in the framework of causal analysis would be quite difficult. Our solution is to fit a causal model like LiNGAM to the residuals, which leads to a straightforward causal interpretation of the analysis of residuals which is logically consistent with the AR model.

4. Interaction of Instantaneous and Lagged Effects

Here we present some theoretical examples of how the instantaneous and lagged effects interact based on the formula in (5).

An instantaneous effect may seem to be lagged Consider first the case where the instantaneous and lagged matrices are as follows:

$$\mathbf{B}_0 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \qquad \mathbf{B}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.9 \end{pmatrix} \quad (9)$$

That is, there is an instantaneous effect $x_2 \to x_1$, and no lagged effects (other than the purely autoregressive $x_i(t-1) \to x_i(t)$). Now, if an AR(1) model is estimated for data coming from this model, without taking the instantaneous effects into account, we get the autoregressive matrix

$$\mathbf{M}_1 = (\mathbf{I} - \mathbf{B}_0)^{-1}\mathbf{B}_1 = \begin{pmatrix} 0.9 & 0.9 \\ 0 & 0.9 \end{pmatrix} \quad (10)$$

Thus, the effect $x_2 \to x_1$ seems to be lagged although it is, actually, instantaneous.

Spurious effects appear Consider three variables with the instantaneous effects $x_1 \to x_2$ and $x_2 \to x_3$, and no lagged effects other than $x_i(t-1) \to x_i(t)$, as given by

$$\mathbf{B}_0 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{B}_1 = \begin{pmatrix} 0.9 & 0 & 0 \\ 0 & 0.9 & 0 \\ 0 & 0 & 0.9 \end{pmatrix} \quad (11)$$

If we estimate an AR(1) model for the data coming from this model, we obtain

$$\mathbf{M}_1 = (\mathbf{I} - \mathbf{B}_0)^{-1}\mathbf{B}_1 = \begin{pmatrix} 0.9 & 0 & 0 \\ 0.9 & 0.9 & 0 \\ 0.9 & 0.9 & 0.9 \end{pmatrix} \quad (12)$$

This means that the estimation of the simple autoregressive model leads to the inference of a direct lagged effect $x_1 \to x_3$, although no such direct effect exists in the model generating the data, for any time lag.

Causal ordering is not changed A more reassuring result is the following: if the data follows the same causal ordering for all time lags, that ordering is not contradicted by the neglect of instantaneous effect. A rigorous definition of this property is the following.

Theorem 1 *Assume that there is an ordering* $i(j), j = 1 \ldots n$ *of the variables such that no effect goes backward,*[3] *i.e.*

$$\mathbf{B}_\tau(i(j-\delta), i(j)) = 0 \text{ for } \delta > 0, \tau \geq 0, 1 \leq j \leq n \quad (13)$$

Then, the same property applies to the $\mathbf{M}_\tau, \tau \geq 1$ *as well. Conversely, if there is an ordering such that (13) applies to* $\mathbf{M}_\tau, \tau \geq 1$ *and* $\mathbf{B}_0$*, then it applies to* $\mathbf{B}_\tau, \tau \geq 1$ *as well.*

The proof of the theorem is based on the fact that when the variables are ordered in this way (assuming such an order exists), all the matrices $\mathbf{B}_\tau$ are lower-triangular. The same applies to $\mathbf{I} - \mathbf{B}_0$. Now, the product of two lower-triangular matrices is lower-triangular; in particular the $\mathbf{M}_\tau$ are also lower-triangular according to (5), which proves the first part of the theorem. The converse part follows from solving for $\mathbf{B}_\tau$ in (5) and the fact that the inverse of a lower-triangular matrix is lower-triangular.

What this theorem means is that if the variables really follow a single "causal ordering" for all time lags, that ordering is preserved even if instantaneous effects are neglected and a classic AR model is estimated for the data. Thus, there is some limit to how (5) can change the causal interpretation of the results.

5. Towards a Generalization of Granger Causality

The classic interpretation of causality in instantaneous Bayesian network models would be that x_i causes x_j if the (j, i)-th coefficient in $\mathbf{B}_0$ is non-zero. In the time series context, this is related to Granger causality (Granger, 1969), which formalizes causality as the ability to reduce prediction error. A simple operational definition of Granger causality can be based on the autoregressive coefficients $\mathbf{M}_\tau$: If at least one of the coefficients from $x_i(t-\tau), \tau \geq 1$ to $x_j(t)$ is (significantly) non-zero, then x_i Granger-causes x_j. This is because then the variable x_i reduces the prediction error in x_j in the mean-square sense if it is included in the set of predictors, which is the very definition of Granger causality.

In light of the results in this paper, we propose a definition which combines the two aspects: A variable x_i causes x_j if at least one of the coefficients $\mathbf{B}_\tau(j, i)$, giving the effect from $x_i(t-\tau)$ to $x_j(t)$, is (significantly) non-zero for $\tau \geq 0$. The condition for τ is different from Granger causality since the value $\tau = 0$, corresponding to instantaneous effects, is included. Moreover, since estimation of the instantaneous effects changes the estimates of the lagged ones, the lagged effects used in our definition are different from those usually used with Granger causality.

A more general formulation of this definition, which is in line with the general formulation of Granger causality, is that the error in the "prediction" of $x_j(t)$ is reduced when $x_i(t-1), x_i(t-2), \ldots$ *and* $x_i(t)$ are included in the set of predictors. Here, we use a rather unconventional definition of the word "prediction" because we include instantaneous effects.

6. Simulations

To verify the validity of our method, we first performed experiments with artificial data. In the experiments, we created data in the following manner using the LiNGAM code package[4]:

1. We randomly constructed a strictly lower-triangular matrix (i.e. zero entries above and on the diagonal), $\mathbf{B}_0$, for the instantaneous causal model so that the standard deviations of the innovations n_i owing to parent innovations will be in the interval $[0.5, 1.5]$. The number of observed time-series was $n = 10$. Both fully connected (no zeros in the strictly lower triangular part) and sparse networks (many zeros) were created. We also randomly selected the standard deviations of the external influences e_i from the interval $[0.5, 1.5]$.

2. Next, we generated data with various lengths of the time series (300, 500 and 1,000) by independently drawing the external influences e_i from various non-Gaussian distributions with zero mean and unit variance[5]. The values of the innovations n_i were generated according to the assumed instantaneous recursive process. This is straightforward because $\mathbf{B}_0$ is lower-triangular, so we just generate the n_i in the order $n_1, n_2 \ldots$ as is typical in acyclic networks, e.g. (Shimizu et al., 2006).

3. We randomly permuted the order of the innovations n_i to hide the causal order with which the data was generated. We also permuted $\mathbf{B}_0$ as well

[3] In the purely instantaneous case, existence of such an ordering is equivalent to acyclicity of the effects as noted in Section 2.1.

[4] http://www.cs.helsinki.fi/group/neuroinf/lingam/

[5] We first generated a gaussian variable z with zero mean and unit variance and subsequently transformed it to a non-Gaussian variable by $e_i = \text{sign}(z)|z|^q$. The nonlinear exponent q was selected to lie in $[0.5, 0.8]$ or $[1.2, 2.0]$. The former gave a sub-gaussian variable, and the latter a super-gaussian variable. Finally, the transformed variable was standardized to have zero mean and unit variance.

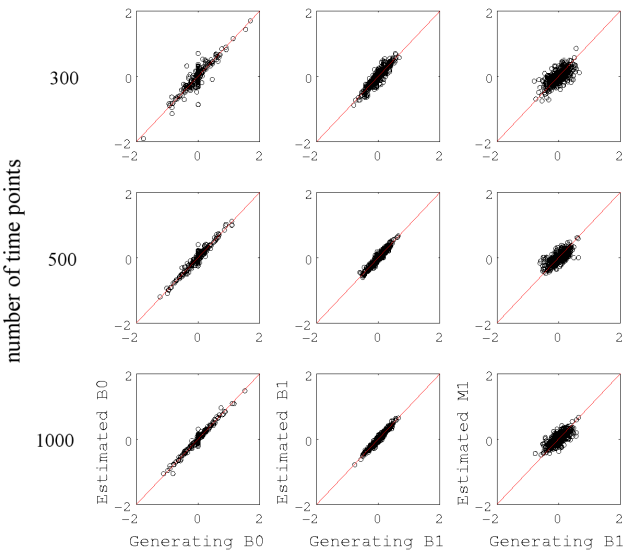


Figure 1. Simulations on artificial data. Left column: Scatterplots of the estimated elements of $\mathbf{B}_0$ versus the generating values. Center column: Scatterplots of the estimated elements of $\mathbf{B}_1$ versus the generating values. Right column: Scatterplots of the estimated elements of $\mathbf{M}_1$ versus those of $\mathbf{B}_1$.The number of observed signals was 10. Five data sets were generated for each scatterplot.

as the variances of the external influences e_i to match the new order.

4. We randomly generated a first-order autoregressive matrix $\mathbf{M}_1$ so that the spectral norm of the matrix was less than 0.99 to ensure the stability of the autoregressive process.

5. The values of the observed signals $x_i(t)$ were generated according to the assumed first-order autoregressive process.

6. Finally, we fed the data to our estimation method. Here we told the method that the generating autoregressive order was 1.

Figure 1 gives the scatterplots of the elements of the estimated parameters versus the generating ones. The left column is for the scatterplots of the estimated causal effects in $\mathbf{B}_0$ versus the generating values. The center column is for the scatterplots of the estimated causal effects in $\mathbf{B}_1$ versus the generating values. The right column is for the scatterplots of the estimated autoregressive coefficients in $\mathbf{M}_1$ versus the generating values of the causal effects in $\mathbf{B}_1$ (here, the estimation was invalid because instantaneous effects were ignored).

For the scatterplots in the left and center columns, the estimation worked well when the sample size grew, as evidenced by the grouping of the data points onto the main diagonal, although for the small sample size 300 the estimation was often inaccurate. On the other hand, the scatterplots in the right column confirmed that the causal effects were not correctly estimated by the ordinary autoregressive coefficients when instantaneous influences existed since the data points were not very close to the main diagonal.

7. Experiments on Real Data

7.1. Financial Data

As a first illustration of the applicability of the method on real data, we analyzed a dataset from a time series repository on the Internet.[6] The data consisted of two observed signals, x_1: weekly closing price of Toyota stock and x_2: weekly closing rate of exchange of Japanese Yen to U.S. Dollar in 2007. The number of time points was 50. The maximum, minimum and mean of x_1 were 8,230, 5,870 and 7,102 (JPY). Those of x_2 were 123.86, 108.51 and 117.72 (JPY).

We analyzed the data using our method with autoregressive order of 1. The estimated first-order autoregressive matrix $\mathbf{M}_1$ and residual correlation matrix were as follows:

$$\mathbf{M}_1 = \begin{pmatrix} 0.95 & -4.22 \\ 0.0008 & 0.78 \end{pmatrix} \tag{14}$$

$$\mathrm{corr}(\mathbf{n}) = \begin{pmatrix} 1.00 & 0.66 \\ 0.66 & 1.00 \end{pmatrix}$$

The relatively strong correlation between the residuals implied that there would be some dependency that had not been modeled by the AR model. Thus, we fitted the instantaneous causal model to the residuals, as proposed above. The estimated instantaneous causal effect matrix $\mathbf{B}_0$ and resulting lagged causal effect matrix $\mathbf{B}_1$ were as follows:

$$\mathbf{B}_0 = \begin{pmatrix} 0 & 56.04 \\ 0.0027 & 0 \end{pmatrix} \tag{15}$$

$$\mathbf{B}_1 = \begin{pmatrix} 0.91 & -48.01 \\ -0.0018 & 0.79 \end{pmatrix} \tag{16}$$

The matrix $\mathbf{B}_0$ is very close to be upper-triangular, which implied that the model was really acyclic (because switching the order of the variables would make $\mathbf{B}_0$ lower-triangular). Further, the instantaneous effect $x_2 \to x_1$ in $\mathbf{B}_0$ was one order of magnitude larger than the lagged effect in $\mathbf{M}_1$ and thus the lagged co-

[6] Yahoo! Japan Finance: http://quote.yahoo.co.jp/

efficients in $\mathbf{M}_1$ are quite different from those in $\mathbf{B}_1$, due to the formula in (5).

Figure 2 shows a graphical representation of the estimated model for financial data. First, it implies that a higher value of the yen (x_2) had a negative lagged effect (-48.01) on the price of Toyota stock (x_1). This would be reasonable since Toyota sells many cars abroad, and a higher value of the yen would increase the cost price and decrease the earning. Interestingly, it was also implied that a higher value of the yen had a positive instantaneous effect (56.04) on the price of Toyota stock. In other words, for weeks where values of the yen one week before were the (approximately) same, if the yen got more expensive (due to some reason other than the value of the yen one week before, perhaps a U.S. recession, for example) then the price of Toyota stock would get more expensive. It would be interesting to further study the economic mechanism with more extensive data.

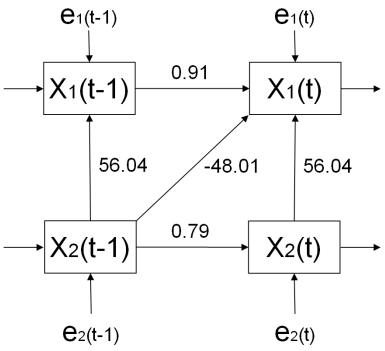


Figure 2. A graphical representation of the model estimated in Section 7.1. The x_1 and x_2 denote weekly closing price of Toyota stock in 2007 and weekly closing rate of exchange of Japanese Yen to U.S. Dollar in 2007, respectively. The arrow from $x_1(t-1)$ to $x_2(t)$ was omitted since the estimated strength was very close to zero (-0.0018).

7.2. Magnetoencephalographic Data

As a second illustration of the applicability of the method on real data, we applied it on magnetoencephalography (MEG), i.e. measurements of the electric activity in the brain. The raw data consisted of the 306 MEG channels measured by the Vectorview helmet-shaped neuromagnetometer (Neuromag Ltd., Helsinki, Finland) in a magnetically shielded room at the Brain Research Unit, Low Temperature Laboratory, Helsinki University of Technology. The sampling frequency was 600 Hz. The measurements consisted of 300 seconds of resting state brain activity from the experiment of (Ramkumar et al., 2007). The subject was sitting with eyes closed, and did not perform any specific task nor was there any specific sensory stimulation. The channels were first linearly projected to the signal space to reduce noise (Uusitalo & Ilmoniemi, 1997). In this illustrative experiment, we only consider a single (gradiometer) channel in the right occipital cortex near the midline.

We considered the interaction of about 10 Hz (alpha) and about 20 Hz (beta) oscillations commonly observed in electromagnetic recordings of spontaneous brain activity. We first computed the amplitudes of the oscillations by dividing the data into windows of length of 0.25 seconds, performing fast Fourier transform inside each of them, and computing the total Fourier amplitudes (unweighted Euclidean norm of the Fourier coefficients) in the frequency ranges of $8 \ldots 12$Hz (alpha range, denoted by x_1) and $15 \ldots 25$Hz (beta range, denoted by x_2). Thus we obtained two time series of 1,200 points.

We fitted our model, with autoregressive order of 1 to the data. The obtained matrices are

$$\mathbf{M}_1 = \begin{pmatrix} 0.23381 & 0.14551 \\ 0.10838 & 0.14314 \end{pmatrix} \tag{17}$$

$$\mathbf{B}_0 = \begin{pmatrix} 0 & -0.65768 \\ 0.56722 & 0 \end{pmatrix} \tag{18}$$

$$\mathbf{B}_1 = \begin{pmatrix} 0.30509 & 0.23965 \\ -0.024244 & 0.060608 \end{pmatrix} \tag{19}$$

What we see is that the instantaneous model is far from trivial: the effects in $\mathbf{B}_0$ are relatively strong. This is also reflected in $\mathbf{B}_1$ which is now rather different from $\mathbf{M}_1$. Thus, the interpretation of the autoregressive matrices using just the autoregressive model (i.e. $\mathbf{M}_1$) or the combined model (i.e. $\mathbf{B}_1$) are quite different. In the classic autoregressive case (based on $\mathbf{M}_1$), the lagged effect $x_1 \to x_2$ is relatively strongly positive whereas in the combined model it is quite weak. In fact, that effect is now modelled as an instantaneous effect in $\mathbf{B}_0$. Even more interesting is that the instantaneous model has a strong negative effect $x_2 \to x_1$ which is not visible at all in the purely autoregressive matrix $\mathbf{M}_1$. Thus, the results illustrate how the interpretation of causal effects (and even of the lagged ones) can change drastically when including the instantaneous effects.

Using an autoregressive order of 2 did not change the results. We also ran the method many times to exclude the problem of the ICA estimation algorithm (used in LiNGAM estimation) getting stuck in local minima (Himberg et al., 2004), and the result was found to be robust with respect to that manipulation.

One problem with this experiment is that the causal model estimated by LiNGAM is far from acyclic. Here, we can justify the procedure by using the theory of cyclic model estimation proposed by (Lacerda et al., 2008); the estimation here gives the only "stable" model according to that theory. Performance of LiNGAM estimation methods in the case of cyclic models, and the possible need for new methods for estimating cyclic models are future research topics of great practical importance. However, as discussed above, they are separate from the main contribution of our paper in the sense that we can use any such new method to estimate the instantaneous model in our framework.

8. Conclusion

We showed how non-Gaussianity enables estimation of a causal discovery model in which the linear effects can be either instantaneous or time-lagged. Like in the purely instantaneous case (Shimizu et al., 2006), non-Gaussianity makes the model identifiable without explicit prior assumptions on existence or non-existence of given causal effects. The classic assumption of acyclicity is sufficient although probably not necessary. From the practical viewpoint, an important implication is that considering instantaneous effects changes the coefficient of the time-lagged effects as well.

References

Bollen, K. A. (1989). *Structural equations with latent variables*. John Wiley & Sons.

Comon, P. (1994). Independent component analysis—a new concept? *Signal Processing*, *36*, 287–314.

Demiralp, S., & Hoover, K. D. (2003). Searching for the causal structure of a vector autoregression. *Oxford Bulletin of Economics and Statistics*, *65 (supplement)*, 745–767.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, *37*, 424–438.

Himberg, J., Hyvärinen, A., & Esposito, F. (2004). Validating the independent components of neuroimaging time-series via clustering and visualization. *NeuroImage*, *22*, 1214–1222.

Hyvärinen, A., Karhunen, J., & Oja, E. (2001). *Independent component analysis*. Wiley Interscience.

Kim, J., Zhu, W., Chang, L., Bentler, P. M., & Ernst, T. (2007). Unified structural equation modeling approach for the analysis of multisubject, multivariate functional MRI data. *Human Brain Mapping*, *28*, 85–93.

Lacerda, G., Spirtes, P., Ramsey, J., & Hoyer, P. O. (2008). Discovering cyclic causal models by independent components analysis. *Proc. 24th Conf. on Uncertainty in Artificial Intelligence (UAI2008)*. Helsinki, Finland.

Moneta, A., & Spirtes, P. (2006). Graphical models for the identication of causal structures in multivariate time series models. *Proc. Joint Conference on Information Sciences*. Kaohsiung, Taiwan.

Opgen-Rhein, R., & Strimmer, K. (2007). From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC Systems Biology*, *1*.

Pearl, J. (2000). *Causality: Models, reasoning, and inference*. Cambridge University Press.

Ramkumar, P., Parkkonen, L. T., He, B. J., Raichle, M. E., Hämäläinen, M. S., & Hari, R. (2007). Identification of stimulus-related and intrinsic networks by spatial independent component analysis of MEG signals. Abstract presented at the Society for Neuroscience Meeting, San Diego, California.

Richardson, T. S., & Spirtes, P. (1999). Automated discovery of linear feedback models. In C. Glymour and G. Cooper (Eds.), *Computation, causation and discovery*, 253–302. The MIT Press.

Roebroeck, A., Formisano, E., & Goebel, R. (2005). Mapping directed influence over the brain using granger causality and fMRI. *NeuroImage*, *25*, 230–242.

Shimizu, S., Hoyer, P. O., Hyvärinen, A., & Kerminen, A. (2006). A linear non-Gaussian acyclic model for causal discovery. *J. of Machine Learning Research*, *7*, 2003–2030.

Spirtes, P., Glymour, C., & Scheines, R. (1993). *Causation, prediction, and search*. Springer-Verlag.

Swanson, N. R., & Granger, C. W. J. (1997). Impulse response functions based on a causal approach to residual orthogonalization in vector autoregression. *J. of the Americal Statistical Association*, *92*, 357–367.

Uusitalo, M. A., & Ilmoniemi, R. J. (1997). Signal-space projection method. *Med. Biol. Eng.*, *32*, 35–42.

Hierarchical Model-Based Reinforcement Learning: R-MAX + MAXQ

Nicholas K. Jong NKJ@CS.UTEXAS.EDU
Peter Stone PSTONE@CS.UTEXAS.EDU
The University of Texas at Austin, 1 University Station, Austin, TX 78712 USA

Abstract

Hierarchical decomposition promises to help scale reinforcement learning algorithms naturally to real-world problems by exploiting their underlying structure. Model-based algorithms, which provided the first finite-time convergence guarantees for reinforcement learning, may also play an important role in coping with the relative scarcity of data in large environments. In this paper, we introduce an algorithm that fully integrates modern hierarchical and model-learning methods in the standard reinforcement learning setting. Our algorithm, R-MAXQ, inherits the efficient model-based exploration of the R-MAX algorithm and the opportunities for abstraction provided by the MAXQ framework. We analyze the sample complexity of our algorithm, and our experiments in a standard simulation environment illustrate the advantages of combining hierarchies and models.

1. Introduction

Reinforcement Learning (RL) algorithms tackle a very challenging problem: how to find rewarding behaviors in unknown environments (Sutton & Barto, 1998). An important goal of RL research is to generalize these algorithms to structured representations and to learn from limited experience. In this paper, we develop an algorithm that integrates two important branches of RL research that, despite their popularity, have rarely been studied in tandem.

The first of these two branches is hierarchical RL. Humans cope with the extraordinary complexity of the real world in part by thinking hierarchically, and we would like to imbue our learning algorithms with the same faculty. In the RL community, this impetus has taken shape as work on temporal abstraction, in which temporally extended abstract actions allow agents to reason above the level of primitive actions (Barto & Mahadevan, 2003). The advantages of such methods include the ability to incorporate prior knowledge and the creation of opportunities for state abstraction. Recent work in the automatic discovery of hierarchy has focused on the ability to focus exploration in novel regions of the state space (Şimşek & Barto, 2004).

The second branch is model-based RL, which directly estimates a model of the environment and then plans with this model. Early work demonstrated that summarizing an agent's experience into a model could be an efficient way to reuse data (Moore & Atkeson, 1993), and later work utilized the uncertainty in an agent's model to guide exploration, yielding the first (probabilistic) finite bounds on the amount of data required to learn near-optimal behaviors in the general case (Kearns & Singh, 1998; Kakade, 2003).

Few RL researchers have tried to combine these two approaches, despite the intuitive appeal of learning hierarchical models of the world. Prior work includes adaptations to the average-reward formulation (Seri & Tadepalli, 2002) and to deterministic domains (Diuk et al., 2006). In this paper, we introduce an algorithm that fully integrates modern hierarchical-decomposition and model-learning methods in the standard setting of discounted rewards and stochastic dynamics. Section 2 details how we decompose high-level models into lower-level models. Section 3 presents our algorithm, which joins our model decomposition with the R-MAX approach to learning primitive models. In Section 4, we formally analyze our algorithm, R-MAXQ. Section 5 describes our empirical results. In Section 6 we discuss related work more fully, and in Section 7 we conclude.

2. Hierarchies of Models

We begin by describing our recursive action decomposition, which defines how we plan at the high level given learned models of primitive actions. Section 3 presents a complete algorithm obtained by combining this decomposition with a particular way of learning primitive models.

We adopt the standard semi-Markov decision process (SMDP) formalism for describing temporally extended actions (Sutton et al., 1999), but we modify the notation to

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

better reflect the recursive nature of hierarchical RL. We define an SMDP as the conjunction $\langle S, A\rangle$ of a finite state space S and a finite action set A. Each action $a \in A$ is associated with a transition function P^a and a reward function R^a. For convenience, we use a *multi-time* model (Sutton et al., 1999), so $P^a(s,s') = \sum_{k=1}^{\infty} \gamma^k \Pr(k, s' \mid s, a)$, where $\gamma \in (0,1)$ is a discount factor and $\Pr(k, s' \mid s, a)$ is the probability that executing action $a \in A$ in state $s \in S$ will take exactly k time steps and terminate in state $s' \in S$. Similarly, $R^a(s) = E\left[\sum_{k=0}^{\infty} \gamma^k r_k\right]$, where r_k is the one-step reward earned during the kth time step executing a.

If $a \in A$ is a *primitive action*, then it will always terminate after exactly one time step, so $\sum_{s'} P^a(s,s') = \gamma$ for all $s \in S$. Since we may construe a discount factor of γ as equivalent to terminating a trajectory with probability $1-\gamma$ after each time step, the "missing" transition probability corresponds to the probability of termination.

In the RL setting, each P^a and R^a is initially unknown, but for each $a \in A$ that is a *composite action*, we assume the agent is given a set of terminal states $T^a \subset S$, a set of child actions A^a, and a goal reward function $\tilde{R}^a : T^a \to \mathbb{R}$. A composite action a may be invoked in any state $s \in S \setminus T^a$, and upon reaching a state $s' \in T^a$ it terminates and earns an internal reward of $\tilde{R}^a(s')$. It executes by repeatedly choosing child actions $a' \in A^a$ to invoke. The child actions a' may be primitive or composite. When a' terminates (and assuming a does not terminate), then a selects another child action. (In contrast to the original MAXQ framework, a composite action a only tests for termination upon the termination of a child action a'.) A composite action a selects child actions to maximize the expected sum of the child action rewards $R^{a'}$ and the goal rewards $\tilde{R}^a$.

Given the transition and reward functions for each of the child actions, the optimal policy for the composite action a may be computed using the following system of Bellman equations, for all $s \in S$ and $a' \in A^a$:

$$Q^a(s,a') = R^{a'}(s) + \sum_{s' \in S} P^{a'}(s,s') V^a(s') \quad (1)$$

$$V^a(s) = \begin{cases} \tilde{R}^a(s), & \text{if } s \in T^a \\ \max_{a' \in A^a(s)} Q^a(s,a'), & \text{otherwise,} \end{cases} \quad (2)$$

where $A^a(s) = \left\{ a' \in A^a \mid \text{primitive}(a') \vee s \notin T^{a'} \right\}$. Then the optimal policy $\pi^a : S \to A^a$ is, for all $s \in S$:

$$\pi^a(s) = \text{argmax}_{a' \in A^a(s)} Q^a(s,a'). \quad (3)$$

Dietterich's MAXQ framework computes $Q^a(s,a')$ by decomposing this quantity into $Q^a(s,a') = R^{a'}(s) + C^a(s,a')$, where C^a is a *completion function* that estimates the reward obtained after executing a' but before completing a. It recursively queries the child action for $R^{a'}$ and learns C^a locally using model-free stochastic approximation. Using the learned π^a, it simultaneously learns an external version of C^a that doesn't include the internal goal rewards $\tilde{R}^a$, so that a can report R^a to its own parents.

The key idea behind our model-based approach is to assume that a composite action a can query a child a' for not just $R^{a'}$ but also $P^{a'}$. Then the only unknown quantity in Equation 1 is V^a, which can be computed using standard dynamic programming methods and stored locally. To satisfy our assumption, each action a, whether primitive or composite, must be able to compute both R^a and P^a. Prior research into option models (Sutton et al., 1999) defined Bellman-like equations, for all $s \in S$ and $x \in T^a$:

$$R^a(s) = R^{\pi^a(s)}(s) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s,s') R^a(s') \quad (4)$$

$$P^a(s,x) = P^{\pi^a(s)}(s,x) + \sum_{s' \in S \setminus T^a} P^{\pi^a(s)}(s,s') P^a(s',x), \quad (5)$$

and for all $s \in S$ and $x \in S \setminus T^a$, $P^a(s,x) = 0$. Since P^a is a multi-time model, note that $\sum_{s'} P^a(s,s') < \gamma < 1$, where the "missing" transition probability corresponds to the cumulative $1-\gamma$ probability of terminating (the entire trajectory, not just a) marginalized over the random duration of the execution of a. A key strength of our algorithm is that it takes advantage of models to solve Equations 4 and 5 directly using dynamic programming, instead of using these equations to define update rules for stochastic approximation, as in prior work with option models.

Our decomposition provides a way to compute policies and therefore high-level transition and reward models given lower-level transition and reward models. To ground out this process, models of the primitive actions must be available. However, if R^a and P^a are available for each primitive action a, note that we could compute the optimal policy of the system using standard (non-hierarchical) planning algorithms. Nevertheless, we empirically demonstrate the benefit of using hierarchies in Section 5. The next section first presents our learning algorithm.

3. The R-MAXQ Algorithm

Equations 1–5 show how to compute abstract models from primitive models, but a complete model-based RL algorithm must specify how to estimate the primitive models. We propose combining our hierarchical model decomposition, inspired by the MAXQ value function decomposition, with the primitive models defined by the R-MAX algorithm (Brafman & Tennenholtz, 2002), yielding a new algorithm we call R-MAXQ.

R-MAX defines the transition and reward models for primitive actions as follows. Let $n(s,a)$ denote the number of times primitive action a has executed in state s. Let

$n(s,a,s')$ denote the number of times primitive action a transitioned state s to state s'. Finally, let $r(s,a)$ denote the cumulative one-step reward earned by each execution of primitive action a in state s. Then the primitive transition and reward models are given by:

$$R^a(s) = \begin{cases} \frac{r(s,a)}{n(s,a)}, & \text{if } n(s,a) \geq m \\ V^{\max}, & \text{otherwise,} \end{cases} \quad (6)$$

$$P^a(s,s') = \begin{cases} \frac{n(s,a,s')}{n(s,a)}, & \text{if } n(s,a) \geq m \\ 0, & \text{otherwise,} \end{cases} \quad (7)$$

where $V^{\max}$ is an upper bound on the optimal value function and m is a threshold sample size.[1] Given sufficient data, R-MAX uses the maximum likelihood model, but it otherwise uses an optimistic model that predicts a high-reward terminal transition.[2] By backing up these optimistic rewards through the value function, the learned policy effectively plans to visit insufficiently explored states.

R-MAXQ works in the same way, except it computes a hierarchical value function using its model decomposition instead of a monolithic value function using the standard MDP model. Optimistic rewards propagate not only through the value function V^a at a given composite action a but also up the hierarchy, via each action's computed abstract reward function R^a. Each local policy implicitly exploits or explores by choosing a child action with high apparent value, which combines the child's actual value and possibly some optimistic bonus due to some reachable unknown states. No explicit reasoning about exploration is required at any of the composite actions in the hierarchy: as in R-MAX, the planning algorithm is oblivious to its role in balancing exploration and exploitation in a learning agent. A key advantage of R-MAXQ is that its hierarchy allows it to constrain the agent's policy in a fashion that may reduce unnecessary exploratory actions, as illustrated in Section 5.

Algorithms 1–4 give the R-MAXQ algorithm in detail. All variables are global, except for the arguments a and s, which represent the action and state passed to each subroutine. All global variables are initialized to 0, except that $R^a(s)$ is initialized to $V^{\max}$ for all primitive actions a and states s. Algorithm 1 is the main algorithm, invoked with the root action in the hierarchy and the initial state of the system. MAXQ recursively executes an action a in the current state s, returning the resulting state $s' \in T^a$. Primitive actions execute blindly; composite actions first update their policy and then choose a child action to execute, until some child leaves it in a terminal state.

Algorithm 1 R-MAXQ(a,s)

```
if a is primitive then
  Execute a, obtain reward r, observe state s'
  r(s,a) ← r(s,a) + r                    {record primitive data}
  n(s,a) ← n(s,a) + 1
  n(s,a,s') ← n(s,a,s') + 1
  t ← t + 1
  Return s'
else {a is composite}
  repeat
    COMPUTE-POLICY(a,s)
    s ← R-MAXQ(π^a(s), s)                {recursive execution}
  until s ∈ T^a                          {or episode ends}
  Return s
end if
```

Algorithm 2 updates the policy for composite action a given that the agent is in state s. It first constructs a *planning envelope*: all the states reachable from s (at this node of the hierarchy) and thus relevant to the value of s. Once the planning envelope has been computed and all the child actions' models have been updated on the envelope, the value function and policy could be computed using value iteration. Note that our implementation actually uses prioritized sweeping (Moore & Atkeson, 1993) and aggressive memoization, not shown in our pseudocode, to ameliorate the computational burden of propagating incremental model changes throughout the hierarchy.

Algorithm 2 COMPUTE-POLICY(a,s)

```
if timestamp(a) < t then
  timestamp(a) ← t
  envelope(a) ← ∅
end if
PREPARE-ENVELOPE(a,s)
while not converged do {value iteration}
  for all s' ∈ envelope(a) do
    for all a' ∈ A^a(s') do
      Set Q^a(s',a') using Eq. 1
    end for
    Set V^a(s') using Eq. 2
  end for
end while
π^a(s) ← argmax_{a'∈A^a(s)} Q^a(s,a')   {Eq. 3}
```

Algorithm 3 computes the planning envelope for composite action a by examining the given state's successors under any applicable child action's transition model and recur-

[1] The original Prioritized Sweeping algorithm (Moore & Atkeson, 1993) used the same optimistic one-step model, but its name became identified with its method for propagating changes throughout the value function. The primary contribution of the R-MAX algorithm was a derivation of the appropriate value of m given the desired error bounds.

[2] In effect, setting all the transition probabilities to 0 in Equation 5 gives the "missing" probability all to the implicit terminal state. This trick works properly with the Bellman equations since the terminal state has value 0; the optimism is reflected in the reward for transitioning into this state.

sively adding any new states to the envelope. This computation requires that these models be updated, if necessary.

Algorithm 3 PREPARE-ENVELOPE(a, s)

```
if s ∉ envelope(a) then
  envelope(a) ← envelope(a) ∪ {s}
  for all a' ∈ A^a(s) do
    COMPUTE-MODEL(a', s)
    for all s' ∈ S | P^{a'}(s, s') > 0 do
      PREPARE-ENVELOPE(a, s')
    end for
  end for
end if
```

Algorithm 4 updates the model for an action a at some state s. For composite actions, this requires recursively computing the policy and then solving Equations 4 and 5.

Algorithm 4 COMPUTE-MODEL(a, s)

```
if a is primitive then
  if n(s, a) ≥ m then
    R^a(s) ← r(s,a)/n(s,a)                          {Eq. 6}
    for all s' ∈ S do
      P^a(s, s') ← n(s,a,s')/n(s,a)                 {Eq. 7}
    end for
  end if
else {a is composite}
  COMPUTE-POLICY(a, s)
  while not converged do {dynamic programming}
    for all s' ∈ envelope(a) do
      Set R^a(s') using Eq. 4
      for all x ∈ T^a do
        Set P^a(s', x) using Eq. 5
      end for
    end for
  end while
end if
```

4. Analysis of R-MAXQ

We now provide a very rough sketch of our main theoretical result: R-MAXQ probably follows an approximately optimal policy for all but a finite number of time steps. Unfortunately, this number may be exponential in the size of the hierarchy. This section closes with a brief discussion of the implications of this result.

The original R-MAX algorithm achieves efficient exploration by using an optimistic model. Its model of any given state-action pair is optimistic until it samples that state-action m times. By computing a value function from this optimistic model, the resulting policy implicitly trades off exploration (when the value computed for a given state includes optimistic rewards) and exploitation (when the value only includes estimates of the true rewards). Kakade (2003) bounded the sample complexity of RL by first showing that R-MAX probably only spends a finite number of time steps attempting to reach optimistic rewards (exploring). For the remaining (unbounded) number of time steps, the algorithm exploits its learned model, but its exploitation is near-optimal only if this model is sufficiently accurate. Kakade then bounded the values of m necessary to ensure the accuracy of the model with high probability.

To be precise, let an MDP with finite state space S and finite action space A be given. Let ϵ be a desired error bound, δ the desired probability of failure, and γ the discount factor. Then R-MAX applied to an arbitrary initial state will spend $O\left(\frac{m|S||A|L}{\epsilon}\log\frac{|S||A|}{\delta}\right)$ time steps exploring, with probability greater than $1-\frac{\delta}{2}$, where $L = O\left(\frac{\log\epsilon}{1-\gamma}\right)$. Furthermore, there exists an $m \in O\left(\frac{|S|L^2}{\epsilon^2}\log\frac{|S||A|}{\delta}\right)$ such that when the agent is not exploring, $V^{\pi^*}(s_t) - V^{\pi_t}(s_t) \leq \frac{\epsilon}{1-\gamma}(R^{\max} - R^{\min})$ with probability greater than $1-\frac{\delta}{2}$, where s_t and π_t are the current state and policy at time t, and $R^{\max}$ and $R^{\min}$ bound the reward function.

The hierarchical decomposition used by R-MAXQ complicates an analysis of its sample complexity, but essentially the same argument that Kakade used provides a loose bound. We refer the interested reader to the proof of Kakade (2003) for the gross structure of the argument, and we merely sketch the necessary extensions here. A key lemma is Kakade's ϵ-approximation condition (Lemma 8.5.4). The transition model $\hat{P}$ for an action is an ϵ-approximation for the true dynamics P if for all states $s \in S$, $\sum_{s'\in S}\left|\hat{P}(s,s') - P(s,s')\right| < \epsilon$. The ϵ-approximation condition states that if a model has the correct reward function but only an ϵ-approximation of the transition dynamics for each action, then for all policies π and states $s \in S$, $\left|\hat{V}^{\pi}(s) - V^{\pi}(s)\right| < \frac{\epsilon L}{1-\gamma}$.

Essentially, this condition relates the error bounds in the model approximation to the resulting error bounds in the computed value function. It allows the analysis of R-MAX to determine a sufficient value of m to achieve the desired degree of near optimality. We must extend this condition in two ways to adapt the overall proof to R-MAXQ. First, R-MAXQ violates Kakade's assumption of deterministic reward functions. Define a model reward function $\hat{R}$ to be a λ-approximation of the true reward function R if for all states $s \in S$, $\left|\hat{R}(s) - R(s)\right| < \lambda$. Then it is straightforward to adjust Kakade's derivation of the ϵ-approximation condition to show that the computed value function for any given policy satisfies $s \in S$, $\left|\hat{V}^{\pi}(s) - V^{\pi}(s)\right| < \frac{\epsilon L}{1-\gamma} + \lambda$.

Second, for a given composite action a, we must relate error bounds in the approximations of $R^{a'}$ and $P^{a'}$ for each child $a' \in A^a$ to error bounds in the approximations of R^a and P^a. Since R^a is just the value function for π^a but without the goal rewards (Equation 4), we immediately obtain that the estimated R^a will be an $\left(\frac{\epsilon L}{1-\gamma} + \lambda\right)$-approximation. Equation 7 illustrates that for every $s' \in T^a$, $P^a(\cdot, s')$ can be thought of as a value function estimating the expected cumulative discounted probability of transitioning into s'. The total error in $P^a(s, \cdot)$ will be bounded by the sum of the errors for each $s' \in T^a$, so it can be shown that P^a is an $O\left(\frac{|T^a|\epsilon L}{1-\gamma}\right)$-approximation.

These results bound the errors that propagate up from the primitive actions in the hierarchy, but these bounds seem quite loose. In particular, these bounds can't rule out the possibility that each level of the hierarchy might multiply the approximation error by a factor of $\frac{|T^a|L}{1-\delta}$. Since the amount of data required varies as the inverse square of ϵ, if R-MAX requires m samples of each action at each state to achieve a certain error bound, R-MAXQ may require $m' = O\left(m\left(\frac{TL}{1-\delta}\right)^{2h}\right)$ samples of each primitive action at each state to achieve the same error bound at the root of the hierarchy, where T is the maximum number of reachable terminal states for any composite action and h is the height of the hierarchy: the number of composite tasks on the longest path from the root of the hierarchy to a primitive action (not including the root itself).

By adapting the remainder of Kakade's proof, we can establish that R-MAXQ will probably converge to a (recursively) near-optimal policy, although this guarantee requires exponentially more data than R-MAX in the worst case. We note that this guarantee applies to any choice of hierarchy. It remains to be seen whether it might be possible to derive tighter bounds for specific classes of action hierarchies. Furthermore, as Kakade (2003) notes in his derivation, the ϵ-approximation condition is perhaps unnecessarily stringent, since it gives the worst possible degradation in approximation quality over all possible policies.

In practice, implementations of R-MAX use far smaller values of m than would be required to achieve useful theoretical guarantees. In this vein, we note that running R-MAXQ will result in no more time spent in exploration than running R-MAX with the same value for m. The hierarchical decomposition only weakens the guarantees on the near-optimality of the policy that R-MAXQ exploits. The experiments described in the next section show that a good hierarchy can even reduce the amount of time spent exploring, with no appreciable deterioration in solution quality.

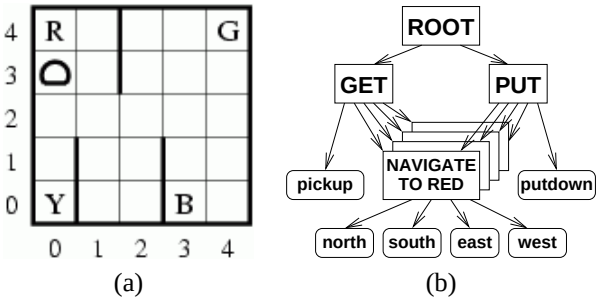


Figure 1. (a) Taxi domain, and (b) an action hierarchy for Taxi

5. Experiments

This section presents our empirical results, which show that R-MAXQ outperforms both of its components, R-MAX and MAXQ-Q. We discuss our findings in detail, to reveal how precisely our algorithm benefits from combining model-based learning and hierarchical decomposition.

For our experiments, we use the familiar Taxi domain (Dietterich, 2000). This domain consists of a 5×5 gridworld with four landmarks, labeled `red`, `blue`, `green`, and `yellow`, illustrated in Figure 1a. The agent is a taxi that must navigate this gridworld to pick up and deliver a passenger. The system has four state variables and six primitive actions. The first two state variables, `x` and `y`, give the coordinates of the taxi in the grid. The third, `passenger`, gives the current location of the passenger as one of the four landmarks or as `taxi`, if the passenger is inside the taxi. The final state variable, `destination`, denotes the landmark where the passenger must go. Four primitive actions, `north`, `south`, `east`, and `west`, each move the taxi in the indicated direction with probability 0.8 and in each perpendicular direction with probability 0.1. The `pickup` action transfers the passenger into the taxi if the taxi is at the indicated landmark. The `putdown` action ends an episode if the passenger is in the taxi and the taxi is at the desired destination. Each episode begins with the taxi in a random location, the passenger at a random landmark, and a destination chosen randomly from the remaining landmarks. Each action incurs a -1 penalty, except that unsuccessful `pickup` and `putdown` actions cost -10, and a successful `putdown` action earns a reward of 20.

The structure of the Taxi domain makes it conducive for research into hierarchical RL. The optimal policy may be described abstractly in four steps. First, navigate to the landmark where the passenger is. Second, pick up the passenger. Third, navigate to the destination landmark. Finally, put down the passenger. Navigation to each of the landmarks constitute reuseable subtasks that hierarchical algorithms can exploit. Dietterich (2000) expressed this domain knowledge in the task hierarchy shown in Figure 1b. This hierarchy defines a navigational composite action for

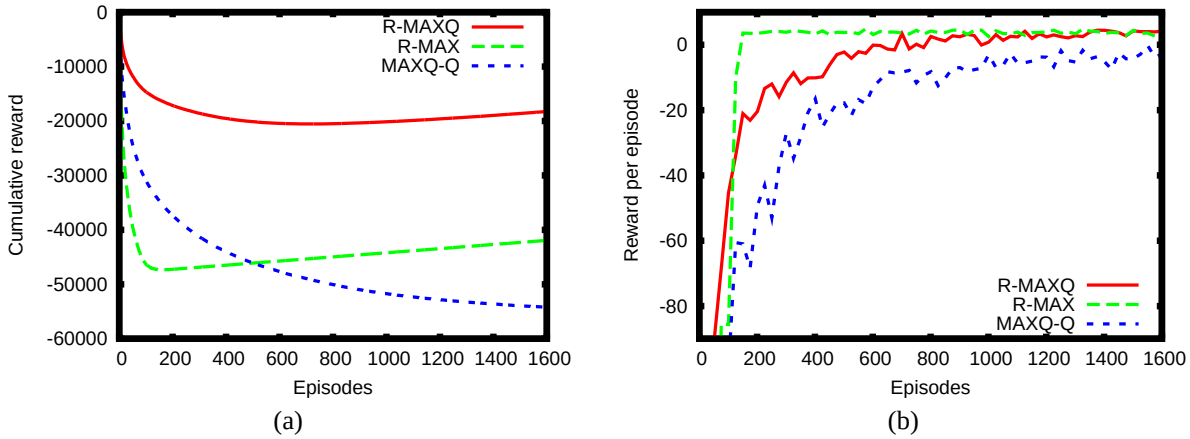


Figure 2. (a) Cumulative and (b) asymptotic performance of R-MAXQ, R-MAX, and MAXQ-Q on the Taxi domain, averaged over 100 independent trials. R-MAXQ and MAXQ-Q utilize the hierarchy shown in Figure 1b, but they do not use any explicit state abstraction.

each of the four landmarks. These actions include the four primitive movement actions as children, and they terminate upon reaching the coordinates corresponding to the respective landmark. The `GET` and `PUT` composite actions each have all four of their navigational composite actions as children, as well as `pickup` or `putdown`, respectively. `GET` terminates when the passenger is in the taxi, and `PUT` terminates only when the episode does. The `ROOT` action only includes `GET` and `PUT` as children, and like `PUT` it defines no terminal states beyond those intrinsic to the domain. All goal reward functions give 0 reward; each action simply minimizes the costs earned before reaching their subgoals.

In our experiments with R-MAX and R-MAXQ we set the threshold sample size at $m = 5$. Preliminary experiments showed that larger values of m did not signicantly improve the final policy, although of course they led to more time spent estimating the model. The only other parameter for these algorithms is the stopping criterion for the dynamic programming steps in Algorithms 2 and 4. In all cases, we ran value iteration until the largest change was smaller than $\epsilon = 0.001$. We provided R-MAXQ and the original MAXQ-Q algorithm with the hierarchy shown in Figure 1b as prior knowledge. For our implementation of MAXQ-Q, we used precisely the hand-tuned parameters Dietterich (2000) optimized for the initial value function, learning rates, and temperature decay (for Boltzmann exploration) for each action in the hierarchy. We conducted 100 independent trials of each condition of our experiments.

5.1. R-MAXQ versus R-MAX

We begin by comparing the performance of R-MAXQ and R-MAX on the Taxi task. Our initial hypothesis was that R-MAXQ would perform no better than R-MAX in the absence of state abstraction, since the model-based ability to plan to explore might subsume the exploratory role that options have played in many model-free RL implementations (Şimşek & Barto, 2004; Singh et al., 2005). Figure 2 reveals that in fact the two algorithms exhibit very different learning curves. In particular, although R-MAX requires many fewer episodes to converge to an optimal policy, R-MAXQ earns much greater total reward.

We had overlooked the fact that the hierarchy used by R-MAXQ doesn't so much guide exploration as it constrains it. In particular, note that the hierarchical agent can never attempt the `putdown` action except at one of the four landmark locations, since the `PUT` action only becomes available when the agent is already at one of these locations, and the four navigational actions keep the agent in this reduced set of states. The agent thus only attempts the `putdown` action in 12 incorrect states, instead of the 396 explored by R-MAX. In addition, R-MAX attempts the `pickup` action in 100 states in which R-MAXQ doesn't, when the passenger is already in the car. Since the penalty for incorrect usage of these actions is -10, R-MAX loses $10(396 - 12 + 100)m = 24200$ reward due to its wasted exploration, accounting for the difference between the two algorithms in Figure 2a. Furthermore, since the `GET` action cannot navigate to an arbitrary location, R-MAXQ can't attempt the `pickup` action in a non-landmark location until some episode randomly starts the agent there. In this case the hierarchy can only postpone, not prevent, wasted exploration. This effect explains the delayed convergence relative to R-MAX: in later episodes R-MAXQ spends time on exploration that R-MAX performed more eagerly.

5.2. R-MAXQ versus MAXQ-Q

Figure 2 also compares R-MAXQ with the original MAXQ-Q algorithm. Of course, this comparison isn't very fair,

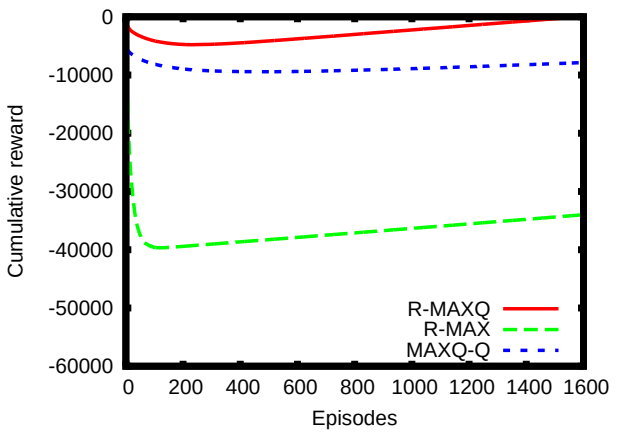


Figure 3. Cumulative performance of R-MAXQ, R-MAX, and MAXQ-Q on the Taxi domain, using state abstraction. (The asymptotic performance is qualitatively similar to that shown in Figure 2b, although with faster convergence.)

since a primary goal of the MAXQ framework was to create opportunities for state abstraction (Dietterich, 2000), which we did not initially exploit. In fact, Dietterich identified the condition described in Section 5.1, which he called shielding, as one that permits abstraction. For a more fair comparison, we allowed our implementation of MAXQ-Q to use all the state abstractions in the Taxi domain identified by Dietterich (2000), along with his optimized parameters.

We applied Dietterich's notion of max node irrelevance to allow R-MAXQ also to enjoy an explicit form of state abstraction as prior knowledge. Each action in the hierarchy abstracts away state variables when our domain knowledge indicates that doing so would not compromise the learned model. However, whereas in MAXQ-Q an action a only reports its abstract reward function R^a to its parents, in R-MAXQ it must also convey the abstract transition function P^a. Thus we only allow a composite action to ignore a state variable if all of its children also ignore that state variable.

In the hierarchy shown in Figure 1b, the four primitive movement actions and the four navigational actions can abstract away the `passenger` and `destination` state variables. `GET` and `pickup` ignore `destination`, and `PUT` and `putdown` ignore `passenger`. However, `ROOT` cannot ignore any state variables. When a child's transition function was more abstract than a parent's model, the parent assumed a very simple dynamic Bayes network (DBN) factorization (Boutilier et al., 1995). For example, $P^{\texttt{north}}$ sets `x` and `y` (each conditional on the previous values of both variables), but `passenger` and `destination` remain constant. Figure 3 compares the performance of the resulting algorithms. Both MAXQ-Q and R-MAXQ learn much faster with state abstraction, with the model-based nature of R-MAXQ continuing to give it an edge.

It is worthwhile to examine more closely how the hierarchy interacts with state abstraction in the Taxi domain. Consider how MAXQ-Q learns the `ROOT` action. The only values stored locally are the completion functions $C^{\texttt{root}}(\cdot, \texttt{GET})$ and $C^{\texttt{root}}(\cdot, \texttt{PUT})$, which have different abstract representations. The latter function is always equal to 0, since after `PUT` terminates there is nothing to complete, since the entire episode has terminated. Meanwhile, to evaluate $C^{\texttt{root}}(s, \texttt{GET})$ the algorithm need only inspect the `passenger` and `destination` variables of s, since the values of these two variables before executing `GET` completely determine the remaining cost of completing `ROOT` after `GET` terminates. Hence, MAXQ-Q only learns 16 values at the `ROOT` node; to compute the value of a state it recursively queries R^a and adds the appropriate completion function (Dietterich, 2000).

R-MAXQ doesn't apply any explicit state abstraction to `ROOT`, but note that after executing either of its two child actions, the result must be one of 12 nonterminal states: with the taxi at one of four landmarks, the passenger in the taxi, and the destination at one of the other three landmarks. Hence, the planning envelope computed in Algorithm 2 will always contain some subset of these 12 states plus the current state. As with MAXQ-Q, the result distribution irrelevance of `GET` allows R-MAXQ to store only a small number of values locally. To compute the value of a state, R-MAXQ also queries one-step values from its children and then adds the appropriate successor state values. In this sense, these 12 states can be thought of as the *completion set* of `ROOT`.

Figure 3 also shows the performance of standard R-MAX with the same DBN factorization as R-MAXQ applied to most of its actions (which are all primitive). Note that in the absence of shielding, `putdown` cannot safely ignore the `passenger` variable. The ability to abstract the primitive models does reduce the amount of exploration that R-MAX must perform, but the improvement is significantly smaller than that of the other algorithms. This result gives more support for motivating hierarchical decomposition with opportunities for state abstraction.

Some preliminary further experiments support the arguments of Jong et al. (2008), who used model-free hierarchical algorithms to suggest that composite actions more reliably improve RL performance when they replace instead of augment primitive actions. We ran R-MAXQ with a hierarchy in which the root's children included all six primitive actions as well as the four navigational composite actions, producing learning curves indistinguishable from those of standard R-MAX in Figure 2. When the root action can execute every primitive action, the planning envelope grows to include too many states. Formalizing the properties of a composite action's completion set may help us understand

how hierarchies can constrain planning envelopes without sacrificing learning performance.

6. Related Work

Other algorithms have combined hierarchical RL with a model-based approach, but not in the standard framework of discounted rewards and stochastic dynamics. Diuk et al. (2006) developed a model-based MAXQ algorithm for deterministic domains, allowing them to quickly sample the effect of a composite action recursively: every action's effect can be represented as a scalar reward and a single successor state. Their algorithm also uses Dietterich's approach to state abstraction, occasionally forcing it to replan, since the effect of a child action may depend on state variables not visible to the parent, making it seem nondeterministic. In contrast, R-MAXQ does not employ explicit state abstraction, allowing it to save the value functions and policies computed during one time step for all future time steps. Our algorithm relies on the choice of hierarchy to yield small planning envelopes, automatically achieving an effective reduction in the size of the state space considered during any one time step.

Seri and Tadepalli (2002) extended the MAXQ framework to average-reward reinforcement learning, resulting in an algorithm that learns a model to facilitate the computation of the bias for each state from the average reward of the current policy. However, the computation of the average reward itself relies on stochastic approximation techniques, and their algorithm does not have any formal guarantees regarding its sample complexity.

7. Conclusions

The R-MAXQ algorithm combines the efficient model-based exploration of R-MAX with the hierarchical decomposition of MAXQ. Although our algorithm does not improve upon the known formal bounds on the sample complexity of RL, it retains a finite-time convergence guarantee. An empirical evaluation demonstrates that even a relatively simple hierarchy can improve the cumulative reward earned by constraining the exploration that the agent performs, both within individual episodes of learning and throughout an agent's experience with its environment. Even in the absence of explicit state abstraction, the structure of an action hierarchy can drastically reduce the effective state space seen by a given composite action during a single episode. This implicit concept of a reduced completion set, mirroring Dietterich's explicitly abstracted completion function, suggests future avenues of research, both for improving the theoretical guarantees on the sample complexity of R-MAXQ and for guiding the discovery of more useful hierarchies.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0237699 and the DARPA Bootstrap Learning program.

References

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete-Event Systems*, *13*, 41–77. Special Issue on Reinforcement Learning.

Boutilier, C., Dearden, R., & Goldszmidt, M. (1995). Exploiting structure in policy construction. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1104–1111).

Brafman, R. I., & Tennenholtz, M. (2002). R-MAX – a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303.

Diuk, C., Strehl, A. L., & Littman, M. L. (2006). A hierarchical approach to efficient reinforcement learning in deterministic domains. *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*.

Jong, N. K., Hester, T., & Stone, P. (2008). The utility of temporal abstraction in reinforcement learning. *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*.

Kakade, S. M. (2003). *On the sample complexity of reinforcement learning*. Doctoral dissertation, University College London.

Kearns, M., & Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 260–268).

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, *13*, 103–130.

Seri, S., & Tadepalli, P. (2002). Model-based hierarchical average-reward reinforcement learning. *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 562–569).

Şimşek, Ö., & Barto, A. G. (2004). Using relative novelty to identify useful temporal abstractions in reinforcement learning. *Proceedings of the Twenty-First International Conference on Machine Learning* (pp. 751–758).

Singh, S., Barto, A. G., & Chentanez, N. (2005). Intrinsically motivated reinforcement learning. *Advances in Neural Information Processing Systems 17*.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Efficient Bandit Algorithms for Online Multiclass Prediction

Sham M. Kakade SHAM@TTI-C.ORG
Shai Shalev-Shwartz SHAI@TTI-C.ORG
Ambuj Tewari TEWARI@TTI-C.ORG
Toyota Technological Institute, 1427 East 60th Street, Chicago, Illinois 60637, USA

Abstract

This paper introduces the Banditron, a variant of the Perceptron [Rosenblatt, 1958], for the multiclass bandit setting. The multiclass bandit setting models a wide range of practical supervised learning applications where the learner only receives partial feedback (referred to as "bandit" feedback, in the spirit of multi-armed bandit models) with respect to the true label (e.g. in many web applications users often only provide positive "click" feedback which does not necessarily fully disclose a true label). The Banditron has the ability to learn in a multiclass classification setting with the "bandit" feedback which only reveals whether or not the prediction made by the algorithm was correct or not (but does not necessarily reveal the true label). We provide (relative) mistake bounds which show how the Banditron enjoys favorable performance, and our experiments demonstrate the practicality of the algorithm. Furthermore, this paper pays close attention to the important special case when the data is linearly separable — a problem which has been exhaustively studied in the full information setting yet is novel in the bandit setting.

1. Introduction

In the conventional supervised learning paradigm, the learner has access to a data set in which the true labels of the inputs are provided. While attendant learning algorithms in this paradigm are enjoying wide ranging success, their effective application to a number of domains, including many web based applications, hinges on being able to learn in settings where the true labels are not fully disclosed, but rather the learning algorithm only receives some partial feedback. Important domains include both the (financially important) sponsored advertising on webpages and recommender systems. The typical setting is: first, a user queries the system; then using the query and other potentially rich knowledge the system has about the user (e.g. past purchases, their browsing history, etc.) the system makes a suggestion (e.g. it presents the user with a few ads they might click on or songs they might buy); finally, the user either positively or negatively responds to the suggestion. Crucially, the system does not learn what would have happened had other suggestions been presented.

We view such problems as naturally being online, "bandit" versions of multiclass prediction problems, and, in this paper, we formalize such a model. In essence, this multiclass bandit problem is as follows: at each round, the learner receives an input $\mathbf{x}$ (say the users query, profile, and other high dimensional information); the learner predicts some class label $\hat{y}$ (the suggestion); then the learner receives the limited feedback of only whether the chosen label was correct or not. In the conventional, "full information" supervised learning model, a true label y (possibly more than one or none at all) is revealed to the learner at each round — clearly unrealistic in the aforementioned applications. In both cases, the learner desires to make as few mistakes as possible. The bandit version of this problem is clearly more challenging, since, in addition to the issues ones faces for supervised learning (e.g. learning a mapping from a high dimensional input space to the label space), one also faces balancing exploration and exploitation.

This paper considers the workhorse of hypothesis spaces, namely linear predictors, in the bandit setting. Somewhat surprisingly, while there has been a staggering number of results on (margin based) linear predictors and much recent work on bandit models, the

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

intersection of these two settings is novel and opens a number of interesting (both theoretical and practical) questions, which we consider. In particular, we pay close attention to the important case where the data are linearly separable, where, in the full information setting, the (efficient) Perceptron algorithm makes a number of mistakes that is asymptotically bounded (so the actual error rate will rapidly converge to 0).

There are a number of related results in the bandit literature. The Exp4 algorithm (for the "experts" setting) of Auer et al. [1998] and the contextual bandit setting of Langford and Zhang [2007] are both bandit settings where the learner has side information (e.g. the input "x") when making a decision — in fact, our setting can be thought of as a special case of the contextual bandit setting[1]. However, these settings consider abstract hypothesis spaces and do not explicitly consider efficient algorithms. Technically related are the bandit algorithms for online convex optimization of Flaxman et al. [2005], Kleinberg [2004], which attempt to estimate a gradient (for optimization) with only partial feedback. However, these algorithms do not apply due to the subtleties of using the non-convex classification loss, which we discuss at the end of Section 2.

This paper provides an *efficient* bandit algorithm, the Banditron, for multiclass prediction using linear hypothesis spaces, which enjoys a favorable mistake bound. We provide empirical results showing our algorithm is quite practical. For the case where the data is linearly separable, our mistake bound is $O(\sqrt{T})$ in T rounds. We also provide results toward characterizing the optimal achievable mistake bound for the linearly separable case (ignoring efficiency issues here) and introduce some important open questions regarding this issue. In the Extensions section, we also discuss update rules which generalize the Winnow algorithm (for L1 margins) and margin-mistake based algorithms to the bandit setting. We also discuss how our algorithm can be extended to ranking and settings where more than one prediction $\hat{y}$ can be presented to the user (e.g. an advertising setting where multiple ads may be presented).

2. Problem Setting

We now formally define the problem of online multiclass prediction in the bandit setting. Online learning is performed in a sequence of consecutive rounds. On round t, the learner is given an instance vector $\mathbf{x}_t \in \mathbb{R}^d$ and is required to predict a label out of a set of k predefined labels which we denote by $[k] = \{1, \ldots, k\}$. We denote the predicted label by $\hat{y}_t$. In the full information case, after predicting the label, the learner receives the correct label associated with $\mathbf{x}_t$, which we denote by $y_t \in [k]$. We consider a bandit setting, in which the feedback received by the learner is $\mathbf{1}[\hat{y}_t \neq y_t]$, where $\mathbf{1}[\pi]$ is 1 if predicate π holds and 0 otherwise. That is, the learner knows if it predicted an incorrect label, but it does not know the identity of the correct label. The learner's ultimate goal is to minimize the number of prediction mistakes, M, it makes along its run, where:

$$M = \sum_{t=1}^{T} \mathbf{1}[\hat{y}_t \neq y_t] \quad .$$

To make M small, the learner may update its prediction mechanism after each round so as to be more accurate in later rounds.

The prediction of the algorithm at round t is determined by a hypothesis, $h_t : \mathbb{R}^d \to [k]$, where h_t is taken from a class of hypotheses $\mathcal{H}$. In this paper we focus on the class of margin based linear hypotheses. Formally, each $h \in \mathcal{H}$ is parameterized by a matrix of weights $W \in \mathbb{R}^{k \times d}$ and is defined to be:

$$h(\mathbf{x}) \;=\; \operatorname*{argmax}_{j \in [k]} \; (W\mathbf{x})_j \quad , \tag{1}$$

where $(W\mathbf{x})_j$ is the jth element of the vector obtained by multiplying the matrix W with the vector $\mathbf{x}$. Since each hypothesis is parameterized by a weight matrix, we refer to a matrix W also as a hypothesis — by that we mean that the prediction is defined as given in Eq. (1). To evaluate the performance of a weight matrix W on an example $(\mathbf{x}, y)$ we check whether W makes a prediction mistake, namely determine if $\arg\max_j (W\mathbf{x})_j \neq y$.

The class of margin based linear hypotheses for multiclass learning has been extensively studied in the full information case [Duda and Hart, 1973, Vapnik, 1998, Weston and Watkins, 1999, Elisseeff and Weston, 2001, Crammer and Singer, 2003]. Our starting point is a simple adaptation of the Perceptron algorithm [Rosenblatt, 1958] for multiclass prediction in the full information case (this adaptation is called Kesler's construction in [Duda and Hart, 1973, Crammer and Singer, 2003]). Despite its age and simplicity, the Perceptron has proven to be quite effective in practical problems, even when compared to state-of-the-art large margin algorithms [Freund and Schapire, 1999]. We denote by W^t the weight matrix used by the Perceptron at round t. The Perceptron starts with the all

[1] The contextual bandit setting can be thought of as a general cost sensitive classification problem with bandit feedback. While their setting is an i.i.d. one, we make no statistical assumptions.

zero matrix $W^1 = \mathbf{0}$ and updates it as follows

$$W^{t+1} \;=\; W^t + U^t \quad , \tag{2}$$

where $U^t \in \mathbb{R}^{k\times d}$ is the matrix defined by

$$U^t_{r,j} \;=\; x_{t,j}\left(\mathbf{1}[y_t = r] - \mathbf{1}[\hat{y}_t = r]\right) . \tag{3}$$

In other words, if there is no prediction mistake (i.e. $y_t = \hat{y}_t$), then there is no update (i.e. $W^{t+1} = W^t$), and if there is a prediction mistake, then $\mathbf{x}_t$ is added to the y_tth row of the weight matrix and subtracted from the $\hat{y}_t$th row of the matrix.

A relative mistake bound can be proven for the multiclass Perceptron algorithm. The difficulty with providing mistake bounds for any (efficient) algorithm in this setting stems from the fact that the classification loss is non-convex. Hence, performance bounds are commonly evaluated using the multiclass *hinge-loss* — what might be thought of as a convex relaxation of the classification loss. In particular, the hinge-loss of W on $(\mathbf{x}, y)$ is defined as follows:

$$\ell(W;(\mathbf{x},y)) \;=\; \max_{r\in[k]\setminus\{y\}} [1 - (W\mathbf{x})_y + (W\mathbf{x})_r]_+ \quad , \tag{4}$$

where $[a]_+ = \max\{a, 0\}$ is the hinge function. The hinge-loss will be zero only if $(W\mathbf{x})_y - (W\mathbf{x})_r \geq 1$ for all $r \neq y$. The difference $(W\mathbf{x})_y - (W\mathbf{x})_r$ is a generalization of the notion of *margin* from binary classification. Let $\hat{y} = \text{argmax}_r (W\mathbf{x})_r$ be the prediction of W. Note that if $\hat{y} \neq y$ then $\ell(W;(\mathbf{x},y)) \geq 1$. Thus, the hinge-loss is a convex upper bound on the zero-one loss function, $\ell(\mathbf{w};(\mathbf{x},y)) \geq \mathbf{1}[\hat{y} \neq y]$.

The Perceptron mistake bound holds for any sequence of examples and compares the number of mistakes made by the Perceptron with the cumulative hinge-loss of any fixed weight matrix $W^\star$, even one defined with prior knowledge of the sequence. Formally, let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$ be a sequence of examples and assume for simplicity that $\|\mathbf{x}_t\| \leq 1$ for all t. Let $W^\star$ be any fixed weight matrix. We denote by

$$L = \sum_{t=1}^{T} \ell(W^\star;(\mathbf{x}_t, y_t)) \quad , \tag{5}$$

the cumulative hinge-loss of $W^\star$ over the sequence of examples and by

$$D = 2\,\|W^\star\|_F^2 = 2\sum_{r=1}^{k}\sum_{j=1}^{d}(W^\star_{i,j})^2 \quad , \tag{6}$$

the *complexity* of $W^\star$. Here $\|\cdot\|_F^2$ denotes the Frobenius norm. Then the number of prediction mistakes of the multiclass Perceptron is at most,

$$M \;\leq\; L + D + \sqrt{L\,D} \quad . \tag{7}$$

Algorithm 1 The Banditron

Parameters: $\gamma \in (0, 0.5)$
Initialize $W^1 = \mathbf{0} \in \mathbb{R}^{k\times d}$
for $t = 1, 2, \ldots, T$ **do**
  Receive $\mathbf{x}_t \in \mathbb{R}^d$
  Set $\hat{y}_t = \arg\max_{r\in[k]} (W^t\mathbf{x}_t)_r$
  $\forall r \in [k]$ define $P(r) = (1-\gamma)\mathbf{1}[r = \hat{y}_t] + \frac{\gamma}{k}$
  Randomly sample $\tilde{y}_t$ according to P
  Predict $\tilde{y}_t$ and receive feedback $\mathbf{1}[\tilde{y}_t = y_t]$
  Define $\tilde{U}^t \in \mathbb{R}^{k\times d}$ such that:
    $\tilde{U}^t_{r,j} = x_{t,j}\left(\frac{\mathbf{1}[y_t=\tilde{y}_t]\mathbf{1}[\tilde{y}_t=r]}{P(r)} - \mathbf{1}[\hat{y}_t = r]\right)$
  Update: $W^{t+1} = W^t + \tilde{U}^t$
end for

A proof of the above mistake bound can be found for example in Fink et al. [2006]. The mistake bound in Eq. (7) consists of three terms: the loss of $W^\star$, the complexity of $W^\star$, and a sub-linear term which is often negligible. In particular, when the data is separable (i.e. $L = 0$), the number of mistakes is bounded by D.

Unfortunately, the Perceptron's update cannot be implemented in the bandit setting as we do not know the identity of y_t. One direction is to work directly with the hinge-loss (which is convex) and try to use the bandit algorithms for online convex optimization of Flaxman et al. [2005], Kleinberg [2004]. In this work, they attempt to find an unbiased estimate of the gradient using only bandit feedback (i.e. using only the loss received as feedback). However, since the only feedback the learner receives is $\mathbf{1}[\hat{y}_t \neq y_t]$, one does not necessarily even know the hinge-loss for the chosen decision, $\hat{y}_t$, due to dependence of the hinge loss on the true label y_t. Hence, the results of Flaxman et al. [2005], Kleinberg [2004] are not directly applicable.

3. The Banditron

We now present the Banditron in Algorithm 1, which is an adaptation of the multiclass Perceptron for the bandit case.

Similar to the Perceptron, at each round we let $\hat{y}_t$ be the best label according to the current weight matrix W^t, i.e. $\hat{y}_t = \text{argmax}_r (W^t\mathbf{x}_t)_r$. Most of the time the Banditron exploits the quality of the current weight matrix by predicting the label $\hat{y}_t$. Unlike the Perceptron, if $\hat{y}_t \neq y_t$, then we can not make an update since we are blind to the identity of y_t. Roughly speaking, it is difficult to learn when we exploit using W^t. For this reason, on some of the rounds we let the algorithm explore (with probability $1-\gamma$) and uniformly predict a random label from $[k]$. We denote by $\tilde{y}_t$ the

predicted label. On rounds in which we explore, (so $\tilde{y}_t \neq \hat{y}_t$), if we additionally receive a positive feedback, i.e. $\tilde{y}_t = y_t$, then we indirectly obtain the full information regarding the identity of y_t, and we can therefore update our weight matrix using this positive instance. The parameter γ controls the exploration-exploitation tradeoff.

The above intuitive argument is formalized by defining the update matrix $\tilde{U}^t$ to be a function of the randomized prediction $\tilde{y}_t$. We emphasize that $\tilde{U}^t$ accesses the correct label y_t only through the indicator $\mathbf{1}[y_t = \tilde{y}_t]$ and is thus adequate for the Bandit setting. As we show later in Lemma 4, the expected value of the Banditron's update matrix $\tilde{U}^t$ is exactly the Perceptron's update matrix U^t. While there a number of other variants which also perform unbiased updates, we have found this one provides the most favorable performance (empirically speaking).

The following theorem provides a bound on the expected number of mistakes the Banditron makes.

Theorem 1. *(Mistake Bound). Assume that for the sequence of examples, $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)$, we have, for all t, $\mathbf{x}_t \in \mathbb{R}^d$, $\|\mathbf{x}_t\| \leq 1$, and $y_t \in [k]$. Let $W^\star$ be any matrix, let L be the cumulative hinge-loss of $W^\star$ as defined in Eq. (5), and let D be the complexity of $W^\star$ (i.e. $D = 2||W^\star||_F^2$). Then the number of mistakes M made by the Banditron satisfies*

$$\mathbb{E}[M] \;\leq\; L + \gamma\, T + 3 \max\left\{ \tfrac{k\, D}{\gamma}\,,\, \sqrt{D\, \gamma\, T} \right\} + \sqrt{\tfrac{k\, D\, L}{\gamma}} \quad .$$

where expectation is taken with respect to the randomness of the algorithm.

Before turning to the proof of Thm. 1 let us first optimize the exploration-exploitation parameter γ in different scenarios. First, assume that the data is separable, that is $L = 0$. In this case, we can obtain a mistake bound of $O(\sqrt{T})$. In fact the following corollary shows that an $O(\sqrt{T})$ bound is achievable whenever the cumulative hinge-loss of $W^\star$ is small enough.

Corollary 2 (Low noise)**.** *Assume that the conditions stated in Thm. 1 hold and that there exists $W^\star$ with fixed complexity D and loss $L \leq O(\sqrt{D\, k\, T})$. Then, by setting $\gamma = \sqrt{k\, D/T}$ we obtain the bound $\mathbb{E}[M] \leq O(\sqrt{k\, D\, T})$.*

Next, let us consider the case where we have a constant (average) noise level of ρ, i.e. there exists $\rho \in (0, 1)$ such that $L \leq \rho T$. In this case,

Corollary 3 (High noise)**.** *Assume that the conditions stated in Thm. 1 hold and that there exists $W^\star$ with fixed complexity D and loss $L \leq \rho\, T$ for a constant $\rho \in (0, 1)$. Then, by setting $\gamma = \rho\, (k\, D/T)^{1/3}$ we obtain the bound $\mathbb{E}[M] \leq \rho T\, (1+\epsilon)$ where $\epsilon = O((k\, D)^{1/3}\, T^{-1/3})$.*

We note that the bound in the above corollary can be also written in an additive form as: $\mathbb{E}[M] - L \leq O(T^{2/3})$. However, since we are not giving proper regret bounds as we compare mistakes to hinge-loss we prefer to directly bound $\mathbb{E}[M]$.

Analysis: To prove Thm. 1 we first show that the random matrix $\tilde{U}^t$ is an unbiased estimator of the update matrix U^t used by the Perceptron. Formally, let $\mathbb{E}_t[\tilde{U}^t]$ be the expected value of $\tilde{U}^t$ conditioned on $\tilde{y}_1, \ldots, \tilde{y}_{t-1}$. Then:

Lemma 4. *Let $\tilde{U}^t$ be as defined in Algorithm 1 and let U^t be as defined in Eq. (3). Then, $\mathbb{E}_t[\tilde{U}^t] = U^t$.*

Proof. For each $r \in [k]$ and $j \in [d]$ we have

$$\begin{aligned} \mathbb{E}_t[\tilde{U}^t_{r,j}] \;&=\; \sum_{i=1}^{k} P(i) x_{t,j} \left(\tfrac{\mathbf{1}[i=y_t]\mathbf{1}[i=r]}{P(r)} - \mathbf{1}[\hat{y}_t = r] \right) \\ &=\; x_{t,j}\, (\mathbf{1}[y_t = r] - \mathbf{1}[\hat{y}_t = r]) \;=\; U^t_{r,j} \quad , \end{aligned}$$

which completes the proof. □

Next, we bound the expected squared norm of $\tilde{U}^t$.

Lemma 5. *Let $\tilde{U}^t$ be as defined in Algorithm 1. Then,*

$$\mathbb{E}_t[\, \|\tilde{U}^t\|_F^2\,] \leq 2\, \|\mathbf{x}_t\|^2 \left(\tfrac{k}{\gamma}\, \mathbf{1}[y_t \neq \hat{y}_t] + \gamma\, \mathbf{1}[y_t = \hat{y}_t] \right) \quad .$$

Proof. We first observe that

$$\|\tilde{U}^t\|_F^2 \;=\; \begin{cases} \|\mathbf{x}_t\|^2 \left(\frac{1}{P(y_t)^2} + 1 \right) & \text{if } \tilde{y}_t = y_t \neq \hat{y}_t \\ \|\mathbf{x}_t\|^2 \left(\frac{1}{P(y_t)} - 1 \right)^2 & \text{if } \tilde{y}_t = y_t = \hat{y}_t \\ \|\mathbf{x}_t\|^2 & \text{if } \tilde{y}_t \neq y_t \end{cases}$$

Therefore, if $y_t \neq \hat{y}_t$ then

$$\begin{aligned} \frac{\mathbb{E}_t[\|\tilde{U}^t\|_F^2]}{\|\mathbf{x}_t\|^2} \;&=\; P(y_t) \left(\tfrac{1}{P(y_t)^2} + 1 \right) + (1 - P(y_t)) \\ &=\; 1 + \tfrac{1}{P(y_t)} = 1 + \tfrac{k}{\gamma} \leq \tfrac{2k}{\gamma} \quad , \end{aligned}$$

and if $y_t = \hat{y}_t$ then

$$\begin{aligned} \frac{\mathbb{E}_t[\|\tilde{U}^t\|_F^2]}{\|\mathbf{x}_t\|^2} \;&=\; P(y_t) \left(\tfrac{1}{P(y_t)} - 1 \right)^2 + (1 - P(y_t)) \\ &=\; \tfrac{1}{P(y_t)} - 1 \leq \tfrac{1}{1-\gamma} - 1 \leq \tfrac{\gamma}{1-\gamma} \leq 2\gamma \quad . \end{aligned}$$

Combining the two cases concludes our proof. □

Equipped with the above two lemmas, we are ready to prove Thm. 1.

Proof of Thm. 1. Throughout the proof we use the notation $\langle W^\star, W^t \rangle := \sum_{r=1}^{k} \sum_{j=1}^{d} W^\star_{r,j} W^t_{r,j}$. We prove the theorem by bounding $\mathbb{E}[\langle W^\star, W^{T+1} \rangle]$ from above and from below starting with a lower bound. We can first use the fact that $W^1 = \mathbf{0}$ to rewrite $\mathbb{E}[\langle W^\star, W^{T+1} \rangle]$ as $\sum_{t=1}^{T} \Delta_t$ where

$$\Delta_t := \mathbb{E}[\langle W^\star, W^{t+1} \rangle] - \mathbb{E}[\langle W^\star, W^t \rangle] \ .$$

Expanding the definition of W^{t+1} and using Lemma 4 we obtain that for all t, $\Delta_t = \mathbb{E}[\langle W^\star, \tilde{U}^t \rangle] = \mathbb{E}[\langle W^\star, U^t \rangle]$. Next, we note that the definition of the hinge-loss given in Eq. (4) implies that the following holds regardless of the value of $\hat{y}_t$

$$\ell(W^\star, (\mathbf{x}_t, y_t)) \ \geq \ \mathbf{1}[\hat{y}_t \neq y_t] - \langle W^\star, U^t \rangle \ .$$

Therefore $\Delta_t \geq \mathbb{E}[\, \mathbf{1}[\hat{y}_t \neq y_t] \,] - \ell(W^\star, (\mathbf{x}_t, y_t))$. Summing over t we obtain the lower bound

$$\mathbb{E}[\langle W^\star, W^{T+1} \rangle] \ = \ \sum_{t=1}^{T} \Delta_t \ \geq \ \mathbb{E}[\hat{M}] - L \ , \qquad (8)$$

where $\hat{M} := \sum_{t=1}^{T} \mathbf{1}[\hat{y}_t \neq y_t]$ and L is as defined in Eq. (5). Next, we show an upper bound on $\mathbb{E}[\langle W^\star, W^{T+1} \rangle]$. Using Cauchy-Schwartz inequality we have $\langle W^\star, W^{T+1} \rangle \leq \|W^\star\|_F \, \|W^{T+1}\|_F$. To ease our notation, we use the shorthand $\|\cdot\|$ for denoting the Frobenius norm. Using the definition of D given in Eq. (6), the concavity of the sqrt function, and Jensen's inequality we obtain that

$$\mathbb{E}[\langle W^\star, W^{T+1} \rangle] \leq \sqrt{\tfrac{D \, \mathbb{E}[\, \|W^{T+1}\|^2 \,]}{2}} \ . \qquad (9)$$

We therefore need to upper bound the expected value of $\|W^{T+1}\|^2$. Expanding the definition of W^{T+1} we get that

$$\begin{aligned} \mathbb{E}[\|W^{T+1}\|^2] &= \mathbb{E}[\|W^T\|^2 + \langle W^T, \tilde{U}^T \rangle + \|\tilde{U}^T\|^2] \\ &= \sum_{t=1}^{T} \Big(\mathbb{E}[\langle W^t, \tilde{U}^t \rangle] + \mathbb{E}[\, \|\tilde{U}^t\|^2 \,] \Big) \ . \end{aligned}$$

Using Lemma 4 we obtain that $\mathbb{E}[\langle W^t, \tilde{U}^t \rangle] = \mathbb{E}[\langle W^t, U^t \rangle] \leq 0$, where the second inequality follows from the definition of U^t and $\hat{y}_t$. Combining this with Lemma 5 and with the assumption $\|\mathbf{x}_t\| \leq 1$ for all t we obtain that

$$\begin{aligned} \mathbb{E}[\|W^{T+1}\|^2] &\leq \sum_{t=1}^{T} \mathbb{E}\left(\tfrac{2k}{\gamma} \mathbf{1}[y_t \neq \hat{y}_t] + 2\gamma \mathbf{1}[y_t = \hat{y}_t] \right) \\ &\leq \tfrac{2k}{\gamma} \, \mathbb{E}[\hat{M}] + 2\,\gamma\, T \ . \end{aligned}$$

Plugging the above into Eq. (9) and using the inequality $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ we get the upper bound

$$\mathbb{E}[\langle W^\star, W^{T+1} \rangle] \leq \sqrt{\tfrac{D\,k\,\mathbb{E}[\hat{M}]}{\gamma}} + \sqrt{D\,\gamma\,T} \ .$$

Comparing the above upper bound with the lower bound given in Eq. (8) and rearranging terms yield

$$\mathbb{E}[\hat{M}] - \sqrt{\tfrac{D\,k\,\mathbb{E}[\hat{M}]}{\gamma}} - \left(L + \sqrt{D\,\gamma\,T} \right) \leq 0 \ .$$

Standard algebraic manipulations give the bound

$$\mathbb{E}[\hat{M}] \ \leq \ L + \sqrt{\tfrac{D\,k\,L}{\gamma}} + 3 \max\left\{ \tfrac{D\,k}{\gamma} \, , \, \sqrt{D\,\gamma\,T} \right\} \ .$$

Finally, our proof is concluded by noting that in expectation we are exploring no more than γT of the rounds and thus $\mathbb{E}[M] \leq \mathbb{E}[\hat{M}] + \gamma T$. □

4. Mistake Bounds Under Separability

In this section we present results towards characterizing the optimal achievable rate for the case where the data is separable. Here, in the full-information setting, the mistake bound of the Perceptron algorithm is finite and bounded by D. We now present an (inefficient) algorithm showing that the achievable mistake bound in the bandit setting is also finite — thus the Banditron's mistake bound of $O(\sqrt{T})$ leaves significant room for improvement (though the algorithm is quite simple and has reasonable performance, which we demonstrate in the next section).

First, as a technical tool, we make the interesting observation that the halving algorithm (generalized to the multiclass setting) is also applicable to the bandit setting. The algorithm is as follows: Let $\mathcal{H}'$ be the current set of "active" experts, which is initialized to the full set, i.e. $\mathcal{H} = \mathcal{H}'$ at $t = 1$. At each round t, we predict using the majority prediction r (i.e. the prediction $r \in [k]$ which the most hypotheses in $\mathcal{H}'$ predict). If we are correct, we make no update. If we are incorrect, we remove from the active set, $\mathcal{H}'$, those $h \in \mathcal{H}'$ which predicted the incorrect label r. Crucially, this (generalized) halving algorithm is implementable with only the bandit feedback that we receive. This algorithm enjoys the following mistake bound.

Lemma 6. *(Halving Algorithm). The halving algorithm (in the bandit setting) makes at most $k \ln |\mathcal{H}|$ mistakes on any sequence in which there exists some hypothesis in $\mathcal{H}$ which makes no mistakes.*

Proof. Whenever the algorithm makes a mistake, the size of active set is reduced by at least a $1 - 1/k$ fraction, since majority prediction uses a fraction of hypothesis (from the active set) that is at least $1/k$. Since the algorithm never removes a perfect hypothesis from the active set, the maximal number of mistakes, M, that can occur until $\mathcal{H}'$ consists of only perfect

hypotheses satisfies $(1-1/k)^M|\mathcal{H}| \geq 1$. Using the inequality $(1-1/k) \leq e^{-1/k}$ and solving for M leads to the claim. □

Using this, the following theorem shows that the achievable bound for the number of mistakes is asymptotically finite. Unfortunately, the result has a dimensionality dependence on d. The algorithm essentially uses the margin condition to construct an appropriately sized cover for $\mathcal{H}$, the set of all linear hypotheses, and runs the halving algorithm on this cover.

Theorem 7. *There exists a deterministic algorithm (in the bandit setting), taking D as input, which makes at most $O(k^2 d \ln(Dd))$ mistakes on any sequence (where $\|\mathbf{x}_t\| \leq 1$) that is linearly separable at margin 1 by some $W^\star$, with $2||W^\star||_F^2 \leq D$.*

Proof. (sketch) Since the margin is 1, it is straightforward to show that if W is a perturbation of $W^\star$ which satisfies $||W^\star - W||_\infty \leq O(\frac{1}{\sqrt{d}})$, then the data is still linearly separable under W. By noting that each coordinate in $W^\star$ is (rather crudely) bounded by $\sqrt{D}$, there exists a discretized grid of $\mathcal{H}$ of size $O(\sqrt{Dd})^{kd}$ which contains a linear separator. The algorithm simply runs the halving algorithm on this cover. □

This result is in stark contrast to the Perceptron mistake bound which has no dependence on the dimension d. We now provide a mistake bound with no dependence on the dimension. Unfortunately, it is not asymptotically finite, as it is has a rather mild dependence on the time — it is $O(D \ln T)$ (ignoring k and higher order terms), while the Perceptron mistake bound is $O(D)$.

Theorem 8. *There exists a randomized algorithm (in the bandit setting), taking as inputs D, T and $\delta > 0$, such that with probability greater than $1-\delta$ the algorithm makes at most $O(k^2 D \ln \frac{T+k}{\delta}(\ln D + \ln\ln \frac{T+k}{\delta}))$ mistakes on any T length sequence (where $\|\mathbf{x}_t\| \leq 1$) that is linearly separable at margin 1 by some $W^\star$, with $2||W^\star||_F^2 \leq D$.*

The algorithm first constructs a random projection operator which projects any $\mathbf{x}$ into a space of dimension $d' = O(D \ln \frac{T+k}{\delta})$, and then it runs the previous algorithm in this lower dimensional space. The proof essentially consists of using the results in Arriaga and Vempala [2006] to argue that the (multiclass) margin is preserved under this random projection.

Proof. (sketch) It is simpler to rescale $W^\star$ such that $||W^\star||_F = 1$ and the margin is $1/\sqrt{D}$. Consider the $T+k$ points x_1 to x_T and the (row) vectors $W_1^\star, \ldots W_k^\star$, whose norms are all bounded by 1. Let P be a matrix of dimension $d' \times d$, where each entry of P is independently sampled from $U(-1,1)$. Define the projection operator $\Pi(v) = \frac{1}{\sqrt{d'}} Pv$. Corollary 2 of Arriaga and Vempala [2006] (essentially a result from the JL lemma) shows that if $d' = O(D \ln \frac{T+k}{\delta})$ then this projection additively preserves the inner products of these points up to $\frac{1}{3\sqrt{D}}$, i.e. $|\Pi(W_r^\star)\cdot\Pi(\mathbf{x}_t) - W_r^\star \cdot \mathbf{x}_t| \leq \frac{1}{3\sqrt{D}}$. It follows that, after the projection, the data is linearly separable with margin at least $\frac{1}{3\sqrt{D}}$. Letting $\Pi(W^\star)$ denote the matrix where each row of $W^\star$ has been projected, then, also by the JL lemma, the norm $||\Pi(W^\star)||_F$ will (rather crudely) be bounded by 2 (recall $||W^\star||_F = 1$). Hence, the projected data is linearly separable at margin $1/(3\sqrt{D})$ by a weight matrix that has norm $O(1)$, which is identical to being separable at margin 1 with weight vector of complexity $O(D)$. The algorithm is to first create a random projection matrix (which can be constructed without knowledge of the sequence) and then we can run the previous algorithm on the lower dimensional space d'. Since we have shown that the margin is preserved (up to a constant) in the lower dimensional space, the result follows from the previous Theorem 7, with d' as the dimensionality. □

We discuss open questions in the Extensions section.

5. Experiments

In this section, we report some experimental results for the Banditron algorithm on synthetic and real world data sets. For each data set, we ran Banditron for a wide range of values of the exploration parameter γ. For each value of γ, we report the average error rate, where the averaging is over 10 independent runs of Banditron.

The results are shown on Fig. 1. Each column corresponds to one data set. The top figures plot the error rates of Banditron (for the best value of γ) and Perceptron as a function of the number of examples seen. We show these on a log-log scale to get a better visual indication of the asymptotics of these algorithms. The bottom figures plot the final error rates on the complete data set as a function of γ. As expected, setting γ too low or too high leads to higher error rates.

The first data set, denoted by SynSep, is a 9-class, 400-dimensional synthetic data set of size 10^6. The idea is to have a simple simulation of generating a text document. The coordinates represent different words in a small vocabulary of size 400. See the caption of Figure 1 for details. We ensure, by construction, that

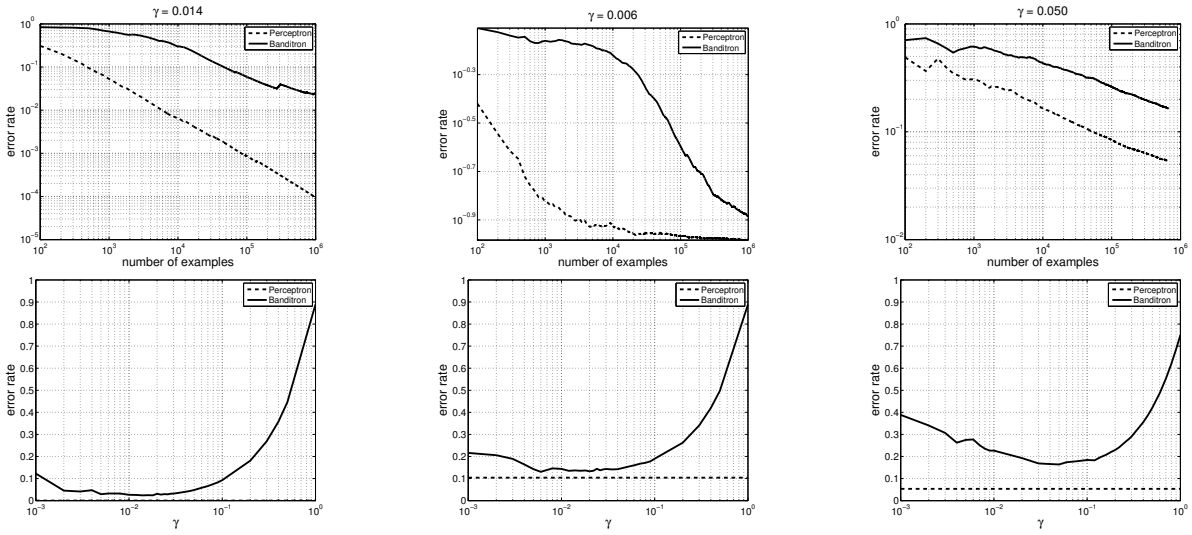


Figure 1. Error rates of Perceptron (dashed) and Banditron (solid) on the SYNSEP (left), SYNNONSEP (middle), and REUTERS4 (right) data sets. The 9-class synthetic data sets are generated as follows. We fix 9 bit-vectors $v_1, \ldots, v_9 \in \{0,1\}^{400}$ each of which has 20 to 40 bits turned on in its first 120 coordinates. The supports of some of these vectors overlap. The vectors v_i correspond to 9 topics where topic i has "keywords" that correspond to the bits turned on in v_i. To generate an example, we randomly choose a v_i and randomly turn off 5 bits in its support. Further, we randomly turn on 20 additional bits in the last 280 coordinates. The last 280 coordinates thus correspond to common words that can appear in a document from any topic.

SYNSEP is linearly separable. The left plots in Figure 1 show the results for this data set. Since this is a separable data set, Perceptron makes a bounded number of mistakes and its error rate plot falls at a rate of $1/T$ yielding a slope of -1 on a log-log plot. Corollary 2 predicts that error rate for Banditron should decay faster than $1/\sqrt{T}$ and we indeed see a slope of approximately -0.55 on the log-log plot. The second data set, denoted by SYNNONSEP, is constructed in the same way as SYNSEP except that we introduce 5% label noise. This makes the data set non-separable. The middle plots in Fig. 1 show the results for SYNNONSEP. The Perceptron error rate decays till it drops to 10% and then becomes constant. Banditron does not decay appreciably till 10^4 examples after which it falls rapidly to its final value of $10^{-0.89} = 13\%$.

We construct our third data set REUTERS4 from the Reuters RCV1 collection. Documents in the Reuters data set can have more than one label. We restrict ourselves to those documents that have exactly one label from the following set of labels: {CCAT, ECAT, GCAT, MCAT}. This gives us a 4-class data set of size 673,768 which includes about 84% of the documents in the original Reuters data set. We do this because the model considered in this paper assumes that every instance has a single true label. See the Extensions section for a discussion about dealing with multiple labels. We represent each document using bag-of-words, which leads to 346,810 dimensions. The right plots in Fig. 1 show the results for REUTERS4. The final error rates for Perceptron and Banditron ($\gamma = 0.05$) are 5.3% and 16.3% respectively. However, it is clear from the top plot that as the number of examples grows, the error rate of Banditron is dropping at a rate comparable to that of Perceptron.

6. Extensions and Open Problems

We now discuss a few extensions of the Banditron algorithm and some open problems. These extensions may possibly improve the performance of the algorithm and also broaden the set of applications that can be tackled by our approach. Due space constraints, we confine ourselves to a rather high level overview.

Label Ranking: So far we assumed that each instance vector is associated with a *single* correct label and we must correctly predict this particular label. In many applications this binary dichotomy is inadequate as each label is associated with a degree of relevance, which reflects to what extent it is relevant to the instance vector in hand. Furthermore, it is sometime natural to predict a subset of the labels rather than a single label. For example, consider again the problem of sponsored advertising on webpages described in the Introduction. Here, the system presents the user with a few ads. If the user positively responds to one

of the suggestions (say by a "click"), this implies that the user prefers this suggestion over the other suggestions, but it does not necessarily mean that the other suggestions are completely wrong.

We now briefly discuss a possible extension of the Banditron algorithm for this case (using techniques from Crammer et al. [2006]). On each round, we first find the r top ranked labels (where ranking is according to $\langle \mathbf{w}_r, \mathbf{x}_t \rangle$). With probability $1-\gamma$ we exploit and predict these labels. With probability γ we explore and randomly change one of the top ranked labels with another label which is ranked lower by our model. If we are exploring and the user chooses the replaced label, then we obtain a feedback that can be used for improving our model. The Banditron analysis can be generalized to this case, leading to bounds on the number of rounds in which the user negatively responds to our advertisement system.

Multiplicative Updates and Margin-Based Updates: While deriving the Banditron algorithm, our starting point was the Perceptron, which is an extremely simple online learning algorithm for the full information case. Over the years, many improvements of the Perceptron were suggested (see for example Shalev-Shwartz and Singer [2007] and the references therein). It is therefore interesting to study which algorithms can be adapted to the Bandit setting. We conjecture that it is relatively straightforward to adapt the multiplicative update scheme [Littlestone, 1988, Kivinen and Warmuth, 1997] to the bandit setting while achieving mistake bounds similar to the mistake bounds we derived for the Banditron. It is also possible to adapt margin-based updates (i.e. updating also when there is no prediction mistake but only a margin violation) to the bandit setting. Here, however, it seems that the resulting mistake bounds for the low noise case are inferior to the bound we obtained for the Banditron.

Achievable Rates and Open Problems: The immediate question is how to improve our rate of $O(T^{2/3})$ to $O(\sqrt{T})$ in the general setting with an efficient algorithm. We conjecture this is at least possible by some (possibly inefficient) algorithm. Important open questions in the separable case are: What is the optimal mistake bound? In particular, does there exist a finite mistake bound which has no dimensionality dependence? Furthermore, are there efficient algorithms which achieve the mistake bound of $O(D \ln T)$, provided in Theorem 8 (or better)? Practically speaking, this last question is of the most importance, as then we would have an algorithm that actually achieves a very small mistake bound in certain cases.

References

R.I. Arriaga and S. Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Mach. Learn.*, 63(2), 2006.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual FOCS*, 1998.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, Mar 2006.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

A. Elisseeff and J. Weston. A kernel method for multi-labeled classification. In *Advances in Neural Information Processing Systems 14*, 2001.

M. Fink, S. Shalev-Shwartz, Y. Singer, and S. Ullman. Online multiclass learning by interclass hypothesis sharing. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

A. Flaxman, A. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 385–394, 2005.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37 (3):277–296, 1999.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

R.D. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. *NIPS*, 2004.

J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. *NIPS*, 2007.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

S. Shalev-Shwartz and Y. Singer. A primal-dual perspective of online learning algorithms. *Machine Learning Journal*, 2007.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, April 1999.

Large Scale Manifold Transduction

Michael Karlen[*†] MICHAEL.KARLEN@GMAIL.COM
Jason Weston[*] JASONW@NEC-LABS.COM
Ayse Erkan[*‡] NAZ@CS.NYU.EDU
Ronan Collobert[*] COLLOBER@NEC-LABS.COM
(∗) NEC Labs America, 4 Independence Way, Princeton, NJ 08540 USA
(†) Ecóle Polytechnique Fédérale de Lausanne, CH-1015, Lausanne, Switzerland
(‡) New York University, Computer Science Department, 715 Broadway New York, NY 10003 USA

Abstract

We show how the regularizer of Transductive Support Vector Machines (TSVM) can be trained by stochastic gradient descent for linear models and multi-layer architectures. The resulting methods can be trained *online*, have vastly superior training and testing speed to existing TSVM algorithms, can encode prior knowledge in the network architecture, and obtain competitive error rates. We then go on to propose a natural generalization of the TSVM loss function that takes into account neighborhood and manifold information directly, unifying the two-stage Low Density Separation method into a single criterion, and leading to state-of-the-art results.

1. Introduction

Several methods for improving discriminative classifiers using unlabeled data have been developed in the last few years. Perhaps the two most popular ways of utilizing the unlabeled data are:

(i) maximizing the *margin* on the unlabeled data as in Transductive Support Vector Machines (TSVM) so that the decision rule lies in a region of low density; and

(ii) learning the *cluster* or *manifold* structure from the unlabeled data as in cluster kernels (Chapelle et al., 2003), label propagation (Zhu & Ghahramani, 2002), and Laplacian SVMs (Belkin et al., 2006).

Both approaches can be seen as making the same *structure assumption* on the data, that the cluster or manifold structure in the data is correlated with the class labels of interest.

The Low Density Separation algorithm (LDS) (Chapelle & Zien, 2005) is a two-stage algorithm that combines both of these approaches, with improved results over using only one of the techniques, however the combination method is somewhat *ad-hoc*.

A serious problem with all these methods is that they suffer from an inability to scale to very large datasets, apart from in the linear case (Sindhwani & Keerthi, 2006). This is ironic because the potential gain of semi-supervised learning lies in the vast amounts of readily available unlabeled data. This performance gain is never attained simply because of the computational burden of calculating the result. In the conclusion of the article describing the LDS algorithm the authors state:

"We observe that the time (and to some degree, also space) complexities of all methods investigated here prohibit the application to really large sets of unlabeled data, say, more than a few thousand. Thus, work should also be devoted to improvements of the computational efficiency of algorithms, ideally of LDS."

In this work we propose a new method for semi-supervised learning which features the following improvements over existing approaches:

- A new regularizer for semi-supervised learning is proposed, that is a unification of the approaches in both *margin-based* and *manifold-based* regularization. As such it represents a clean version of

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

the LDS method in a single objective, rather than an *ad-hoc* two-stage approach. Experimental results show its good performance.

- We train our system using stochastic gradient descent and choose linear or multi-layer architectures rather than kernel methods. This results in far faster training and testing times than existing methods, and also allows semi-supervised learning to be performed *online*. Our method can easily scale to *millions* of examples.

- We show it is also possible to encode domain knowledge into our multi-layer architecture approach, resulting in excellent generalization performance. This is demonstrated by training semi-supervised convolutional networks for image data.

The rest of the article is as follows. Section 2 describes in detail existing margin and manifold based regularization approaches, and scalability of the resulting algorithms. Section 3 describes our proposed approach, Section 4 compares it experimentally to existing methods, and Section 5 concludes.

2. Existing Approaches

As stated in the introduction, two of the most popular loss functions (regularizers) for using unlabeled data are *margin*-based regularization as in TSVMs and *manifold*-based regularization. We will discuss each of these in turn.

2.1. TSVMs

The Transductive Support Vector Machine (TSVM) is an algorithm originally proposed by Vapnik (1998) to take advantage of both a labeled training set and an unlabeled test set during prediction time. It was named that way because Vapnik proved bounds on generalization performance given the availability of the test set that were superior to induction based on using the labeled training set alone. The idea of the algorithm was:

(i) Choose a nested set of functions $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \ldots$ of increasing capacity.

(ii) For each possible labeling of the test examples, find the smallest subset $\mathcal{F}_k$ that can classify both training and testing data correctly.

(iii) Choose the labeling which required the smallest index k.

In terms of actual implementation it is known that the notion of margin – the distance of examples from the classifier's decision rule – is connected to the concept of capacity (Vapnik, 1998), so a simple algorithm is the following: *choose the decision rule that maximizes the margin on both labeled and unlabeled examples.*

The Support Vector Machine (Vapnik, 1998) for two-class classification already implements a margin based capacity control on *labeled* examples, using an optimization problem of the following form:

$$\min_{w,b} \gamma||w||^2 + \sum_{i=1}^{L} \ell(f(x_i), y_i) \tag{1}$$

where the family of functions are

$$f(x) = w \cdot x + b \tag{2}$$

and $\{(x_i, y_i), \ldots, (x_L, y_L)\} \subset \mathbb{R}^d \times \{\pm 1\}$ are the labeled training examples, and the loss function $\ell(\cdot, \cdot)$ is the so-called hinge loss:

$$\ell(f(x), y) = \max(0, 1 - yf(x)). \tag{3}$$

To implement Transductive SVMs it is (almost) sufficient to take the SVM optimization problem (1) and add an extra term for the unlabeled examples:

$$\min_{w,b} \gamma||w||^2 + \sum_{i=1}^{L} \ell(f(x_i), y_i) + \lambda \sum_{i=1}^{U} \ell^*(f(x_i^*)) \tag{4}$$

where the U unlabeled examples use the so-called symmetric hinge loss function

$$\ell^*(f(x^*)) = \max(0, 1 - |f(x^*)|) \tag{5}$$

which, intuitively speaking, pushes the unlabeled examples far from the margin: the absolute value is necessary in equation (5) because one does not know which side of the hyperplane those examples should lie on, unlike the labeled examples, so effectively the classifier trains on its own predictions. This notion of *self-learning* (Chapelle et al., 2006) can cause disastrous consequences in some cases: especially when the dimensionality $d \gg L$ one might be able to classify all unlabeled examples as belonging to one class whilst still classifying the labeled data correctly, giving a low value of the objective function, but nonsense results. This is solved by introducing a so-called balancing constraint which attempts to keep some of the unlabeled examples in each class.

Many researchers seem to be believe that the TSVM objective function is a good choice for semi-supervised

learning. However, finding a solution to the non-convex problem is far from easy, and thus several implementations have been attempted thus far. We will now describe some of those specific implementations, and their key differences.

S^{3}VM The authors of (Bennet & Demiriz, 1998) proposed to use mixed integer programming to find the labeling with the lowest objective function. The optimization appears intractable for large datasets, as "the solver failed due to excessive branching" in those cases. Only the linear case was considered, and no balancing constraint was used.

SVMLight-TSVM In (Joachims, 1999) a heuristic algorithm was proposed that at first fixes the labels of the unlabeled examples and then iteratively switches those labels to improve the TSVM objective function, solving a convex SVM objective function at each step. The nonlinear case is implemented by solving in the dual, resulting in a kernel model of the form:

$$f(x) = \sum_{i=1}^{L} \alpha_i y_i K(x_i, x) + \sum_{i=1}^{U} \alpha_i^* K(x_i^*, x) + b \quad (6)$$

A balancing constraint enforces that the fraction of positive and negatives assigned to the unlabeled data should be the same fraction as found in the labeled data. According to the proof of convergence, the algorithm at worst case could look at all 2^U labelings, but this is rather unlikely. The algorithm can deal with a few thousand examples in the nonlinear case in practice, but is faster in the linear case.

VS^{3}VM In (Fung & Mangasarian, 2001) a concave-convex minimization approach was proposed that solves successive convex problems, usually requiring only 5-7 linear programs, where they chose the L_1 norm of w as a regularizer instead of the L_2 norm. They studied the linear case, with no balancing constraint. This method will scale like the linear solver used in each iteration.

∇TSVM More recently, the authors of (Chapelle & Zien, 2005) proposed to optimize TSVM by gradient descent in the primal. For the nonlinear case, Kernel PCA has to be performed so that optimization in the primal is possible. This algorithm is faster than SVMLight-TSVM at least for small datasets (Collobert et al., 2006), but still has cubic complexity $O((U+L)^3)$. This method also requires one to store the entire kernel matrix of $(U+L)^2$ elements in memory, which clearly becomes infeasible for large datasets. The authors introduced a balancing constraint that is amenable to gradient descent:

$$\frac{1}{U}\sum_{i=1}^{U} f(x_i^*) \;=\; \frac{1}{L}\sum_{i=1}^{L} y_i \,. \quad (7)$$

CCCP-TSVM The authors of (Collobert et al., 2006) proposed to apply the Concave-Convex procedure for non-convex problems to TSVMs, which can be seen as a nonlinear extension of VS^{3}VMs. It uses the same balancing constraint as ∇TSVM. This implementation is over 100 times faster than SVMLight-TSVM and 50 times faster than ∇TSVM (for $L+U=2000$), and appears to scale better as well. It has empirically quadratic complexity because it relies on the sparsity of the SVM solution for improved speed and memory requirements. However, it still takes around 40 hours on a modern machine to solve a problem with 60,000 unlabeled examples in the nonlinear case.

Large Scale Linear TSVMs The authors of (Sindhwani & Keerthi, 2006) recently proposed a large scale TSVM method for the linear case. They focused on text problems with large sparse feature vectors and train the model (2) directly in the primal. In particular, they use a label switching heuristic like SVMLight-TSVM, but switch multiple labels at once.

In the nonlinear case things are not so easy. One is restricted in the quest to reduce training time by the prediction speed of the model (6) . Moreover, computation grows as the training data grows (Steinwart & Scovel, 2005). Even if one tries tricks to keep a fixed number of basis functions these methods are still slow compared to multi-layer models (Burges, 1996).

2.2. Manifold-based regularization

A separate direction of research in semi-supervised learning is manifold-learning based regularization. The main idea in these approaches is to find a representation of the data which collapses points lying in the same manifold so that a classification algorithm can easily predict that they share the same class label.

These methods can be split into two categories: those which treat this as a two-stage problem: (i) learn an embedding and (ii) train a classifier in this new space, and those which try to do everything in a single step.

To train a two-stage classifier, in the first stage one employs any manifold-learning algorithm such as Isomap (Tenenbaum et al., 2000), Laplacian Eigenmaps (Belkin & Niyogi, 2003) or spectral clustering (Ng et al., 2002). The authors of (Chapelle et al., 2003) use such methods to build a kernel for SVMs

and call these kernels "cluster kernels". The "graph"-SVM method proposed in (Chapelle & Zien, 2005) also builds a kernel for SVM. In this method one embeds in a space where distances are the shortest paths on the graph weighted with the original distance measure, similar to the Isomap algorithm. Thus, points connected by regions of high density are close to each other in the new space.

To learn a single stage classifier, one has to introduce a regularizing term in the objective function which directly encodes behavior such as that described in the previous paragraph. The Laplacian Eigenmaps embedding algorithm in particular employs an objective function that is easily encoded in a classifier:

$$\sum_{ij} W_{ij}||f(x_i) - f(x_j)||^2 \tag{8}$$

Such a regularizer has been used both to generalize a Parzen-windows (Duda & Hart, 1973) type classifier resulting in a method called label propagation (Zhu & Ghahramani, 2002), and in SVMs. The SVM method is called Laplacian SVMs (LapSVM) (Sindhwani et al., 2005) and minimizes:

$$\min_{w,b} \sum_{i=1}^{L} \ell(f(x_i), y_i) + \gamma||w||^2 + \lambda \sum_{i,j=1}^{U} W_{ij}||f(x_i^*) - f(x_j^*)||^2 \tag{9}$$

We speculate here that forcing the Euclidean distance to be small if two points are assumed to be the same label might be a little stringent as for prediction it is only the sign of $f(x^*)$ that is important. Moreover, we also note that the lack of balancing constraint might mean in high dimensions that all the unlabeled examples can collapse to a single prediction.

In contrast, the LDS method (Chapelle & Zien, 2005) proposes to use both TSVM and manifold regularizers at once in a two-stage method. First, the Isomap-like embedding method of "graph"-SVM is used whereby data is clustered. Then, in the new embedding space, ∇TSVM is applied. The authors found that using both regularizers at once was better than using one type of regularizer alone.

In summary, we have discussed several algorithms which use two main types of regularizer: a clustering or an embedding that takes into account *structure* in the unlabeled data. Indeed TSVM is a kind of large margin clustering as has been exploited in (Xu et al., 2005) and is strongly related to classical techniques like competitive learning (Duda & Hart, 1973). In (Chapelle & Zien, 2005) the authors speculate that manifold-based regularization has a stabilizing effect on TSVM optimization. Without such neighborhood-based regularization TSVMs only compare unlabeled examples to the existing model, and not to each other. Using both approaches as in LDS is thus a smart idea, however it suffers from two problems: (i) the two-stage approach seems *ad-hoc* and (ii) the method is slow.

In the next Section we propose a new approach which remedies these problems.

3. Proposed Approach

We propose the following algorithm, named Manifold Transduction: minimize

$$\frac{1}{L}\sum_{i=1}^{L} \ell(f(x_i), y_i) + \frac{\lambda}{U^2} \sum_{i,j=1}^{U} W_{ij}\ \ell\big(f(x_i^*), y^*(\{i,j\})\big) \tag{10}$$

where

$$y^*(N) = \text{sign}(\sum_{k \in N} f(x_k^*)) \tag{11}$$

where the edge weights W_{ij} define pairwise similarity relationships between unlabeled examples x^*.

This objective, like TSVMs objective, is non-convex and there is no simple optimization scheme for solving it even for linear models such as kernel machines. Because of this fact, and the scalability problems with nonlinear kernel methods, we propose several novel algorithmic choices in its implementation:

(i) We minimize this function in the primal by stochastic gradient descent. This makes *online* semi-supervised learning possible for the first time.

(ii) In the nonlinear case we employ a multi-layer architecture to define $f(x)$. This makes both training and testing far faster than competing kernel methods such as TSVM.

(iii) We also make a specific recommendation for the implementation of an *online* balancing constraint.

We will now study this algorithm, and explain the reason for these choices in detail.

3.1. Objective function

In (10) we propose a new loss function for *unlabeled examples*:

$$\ell^*(f(x_i^*)) = \ell\Big(f(x_i^*), y^*(N)\Big) \tag{12}$$

where N is a set of examples that one believes share the same label, e.g. a set of neighboring examples. The

function y^* predicts the label of that set by taking the mean prediction.

For both labeled and unlabeled training data we use the hinge loss (3) as in SVMs.

In equation (10) we consider pairs of examples, weighted by the graph W_{ij}. If $W_{ii} = 1$ and $W_{ij} = 0$ for $i \neq j$ then we recover the TSVM loss function:

$$\ell^*(f(x_i^*)) = \ell\Big(f(x_i^*), \text{sign}(f(x_i^*))\Big) \qquad (13)$$

because we do not take neighborhood information into account.

Setting $W_{ij} = 1$ if x_i^* is among the k-nearest neighbors of x_j^*, and zero otherwise, our algorithm becomes a natural generalization of TSVM that regularizes using neighborhood information. This is a similar regularizer to the neighborhood-based manifold regularizers of Section 2.2 but based on *clustering* rather than *embedding*.

We make the assumption that if two examples are neighbors then they have the same *class label*, whereas manifold-based regularization assumes they are close in an *embedding space*. Our constraint is not as strict, but captures the prior we wish to encode. For example, if one class of data has more variance than the other, then the regularization of (9) might focus on that class, and ignore the other.

Extensions of our algorithm are also possible. First, in the multi-class case where $f(x^*)$ outputs a c-dimensional vector, we can define $y^*(N) = \text{argmax} \sum_{k \in N} f(x_k^*)$. Further, if the set N contains more than two examples then our algorithm takes into account a neighborhood in analogy to k-nearest neighbor. This is not easily possible with the approach of (9) which is limited to pairs.

3.2. Model: Multi-Layer Architecture

As already discussed, the issue that makes all the previously described algorithms computationally expensive in the nonlinear case is their choice of the kernel expansion (6). Instead we propose to use a multi-layer model of the form:

$$f(x) = \sum_{i=1}^{d} w_i^0 \; h_i(x) + b$$

where typically one chooses hidden units

$$h_i(x) = S\left(\sum_j w_j^i \; x_j + b^i\right)$$

Algorithm 1 Online Manifold Transduction

Input: labeled data (x_i, y_i) and unlabeled data x_i^*
repeat
  Pick a random labeled example (x_i, y_i)
  Make a gradient step to optimize $\ell(f(x_i), y_i)$
  Pick a random unlabeled example x_i^*
  Pick a random neighbor x_j^* of x_i^*
  Predict label $y^* = y^*(\{i, j\})$
  if fraction of recent assignments to class $y^* < p_{est}(y^*)$ (see Section 3.4) **then**
    Make a gradient step for $\ell(f(x_i^*), y^*)$
  end if
until stopping criteria is met.

where S is a non-linear squashing function. We use the Hard Tanh function:

$$S(x) = \begin{cases} 1 & \text{if } x \geq 1 \\ -1 & \text{if } x \leq -1 \\ x & \text{otherwise.} \end{cases}$$

In the multi-class case we define one output $f_i(x)$ for each class, but each function f_i shares the same hidden units h_j, as is often done in neural network models.

The flexibility of using multi-layer architectures also allows us to encode prior knowledge into our model. For example, convolutional neural networks (CNNs) (LeCun et al., 1998) have several layers of image patch based feature maps applied across the input image. Such networks have been shown to perform very well in digit, face and 3D object detection tasks.

3.3. Optimization: Stochastic Gradient

We optimize our objective *online*, in the primal, using stochastic gradient descent. Recent experimental comparisons show this approach often outperforms sophisticated optimizer schemes (Bottou, 2007). To simplify the hyperparameters we fix $\lambda = 1$ in our experiments, yielding the method described in Algorithm 1. If the model is multi-layered then we use backpropagation (see, e.g. (Duda & Hart, 1973)) during the gradient step. A typical stopping criteria is to use a validation set or to measure the objective function value.

3.4. Balancing Constraint

To implement a balancing constraint while learning *online* we keep a cache of (arbitrarily) the last $25c$ predictions $f(x_i^*)$ where c is the number of classes. This is dependent on c because if c is large the cache must also be large or the estimates will be too poor. We then try to make the next prediction *balanced* assuming we

have a fixed estimate $p_{est}(y)$ of the probability of each class, which without further information, can be estimated from the labeled data: $p_{trn}(y=i) = \frac{|\{i:y_i=i\}|}{L}$. We consider two alternatives:

1. **∇bal** Adding the term (7) to the objective function multiplied by a scaling factor as in ∇TSVMs. The disadvantage of such an approach is that the scaling factor is a further hyperparameter.
2. **ignore–bal** Count how many examples in the cache have been attributed to each class. If the next unlabeled example x^* is given a label y^* by the model that already has too many examples assigned to it, then we simply do not make a gradient step for this example.

We note that the quality of p_{trn} depends on the ratio of labeled examples L to the number of classes c, not the input dimensionality d. Thus it may be a good estimate in many real datasets. However, because in some of the small datasets used in (Chapelle & Zien, 2005) it is a poor estimate we consider improving this estimate by taking into account that we have access to unlabeled data. We suggest the following simple method p_{knn}: label the k nearest neighbors of each labeled example with its label. If k is large enough some labeled points will label the same examples, and so when we count the number of points assigned to each class, we achieve a smoothed version of p_{trn}.

4. Experiments

4.1. Small Scale Datasets

We first report results on three small-scale datasets, summarized in Table 1. We follow the methodology in (Chapelle & Zien, 2005; Collobert et al., 2006) and report the best mean test error for a *fixed* set of hyperparameters over 10 splits of the data. For our method, we test standard transduction (our regularizer with no neighborhood information), called TNN (Transductive Neural Network), and our method *with* neighborhood information, called ManTNN (Manifold Transduction Neural Network). We also compute a baseline Neural Network (NN).

For NN, TNN and ManTNN we fixed 50000 iterations of Algorithm 1 and for ManTNN we chose 10 nearest neighbors for all datasets. We also choose not to minimize $\ell(f(x_i^*), y^*)$ for the first $10L$ iterations so that the classifier first finds a good model with labeled data alone before using the unlabeled data. We thus have two free parameters: the choice of hidden units {0,50,100,150,200} and the choices of learning rate { 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005 }.

Table 1. Datasets used in the experiments. The first three are small-scale datasets using the same experimental setup as found in (Chapelle & Zien, 2005). Mnist1h and Mnist1k use the same experimental setup as in (Collobert et al., 2006). Mnist1k+Invar uses shifted versions of digits to make an unlabeled set of 630,000 examples.

data set	classes	dims	points	labeled
g50c	2	50	500	50
Text	2	7511	1946	50
Uspst	10	256	2007	50
Mnist1h	10	784	70k	100
Mnist1k	10	784	70k	1000
Mnist1k+Invar	10	784	630k	1000

Table 2. Test Error for various methods for enforcing the balancing constraint, see Section 3.4 for explanation. The "no bal" method does not use a balancing constraint. p_{trn} and p_{tst} balance using the training and testing set distributions respectively, and p_{knn} estimates the true distribution using a k-nn based method on the unlabeled data.

	Uspst			g50c		
	p_{trn}	p_{knn}	p_{tst}	p_{trn}	p_{knn}	p_{tst}
TNN						
no bal	22.3	–	–	6.5	–	–
∇bal	30.4	29.3	29.4	6.5	6.5	6.5
ignore-bal	19.1	16.1	12.5	6.1	6.3	6.3
ManTNN						
ignore-bal	15.6	11.9	8.5	5.9	5.7	5.5

Table 3. Transductive (T-) and Manifold Transduction (ManT-) versions of Neural Networks (NN) as well as a baseline NN are compared to existing methods on Small-Scale Datasets. Following (Chapelle & Zien, 2005) all methods apart from those marked (*) have test error rates reported for a *fixed* set of hyperparameters averaged over 10 splits, where that fixed set is chosen using the test error itself. All methods have 2 free hyperparameters. In comparison, the methods marked (*) have parameters optimized on each split using 5-fold cross-validation.

	g50c	Text	Uspst
SVM	8.32	18.86	23.18
SVMLight-TSVM	6.87	7.44	26.46
CCCP-TSVM	5.62	7.97	16.57
∇TSVM	5.80	5.71	17.61
LapSVM$^{(*)}$	5.4	10.4	12.7
LDS$^{(*)}$	5.4	5.1	15.8
Label propagation	17.30	11.71	21.30
graph	8.32	10.48	16.92
NN	8.54	15.87	24.57
TNN	6.34	6.11	16.06
ManTNN	5.66	5.34	11.90

Balancing constraint comparison We first compare the balancing constraint methods ∇bal and ignore-bal and three different strategies for computing the class distribution p_{trn}, p_{knn} and p_{tst} as described in Section 3.4. p_{trn} measures the training set distribution, p_{knn} estimates the unlabeled set distribution (which is what we are really interested in) by using a k-nn like method, and p_{tst} is the true distribution of the unlabeled data, which is inaccessible in a real situation. A comparison on two of the datasets is given in Figure 2 (results are similar for "text").

We could not get ∇bal to work well in an online situation whereas the simple ignore-bal heuristic gives good results. Due to the small dataset size the difference in test error between using p_{tst} and p_{trn} is actually quite large. Using p_{knn} seems to be a better estimate than p_{trn}. We therefore adopt ignore-bal and the p_{knn} method in the following experiments.

Comparison with other methods We compare TNN and ManTNN to several TSVM implementations as well as label propagation, graph, LapSVM and LDS on the small scale datasets. The results given in Table 3 show that both TNN and ManTNN are competitive with existing approaches, and ManTNN, which includes the manifold-based transductive regularizer, outperforms TNN, which uses transduction alone.

4.2. Large Scale Dataset: MNIST

We then compared our method to SVMs and TSVMs on a "semi-supervised version" of the MNIST digit database, following (Collobert et al., 2006), using either 100 or 1000 labeled examples and 70000 unlabeled examples, and a validation set of 1000 examples for choosing parameters. Error rates are measured on the MNIST test set.

We used a two-layer neural network as before (baseline NN, TNN, ManTNN), choosing from the same set of hidden units and learning rates. We use the validation set as a stopping criteria for Algorithm 1.

We also applied convolutional networks (CNNs), and transductive versions of them, to this task. We chose an architecture similar to (LeCun et al., 1998). There are 6 layers. The first is six 3x3 spatial convolutions (outputting 26x26x6 features to the next layer). The second is six spatial 2x2 spatial subsamplings (outputting 13x13x6 features). The third is sixteen 4x4 spatial convolutions (outputting 10x10x16 features). The fourth is sixteen 2x2 spatial subsamplings (giving 5x5x16 features). The fifth is fifty 5x5 spatial convolutions (giving 1x1x50 features). This is followed by a standard fully connected layer with n hidden units (chosen as in the two-layer net), followed by a linear layer yielding the final 10 outputs (class predictions). CNNs encode prior knowledge about spatial features within the image, which should give improved accuracy over a standard NN.

Table 4. Results on Large-Scale Datasets: MNIST with 100 or 1000 labels and 70,000 unlabeled examples. Test Error is reported for Transductive (T-) and Manifold Transduction (ManT-) versions of Neural Networks (NN) and convolutional networks (CNN), and compared to SVMs and TSVMs. *ManTCNN (p_{tst})* uses the test distribution as the balancing constraint, which if this information were available, would give improved performance.

	Mnist1h	Mnist1k
SVM	23.44	7.77
CCCP-TSVM	16.81	5.38
NN	25.81	10.70
TNN	18.02	6.66
ManTNN	7.30	2.88
CNN	22.98	6.45
TCNN	13.01	3.50
ManTCNN	6.65	2.15
ManTCNN (p_{tst})	*1.96*	*1.87*

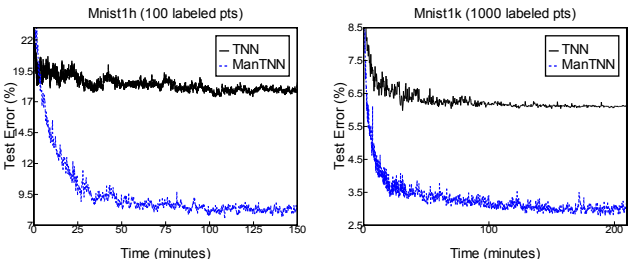


Figure 1. Test error versus training times for the TNN and ManTNN algorithms on Mnist (100 or 1000 labeled examples, 70000 unlabeled examples). These results compare favourably with the time to train the fastest TSVM algorithm (Collobert et al., 2006) which took 41.9 hours on the same machine.

The results are given in Table 4. The baseline NN performs slightly worse than SVM, but CNNs perform slightly better. Applying transduction, TNN is slightly worse than TSVMs, but TCNN is slightly better. Manifold Transduction outperforms all these methods, with ManTNN and ManTCNN performing almost as well as each other. The last row in the table shows ManTCNN trained with the true balancing constraint (knowing the test distribution). It appears that for only 100 labeled examples knowing this distribution could make results even better, although with 1000 labeled examples this is less important.

Training time for TNNs and ManTNNs are given in Figure 1. The results are shown for the best choice of hidden units and learning rate as chosen on the validation set. TNNs take around one hour to reach convergence, and ManTNNs (omitting the time to compute neighbors for ManTNN) take slightly longer. These times should be compared to the fastest TSVM implementation, CCCP-TSVMs, which took 41.9 hours on the same machine. Our code is not particularly optimized and is written in a scripting language with a C++ back-end. On MNIST, a nonlinear TNN with 200 hidden units can process 1 million unlabeled examples in an *online* fashion in 12.5 minutes.

Mnist1k+Invar We also performed experiments on MNIST1k with a larger unlabeled set of 630,000 examples by translating the original set by at most one pixel in each direction. TNN achieves a test error of 5.23% on the original test set, when choosing the training iteration that gives the minimum validation error, and ManTNN achieves a test error of 2.43%. Both methods outperform their counterparts trained with less unlabeled data using MNIST1k. Training time took 4.47 hours and 3.96 hours respectively for the two algorithms, including the computation time for generating the invariances.

5. Conclusions

In this article we introduced a large scale *nonlinear* method that elegantly combines the two main regularization principles for discriminative semi-supervised learning: transduction and neighborhood-based (manifold-based) regularization. Our future work will be to apply this approach to real large-scale nonlinear problems e.g. applications in vision and natural language processing.

References

Belkin, M., & Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: a geometric framework for learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, *7*, 2399–2434.

Bennet, K., & Demiriz, A. (1998). Semi-Supervised Support Vector Machines. *NIPS 12, 1998*. MIT Press, Cambridge, MA.

Bottou, L. (2007). `http://leon.bottou.org/projects/sgd`.

Burges, C. (1996). Simplified Support Vector Decision Rules. *ICML*, 71–77.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Adaptive computation and machine learning. Cambridge, Mass., USA: MIT Press.

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. *NIPS 15* (pp. 585–592). Cambridge, MA, USA: MIT Press.

Chapelle, O., & Zien, A. (2005). Semi-supervised classification by low density separation. *AISTATS* (pp. 57–64).

Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Large scale transductive svms. *Journal of Machine Learning Research*, *7*, 1687–1712.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley & Sons.

Fung, G., & Mangasarian, O. (2001). Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, *15*, 29–44.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *International Conference on Machine Learning, ICML*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*.

Ng, A. Y., Jordan, M., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press.

Sindhwani, V., & Keerthi, S. S. (2006). Large scale semi-supervised linear SVMs. *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 477–484). New York, NY, USA: ACM Press.

Sindhwani, V., Niyogi, P., & Belkin, M. (2005). Beyond the point cloud: from transductive to semi-supervised learning. *International Conference on Machine Learning, ICML*.

Steinwart, I., & Scovel, C. (2005). Fast rates to bayes for kernel machines. *NIPS*, *17*, 1345–1352.

Tenenbaum, J., de Silva, V., & Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Vapnik, V. N. (1998). *Statistical learning theory*. John Wiley and Sons, New York.

Xu, L., Neufeld, J., Larson, B., & Schuurmans, D. (2005). Maximum margin clustering. *Advances in Neural Information Processing Systems*, *17*, 1537–1544.

Zhu, X., & Ghahramani, Z. (2002). *Learning from labeled and unlabeled data with label propagation* (Technical Report CMU-CALD-02-107). Carnegie Mellon University.

Non-Parametric Policy Gradients: A Unified Treatment of Propositional and Relational Domains

Kristian Kersting KRISTIAN.KERSTING@IAIS.FRAUNHOFER.DE
Dept. of Knowledge Discovery, Fraunhofer IAIS, Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Kurt Driessens KURT.DRIESSENS@CS.KULEUVEN.BE
Dept. of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium

Abstract

Policy gradient approaches are a powerful instrument for learning how to interact with the environment. Existing approaches have focused on propositional and continuous domains only. Without extensive feature engineering, it is difficult – if not impossible – to apply them within structured domains, in which e.g. there is a varying number of objects and relations among them. In this paper, we describe a non-parametric policy gradient approach – called NPPG – that overcomes this limitation. The key idea is to apply Friedmann's gradient boosting: policies are represented as a weighted sum of regression models grown in an stage-wise optimization. Employing off-the-shelf regression learners, NPPG can deal with propositional, continuous, and relational domains in a unified way. Our experimental results show that it can even improve on established results.

1. Introduction

Acting optimally under uncertainty is a central problem of artificial intelligence. If an agents learns to act solely on the basis of the rewards associated with actions taken, this is called reinforcement learning (Sutton & Barto, 1998). More precisely, the agent's learning task is to find a policy for action selection that maximizes its reward over the long run.

The dominant reinforcement learning (RL) approach for the last decade has been the value-function approach. An agent uses the reward it occasionally receives to estimate a value-function indicating the expected value of being in a state or of taking an action in a state. The policy is represented only implicitly, for instance as the policy that selects in each state the action with highest estimated value. As Sutton et al. (2000) point out, the value function approach, however, has several limitations. First, it seeks to find deterministic policies, whereas in real world applications the optimal policy is often stochastic, selecting different actions with specific probabilities. Second, a small change in the value-function parameter can push the value of one action over that of another, causing a discontinuous change in the policy, the states visited, and overall performance. Such discontinuous changes have been identified as a key obstacle to establishing convergence assurances for algorithms following the value-function approach (Bertsekas & Tsitsiklis, 1996). Finally, value-functions can often be much more complex to represent than the corresponding policy as they encode information about both the size and distance to the appropriate rewards. Therefore, it is not surprising that so called policy gradient methods have been developed that attempt to avoid learning a value function explicitly (Williams, 1992; Baxter et al., 2001; Konda & Tsitsiklis, 2003). Given a space of parameterized policies, they compute the gradient of the expected reward with respect to the policy's parameters, move the parameters into the direction of the gradient, and repeat this until they reach a local optimum. This direct approximation of the policy overcomes the limitations of the value function approach stated above. For instance, Sutton et al. (2000) show convergence even when using function approximation.

Current policy gradient methods have focused on propositional and continuous domains assuming the environment of the learning agent to be representable as a vector-space. Nowadays, the role of structure and relations in the data, however, becomes more and

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

more important (Getoor & Taskar, 2007): information about one object can help the agent to reach conclusions about other objects. Such domains are hard to represent meaningfully using a fixed set of features. Therefore, relational RL approaches have been developed (Džeroski et al., 2001), which seek to avoid explicit state and action enumeration as – in principle – traditionally done in RL through a symbolic representation of states and actions. Existing relational RL approaches, however, have focused on value functions and, hence, suffer from the same problems as their propositional counterparts as listed above.

In this paper, we present the first model-free policy gradient approach that deals with relational and propositional domains. Specifically, we present a non-parametric approach to policy gradients, called NPPG. Triggered by the observation that finding many rough rules of thumb of how to change the way to act can be a lot easier than finding a single, highly accurate policy, we apply Friedmann's (2001) gradient boosting. That is, we represent policies as weighted sums of regression models grown in a stage-wise optimization. Such a functional gradient approach has recently been used to efficiently train conditional random fields for labeling (relational) sequences using boosting (Dietterich et al., 2004; Gutmann & Kersting, 2006) and for policy search in continuous domains (Bagnell & Schneider, 2003). In contrast to the supervised learning setting of the sequence labeling task, feedback on the performance is received only at the end of an action sequence in the policy search setting. The benefits of a boosting approach to functional policy gradients are twofold. First, interactions among states and actions are introduced only as needed, so that the potentially infinite search space is not explicitly considered. Second, existing off-the-shelf regression learners can be used to deal with propositional and relational domains in a unified way. To the best of the authors' knowledge, this is the first time that such a unified treatment is established. As our experimental results show, NPPG can even significantly improve upon established results in relational domains.

We proceed as follows. We will start off by reviewing policy gradients and their mathematical background. Afterwards, we will develop NPPG in Section 3. In Section 4, we will present our experimental results. Before concluding, we will touch upon related work.

2. Policy Gradients

Policy gradient algorithms find a locally optimal policy starting from an arbitrary initial policy using a gradient ascent search through an explicit policy space. Consider the standard RL framework (Sutton & Barto, 1998), where an agent interacts with a Markov Decision Process (MDP). The MDP is defined by a number of states $s \in \mathcal{S}$, a number of actions $a \in \mathcal{A}$, state-transition probabilities $\delta(s,a,s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ that represent the probability that taking action a in state s will result in a transition to state s' and a reward function $r(s,a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. When the reward function is nondeterministic, we will use the expected rewards $R(s,a) = E_{s,a}[r(s,a)]$, where $E_{s,a}$ denotes the expectation over all states s and actions a. The state, action, and reward at time t are denoted as $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$ and $r_t (or\ R_t) \in \mathbb{R}$.

The agent selects which action to execute in a state following a policy function $\pi(s,a) : \mathcal{S} \times \mathcal{A} \to [0,1]$. Current policy gradient approaches assume that the policy function π is parameterized by a (weight) vector $\theta \in \mathbb{R}^n$ and that this policy function is differentiable with respect to its parameters, i.e., that $\frac{\partial \pi(s,a,\theta)}{\partial \theta}$ exists. A common choice is a Gibbs distribution based on a linear combination of features:

$$\pi(s,a) = e^{\Psi(s,a)} / \left(\sum\nolimits_b e^{\Psi(s,b)} \right) , \qquad (1)$$

where the potential function $\Psi(s,a) = \theta^T \phi_{sa}$ with ϕ_{sa} the feature vector describing state s and action a. This representation guarantees that the policy specifies a probability distribution independent of the exact form of the function Ψ. Choosing a parameterization, creates an explicit policy space $\mathbb{R}^n$ with n equal to size of the parameter vector θ. This space can be traversed by an appropriate search algorithm.

Policy gradients are computed w.r.t. a function ρ that expresses the value of a policy in an environment, $\rho(\pi) = \sum_{s \in \mathcal{S}} d^\pi(s) \sum_{a \in \mathcal{A}} \pi(s,a) \cdot Q^\pi(s,a)$, where Q^π is the usual (possibly discounted) state-action value function, e.g., $Q^\pi(s,a) = E_\pi \left[\sum_{i=0}^{\infty} \gamma^i R_{t+i} | s_t = s, a_t = a \right]$, and $d^\pi(s)$ is the (possibly discounted) stationary distribution of states under policy π. We assume a fully ergodic environment so that $d^\pi(s)$ exists and is independent of any starting state s_0 for all policies.

A property of ρ for both average reward reinforcement learning and episodic tasks with a fixed starting state s_0 is that (Sutton et al., 2000, Theorem 1)

$$\frac{\partial \rho}{\partial \theta} = \sum\nolimits_s d^\pi(s) \sum\nolimits_a \frac{\partial \pi(s,a)}{\partial \theta} \cdot Q^\pi(s,a) . \qquad (2)$$

Important to note is that this gradient does not include the term $\frac{\partial d^\pi(s)}{\partial \theta}$. Eq. (2) allows the computation of an approximate policy gradient through exploration. By sampling states s through exploration of the environment following policy π, the distribution $d^\pi(s)$ is

automatically represented in the generated sample of encountered states. The sum $\sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a)$ then becomes an unbiased estimate of $\frac{\partial \rho}{\partial \theta}$ and can be used in a gradient ascent algorithm.

Of course, the value Q^π is unknown and must be estimated for each visited state-action pair either by using a Monte Carlo approach or by building an explicit representation of Q, although in this latter case care must be taken when choosing the parameterization of the Q-function (Sutton et al., 2000).

3. Non-Parametric Policy Gradients

A drawback of a fixed, finite parameterization of a policy such as in Eq. (1) is that it assumes each feature makes an independent contribution to the policy. Of course it is possible to define more features to capture combinations of the basic features, but this leads to a combinatorial explosion in the number of features, and hence, in the dimensionality of the optimization problem. Moreover, in continuous and in relational environments it is not clear at all which features to choose as there are infinitely many possibilities.

To overcome these problems, we introduce a different policy gradient approach based on Friedmann's (2001) gradient boosting. In our case, the potential function Ψ in the Gibbs distribution[1] of Eq. (1) is represented as a weighted sum of regression models grown in an stage-wise optimization. Each regression model can be viewed as defining several new feature combinations. The resulting policy is still a linear combination of features, but the features can be quite complex. Formally, gradient boosting is based on the idea of functional gradient ascent, which we will now describe.

3.1. Functional Gradient Ascent

Traditional gradient ascent estimates the parameters θ of a policy iteratively as follows. Starting with some initial parameters θ_0, the parameters θ_m in the next iteration are set to the current parameters plus the gradient of ρ w.r.t. to θ, i.e., $\theta_m = \theta_0 + \delta_1 + \ldots + \delta_m$ where $\delta_m = \eta_m \cdot \partial\rho/\partial\theta_{m-1}$ is the gradient multiplied by a constant η_m, which is obtained by doing a line search along the gradient. Functional gradient ascent is a more general approach, see e.g. (Friedman, 2001; Dietterich et al., 2004). Instead of assuming a linear parameterization for Ψ, it just assumes that Ψ will be represented by a linear combination of functions. Specifically, one starts with some initial function Ψ_0, e.g. based on the zero potential, and iteratively adds corrections $\Psi_m = \Psi_0 + \Delta_1 + \ldots + \Delta_m$. In contrast to the standard gradient approach, Δ_m here denotes the so-called functional gradient, i.e., $\Delta_m = \eta_m \cdot E_{s,a}\left[\partial\rho/\partial\Psi_{m-1}\right]$. Interestingly, this functional gradient coincides with what traditional policy gradient approach estimate, namely (2), as the following theorem says.

Theorem 3.1 (Functional Policy Gradient) *For any MDP, in either the average-reward or start-state formulation,*

$$E_{s,a}\left[\frac{\partial \rho}{\partial \Psi}\right] = \sum\nolimits_{s,a} d^\pi(s) \cdot \frac{\partial \pi(s,a)}{\partial \Psi} \cdot Q(s,a)\,. \quad (3)$$

This is a straightforward adaptation of Theorem 1 in (Sutton et al., 2000) and is also quite intuitive: the functional policy gradient indicates how we would like the policy to change in all states and actions in order to increase the performance measure ρ.

Unfortunately, we do not know the distribution $d^\pi(s)$ of how often we visit each state s under policy π. This is, however, easy to approximate from the empirical distribution of states visited when following π:

$$E_{s,a}\left[\frac{\partial \rho}{\partial \Psi}\right] \approx E_{s\sim\pi}\left[\sum\nolimits_a \underbrace{\frac{\partial \pi(s,a)}{\partial \Psi} \cdot Q(s,a)}_{=:f_m(s,a)}\right]\,, \quad (4)$$

where the state s is sampled according to π, denoted as $s \sim \pi$. We now have a set of traning examples from the distribution $d^\pi(s)$, so we can compute the value $f_m(s,a)$ of the functional gradient at each of the training data points for all[2] actions a applicable in s. We can then use these point-wise functional gradients to define a set of training examples $\{(s,a), f_m(s,a)\}$ and then train a function $f_m : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ so that it minimizes the squared error over the training examples. Although the fitted function f_m is not exactly the same as the desired functional gradient in Eq. (3), it will point in the same general direction assuming there are enough training examples. So, taking a step $\Delta_m = \eta_m \cdot f_m$ will approximate the true functional policy gradient ascent.

[1] Other distributions are possible but are subject to future research. For instance, it would be interesting to investigate modeling joint actions of multiple agents in relatioanl domains a long the lines of Guestrin et al. (2002).

[2] Here, an update is made for all actions possible in each state encountered irrespective of which action was actually taken. Alternatively, we can only make an update for the one action actually taken (Baxter et al., 2001). To compensate for the fact that some actions are selected more frequently than others, we divide by the probability of choosing the action, i.e., we use $f_m(s,a)/\pi(s,a)$ as functional gradient training examples and do not run over all actions.

As an example, we will now derive the point-wise functional gradient of a policy parameterized as a Gibbs distribution (1). For the sake of readability, we denote $\Psi(s,a)$ as Ψ_a and $\Psi(s,b)$ as Ψ_b.

Proposition 3.1 *The point-wise functional gradient of ρ with respect to a policy parameterized as a Gibbs distribution* (1) *equals to $Q(s,a) \cdot \frac{\partial \pi(s,a)}{\partial \Psi}$ with*

$$\frac{\partial \pi(s,a)}{\partial \Psi(s',a')} = \begin{cases} \pi(s,a)(1-\pi(s,a)) & \text{if } s = s' \wedge a = a', \\ -\pi(s,a)\pi(s,a') & \text{if } s = s' \wedge a \neq a' \\ 0 & \text{otherwise,} \end{cases}$$

To see this, consider the first case; the other cases can be derived in a similar way. Due to the Gibbs distribution, we can write $(\partial \pi(s,a))/(\partial \Psi(s,a)) =$

$$\frac{\partial}{\partial \Psi_a} \frac{e^{\Psi_a}}{\sum_b e^{\Psi_b}} = \frac{e^{\Psi_a} \cdot \sum_b e^{\Psi_b} - e^{\Psi_a} \cdot \sum_b \partial e^{\Psi_b}/\partial \Psi_a}{\left[\sum_b e^{\Psi_b}\right]^2} .$$

Assuming $(\partial \Psi_b)/(\partial \Psi_a) = 0$, i.e., Ψ_a and Ψ_b are independent, it holds $\sum_b \partial e^{\Psi_b}/\partial \Psi_a = e^{\Psi_a}$ and we can rewrite the state-action gradient as

$$= e^{\Psi_a} \frac{\left(\sum_b e^{\Psi_b} - e^{\Psi_a} \cdot \frac{\sum_b e^{\Psi_b}}{\sum_b e^{\Psi_b}}\right)}{\left[\sum_b e^{\Psi_b}\right]^2} = e^{\Psi_a} \frac{\left(1 - \frac{e^{\Psi_a}}{\sum_b e^{\Psi_b}}\right)}{\sum_b e^{\Psi_b}}$$

which simplifies – due to the definition of $\pi(s,a)$ – to

$$\frac{\partial \pi(s,a)}{\partial \Psi(s,a)} = \pi(s,a)(1-\pi(s,a)) .$$

The key point is the assumption $(\partial \Psi_b)/(\partial \Psi_a) = 0$. This actually means that we model Ψ with k functions f_k, one for each action a, i.e., $\Psi(s,a) = f_a(s)$. In turn, we estimate k regression models f_m^a. In the experiments, however, learning a single regression model f_m did not decrease performance so that we stick here to the conceptually easier variant of learning a single regression model f_m.

We call policy gradient methods that follow the outlined functional gradient approach *non-parametric policy gradients*, or NPPG for short. They are non-parametric because the number of parameters can grow with the number of episodes.

3.2. Gradient Tree Boosting

NPPG as summarized in Alg. 1 describes actually a family of approaches. In the following, we will develop a particular instance, called TreeNPPG, which uses regression tree learners to estimate f_m in line 6.

In TreeNPPG, the policy is represented by sums of regression *trees*. Each regression tree can be viewed as defining several new feature combinations, one corresponding to each path in the tree from the root to a leaf. The resulting policies still have the form of a linear combination of features, but the features can be quite complex. The trees are grown using a regression tree learner such as CART (Breiman et al., 1984), which in principle runs as follows. It starts with the empty tree and repeatedly searches for the best test for a node according to some splitting criterion such as weighted variance. Next, the examples $\mathcal{R}$ in the node are split into $\mathcal{R}_s$ (success) and $\mathcal{R}_f$ (failure) according to the test. For each split, the procedure is recursively applied, obtaining subtrees for the respective splits. As splitting criterion, we use the weighted variance on $\mathcal{R}_s$ and $\mathcal{R}_f$. We stop splitting if the variance in one node is small enough or a depth limit was reached. In leaves, the average regression value is predicted.

We propose to use regression tree learners because a rich variety of variants exists that can deal with finite, continuous and even relational data. Depending on the type of the problem domain at hand, one can instantiate the TreeNPPG algorithm by choosing the appropriate regression tree learner. We will give several examples in the following experimental section.

Algorithm 1: Non-Parametric Policy Gradient

1 Let Ψ_0 be the zero potential (the empty tree)
2 **for** $m = 1$ *to* N **do**
3 Choose a starting state s
4 Generate episodes $\mathcal{E}_m$ starting in s following the policy π_{m-1} with $\pi_{m-1}(s,a) = e^{\Psi_{m-1}(s,a)} / \left(\sum_b e^{\Psi_{m-1}(s,b)}\right)$
5 Generate functional gradient examples $\mathcal{R}_m = \{(s_i, a_{ij}), f_m(s_i, a_{ij})\}$ based on $\mathcal{E}_m$
6 Induce regression model f_m based on $\mathcal{R}_m$
7 Set potential to $\Psi_m = \Psi_{m-1} + \Delta_m$ where $\Delta_m = \eta_m \cdot f_m$ with local step size η_m
8 **return** *final potential* $\Psi = \Psi_0 + \Delta_1 + \ldots + \Delta_m$

4. Experimental Evaluation

Our intention is to investigate how well NPPG works. To this aim, we implemented it and investigated the following questions:

(Q1) Does TreeNPPG work and, if so, are there cases where it yields better results than current state-of-the-art methods? **(Q2)** Is TreeNPPG applicable across finite, continuous, and relational domains?

In the following, we will describe the experiments carried out to investigate the questions and their results.

(Q1) Blocks World: A Relational Domain

As a complex domain for empirical evaluation of TreeNPPG , we consider the well-known *blocks world* (Slaney & Thiébaux, 2001). To be able to compare the performance of TreeNPPG to other relational RL (RRL) systems, we adopt the same experimental environment as used for RRL by e.g. Driessens and Džeroski (2005). We learn in a world with 10 blocks and try to accomplish the $on(A, B)$ goal. This goal is parameterized and appropriate values for A and B are chosen at same the time as a starting state is generated. Thus, the reinforcement learn learns a single policy that stacks any two blocks. Although this is a relatively simple goal in a planning context, both RRL and our TreeNPPG algorithm use a model free approach and only learn from interactions with the environment. For the given setup this means that there are approximately 55.7 million reachable states[3] of which 1.44 million are goal states. The minimal number of steps to the goal is 3.9 on average. The percentage of states for other minimal solution sizes are given in Fig. 2. The agent only receives a reward of 1 if it reaches the goal in the minimal number of steps. The probability of receiving a reward using a random strategy is approximately 1.3%. To counter this difficulty of reaching any reward, we adopted the active guidance approach as proposed by Driessens and Dzeroski (2004), presenting the RL agent with 10% of expert traces during exploration in all experiments.

We apply TreeNPPG to this relational domain by simply employing the relational regression tree learner Tilde (Blockeel & De Raedt, 1998). Rather than using attribute-value or threshold tests in node of the tree, Tilde employs logical queries. Furthermore, a placeholder for domain elements (such as blocks) can occur in different nodes meaning that all occurrences denote the same domain element. Indeed, this slightly complicates the induction process, for example when generating the possible tests to be placed in a node. To this aim, it employs a classical *refinement operator* under θ-subsumption. The operator basically adds a literal, unifies variables, and grounds variables. When a node is to be splitted, the set of all refinements are computed and evaluated according to the chosen heuristic. Except for the representational differences, Tilde uses the same approach to tree-building as the generic tree learner. Because single episodes are too short and thus generate too few examples for Tilde to learn meaningful trees, we postpone calling Tilde until an episode brings the cumulated number of examples over 100. As a language bias for Tilde we employ the same language bias as used in published RRL experiments, including predicates such as *clear*, *on*, *above* and *compheight*[4]. Each learned model Δ_m, updates the potential function Ψ using a step-size $\eta_m = 1$. We count on the self-correcting property of tree boosting to correct over- or under-stepping the target on the next iteration.

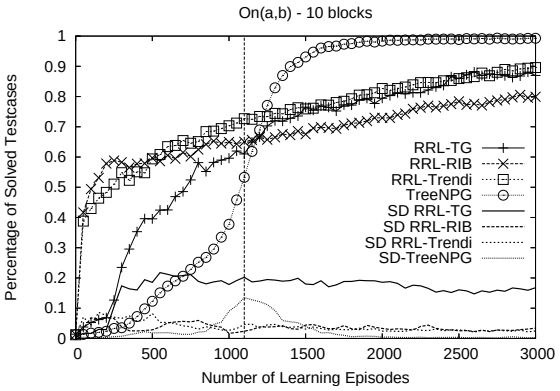


Figure 1. Comparison of the learning curves for Non-Parametric Policy Gradient and various RRL implementations on the on(A,B) task in a world with 10 blocks.

We ran experiments using three versions of the RRL system, i.e, TG (Driessens et al., 2001), RIB (Driessens & Ramon, 2003), and Trendi (Driessens & Džeroski, 2005), which represent the current state-of-the-art of RRL systems, and our TreeNPPG algorithm. After every 50 learning episodes, we fixed the strategy learned by RRL and tested the performance on 1000 randomly generated starting states. For TreeNPPG fixing the strategy is not required as it uses the learned strategy for exploration directly. Fig. 1 shows the resulting learning curves averaged over 10 test-runs as well as the standard deviations of these curves. As shown, TreeNPPG outperforms all tested versions of the RRL system. After approximately 2000 learning episodes, TreeNPPG solves 99% of all presented testcases in the minimal number of steps. With less than 4 steps per learning episode on average, this means that TreeNPPG generalizes the knowledge it collected by visiting less than 8000 states to solve 99% of 54.3 million states. Around this time, TreeNPPG has generated a list of 500 trees on average. One observation that can not be made directly on the shown graph is the stability of the TreeNPPG algorithm. Where the performance of the RRL system using TG

[3]We do not consider states in which the goal has already been satisfied as starting states. Therefore, a number of states where the goal is satisfied are not reachable, i.e., the states where $on(A, B)$ is satisfied and there are extra blocks on top of A.

[4]For a full overview and the semantics of these predicates, we refer to (Driessens & Džeroski, 2005).

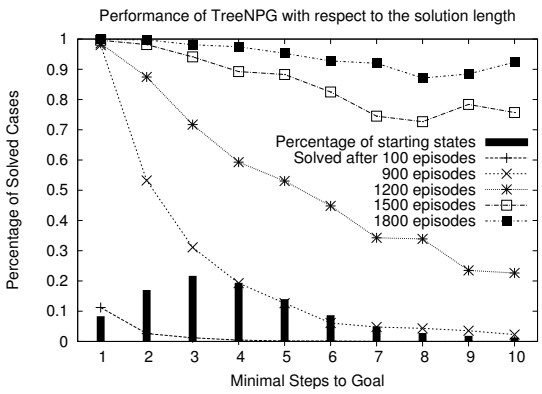


Figure 2. Detailed view of the learning performance of TreeNPPG showing the percentage of solved test-cases with respect to the minimal solution size for varying amounts of learning experience. The graph also shows the percentage of test-cases for each solution length.

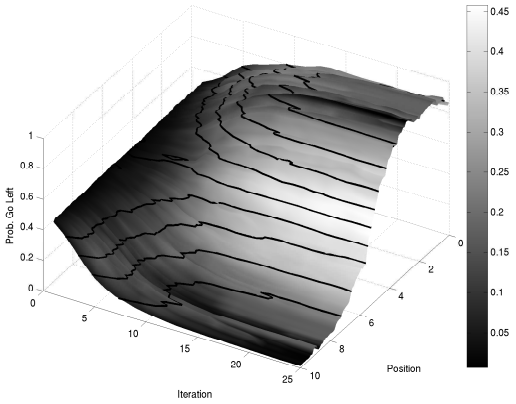


Figure 3. Learning performance of TreeNPPG on the continuous *corridor* task. Shown is how the probability of going *left* at all corridor positions evolves with the number of iterations. The shading indicates the standard deviation over the 30 runs of the experiment.

for regression can vary substantially between experiments, TreeNPPG follows an extremely similar path in each iteration of the experiment. The only variation between experiments is the exact time-frame of the phase transition in the results, as shown by the peak in TreeNPPG's standard deviation curve. Fig. 2 shows the results in more detail, plotting the percentage of solved test-cases with respect to the minimal solution length. As one can see, TreeNPPG gradually learns to solve problems with growing solution sizes. The graph also shows the percentage of test-cases for each solution size.

To summarize, the results clearly show that question **Q1** can be answered affirmatively.

(Q2) Corridor World: A Continuous Domain

To qualitatively test whether TreeNPPG is also applicable in continuous domains, we considered a simple *continuous corridor* domain. The task is to navigate a robot from any position $pos_0 \in [0, 10]$ in a one-dimensional corridor $[0, 10]$ to one of the exists at both ends (0 and 10). At each time $t = 0, 1, 2, \ldots$, the robot can go either one step to the *left* ($a = -1$) or one step to the *right* ($a = 1$); the outcome, however, is uncertain with $pos_{t+1} = pos_t + a + \mathcal{N}(1, 1)$. The robot is given a reward after each step equal to -1 if $pos \in (0, 10)$ and equal to 20 otherwise, i.e., it reaches one of the exists.

Fig. 3 shows how the stochastic policy evolves with the number of iterations, i.e., calls to the tree learner. These results are averaged over 30 reruns. In each run, we selected a random starting position pos_0 uniformly in $(0, 10)$, gathered learning examples from 30 episodes in each iteration, and used a step size $\eta_m = 0.7$. As one can see, the robot gradually learns to go *left* in the left section of the corridor and *right* in the right section. Quite intuitively, the uncertainty is highest in the middle part of the corridor.

We also ran experiments in a grid-world version of this problem. We do not report on the successful results here because finite domains are a special case of the relational and continuous cases. To summarize, question **Q2** can also be answered affirmatively.

5. Related Work

Within reinforcement learning (RL), there are two main classes of solution methods: value-function methods seek to estimate the value of states and actions whereas policy-based methods search for a good policy within some class of policies.

Within policy-based methods, policy gradients have received increased attention, see e.g. (Williams, 1992; Baxter et al., 2001; Guestrin et al., 2002; Konda & Tsitsiklis, 2003; Munos, 2006) as a non-exhausting list. Most closely related to NPPG is the work of Bagnell and Schneider (2003), see also (Bagnell, 2004). They proposed a functional gradient policy algorithms in reproducing kernel hilbert spaces for continuous domains. So far, however, this line of work has not considered (relationally) structured domains and the connection to gradient boosting was not employed.

Within value function approaches, the situation is slightly different. NPPG can be viewed as automatically generating a "variable" propositionalization or discretization of the domain at hand. In this sense, it is akin to tree-based state discretization RL ap-

proaches such as (Chapman & Kaelbling, 1991; McCallum, 1996; Uther & Veloso, 1998) and related approaches. Within this line of research, there have been some boosting methods proposed. Ernst et al. (2005) showed how to estimate Q functions with ensemble methods based on regression trees. Riedmiller (2005) keeps all regression examples and re-weights them according to some heuristic. Both neither consider policy gradients nor relational domains.

Recently, there have been some exciting new developments in combining the rich relational representations of classical knowledge representation with RL. While traditional RL requires (in principle) explicit state and action enumeration, these symbolic approaches seek to avoid explicit state and action enumeration through a symbolic representation of states and actions. Most work in this context, however, has focused on value function approaches. Basically, a number of relational regression algorithms have been developed for use in this RL system that employ relational regression trees (Driessens et al., 2001), relational instance based regression (Driessens & Ramon, 2003), graph kernels and Gaussian processes (Gärtner et al., 2003) and relational model-trees (Driessens & Džeroski, 2005). Finally, there is an increasing number of dynamic programming approaches for solving relational MDPs (Kersting et al., 2004; Sanner & Boutilier, 2005; Wang et al., 2007). In contrast to NPPG, they assume a model of the domain. It would be interesting, however, to combine NPPG with these approaches along the line of (Wang & Dietterich, 2003). A parametric step into this direction has been already taken by Aberdeen (2006).

6. Conclusions

We have introduced the framework of non-parametric policy gradient (NPPG) methods. It seeks to leverage the policy selection problem by approaching it from a gradient boosting perspective. NPPG is fast and straightforward to implement, combines the expressive power of relational RL with the benefits of policy gradient methods, and can deal with finite, continuous, and relational domains in a unified way. Moreover, the experimental results show a significant improvement over established results; for the first time, a (model-free) relational RL approach learns to solve $on(A, B)$ in a world with 10 blocks.

NPPG suggests several interesting directions for future research such as using more advanced regression models, developing actor-critic versions of NPPG estimating a value function in parallel to reduce the variance of the gradient estimates, and exploiting NPPG's ability to learn in hybrid domains with both discrete and continuous variables within real-world domains such as as robotics and network routing. Most interesting, however, is to address the more general problem of learning how to interact with (relationally) structured environments in the presence observation noise. NPPG is naturally applicable in this case and, hence, paves the way towards (model-free) solutions of what can be called relational POMDPs. This is a topic of high current interest since it combines the expressive representations of classical AI with the decision-theoretic emphasis of modern AI.

Acknowledgments The authors would like to thank the anonymous reviewers for their valuable comments. The material is based upon work performed while KK was with CSAIL@MIT and is supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010, and by a Fraunhofer ATTRACT fellowship. KD is a post-doctoral research fellow of the Research Fund - Flanders (FWO).

References

Aberdeen, D. (2006). Policy-gradient methods for planning. *Advances in Neural Information Processing Systems 18* (pp. 9–17).

Bagnell, J. (2004). *Learning decisions: Robustness, uncertainty, and approximation*. Doctoral dissertation, Robotics Institute, Carnegie Mellon University, Pittsburg, Pa, USA.

Bagnell, J., & Schneider, J. (2003). *Policy search in reproducing kernel hilbert space* (Technical Report CMU-RI-TR-03-45). Robotics Institute, Carnegie Mellon University, Pittsburg, Pa, USA.

Baxter, J., Bartlett, P., & Weaver, L. (2001). Experiments with infinite-horizon, policy-gradient estimation. *Journal of Arificial Intellifence Research (JAIR)*, *15*, 351–381.

Bertsekas, D. P., & Tsitsiklis, J. (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.

Blockeel, H., & De Raedt, L. (1998). Top-down induction of first order logical decision trees. *Artificial Intelligence*, *101*, 285–297.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Belmont: Wadsworth.

Chapman, D., & Kaelbling, L. P. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisions. *Proceedings*

of the 12th International Joint Conference on Artificial Intelligence (pp. 726–731) Sydney, Australia.

Dietterich, T., Ashenfelter, A., & Bulatov, Y. (2004). Training conditional random fields via gradient tree boosting. *Proceedings of the 21st International Conference on Machine Learning* (pp. 217–224). Banff, Canada.

Driessens, K., & Džeroski, S. (2005). Combining model-based and instance-based learning for first order regression. *Proceedings of the 22nd International Conference on Machine Learning* (pp. 193–200) Bonn, Germany.

Driessens, K., & Dzeroski, S. (2004). Integrating guidance into relational reinforcement learning. *Machine Learning*, *57*, 271–304.

Driessens, K., & Ramon, J. (2003). Relational instance based regression for relational reinforcement learning. *Proceedings of the 20th International Conference on Machine Learning* (pp. 123–130) Washington, DC, USA.

Driessens, K., Ramon, J., & Blockeel, H. (2001). Speeding up relational reinforcement learning through the use of an incremental first order decision tree learner. *Proceedings of the 12th European Conference on Machine Learning* (pp. 97–108), Freiburg, Germany.

Džeroski, S., De Raedt, L., & Driessens, K. (2001). Relational reinforcement learning. *Machine Learning*, *43*, 7–52.

Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research (JMLR)*, *6*, 503–556.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.

Gärtner, T., Driessens, K., & Ramon, J. (2003). Graph kernels and gaussian processes for relational reinforcement learning. *International Conference on Inductive Logic Programming* (pp. 146–163) Szeged, Hungary.

Getoor, L., & Taskar, B. (2007). *An introduction to statistical relational learning*. MIT Press.

Guestrin, C., Lagoudakis, M., & Parr, R. (2002). Coordinated reinforcement learning. *Proceedings of the 19th International Conference on Machine Learning* (pp. 227–234) Sydney, Australia.

Gutmann, B., & Kersting, K. (2006). TildeCRF: Conditional random fields for logical sequences. *Proceedings of the 17th European Conference on Machine Learning* (pp. 174–185). Berlin, Germany.

Kersting, K., van Otterlo, M., & De Raedt, L. (2004). Bellman goes relational. *Proceedings of the Twenty-First International Conference on Machine Learning (ICML-2004)* (pp. 465–472). Banff, Canada.

Konda, V. R., & Tsitsiklis, J. N. (2003). On actor-critic algorithms. *SIAM J. Control Optim.*, *42*, 1143–1166.

McCallum, A. (1996). *Reinforcement learning with selective perception and hidden state*. Doctoral dissertation, University of Rochester, New York, NY, USA.

Munos, R. (2006). Policy gradient in continuous time. *Journal of Machine Learning Research (JMLR)*, *7*, 771–791.

Riedmiller, M. (2005). Neural fitted Q iteration - First experiences with a data efficient neural reinforcement learning method. *Proceedings of the 16th European Conference on Machine Learning* (pp. 317–328) Porto, Portugal.

Sanner, S., & Boutilier, C. (2005). Approximate linear programming for first-order MDPs. *Proceedings of the 21st conference on Uncertainty in AI (UAI)* (pp. 509-517) Edinburgh, Scotland.

Slaney, J., & Thiébaux, S. (2001). Blocks world revisited. *Artificial Intelligence*, *125*, 119–153.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: The MIT Press.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems 12* (pp. 1057–1063). MIT Press.

Uther, W., & Veloso, M. (1998). Tree based discretization for continuous state space reinforcement learning. *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-96)* (pp. 769–774) Portland, OR, USA.

Wang, C., Joshi, S., & Khardon, R. (2007). First order decision diagrams for relational mdps. *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (pp. 1095–1100). Hyderabad, India: AAAI press.

Wang, X., & Dietterich, T. (2003). Model-based policy gradient reinforcement learning. *Proceedings of the 20th International Conference on Machine Learning* (pp. 776-783) Washington, DC, USA.

Williams, R. (1992). Simple statistical gradient following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

ICA and ISA Using Schweizer-Wolff Measure of Dependence

Sergey Kirshner SERGEY@CS.UALBERTA.CA
Barnabás Póczos POCZOS@CS.UALBERTA.CA
AICML, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2E8

Abstract

We propose a new algorithm for independent component and independent subspace analysis problems. This algorithm uses a contrast based on the Schweizer-Wolff measure of pairwise dependence (Schweizer & Wolff, 1981), a non-parametric measure computed on pairwise ranks of the variables. Our algorithm frequently outperforms state of the art ICA methods in the normal setting, is significantly more robust to outliers in the mixed signals, and performs well even in the presence of noise. Our method can also be used to solve independent subspace analysis (ISA) problems by grouping signals recovered by ICA methods. We provide an extensive empirical evaluation using simulated, sound, and image data.

1. Introduction

Independent component analysis (ICA) (Comon, 1994) deals with a problem of a blind source separation under the assumptions that the sources are independent and that they are linearly mixed. ICA has been used in the context of blind source separation and deconvolution, feature extraction, denoising, and successfully applied to many domains including finances, neurobiology, and processing of fMRI, EEG, and MEG data. For a review on ICA, see Hyvärinen et al. (2001).

Independent subspace analysis (ISA) (also called multi-dimensional ICA and group ICA) is a generalization of ICA that assumes that certain sources depend on each other, but the dependent groups of sources are still independent of each other, i.e., the independent groups are multidimensional. The ISA task has been the subject of extensive research (e.g., Cardoso, 1998; Theis, 2005; Bach & Jordan, 2003; Hyvärinen & Köster, 2006; Póczos & Lőrincz, 2005) and applied, for instance, to EEG-fMRI data.

Our contribution, SWICA, is a new ICA algorithm based on Schweizer-Wolff (SW) non-parametric dependence measure. SWICA has the following properties:

- SWICA performs comparably to other state of the art ICA methods, outperforming them in a large number of test cases.
- SWICA is extremely robust to outliers as it uses *rank* values of the signals rather than their actual values.
- SWICA suffers less from the presence of noise than other algorithms.
- SW measure can be used as the cost function to solve ISA problems by grouping sources recovered by ICA methods.
- SWICA is simple to implement, and the Matlab/C++ code is available for public use.
- On a negative side, SWICA is slower than other methods, limiting its use to sources of moderate dimensions, and it requires more samples to demix sources with near-Gaussian distributions.

The paper is organized as follows. An overview of the ICA and ISA problems and methods is presented in Section 2. Section 3 motivates and describes Schweizer-Wolf dependence measure. Section 4 describes a 2-source version of SWICA, extends it to a d-source problem, describes an application to ISA, and mentions possible approaches for accelerating SWICA. Section 5 provides a thorough empirical evaluation of SWICA to other ICA algorithms under different settings and data types. The paper is concluded with a summary in Section 6.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. ICA and ISA

We consider the following problem. Assume we have d independent 1-dimensional sources (random variables) denoted by $S^1, \ldots, S^d$. We assume each source emits N i.i.d. samples denoted by $\left(s_1^i, \ldots, s_N^i\right)$. Let $\mathbf{S} = \left\{s_i^j\right\} \in \mathbb{R}^{d \times N}$ be a matrix of these samples. We assume that these sources are hidden, and that only a matrix $\mathbf{X}$ of mixed samples can be observed:

$$\mathbf{X} = \mathbf{AS}$$

where $\mathbf{A} \in \mathbb{R}^{d \times d}$. (We further assume that $\mathbf{A}$ has full rank d.) The task is to recover the sample matrix $\mathbf{S}$ of the hidden sources by finding a demixing matrix $\mathbf{W}$

$$\mathbf{Y} \quad = \quad \mathbf{WX} = (\mathbf{WA})\,\mathbf{S},$$

and the estimated sources $Y^1, \ldots, Y^d$ are mutually independent. The solution can be recovered only up to a scale and a permutation of the components; thus we assume that the data has been pre-whitened, and it is sufficient to search for an *orthogonal* matrix $\mathbf{W}$ (e.g., Hyvärinen et al., 2001). Additionally, since jointly Gaussian sources are not identifiable under linear transformations, we assume that no more than one source is normally distributed.

There are many approaches to solving the ICA problem, differing both in the objective function designed to measure the independence between the unmixed sources (sometimes referred to as a contrast function) and the optimization methods for that function. Most commonly used objective function is the mutual information (MI)

$$J(\mathbf{W}) = I\left(Y^1, \ldots, Y^d\right) = \sum_{i=1}^{d} h\left(Y^i\right) - h\left(Y^1, \ldots, Y^d\right) \tag{1}$$

where h is the differential entropy. Alternatively, one can minimize the sum $\sum_{i=1}^{d} h\left(Y^i\right)$ of the univariate entropies as the joint entropy is constant (e.g., Hyvärinen et al., 2001). Neither of these quantities can be evaluated directly, so approximations are used instead. Among effective methods falling in the former category is KernelICA (Bach & Jordan, 2002); RADICAL (Learned-Miller & Fisher, 2003) and FastICA (Hyvärinen, 1999) approximate the sum of the univariate entropies. There are other possible cost functions including maximum likelihood, moment-based methods, and correlation-based methods.

While ICA problems has been well-studied in the above formulation, there are a number of variations of it that are subject of active research. One such formulation is a noisy version of ICA

$$\mathbf{X} = \mathbf{AS} + \boldsymbol{\epsilon} \tag{2}$$

where multivariate noise $\boldsymbol{\epsilon}$ is often assumed normally distributed. Another related problem occurs when the mixed samples $\mathbf{X}$ are corrupted by a presence of outliers. There are many other possibilities that go beyond the scope of this paper.

Of a special note is a generalization of ICA where some of the sources are *dependent*, independent subspace analysis (ISA). For this case, the mutual information and Shannon entropies from Equation 1 would involve multivariate random vectors instead of scalars. Resulting multidimensional entropies are exponentially more difficult to estimate than their scalar counterparts, making ISA problem more difficult than ICA. However, Cardoso (1998) conjectured that the ISA problem can be solved by first preprocessing the mixtures $\mathbf{X}$ by an ICA algorithm and then grouping the estimated components with highest dependence. While the extent of this conjecture is still on open issue, it has been rigorously proven for some distribution types (Szabó et al., 2007). Even without a proof for the general case, a number of algorithms apply this heuristics with success (Cardoso, 1998; Theis, 2007; Bach & Jordan, 2003). There are ISA methods not relying on Cardoso's conjecture (e.g., Hyvärinen & Köster, 2006) although they are susceptible to getting trapped in local minima.

3. Non-parametric Rank-Based Approach

Most of the ICA algorithms use an approximation to mutual information (MI) as their objective functions, and the quality of the solution thus depends on how accurate is the corresponding approximation. The problem with using MI is that without a parametric assumption on the functional form of the joint distribution, MI cannot be evaluated exactly, and numerical estimation can be both inaccurate and computationally expensive. In this section, we explore other measures of pairwise association as possible ICA contrasts. To note, most commonly used measure of correlation, Pearson's linear correlation coefficient, cannot be used as it is invariant under rotations (once the data has been centered and whitened)

Instead, we are focusing on measures of dependence of the *ranks*. Ranks have a number of desirable properties – they are invariant under monotonic transformations of the individual variables, insensitive to outliers,

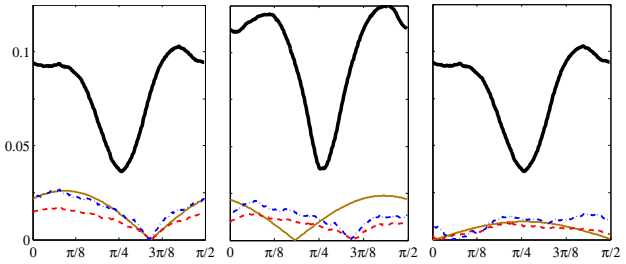


Figure 1. Absolute values of sample versions for Pearson's ρ_p (solid thin, brown), Kendall's τ (dashed, red), Spearman's ρ (dash-dotted, blue), and Schweizer-Wolff σ (solid thick, black) as a function of rotation angle $\left[0, \frac{\pi}{2}\right]$. Data was obtained by rotating by $\frac{\pi}{4}$ 1000 samples from a uniform distribution on $\mathbf{I}^2$ (left), with added outliers (center), and with added noise (right).

and not very sensitive to small amounts of noise. We found that a dependence measure defined on copulas (e.g., Nelsen, 2006), probability distributions on continuous ranks, has the right properties to be used as a contrast for ICA demixing.

3.1. Ranks and Copulas

Let a pair of random variables $(X, Y) \in \mathbb{R}^2$ be distributed according to a bivariate probability distribution P. Assume we are given N samples of (X, Y), $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$. Let the rank $r_x(x)$ be the number of x_i, $i = 1, \ldots, N$ such that $x > x_i$, and let $r_y(y)$ be defined similarly.

Many non-linear dependence measures are based on ranks. Among most commonly used are Kendall's τ and Spearman's ρ rank correlation coefficients. Kendall's τ measures the difference between proportions of concordant pairs ((x_i, y_i) and (x_j, y_j) such that $(x_i - x_j)(y_i - y_j) > 0$) and discordant pairs. Spearman's ρ measures a linear correlation between ranks of $r_x(x)$ and $r_y(y)$. Both τ and ρ have a range of $[-1, 1]$ and are equal to 0 (in the limit) if the X and Y are independent. However, the converse is not true, and both τ and ρ can be 0 even if X and Y are not independent. While they are robust to outliers, neither ρ nor τ make for a good ICA contrast as they provide a noisy estimate for dependence from moderately-sized data sets when the dependence is weak (See Figure 1 for an illustration).

Rank correlations can be extended from samples to distributions with the help of *copulas*, distributions over continuous multivariate ranks. We will devise an effective robust contrast for ICA using a measure of dependence for copulas which is closely related to Spearman's ρ.

Let $\mathbf{I}$ denote a unit interval $[0, 1]$. A bivariate *copula* C is probability function (cdf) defined on a unit square, $C : \mathbf{I}^2 \to \mathbf{I}$ such that its univariate marginals are uniform, i.e., $C(u, 1) = u$, $C(1, v) = v$, $\forall u, v, \in \mathbf{I}$.[1] Let $U = P_x(X)$ and $V = P_y(Y)$ denote the corresponding cdfs for previously defined random variables X and Y. Variables $X = P_x^{-1}(U)$ and $Y = P_y^{-1}(V)$ can be defined in terms of the inverse of marginal cdfs. Then, for $(u, v) \in \mathbf{I}^2$, define C as

$$C(u, v) = P\left(P_x^{-1}(u), P_y^{-1}(v)\right).$$

It is easy to verify that C is a copula. Sklar's theorem (Sklar, 1959) states that such copula exists for any distribution P, and that it is unique on the range of values of the marginal distributions. A copula can be thought of as binding univariate marginals P_x and P_y to make a distribution P.

Copulas can also be viewed as a canonical form of multivariate distributions as they preserve multivariate dependence properties of the corresponding families of distributions. For example, the mutual information of the joint distribution is equal to the negentropy of its copula restricted to the region on which the copula *density* function (denoted in this paper by $c(u, v)$) is defined:

$$\begin{aligned} c(u, v) &= \frac{\partial^2 C(u, v)}{\partial u \partial v} = \frac{p(x, y)}{p_x(x)\, p_y(y)}; \\ I(X, Y) &= \int_{\mathbf{I}^2} c(u, v) \ln c(u, v)\, du dv. \end{aligned}$$

Such negentropy is minimized when $C(u, v) = \Pi(u, v) = uv$. Copula Π is referred to as the *product* copula and is equivalent to variables U and V (and the original variables X and Y) being mutually independent. This copula will play a central part in definition of contrasts in the next subsection.

Copulas can also be viewed as a joint distribution over univariate *ranks*, and therefore, preserve all of the rank statistics of the corresponding multivariate distributions; rank based statistics can be expressed in terms of the copula alone. For example, Spearman's ρ has a convenient functional form in terms of the corresponding copulas (e.g., Nelsen, 2006):

$$\rho = 12 \int_{\mathbf{I}^2} (C(u, v) - \Pi(u, v))\, du dv. \qquad (3)$$

[1] While we restrict our attention to bivariate copulas, many of the definitions and properties described in this section can be extended to a d-variate case.

As the true distribution P and its copula C are not known, the rank statistics can be estimated from the available samples using an *empirical copula* (Deheuvels, 1979). For a data set $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, an empirical copula C_N is given by

$$C_N\left(\frac{i}{N}, \frac{j}{N}\right) = \frac{\#\text{ of } (x_k, y_k) \text{ s.t. } x_k \leq x_i \text{ and } y_k \leq y_j}{N}. \quad (4)$$

Well-known sample versions of several non-linear dependence measures can be obtained using an empirical copula (e.g., Nelsen, 2006). For example, sample version r of Spearman's ρ appears to be a grid integration evaluation of its expression in terms of a copula (Equation 3):

$$r = \frac{12}{N^2 - 1} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(C_N\left(\frac{i}{N}, \frac{j}{N}\right) - \frac{i}{N} \times \frac{j}{N} \right). \quad (5)$$

3.2. Schweizer-Wolff σ and κ

Part of the problem with Kendall's τ and Spearman's ρ as a contrast for ICA is a property that their value may be 0 even though the corresponding variables X and Y are not independent. Instead, we suggest using Schweizer-Wolff σ, a measure of dependence between two continuous random variables (Schweizer & Wolff, 1981):

$$\sigma = 12 \int_{\mathbf{I}^2} |C(u, v) - uv| \, \mathrm{d}u\mathrm{d}v. \quad (6)$$

σ can be viewed as an L_1 norm between a copula for the distribution and a product copula. It has a range of $[0, 1]$, with an important property that $\sigma = 0$ if and only if the corresponding variables are mutually independent, i.e., $C = \Pi$. The latter property suggests an ICA algorithm for a pair of variables: pick a rotation angle such that the corresponding demixed data set has its σ minimized. A sample version of σ is similar to that of ρ (Equation 5):

$$s = \frac{12}{N^2 - 1} \sum_{i=1}^{N} \sum_{j=1}^{N} \left| C_N\left(\frac{i}{N}, \frac{j}{N}\right) - \frac{i}{N} \times \frac{j}{N} \right|. \quad (7)$$

We note that other measures of dependence can be potentially used as an ICA contrast. We also experimented with an L_∞ version of σ, $\kappa = 4 \sup_{\mathbf{I}^2} |C(u, v) - uv|$, a dependence measure similar to Kolmorogov-Smirnov univariate statistic (Schweizer & Wolff, 1981), with results similar to σ.

4. SWICA: A New Algorithm for ICA and ISA

In this section, we present a new algorithm for ICA and ISA demixing. The algorithm uses Schweizer-Wolff σ estimates as a contrast in demixing pairs of variables; we named this algorithm Schweizer-Wolff contrast for ICA, or SWICA for short.

4.1. 2-dimensional Case

First, we tackle the case of a two-dimensional signal $\mathbf{S}$ mixed with a 2×2 matrix $\mathbf{A}$. We, further assume $\mathbf{A}$ is orthogonal (otherwise achievable by whitening). The problem is then reduced to finding a demixing rotation matrix $\mathbf{W} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$.

For the objective function, we use s (Equation 7) computed on $2 \times N$ matrix $\mathbf{Y} = \mathbf{WX}$ of rotated samples. Given an angle θ, $s(\boldsymbol{Y}(\theta))$ can be computed by first sorting each of the rows of $\mathbf{Y}(\theta)$ and computing row ranks for each entry of $\mathbf{Y}(\theta)$, then computing an empirical copula C_N (Equation 4) for ranks of $\mathbf{Y}$, and finally computing $s(\mathbf{Y}(\theta))$ (Equation 7). The solution is then found by finding angle θ minimizing $s(\mathbf{Y}(\theta))$. Similar to RADICAL (Learned-Miller & Fisher, 2003), we find such solution by searching over K values of θ in the interval $\left[0, \frac{\pi}{2}\right)$. This algorithm is outlined in Figure 2.

4.2. d-dimensional Case

A d-dimensional linear transformation described by a $d \times d$ orthogonal matrix $\mathbf{W}$ is equivalent to a composition of 2-dimensional rotations (called *Jacobi* or *Givens* rotations) (e.g., Comon, 1994). The transformation matrix itself can be written as a product of corresponding rotation matrices, $\mathbf{W} = \mathbf{W}_L \times \ldots \times \mathbf{W}_1$ where each matrix $\mathbf{W}_l$, $l = 1, \ldots, L$ is a rotation matrix (by angle θ_l) for some pair of dimensions (i, j). Thus a d-dimensional ICA problem can be solved by solving 2-dimensional ICA problems in succession. Given a current demixing matrix $\mathbf{W}_c = \mathbf{W}_l \times \ldots \times \mathbf{W}_1$ and a current version of the signal $\mathbf{X}_c = \mathbf{W}_c\mathbf{X}$, we find an angle θ corresponding to $\mathrm{SWICA}\left(\mathbf{X}_c^{(i,j)}, K\right)$. Taking an approach similar to RADICAL, we perform a fixed number of successive sweeps through all possible pairs of dimensions (i, j).

We should note that while d-dimensional SWICA is not guaranteed to converge, it converges in practice a vast majority of the time. A likely explanation is that each 2-dimensional optimization finds a transfor-

Algorithm SWICA($\mathbf{X}, K$)
Inputs: $\mathbf{X}$, a $2 \times N$ matrix where rows are mixed signals (centered and whitened), K equispaced evaluation angles in the $[0, \pi/2)$ interval

For each of K angles θ in the interval $[0, \pi/2)$ ($\theta = \frac{\pi k}{2K}, k = 0, \ldots, K-1$.)

- Compute rotation matrix

$$\mathbf{W}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

- Compute rotated signals $\mathbf{Y}(\theta) = \mathbf{W}(\theta)\mathbf{X}$.
- Compute $s(\mathbf{Y}(\theta))$, a sample estimate of σ (Equation 7)

Find best angle $\theta_m = \arg\min_\theta s(\mathbf{Y}(\theta))$

Output: Rotation matrix $\mathbf{W} = \mathbf{W}(\theta_m)$, demixed signal $\mathbf{Y} = \mathbf{Y}(\theta_m)$, and estimated dependence measure $s = s(\boldsymbol{Y}(\theta_m))$

Figure 2. Outline of SWICA algorithm (2-d case).

mation that reduces the sum of entropies for the corresponding dimensions, reducing the overall sum of entropies. In addition to this, Learned-Miller and Fisher (2003) suggest that the minimization of the overall sum of entropies in this fashion (by changing only two terms in the sum) may make it easier to escape local minima.

4.3. Complexity Analysis and Acceleration Tricks

2-dimensional SWICA requires a search over K angles. For each angle, we first sort the data to compute the ranks of each data point ($\mathcal{O}(N \log N)$), and then use these ranks to compute s by computing the empirical copula and summing over the $N \times N$ grid (Equation 7), requiring $\mathcal{O}(N^2)$ additions. Therefore, running time complexity of 2-d SWICA is $\mathcal{O}(KN^2)$. Each sweep of a d-dimensional ICA problem solves a 2-dimensional ICA problem for each pair of variables, $\mathcal{O}(d^2)$ of them; S sweeps would have $\mathcal{O}(Sd^2KN^2)$ complexity. In our experiments, we employed $K = 180, S = 1$ for $d = 2$, and $K = 90, S = d$ for $d > 2$.

The most expensive computation in SWICA is $\mathcal{O}(N^2)$ needed to compute $s(\mathbf{Y}(\theta))$. Reducing this complexity, either by approximation, or perhaps, by an efficient rearrangement of the sum, is left to future research. We used several other tricks to speed up the computation. One, for large N ($N > 2500$) we estimated s using only N_s^2 ($N_s = \lfloor \frac{N}{\lceil \frac{N}{2500} \rceil} \rfloor$) terms in the sum corresponding to equispaced gridpoints on $\mathbf{I}^2$. Two, when searching for θ minimizing $s(\mathbf{Y}(\theta))$, it is unnecessary to sum over all N^2 terms when evaluating a candidate θ if a partial sum already results in a value of $s(\mathbf{Y}(\theta))$ larger than the current best. This optimization translates into a 2-fold speed increase in practice. Three, it is unnecessary to complete all S sweeps if the algorithm already converged. One possible measure of convergence is the Amari error (Equation 8) measured for the cumulative rotation matrix for the most recent sweep.

4.4. Using Schweizer-Wolff σ for ISA

Following Cardoso's conjecture, ISA problems can be solved by first finding a solution to an ICA problem, and then by grouping resulting sources that are *not* independent (Cardoso, 1998). We propose employing Schweizer-Wolff σ to measure dependence of sources for an ICA solution as it provides a computationally effective alternative to mutual information, commonly used measure of source dependence. Note that ICA solution, the first step, can be obtained using any approach, e.g., FastICA due to its computational speed for large d. One commonly used trick for grouping the variables is to use a non-linear transformation of the variables to "amplify" their dependence as independent variables remain independent under such transformations.[2]

5. Experiments

For the experimental evaluation of SWICA, we considered several settings. For the evaluation of the quality of demixing solution matrix $\mathbf{W}$, we computed the Amari error (Amari et al., 1996) for the resulting transformation matrix $\mathbf{B} = \mathbf{WA}$. Amari error $r(\mathbf{B})$ measures how different matrix $\mathbf{B}$ is from a permutation matrix, and is defined as

$$\alpha \sum_{i=1}^{d} \left(\frac{\sum_{j=1}^{d} |b_{ij}|}{\max_j |b_{ij}|} - 1 \right) + \alpha \sum_{j=1}^{d} \left(\frac{\sum_{i=1}^{d} |b_{ij}|}{\max_i |b_{ij}|} - 1 \right). \tag{8}$$

where $\alpha = 1/(2d(d-1))$. $r(\mathbf{B}) \in [0, 1]$, and $r(\mathbf{B}) = 0$ if and only if $\mathbf{B}$ is a permutation matrix. We compared SWICA to FastICA (Hyvärinen, 1999), KernelICA-KGV (Bach & Jordan, 2002), RADICAL (Learned-Miller & Fisher, 2003), and JADE (Cardoso, 1999).

[2]Such transformations are at the core of the KernelICA and JADE ICA algorithms.

For the simulated data experiments, we used 18 different one-dimensional densities to simulate sources. These test-bed densities (and some of the experiments below) were proposed by Bach and Jordan (2002) to test KernelICA and by Learned-Miller and Fisher (2003) to evaluate RADICAL; we omit the description of these densities due to lack of space as they can be looked up in the above papers.

Table 1 summarizes the medians of the Amari errors for 2-dimensional problems where both sources had the same distribution. Samples from these sources were then transformed by a random rotation, and then demixed using competing ICA algorithms. SWICA outperforms its competitors in 8 out of 18 cases, and performs comparably in several other cases. However, it performs poorly when the joint distribution for the sources is close to a Gaussian (e.g., (d) t-distribution with 5 degrees of freedom). One possible explanation for why SWICA performs worse than its competitors for these cases is that by using ranks instead of the actual values, SWICA is discarding some of the information that may be essential to separating such sources. However, given larger number of samples, SWICA is able to separate near-Gaussian sources (data not shown due to space constraints). SWICA also outperformed other methods when sources were not restricted to come from the same distribution (Table 2) and proved effective for multi-dimensional problems ($d = 4, 8, 16$).

Figure 3 summarizes the performance of ICA algorithms in the presence of outliers for the d-source case ($d = 2, 4, 8$). Distributions for the sources were chosen at random from the 18 distributions from the experiment in Table 1. The sources were mixed using a random rotation matrix. The mixed sources were then corrupted by adding $+5$ or -5 to a single component for a small number of samples. SWICA significantly outperforms the rest of the algorithms as the contrast used by SWICA is insensitive to minor changes in the sample ranks introduced by a small number of outliers. For $d = 2$, we tested SWICA further by significantly increasing the number of outliers; the performance was virtually unaffected when the proportion of the outliers was below 20%. SWICA is also less sensitive to noise than other ICA methods (Figure 4).

We further tested SWICA on sound and image data. We mixed $N = 1000$ samples from 8 sound pieces of an ICA benchmark[3] by a random orthogonal 8×8 matrix. Then we added 20 outliers to this mixture in the same way as in the previously described outlier experiment and demixed them using ICA algorithms. Figure 5 shows that SWICA outperforms other methods on this task. For the image experiment, we used 4 natural images[4] of size 128×256. The pixel intensities we normalized in the [0, 255] interval. Each image was considered as a realization of a stochastic variable with 32768 sample points. We mixed these 4 images by a 4×4 random orthogonal mixing matrix, resulting in a mixture matrix of size 4×32768. Then we added large $+2000$ or -2000 outliers to 3% randomly selected points of these mixture, and then selected at random 2000 samples from the 32768 vectors. We estimated the demixing matrix $\mathbf{W}$ using only these 2000 points,

Table 1. The Amari errors (multiplied by 100) for two-component ICA with 1000 samples. Each entry is the median of 100 replicates for each pdf, (a) to (r). The lowest (best) entry in each row is boldfaced.

pdf	SWICA	FastICA	RADICAL	KernelICA	JADE
a	3.74	3.01	2.18	**2.09**	2.67
b	2.39	4.87	**2.31**	2.50	3.47
c	**0.79**	1.91	1.60	1.54	1.63
d	10.10	5.63	4.10	5.05	**3.94**
e	**0.47**	4.75	1.43	1.21	3.27
f	**0.78**	2.85	1.39	1.34	2.77
g	**0.74**	1.49	1.19	1.11	1.19
h	3.66	5.32	4.01	3.54	**3.36**
i	10.21	7.38	6.95	7.70	**6.41**
j	**0.86**	4.64	1.29	1.21	3.38
k	**2.10**	5.58	2.65	2.38	3.53
l	4.09	7.68	**3.61**	3.65	5.21
m	**1.11**	3.41	1.43	1.23	2.58
n	2.08	4.05	2.10	**1.56**	4.07
o	5.07	3.81	2.86	2.92	**2.78**
p	**1.24**	2.92	1.81	1.53	2.70
q	3.01	12.84	2.30	**1.67**	10.78
r	3.32	4.30	3.06	**2.65**	3.32

Table 2. The Amari errors (multiplied by 100) for d-component ICA with N samples. Each entry is the median of 1000 replicates for $d = 2$ and 100 for $d = 4, 8, 16$. Source densities were chosen uniformly at random from (a)-(r). The lowest (best) entry in each row is boldfaced.

d	N	SWICA	FastICA	RADICAL	KernelICA	JADE
2	1000	**1.53**	4.31	2.13	1.97	3.47
4	2000	**1.31**	3.74	1.72	1.66	2.83
8	5000	**1.20**	2.58	1.31	1.25	2.25
16	10000	1.16	1.92	**0.93**	6.69	1.76

[3] http://www.cis.hut.fi/projects/ica/cocktail/cocktail_en.cgi [4] http://www.cis.hut.fi/projects/ica/data/images/

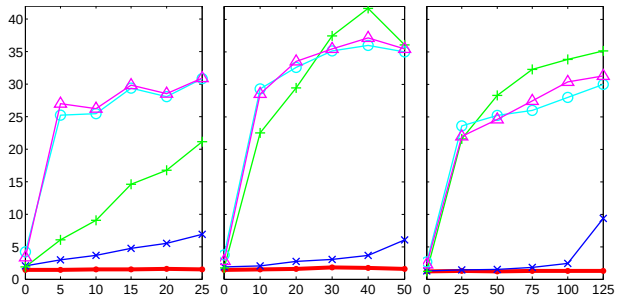


Figure 3. Amari errors (multiplied by 100) for 2-d (left), 4-d (center), and 8-dimensional (right) ICA problem in the presence of outliers. The plot shows the median values over $R = 1000, 100, 100$ replicas of $N = 1000, 2000, 5000$ samples for $d = 2, 4, 8$, respectively. Legend: Swica – red dots (thick), RADICAL – blue x's, KernelICA – green pluses, FastICA – cyan circles, JADE – magenta triangles. The x-axis shows the number of outliers.

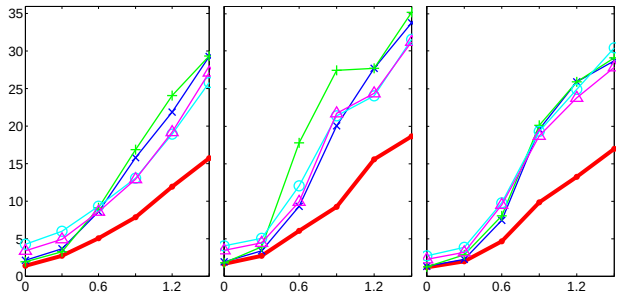


Figure 4. Amari errors (multiplied by 100) for 2-d (left), 4-d (center), and 8-dimensional (right) ICA problems in the presence of independent Gaussian noise applied to mixed sources. The plot shows the median values of $R = 1000, 100, 100$ replicas of $N = 1000, 2000, 5000$ samples for $d = 2, 4, 8$, respectively. The abscissa shows the variance of the Gaussian noise, $\sigma^2 = (0, 0.3, 0.6, 0.9, 1.2, 1.5)$. The legend is the same as in Figure 3.

and then recovered the hidden sources for all 32768 samples using this matrix. SWICA significantly outperformed other methods. Figure 7 shows an example of the demixing achieved by different ICA algorithms.

Finally, we applied Schweizer-Wolff σ in an ISA setting. We used 6 3-dimensional sources where each variable was sampled from a geometric shape (Figure 6a), resulting in 18 univariate hidden sources. These sources ($N = 1000$ samples) were then mixed with a random 18×18 orthogonal matrix (Figure 6b). Applying Cardoso's conjecture, we first processed the mixed sources using FastICA, and then clustered the recovered sources using σ computed on their absolute values (a non-linear transformation) (Figure 6c). The hidden subspaces were recovered with high precision as indicated by the Hinton diagram of **WA** (Figure 6d).

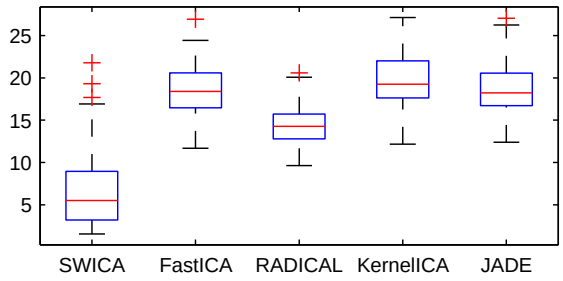


Figure 5. Box plot of Amari errors (multiplied by 100) for the mixed sounds with outliers. Plot was computed over $R = 100$ replicas.

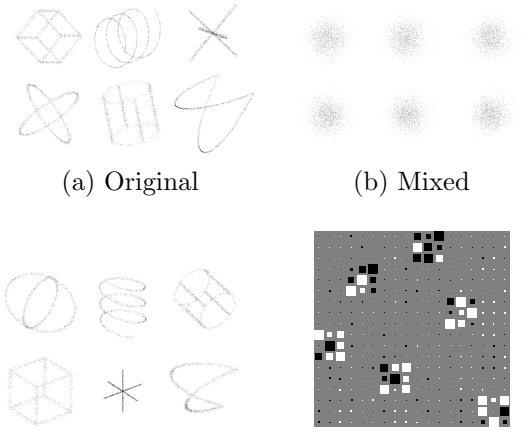


Figure 6. ISA experiment for 6 3-dimensional sources.

6. Conclusion

We proposed a new ICA and ISA method, SWICA, based on a non-parametric rank-based estimate of the dependence between pairs of variables. Our method frequently outperforms other state of the art ICA algorithms, is very robust to outliers, and only moderately sensitive to noise. On the other hand, it is somewhat slower than other ICA methods, and requires more samples to separate near-Gaussian sources. In the future, we plan to investigate possible accelerations to the algorithm, and statistical characteristics of the source distributions that affect the contrast.

Acknowledgements

This work has been supported by the Alberta Ingenuity Fund through the Alberta Ingenuity Centre for Machine Learning.

(a) Original (b) Mixed (c) SWICA (d) FastICA (e) RADICAL

Figure 7. Separation of outlier-corrupted mixed images. (a) The original images. (b) the mixed images corrupted with outliers. (c)-(e) The separated images using SWICA, FastICA, and RADICAL algorithms, respectively. The Amari error of the SWICA, FastICA, Radical was 0.10, 0.30, 0.29 respectively. The quality of the KernelICA and JADE was similar to that of FastICA and RADICAL.

References

Amari, S., Cichocki, A., & Yang, H. (1996). A new learning algorithm for blind source separation. *NIPS* (pp. 757–763).

Bach, F. R., & Jordan, M. I. (2002). Kernel independent component analysis. *JMLR*, *3*, 1–48.

Bach, F. R., & Jordan, M. I. (2003). Beyond independent components: Trees and clusters. *JMLR*, *4*, 1205–1233.

Cardoso, J.-F. (1998). Multidimensional independent component analysis. *Proc. ICASSP'98, Seattle, WA*.

Cardoso, J.-F. (1999). High-order contrasts for independent component analysis. *Neural Computation*, *11*, 157–192.

Comon, P. (1994). Independent component analysis, a new concept? *Signal Proc.*, *36*, 287–314.

Deheuvels, P. (1979). La fonction de dépendance empirique et ses propriétés, un test non paramétrique d'indépendance. *Bulletin de l'Académie Royale de Belgique, Classe des Sciences*, 274–292.

Hyvärinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. on Neural Networks*, 626–634.

Hyvärinen, A., Karhunen, J., & Oja, E. (2001). *Independent component analysis*. New York: John Wiley.

Hyvärinen, A., & Köster, U. (2006). FastISA: A fast fixed-point algorithm for independent subspace analysis. *Proc. of ESANN*.

Learned-Miller, E. G., & Fisher, J. W. (2003). ICA using spacings estimates of entropy. *JMLR*, *4*, 1271–1295.

Nelsen, R. B. (2006). *An introduction to copulas*. Springer Series in Statistics. Springer. 2nd edition.

Póczos, B., & Lőrincz, A. (2005). Independent subspace analysis using geodesic spanning trees. *Proc. of ICML-2005* (pp. 673–680).

Schweizer, B., & Wolff, E. F. (1981). On nonparametric measures of dependence for random variables. *The Annals of Statistics*, *9*, 879–885.

Sklar, A. (1959). Fonctions de répartition à n dimensions et leures marges. *Publications de l'Institut de Statistique de L'Université de Paris*, *8*, 229–231.

Szabó, Z., Póczos, B., & Lőrincz, A. (2007). Undercomplete blind subspace deconvolution. *JMLR*, *8*, 1063–1095.

Theis, F. J. (2005). Blind signal separation into groups of dependent signals using joint block diagonalization. *Proc. of ISCAS*. (pp. 5878–5881).

Theis, F. J. (2007). Towards a general independent subspace analysis. *Proc. of NIPS 19* (pp. 1361–1368).

Unsupervised Rank Aggregation with Distance-Based Models

Alexandre Klementiev KLEMENTI@UIUC.EDU
Dan Roth DANR@UIUC.EDU
Kevin Small KSMALL@UIUC.EDU
University of Illinois at Urbana-Champaign, 201 N Goodwin Ave, Urbana, IL 61801 USA

Abstract

The need to meaningfully combine sets of rankings often comes up when one deals with ranked data. Although a number of heuristic and supervised learning approaches to rank aggregation exist, they require domain knowledge or supervised ranked data, both of which are expensive to acquire. In order to address these limitations, we propose a mathematical and algorithmic framework for learning to aggregate (partial) rankings without supervision. We instantiate the framework for the cases of combining permutations and combining top-k lists, and propose a novel metric for the latter. Experiments in both scenarios demonstrate the effectiveness of the proposed formalism.

1. Introduction

Consider the scenario where each member of a panel of judges independently generates a (partial) ranking over a set of items while attempting to reproduce a true underlying ranking according to their level of expertise. This setting motivates a fundamental machine learning and information retrieval (IR) problem - the necessity to meaningfully aggregate preference rankings into a joint ranking. The IR community refers to this as *data fusion*, where a joint ranking is derived from the outputs of multiple retrieval systems. For example, in *meta-search* the aim is to aggregate Web search query results from several engines into a more accurate ranking. In many natural language processing applications, such as *machine translation*, there has been an increased interest in combining the results of multiple systems built on different principles in an effort to improve performance (Rosti et al., 2007).

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

One impediment to solving *rank aggregation* tasks is the high cost associated with acquiring full or partial preference information, making supervised approaches of limited utility. For data fusion, efforts to overcome this difficulty include applying domain specific heuristics (Shaw & Fox, 1994) or collecting such preference information indirectly (e.g. using clickthrough data (Joachims, 2002)). In order to address this limitation, we propose a general *unsupervised learning* framework for (partial) rank aggregation.

Analyzing ranked data is an extensively studied problem in statistics, information retrieval, and machine learning literature. (Mallows, 1957) introduced a distance-based model for fully ranked data and investigated its use with Kendall's and Spearman's metrics. The model was later generalized to other distance functions and for use with partially ranked data (Critchlow, 1985). (Lebanon & Lafferty, 2002) proposed a multi-parameter extension, where multiple modal rankings (e.g. expert opinions) are available and use their formalism for supervised ensemble learning; they also analyzed their model for partially ranked data (Lebanon & Lafferty, 2003).

The first key contribution of our work is the derivation of an EM-based algorithm for learning the parameters of the extended Mallows model without supervision. We instantiate the model with appropriate distance functions for two important scenarios: combining permutations and combining top-k lists. In the context of defining distances between rankings, various metrics have been proposed and analyzed (Critchlow, 1985; Estivill-Castro et al., 1993). Distances over top-k lists, i.e. rankings over the k most preferable objects, receive particular attention in the IR community (Fagin et al., 2003). (Fligner & Verducci, 1986) show that a class of distance functions between full rankings, such as Kendall's and Cayley's metrics, decompose into a sum of independent components allowing for efficient parameter estimation of the standard Mallows model.

A second key contribution of our work is the derivation

of a novel decomposable distance function for top-k lists. We show it to be a generalization of the Kendall metric and demonstrate that it can be decomposed, enabling us to estimate the parameters of the extended Mallows model efficiently.

Among recent work, (Busse et al., 2007) propose a method for clustering heterogeneous rank data based on the standard Mallows model. More directly related, many heuristics as well as a number of supervised learning approaches (Liu et al., 2007) exist for rank aggregation, although few *learn* to combine rankings without any supervision. (Klementiev et al., 2007) frame unsupervised rank aggregation as an optimization problem specifically for top-k lists, which relies on user-tuned parameters, a form of implicit supervision, whereas we describe a general unsupervised framework that can be instantiated to top-k lists in addition to other settings.

The remainder of the paper is organized as follows: section 2 formalizes distance-based ranking models and introduces relevant notation. Section 3 derives our EM-based algorithm for learning model parameters and specifies the requirements for efficient learning and inference. Section 4 instantiates the framework for two common scenarios: permutations (full rankings) and top-k lists. Section 5 experimentally demonstrates the model's effectiveness in both cases. Finally, section 6 concludes the work and gives ideas for future directions.

2. Distance-Based Ranking Models

2.1. Notation and Definitions

Let $\{x_1, \ldots, x_n\}$ be a set of objects to be ranked, i.e. assigned rank-positions $1, \ldots, n$, by a judge. We denote the resulting permutation $\pi = (\pi(1), \ldots, \pi(n))$, where $\pi(i)$ is the rank assigned to object x_i. Correspondingly, we use $\pi^{-1}(j)$ to denote the index of the object assigned to rank j.

Let $\mathcal{S}_n$ be the set of all $n!$ permutations over n items, and let $d : \mathcal{S}_n \times \mathcal{S}_n \to \mathbb{R}$ be a distance function between two permutations. We will require $d(\cdot, \cdot)$ to be a *right-invariant metric* (Diaconis & Graham, 1977): in addition to the usual properties of a metric, we will also require that the value of $d(\cdot, \cdot)$ does not depend on how the set of objects is indexed. In other words, $d(\pi, \sigma) = d(\pi\tau, \sigma\tau)\ \forall \pi, \sigma, \tau \in \mathcal{S}_n$, where $\pi\tau$ is defined by $\pi\tau(i) = \pi(\tau(i))$.

In particular, note that $d(\pi, \sigma) = d(\pi\pi^{-1}, \sigma\pi^{-1}) = d(e, \sigma\pi^{-1})$, where $e = (1, \ldots, n)$ is the identity permutation. That is, the value of d does not change if we re-index the objects such that one of the permutations becomes e and the other $\nu = \sigma\pi^{-1}$. Borrowing the notation from (Fligner & Verducci, 1986) we abbreviate $d(e, \nu)$ as $D(\nu)$. In a later section, when we define ν as a random variable, we may treat $D(\nu) = D$ as a random variable as well: whether it is a distance function or a r.v. will be clear from the context.

2.2. Mallows Models

While a large body of work on ranking models exists in statistics literature, of particular interest to us are the distance based conditional models first introduced in (Mallows, 1957). Let us give a brief review of the formalism and elucidate some of the its properties relevant to our work. The model generates a judge's rankings according to:

$$p(\pi|\theta, \sigma) = \frac{1}{Z(\theta, \sigma)} \exp(\theta\ d(\pi, \sigma)) \tag{1}$$

where $Z(\theta, \sigma) = \sum_{\pi \in \mathcal{S}_n} \exp(\theta\ d(\pi, \sigma))$ is a normalizing constant. The parameters of the model are $\theta \in \mathbb{R}$, $\theta \leq 0$ and $\sigma \in \mathcal{S}_n$, referred to as the dispersion and the location parameters, respectively. The distribution's single mode is the modal ranking σ; the probability of ranking π decreases exponentially with distance from σ. When $\theta = 0$, the distribution is uniform, and it becomes more concentrated at σ as θ decreases.

One property of (1) is that the normalizing constant $Z(\theta, \sigma)$ does not depend on σ due to the right invariance of the distance function:

$$Z(\theta, \sigma) = Z(\theta) \tag{2}$$

Let us denote the moment generating function of D under (1) as $M_{D,\theta}(t)$, and as $M_{D,0}(t)$ under the uniform distribution ($\theta = 0$). Since (1) is an exponential family,

$$M_{D,\theta}(t) = \frac{M_{D,0}(t + \theta)}{M_{D,0}(\theta)}$$

Therefore,

$$\begin{aligned} E_\theta(D) &= \frac{1}{M_{D,0}(\theta)} \frac{dM_{D,0}(t+\theta)}{dt}\bigg|_{t=0} \\ &= \frac{d \ln(M_{D,0}(t))}{dt}\bigg|_{t=\theta} \end{aligned} \tag{3}$$

(Fligner & Verducci, 1986) note that if a distance function can be expressed as $D(\pi) = \sum_{i=1}^{m} V_i(\pi)$, where

$V_i(\pi)$ are independent (with π uniformly distributed) with m.-g.f. $M_i(t)$, then $M_{D,0}(t) = \prod_{i=1}^{m} M_i(t)$. Consequently, (3) gives:

$$E_\theta(D) = \frac{d}{dt} \sum_{i=1}^{m} \ln M_i(t) \Big|_{t=\theta} \quad (4)$$

We will call such distance functions *decomposable* and will later use (4) in section 4 in order to estimate θ efficiently.

2.3. Extended Mallows Models

(Lebanon & Lafferty, 2002) propose a natural generalization of the Mallows model to the following conditional model:

$$p(\pi|\boldsymbol{\theta}, \boldsymbol{\sigma}) = \frac{1}{Z(\boldsymbol{\theta}, \boldsymbol{\sigma})} p(\pi) \exp\left(\sum_{i=1}^{K} \theta_i \; d(\pi, \sigma_i) \right) \quad (5)$$

where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_K) \in \mathcal{S}_n^K$, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K) \in \mathbb{R}^K$, $\boldsymbol{\theta} \leq \mathbf{0}$, $p(\pi)$ is a prior, and normalizing constant $Z(\boldsymbol{\theta}, \boldsymbol{\sigma}) = \sum_{\pi \in \mathcal{S}_n} p(\pi) \exp(\sum_{i=1}^{K} \theta_i \; d(\pi, \sigma_i))$.

The rankings σ_i may be thought of as votes of K individual judges, e.g. rankings returned by multiple search engines for a particular query in the meta-search setting. The free parameters θ_i represent the degree of expertise of the individual judges: the closer the value of θ_i to zero, the less the vote of the i-th judge affects the assignment of probability.

Under the right-invariance assumption on d, we can use property (2) to derive the following generative story underlying the extended Mallows model:

$$p(\pi, \boldsymbol{\sigma}|\boldsymbol{\theta}) = p(\pi) \prod_{i=1}^{K} p(\sigma_i|\theta_i, \pi) \quad (6)$$

That is, π is first drawn from prior $p(\pi)$. $\boldsymbol{\sigma}$ is then made up by drawing $\sigma_1 \ldots \sigma_K$ *independently* from K Mallows models $p(\sigma_i|\theta_i, \pi)$ with the *same* location parameter π.

It is straightforward to generalize both Mallows models (Critchlow, 1985), and the extended Mallows models to *partial rankings* by constructing appropriate distance functions. We will assume this more general setting in the following section.

3. Learning and Inference

In this section, we derive the general formulation of Expectation Maximization algorithm for parameter estimation of the extended Mallows models (5), and suggest a class of distance functions for which learning can be done efficiently. We then describe an inference procedure for the model.

3.1. EM Background and Notation

Let us start with a brief overview of Expectation-Maximization (Dempster et al., 1977) mostly to introduce some notation. EM is a general method of finding maximum likelihood estimate of parameters of models which depend on unobserved variables. The EM procedure iterates between:

E step: estimate the expected value of complete data log-likelihood with respect to unknown data $\mathcal{Y}$, observed data $\mathcal{X}$, and current parameter estimates θ':

$$T(\theta, \theta') \quad = \quad E[\log p(\mathcal{X}, \mathcal{Y}|\theta)|\mathcal{X}, \theta']$$

M step: choose parameters that maximize the expectation computed in the E step:

$$\theta' \leftarrow \operatorname*{argmax}_{\theta} T(\theta, \theta')$$

In our setting, the $K > 2$ experts generate votes $\boldsymbol{\sigma}$ corresponding to the unobserved true ranking π. We will see multiple instances of $\boldsymbol{\sigma}$ so the observed data we get are ranking vectors $\mathcal{X} = \{\boldsymbol{\sigma}^{(j)}\}_{j=1}^{Q}$ with the corresponding true (unobserved) rankings $\mathcal{Y} = \{\pi^{(j)}\}_{j=1}^{Q}$.

In the meta-search example, $\sigma_i^{(j)}$ is the ranking of the i-th (of the total of K) search engine for the j-th (of the total of Q) query. The (unknown) true ranking corresponding to the j-th query is denoted as $\pi^{(j)}$.

3.2. EM Derivation

We now use the generative story (6) to derive the following propositions (proofs omitted due to space constraints):

Proposition 1. *The expected value of the complete data log-likelihood under* (5) *is:*

$$T(\boldsymbol{\theta}, \boldsymbol{\theta'}) = \sum_{(\pi^{(1)}, \ldots, \pi^{(Q)}) \in \mathcal{S}_n^Q} \mathcal{L}_{\boldsymbol{\theta}} \; \mathcal{U}_{\boldsymbol{\theta'}} \quad (7)$$

where the complete data log-likelihood $\mathcal{L}_{\boldsymbol{\theta}}$ *is:*

$$\mathcal{L}_{\boldsymbol{\theta}} = \sum_{j=1}^{Q} \log p(\pi^{(j)}) - Q\sum_{i=1}^{K} \log Z(\theta_i) + \sum_{j=1}^{Q}\sum_{i=1}^{K} \theta_i \; d(\pi^{(j)}, \sigma_i^{(j)})$$

and the marginal distribution of the unobserved data $\mathcal{U}_{\boldsymbol{\theta'}}$ *is:*

$$\mathcal{U}_{\boldsymbol{\theta'}} = \prod_{j=1}^{Q} p\left(\pi^{(j)} | \boldsymbol{\theta'}, \boldsymbol{\sigma}^{(j)}\right)$$

Proposition 2. *$T(\boldsymbol{\theta}, \boldsymbol{\theta'})$ is maximized by $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$ such that:*

$$E_{\theta_i}(D) = \sum_{\substack{(\pi^{(1)},\ldots,\pi^{(Q)}) \\ \in \mathcal{S}_n^Q}} \left(\frac{1}{Q}\sum_{q=1}^{Q} d(\pi^{(q)}, \sigma_i^{(q)})\right) \mathcal{U}_{\boldsymbol{\theta'}} \quad (8)$$

That is, on each iteration of EM, we need to evaluate the right-hand side (RHS) of (8) and solve the LHS for θ_i for each of the K components.

3.3. Model Learning and Inference

At first, both evaluating the RHS of (8) and solving the LHS for θ_i seem quite expensive ($> n!$). While true in general, we can make the learning tractable for a certain type of distance functions.

In particular, if a distance function can be decomposed into a sum of independent components under the uniform distribution of π (see section 2.2), property (4) may enable us to make the estimation of the LHS efficient. In Section 4, we show two examples of such distance functions (for permutations and top-k lists).

In order to estimate the RHS, we use the Metropolis algorithm (Hastings, 1970) to sample from (5). The chain proceeds as follows: denoting the most recent value sampled as π_t, two indices $i, j \in \{1, \ldots, n\}$ are chosen at random and the objects $\pi_t^{-1}(i)$ and $\pi_t^{-1}(j)$ are transposed forming π'_t. If $a = p(\pi'_t|\boldsymbol{\theta}, \boldsymbol{\sigma})/p(\pi_t|\boldsymbol{\theta}, \boldsymbol{\sigma}) \geq 1$ the chain moves to π'_t. If $a < 1$, the chain moves to π'_t with probability a; otherwise, it stays at π_t. (Diaconis & Saloff-Coste, 1998) show quick convergence for Mallows model with Cayley's distance. While no convergence results are known for the extended Mallows model with arbitrary distance, we found experimentally that the MC chain converges rapidly with the two distance functions used in this work ($10n$ steps in experiments of Section 5).

As the chain proceeds, we update the distance value with the incremental change due to a single transposition, instead of recomputing it from scratch, resulting in substantial savings in computation.

Alternatively, we also found (Section 5.1) that a combination of rankings σ_i weighted by $\exp(-\theta_i)$ provides a reasonable and quick estimate for evaluating the RHS.

Sampling or the suggested alternative RHS estimation used during training is also used for model inference.

4. Model Application

Overcoming the remaining hurdle (the LHS estimation) in learning the model efficiently depends on the definition of a distance function. We now consider two particular types of (partial) rankings: permutations, and top-k lists. The latter is the case when each judge specifies a ranking over k most preferable objects out of n. For instance, a top-10 list may be associated with the 10 items on the first page of results returned by a web search engine. For both permutations and top-k lists, we show distance functions which satisfy the decomposability property (Section 2.2), which, in turn, allows us to estimate the LHS of (8) efficiently.

4.1. Combining Permutations

Kendall's tau distance (Kendall, 1938) between permutations π and σ is a right-invariant metric defined as the minimum number of pairwise adjacent transpositions needed to turn one permutation into the other. Assuming that one of the permutations, say σ, is the identity permutation e (we can always turn one of the permutations into e by re-indexing the objects without changing the value of the distance, see Section 2.1), it can be written as:

$$D_K(\pi) = \sum_{i=1}^{n-1} V_i(\pi)$$

where[1] $V_i(\pi) = \sum_{j>i} I(\pi^{-1}(i) - \pi^{-1}(j))$. V_i are independent and uniform over integers $[0, n-i]$ (Feller, 1968) with m.-g.f. $M_i(t) = \frac{1}{n-i+1}\sum_{k=0}^{n-i} e^{tk}$. Following (Fligner & Verducci, 1986), equation (4) gives:

$$E_\theta(D_K) = \frac{ne^\theta}{1-e^\theta} - \sum_{j=1}^{n} \frac{je^{\theta j}}{1-e^{\theta j}} \quad (9)$$

$E_\theta(D_K)$ is monotone decreasing, so line search for θ will converge quickly.

[1] $I(x) = 1$ if $x > 0$, and 0 otherwise.

4.2. Combining Top-k Lists

We now propose an extension of the Kendall's tau distance to top-k lists, i.e. the case where π and σ indicate preferences over different (possibly, overlapping) subsets of $k \leq n$ objects.

Let us denote by F_π and F_σ the elements in π and σ respectively, noting that $|F_\pi| = |F_\sigma| = k$. We define $Z = F_\pi \cap F_\sigma$, $|Z| = z$, $P = F_\pi \setminus F_\sigma$, and $S = F_\sigma \setminus F_\pi$ (note that $|P| = |S| = k - z = r$). We treat π and σ as rankings, which for us will mean that the smallest index will indicate the top, i.e. contain the most preferred object. For notational convenience, let us now define the *augmented ranking* $\tilde{\pi}$ as π augmented with the elements of S assigned the same index $(k+1)$, one past the bottom of the ranking as shown on Figure 1 ($\tilde{\sigma}$ is defined similarly). We will slightly abuse our notation and denote $\tilde{\pi}^{-1}(k+1)$ to be the set of elements in position $(k+1)$.

Kendall's tau distance D_K is naturally extended from permutations to augmented rankings.

Definition 1. *Distance $\tilde{D}_K(\tilde{\pi}, \tilde{\sigma})$ between augmented rankings $\tilde{\pi}$ and $\tilde{\sigma}$ is the minimum number of adjacent transpositions needed to turn $\tilde{\pi}$ into $\tilde{\sigma}$.*

It can be shown that $\tilde{D}_K(\tilde{\pi}, \tilde{\sigma})$ is a right-invariant metric, thus we will again simplify the notation denoting it as $\tilde{D}_K(\tilde{\pi})$. This distance can be decomposed as:

$$\tilde{D}_K(\tilde{\pi}) = \sum_{\substack{i=1 \\ \tilde{\pi}^{-1}(i) \in Z}}^{k} \tilde{V}_i(\tilde{\pi}) + \sum_{\substack{i=1 \\ \tilde{\pi}^{-1}(i) \notin Z}}^{k} \tilde{U}_i(\tilde{\pi}) + \frac{r(r+1)}{2}$$

where

$$\begin{aligned} \tilde{V}_i(\tilde{\pi}) &= \sum_{\substack{j=i \\ \tilde{\pi}^{-1}(j) \in Z}}^{k} I(\tilde{\pi}^{-1}(i) - \tilde{\pi}^{-1}(j)) + \\ &\quad \sum_{j \in \tilde{\pi}^{-1}(k+1)} I(\tilde{\pi}^{-1}(i) - j) \\ \tilde{U}_i(\tilde{\pi}) &= \sum_{\substack{j=i \\ \tilde{\pi}^{-1}(j) \in Z}}^{k} 1 \end{aligned}$$

Decomposing $\tilde{D}_K(\tilde{\pi})$, the second term is the minimum number of adjacent transpositions necessary to bring the r elements not in Z (grey boxes on Figure 1) to the bottom of the ranking. The third term is the minimum number of adjacent transpositions needed to switch them with the elements in $\tilde{\pi}^{-1}(k+1)$, which would then appear in the correct order in the bottom r positions. Finally, the first term is the adjacent transpositions necessary to put the k elements now in the list in the natural order.

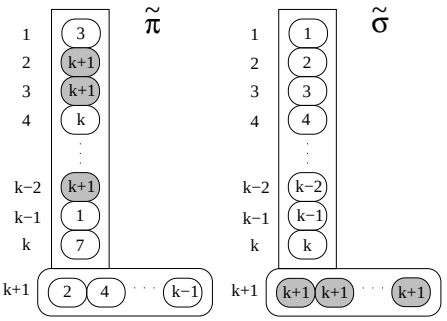


Figure 1. An example of augmented permutations $\tilde{\pi}$ (left) and identity augmented permutation $\tilde{\sigma}$ (right, in natural order). Grey boxes are objects in π but not in σ. $\tilde{D}_K(\tilde{\pi})$ is the minimum number of adjacent transpositions needed to turn $\tilde{\pi}$ into $\tilde{\sigma}$: namely, bring all grey boxes into the position $k+1$ and put the remaining k objects in their natural order.

It can be shown that the random variable summands comprising $\tilde{D}_K(\tilde{\pi})$ are independent when $\tilde{\pi}$ is uniformly distributed. Furthermore, $\tilde{V}_i$ and $\tilde{U}_j$ are uniform over integers $[0, k-i]$ and $[0, z]$, with moment generating functions $\frac{1}{k-i+1}\sum_{j=0}^{k-i} e^{tj}$ and $\frac{1}{z+1}\sum_{j=0}^{z} e^{tj}$, respectively. Assuming $z > 0$, and $r > 0$ equation (4) gives:

$$E_\theta(\tilde{D}_K) = \frac{ke^\theta}{1-e^\theta} - \sum_{j=r+1}^{k} \frac{je^{j\theta}}{1-e^{j\theta}} + \frac{r(r+1)}{2} - r(z+1)\frac{e^{\theta(z+1)}}{1-e^{\theta(z+1)}} \qquad (10)$$

If $r = 0$ (i.e. the augmented rankings are over the same objects), both the distance and the expected value reduce to the Kendall distance results. Also, if $z = 0$ (i.e. the augmented rankings have no objects in common), $\tilde{D}_K = E_\theta(\tilde{D}_K) = k(k+1)/2$, which is the smallest number of adjacent transpositions needed to move the $r = k$ objects in $\tilde{\pi}^{-1}(k+1)$ into the top k positions.

$E_\theta(\tilde{D}_K)$ is decreasing monotonically, so we can again use line search to find the value of θ. Notice that the expected value depends on the value of z (the number of common elements between the two permutations). We will compute the average value of z as we estimate the RHS of (8) and use it to solve the LHS for θ.

5. Experimental Evaluation

We demonstrate the effectiveness of our approach for permutations and top-k lists considered in Section 4.

5.1. Permutations

We first consider the scenario of aggregating permutations. For this set of experiments, the votes of $K = 10$ individual experts were produced by sampling standard Mallows models (1), with the same location parameter $\sigma^* = e$ (an identity permutation over $n = 30$ objects), and concentration parameters $\theta^*_{1,2} = -1.0$, $\theta^*_{3,..,9} = -0.05$, and $\theta^*_{10} = 0$ (the latter generating all permutations uniformly randomly). The models were sampled 10 times, resulting in $Q = 10$ lists of permutations (one for each "query"), which constituted the training data.

In addition to the sampling procedure described in Section 3.3 to estimate the RHS of (8), we also tried the following weighted Borda count approximation. For each "query" q, we took the K votes and mixed them into a single permutation $\hat{\sigma}_q$ as follows: a score for each of the n objects is computed as a weighted combination of ranks assigned to that object by individual judges. The aggregate permutation $\hat{\sigma}_q$ is obtained by sorting the objects according to their resulting scores. The weights are computed using the current values of the model parameters as $\exp(-\theta_i)$. The rationale is that the smaller the absolute value of θ_i, the lower the relative quality of the ranker, and the less it should contribute to the aggregate vote. Finally, the RHS for the i-th component is computed as the distance from its vote to $\hat{\sigma}_q$ averaged over all Q queries.

We also tried using the true permutation σ^* in place of $\hat{\sigma}_q$ to see how well the learning procedure can do.

At the end of each EM iteration, we sampled the current model (5), and computed the Kendall's tau distance between the generated permutation to the true σ^*. Figure 2 shows the model performance when sampling and the proposed approximation are used to estimate the RHS. Although the convergence is much faster with the approximation, the model trained with the sampling method achieves better performance approaching the case when the true permutation is known.

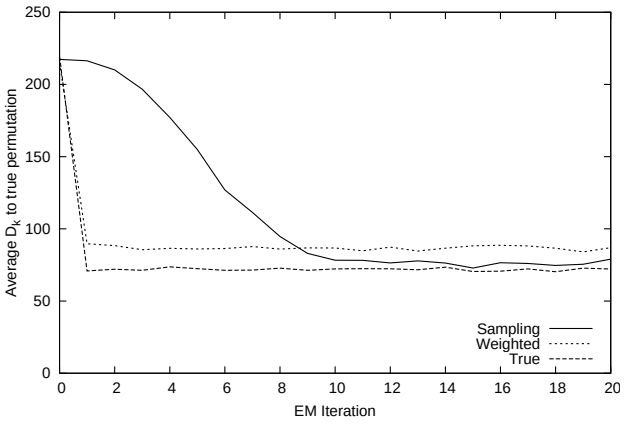


Figure 2. Permutations: learning performance of the model (averaged over 5 runs) when RHS is estimated using sampling (Sampling), the proposed weighted Borda count approximation (Weighted), or the true permutation (True). Although the convergence is much faster with the approximation, model trained with the sampling method achieves better performance.

5.2. Top-k lists

In order to estimate the model's performance in the top-k list combination scenario, we performed data fusion experiments using the data from the ad-hoc retrieval shared task of the TREC-3 conference (Harman, 1994). Our goal here is to examine the behavior of our approach as we introduce poor judges into the constituent ranker pool. In this shared task, 40 participants submitted top-1000 ranking over a large document collection for each of the 50 queries. For our experiments, we used top-100 ($k = 100$) rankings from $K = 38$ of the participants (two of the participants generated shorter rankings for some of the queries and were not used) for all $Q = 50$ queries. We replaced a specific number $K_r \in [0, K]$ of the participants with random rankers (drawing permutations of k documents from the set of documents returned by all participants for a given query uniformly randomly). We then used our algorithm to combine top-k lists from K_r random rankers and $(K - K_r)$ participants chosen at random.

We measure performance using the precision in top-$\{10, 30\}$ documents as computed by *trec_eval*[2] from the TREC conference series. As a baseline, we use $CombMNZ_{rank}$ suggested in (Klementiev et al., 2007). It is a variant of a commonly used $CombMNZ$ (Shaw & Fox, 1994). Given a query q for each document x in the collection it computes a score $N_x \times \sum_{i=1}^{K}(k - r_i(x, q))$, where $r_i(x, q)$ is the rank of the document x in the ranking returned by participant i for the query q, and N_x is the number of participants which place x in their top-k rankings. The aggregate

[2]Available at http://trec.nist.gov/

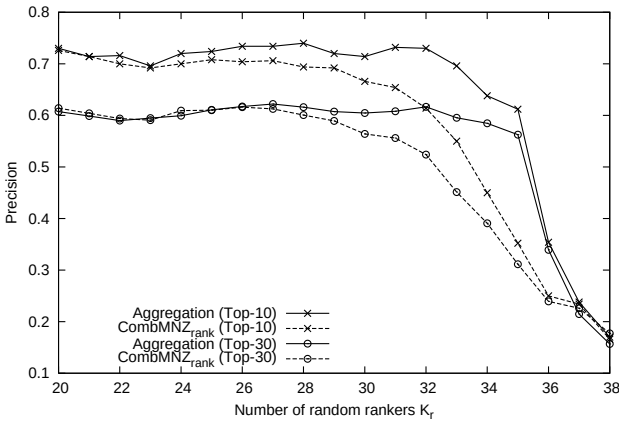


Figure 3. Top-k lists: precision of the aggregate ranker as a function of the number of random component rankers K_r in top 10 and top 30 documents. Our algorithm learns to discount the random components without supervision substantially improving over $CombMNZ_{rank}$.

ranking is obtained by sorting documents according to their scores. Intuitively, the more component rankers rank a document highly the higher it appears in the aggregate ranking.

Figure 3 shows that our algorithm learns to discount the random components *without supervision* substantially improving over the baseline as $K_r \to K$.

We also compared our results with the ULARA algorithm (Klementiev et al., 2007). These results were not included since we found ULARA to be too sensitive to user-defined parameters (an implicit form of supervision) with results varying between competitive with our model to comparable with $CombMNZ_{rank}$.

5.3. Model Dispersion Parameters

In order to demonstrate the relationship between the learned dispersion parameters of the model, $\boldsymbol{\theta}$, and the relative performance of the constituent rankers, we also conducted a meta-search experiment. First, we generated $Q = 50$ queries which result in an unambiguous most relevant document and submitted them to $K = 4$ commercial search engines. For each engine, we kept the 100 highest ranked documents (10 pages of 10 documents each) after removing duplicates, and unified URL formatting differences between engines. We measure performance with Mean Reciprocal Page Rank ($MRPR$), which we define as mean reciprocal rank of the page number on which the correct document appears.

Table 1 shows $MRPR$ of the four search engines and their corresponding model parameters. As expected, the results suggest a correlation between the magnitude of the dispersion parameters and the relative system performance, implying that their values may also be used for unsupervised search engine evaluation. Finally, our model achieves $MRPR = 0.92$ beating all of the constituent rankers.

Table 1. $MRPR$ of the four search engines and their corresponding model parameters; the results suggest a correlation between the magnitude of the dispersion parameters and the relative system performance.

	$S1$	$S2$	$S3$	$S4$
θ	-0.065	0.0	-0.066	-0.049
$MRPR$	0.86	0.43	0.82	0.78

6. Conclusions and Future Work

We propose a formal mathematical and algorithmic framework for aggregating (partial) rankings without supervision. We derive an EM-based algorithm for the extended Mallows model and show that it can be made efficient for the right-invariant decomposable distance functions. We instantiate the framework and experimentally demonstrate its effectiveness for the important cases of combining permutations and combining top-k lists. In the latter case, we introduce the notion of augmented permutation and a novel decomposable distance function for efficient learning.

A natural extension of the current work is to instantiate our framework for other types of partial rankings, as well as to cases where ranking data is not of the same type. The latter is of practical significance since often preference information available is expressed differently by different judges (e.g. top-k rankings of different lengths).

Another direction for future work is to extend the rank aggregation model to accommodate position dependence. In IR, more importance is generally given to results appearing higher in the rankings. Within our framework one may be able to design a distance function reflecting this requirement. Additionally, the quality of votes produced by individual components may depend on the rank, e.g. in the top-k scenario some rankers may be better at choosing few most relevant objects, while others may tend to have more relevant objects in the k selected but may not rank them well relative to one another. This case may be modeled by adding a dependency on rank to the dispersion parameters of the model.

In addition, this framework appears promising for a number of applications. Besides the NLP problems mentioned before, such as learning to combine output from multiple machine translation systems, one interesting setting may be *domain adaptation*. Here, the task is to adapt a hypothesis trained with ample labeled data from one input distribution to a second distribution where minimal training data is available. When the hypothesis is a trained aggregate ranker, we expect the relative expertise of its components to change and can use our approach to reweigh them accordingly.

Acknowledgments

We would like to thank Ming-Wei Chang, Sariel Har-Peled, Vivek Srikumar, and the anonymous reviewers for their valuable suggestions. This work is supported by NSF grant ITR IIS-0428472, DARPA funding under the Bootstrap Learning Program and by MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

References

Busse, L. M., Orbanz, P., & Buhmann, J. M. (2007). Cluster analysis of heterogeneous rank data. *Proc. of the International Conference on Machine Learning (ICML)*.

Critchlow, D. E. (1985). *Metric methods for analyzing partially ranked data*, vol. 34 of *Lecture Notes in Statistics*. Springer-Verlag.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, *39*, 1–38.

Diaconis, P., & Graham, R. L. (1977). Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society*, *39*, 262–268.

Diaconis, P., & Saloff-Coste, L. (1998). What do we know about the Metropolis algorithm? *Journal of Computer and System Sciences*, *57*, 20–36.

Estivill-Castro, V., Mannila, H., & Wood, D. (1993). Right invariant metrics and measures of presortedness. *Discrete Applied Mathematics*, *42*, 1–16.

Fagin, R., Kumar, R., & Sivakumar, D. (2003). Comparing top k lists. *SIAM Journal on Discrete Mathematics*, *17*, 134–160.

Feller, W. (1968). *An introduction to probability theory and its applications*, vol. 1. John Wiley and Sons, Inc.

Fligner, M. A., & Verducci, J. S. (1986). Distance based ranking models. *Journal of the Royal Statistical Society*, *48*, 359–369.

Harman, D. (1994). Overview of the third Text REtrieval Conference (TREC-3).

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, *57*, 97–109.

Joachims, T. (2002). Unbiased evaluation of retrieval quality using clickthrough data. *SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, *30*, 81–93.

Klementiev, A., Roth, D., , & Small, K. (2007). An unsupervised learning algorithm for rank aggregation. *Proc. of the European Conference on Machine Learning (ECML)* (pp. 616–623).

Lebanon, G., & Lafferty, J. (2002). Cranking: Combining rankings using conditional probability models on permutations. *Proc. of the International Conference on Machine Learning (ICML)*.

Lebanon, G., & Lafferty, J. (2003). Conditional models on the ranking poset. *The Conference on Advances in Neural Information Processing Systems (NIPS)* (pp. 431–438).

Liu, Y.-T., Liu, T.-Y., Qin, T., Ma, Z.-M., & Li, H. (2007). Supervised rank aggregation. *Proc. of the International World Wide Web Conference (WWW)*.

Mallows, C. L. (1957). Non-null ranking models. *Biometrika*, *44*, 114–130.

Rosti, A.-V. I., Ayan, N. F., Xiang, B., Matsoukas, S., Schwartz, R., & Dorr, B. J. (2007). Combining outputs from multiple machine translation systems. *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)* (pp. 228–235).

Shaw, J. A., & Fox, E. A. (1994). Combination of multiple searches. *Text REtrieval Conference (TREC)* (pp. 243–252).

On Partial Optimality in Multi-label MRFs

Pushmeet Kohli[1] PKOHLI@MICROSOFT.COM
Alexander Shekhovtsov[2] SHEKHOVT@CMP.FELK.CVUT.CZ
Carsten Rother[1] CARROT@MICROSOFT.COM
Vladimir Kolmogorov[3] VNK@ADASTRAL.UCL.AC.UK
Philip Torr[4] PHILIPTORR@BROOKES.AC.UK

[1]Microsoft Research Cambridge, [2]Czech Technical University in Prague, [3]University College London, [4]Oxford Brookes University

Abstract

We consider the problem of optimizing multi-label MRFs, which is in general NP-hard and ubiquitous in low-level computer vision. One approach for its solution is to formulate it as an integer linear programming and relax the integrality constraints. The approach we consider in this paper is to first convert the multi-label MRF into an equivalent binary-label MRF and then to relax it. The resulting relaxation can be efficiently solved using a maximum flow algorithm. Its solution provides us with a partially optimal labelling of the binary variables. This partial labelling is then easily transferred to the multi-label problem. We study the theoretical properties of the new relaxation and compare it with the standard one. Specifically, we compare tightness, and characterize a subclass of problems where the two relaxations coincide. We propose several combined algorithms based on the technique and demonstrate their performance on challenging computer vision problems.

1. Introduction

One of the major advances in computer vision in the past few years has been the development of efficient deterministic algorithms for solving discrete labeling problems. Labeling problems occur in many places from dense stereo and image segmentation (Boykov et al., 2001; Szeliski et al., 2006) to the use of pictorial structures for object recognition (Felzenszwalb & Huttenlocher, 2000). They can be shown to be equivalent to the problem of estimating the maximum a posterior (MAP) solution in graphical models such as Markov Random Fields (MRF) and Conditional Random Fields (CRF), which is also often referred to as *energy minimization*.

A number of powerful algorithms are present in the literature which deal with the problem. For certain subclasses of the problem, it is possible to compute the exact solution in polynomial time: MRFs of bounded tree-width, *e.g.* (Lauritzen, 1998); with convex pairwise potentials (Ishikawa, 2003); with submodular potentials of binary (Hammer, 1965; Kolmogorov & Zabih, 2004) or multi-label (Schlesinger & Flach, 2000; Kovtun, 2004) variables; with permuted submodular potentials (Schlesinger, 2007). However, the problem of minimizing a general energy function is NP-hard. Nevertheless it is just these sorts of general energies that occur in many vision problems and making progress towards their solution is of paramount importance.

Energy Minimization as a Linear Program General discrete energy minimization can be seen as an integer programming problem. Dropping integrality constraints leads to an attractive linear programming relaxation (LP-1). Unfortunately, linear programs arising from this scheme in vision applications are of very large scale, and are not practical to solve with general methods. A number of algorithms have been developed (Schlesinger, 1976; Koval & Schlesinger, 1976; Wainwright et al., 2003; Kolmogorov, 2006; Werner, 2007) which attempt to solve this relaxation by exploiting the special structure of the problem. However, their common drawback is that they may converge to a suboptimal point. Other developed methods for energy minimization include local search algorithms (Boykov et al., 2001), primal-dual method (Komodakis & Tziritas, 2005), subgradient methods (Schlesinger & Giginyak, 2007; Komodakis et al., 2007). Some local search algorithms provide

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

approximation ratio guarantee for (semi-)metric energies (Boykov et al., 2001; Komodakis & Tziritas, 2005). Finally, there are methods which output a partial assignment of labels with guaranteed optimality certificate for binary (Hammer et al., 1984; Boros et al., 1991; Boros et al., 2006; Rother et al., 2007) and multi-label (Kovtun, 2003; Kovtun, 2004) problems. Dead-end elimination is another related method for identifying non-optimal assignments based on local sufficient conditions, it was originally proposed (Desmet et al., 1992) for predicting and designing the structures of proteins. In this paper we develop a novel method for obtaining partial optimal solutions of functions of multi-label variables.

Our method works by first transforming the multi-label energy function to a function involving binary variables (Schlesinger & Flach, 2006). This binary energy is then minimized by applying the *roof dual* relaxation (Boros & Hammer, 2002; Hammer et al., 1984), which can be solved efficiently using a single graph cut. More importantly, solving the roof dual relaxation results in an assignment of a subset of the variables which is guaranteed to be valid for any optimal solution. This partially optimal solution can be used to constrain the state space of the original multi-label problem. As we show, this approach may be viewed as a different LP-relaxation of the multi-label energy minimization problem (LP-2).

Comparing LP-1 with LP-2 We present a number of theoretical results studying properties of LP-2 and relating LP-2 with LP-1. Our first result is that LP-1 is a tighter relaxation of the energy minimization problem compared to LP-2. We show in the paper that solutions of LP-1 necessarily satisfy the constraints derived by LP-2, so additional guarantees for methods based on LP-1 may follow. We also identify a subclass of problems for which LP-2 is as tight as LP-1. Thus, for problems of this subclass LP-1 can be solved exactly and efficiently using combinatorial methods.

It was recently demonstrated that the roof dual relaxation performs well for a number of computer vision applications (Kolmogorov & Rother, 2007; Rother et al., 2007) which are naturally formulated as a binary energy minimization. However, it turns out that when multi-label problems are formulated as binary energy minimization the roof dual relaxation leaves many variables unassigned. We therefore use recently proposed enhancements of the roof dual technique called "probing" (Boros et al., 2006; Rother et al., 2007) which can often help resolve these ambiguities. Our last contribution is providing an alternative formulation of LP-2: we prove that it is equivalent to computing a decomposition of the energy into submodular and supermodular parts so that the sum of the lower bounds for each part is maximized. Precise details of this theoretical result are given in (Shekhovtsov et al., 2008, section 10).

Although this is primarily a theoretical paper, we have performed a number of experiments with various energy models arising in vision applications, including image restoration, object-based segmentation, and image stitching. Our experiments show that the proposed method outperforms the competing method of (Kovtun, 2003) by labelling many more random variables. We also demonstrate that it may help solving difficult problems by reducing their state space and applying other methods to the reduced problem.

2. Energy Minimization

Let $\mathcal{L} = \{1 \dots K\}$ be a set of labels. Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, where the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is antisymmetric and antireflexive, *i.e.* $(s,t) \in \mathcal{E} \Rightarrow (t,s) \notin \mathcal{E}$. We denote an ordered pair $(s,t) \in \mathcal{E}$ simply by st. Let each graph node $s \in \mathcal{V}$ be assigned a label $x_s \in \mathcal{L}$ and let a *labeling* (or *configuration*) be defined as $\mathrm{x} = \boldsymbol{\{} x_s \,|\, s \in \mathcal{V} \boldsymbol{\}}$[1]. Let $\boldsymbol{\{}\theta_s(i) \in \mathbb{R} \,|\, i \in \mathcal{L},\, s \in \mathcal{V}\boldsymbol{\}}$ be *univariate* potentials and $\boldsymbol{\{}\theta_{st}(i,j) \in \mathbb{R} \,|\, i,j \in \mathcal{L},\, st \in \mathcal{E}\boldsymbol{\}}$ be *pairwise* potentials of a random field. Let in addition θ_{const} be a *constant* term, and let a concatenated vector of potentials, including the constant term, be denoted as $\theta = (\theta, \theta_{\text{const}}) \in \Omega = \mathbb{R}^{\mathcal{I} \cup \{\text{const}\}}$, where set of indices

$$\mathcal{I} = \{(s,i) \,|\, s \in \mathcal{V},\, i \in \mathcal{L}\} \cup \{(st, ij) \,|\; st \in \mathcal{E},\, i,j \in \mathcal{L}\} \tag{1}$$

corresponds to all univariate and pairwise terms. Notation $\theta_{\mathcal{I}}$ will thus refer to all components of θ but the constant term. The *energy* of a configuration x of the random field is defined as:

$$E(\mathrm{x}|\theta) = \sum_{s \in \mathcal{V}} \theta_s(x_s) + \sum_{st \in \mathcal{E}} \theta_{st}(x_s, x_t) + \theta_{\text{const}}. \tag{2}$$

2.1. LP-relaxation

We will study two different relaxations of minimization of (2). Both relaxations can be written using the same formulation but will differ in the graph, number of labels and potential vector involved.

Energy function (2) can be conveniently written using a scalar product in Ω as $E(\mathrm{x}|\theta) = \langle \mu(\mathrm{x}), \theta \rangle$, where $\mu(\mathrm{x}) \in \{0,1\}^{\mathcal{I} \cup \{\text{const}\}}$ is defined by $\mu(\mathrm{x})_s(i) = \delta_{\{x_s = i\}}$, $\mu(\mathrm{x})_{st}(i,j) = \delta_{\{x_s = i\}}\delta_{\{x_t = j\}}$ and $\mu(\mathrm{x})_{\text{const}} = 1$. In this notation minimization of E is expressed as:

$$\min_{\mathrm{x} \in \mathcal{L}^{\mathcal{V}}} \langle \mu(\mathrm{x}), \theta \rangle. \tag{3}$$

[1] Notation $\boldsymbol{\{} x_s | s \in S \boldsymbol{\}}$ (bold brackets), where S is a finite set, will stand for the concatenated vector of variables x_s, rather than the set of their values.

The LP-relaxation of (3) reviewed in, *e.g.*, (Werner, 2007) is constructed as follows. For each variable x_s, a set of relaxed variables $\mu_s(i) \in [0,1]$, $i \in \mathcal{L}$ is introduced. These variables are required to satisfy the *normalization* constraints

$$\sum_{i\in\mathcal{L}} \mu_s(i) = 1, \quad \forall s \in \mathcal{V}. \tag{4}$$

Further, for each pair (x_s, x_t), $st \in \mathcal{E}$ the relaxed variables $\mu_{st}(i,j) \in [0,1]$, $i,j \in \mathcal{L}$ are introduced which must satisfy the *marginalization* constraints:

$$\begin{aligned} \sum_{j'\in\mathcal{L}} \mu_{st}(i,j') &= \mu_s(i), \quad \forall st \in \mathcal{E}, \forall i \in \mathcal{L}, \\ \sum_{i'\in\mathcal{L}} \mu_{st}(i',j) &= \mu_t(j), \quad \forall st \in \mathcal{E}, \forall j \in \mathcal{L}. \end{aligned} \tag{5}$$

The concatenated vector $\mu \in \Omega$ satisfying these constraints will be called a *relaxed* labeling. Indeed, a vector μ with all integral components uniquely represents a labeling x. When the integrality constraints are dropped we get the following relaxation of (3):

$$\min_{\mu\in\Lambda_{G,\mathcal{L}}} \langle \mu, \theta \rangle, \tag{6}$$

where $\Lambda_{G,\mathcal{L}} = \left\{ \mu \in \Omega_+ \,|\, A\mu_{\mathcal{I}} = \mathbf{0},\ B\mu_{\mathcal{I}} = \mathbf{1},\ \mu_{\text{const}} = 1 \right\}$ is called the *local* (Wainwright et al., 2003) polytope of graph G. Here set Ω_+ denotes vectors from Ω with all nonnegative components, equalities $A\mu_{\mathcal{I}} = \mathbf{0}$ express marginalization constraints (5), and equalities $B\mu_{\mathcal{I}} = \mathbf{1}$ express normalization constraints (4).

2.2. Binary Energy Minimization

Energy minimization problems with 2 labels ($|\mathcal{L}| = 2$) are conveniently described in terms of *binary* (or Boolean) variables, *i.e.* with set of labels being $\mathbb{B} = \{0,1\}$. For clarity we will denote binary configurations by z. Univariate and pairwise terms of (2) can be written as:

$$\begin{aligned} &\theta_s(z_s) = \theta_s(1)z_s + \theta_s(0)(1-z_s), \\ &\theta_{st}(z_s,z_t) = \theta_{st}(1,1)z_sz_t + \theta_{st}(0,1)(1-z_s)z_t \\ &+\theta_{st}(1,0)z_s(1-z_t) + \theta_{st}(0,0)(1-z_s)(1-z_t). \end{aligned} \tag{7}$$

Expanding brackets in (7) it is clear that (2) can be transformed to the form:

$$E(\mathrm{z}|\eta) = \sum_s \eta_s z_s + \sum_{st} \eta_{st} z_s z_t + \eta_{\text{const}}, \tag{8}$$

which is a quadratic function of binary variables. Functions of the form $\mathbb{B}^{\mathcal{V}} \mapsto \mathbb{R}$ are called *pseudo-Boolean* (Boros & Hammer, 2002) and minimization (or maximization) of (8) is called quadratic Pseudo-boolean optimization.

Calculating coefficients η from θ is equivalent to choosing the reparametrization $\hat{\theta} \equiv \theta$ with non-zero elements being only $\hat{\theta}_s(1)$, $\hat{\theta}_{st}(1,1)$ and $\hat{\theta}_{\text{const}}$.

It is easy to see that pseudo-Boolean optimization, energy minimization with 2 labels and the MIN-CUT problem are all equivalent problems, and can be simply converted one into another. It is also known that polynomially solvable MIN-CUT problems (those having weights of all edges in the graph non-negative) correspond to quadratic pseudo-Boolean problems with all weights η_{st} being non-positive, which is equivalent to the condition of submodularity of $E(\cdot|\eta)$.

2.3. LP-relaxation of Binary Problems

As shown in (Hammer et al., 1984), the LP relaxation (6) for the case of binary variables has special properties, which in general do not hold in the multi-label case. First, there exists a *half-integral* optimal relaxed labeling μ^*, i.e. all components are 0, $\frac{1}{2}$ or 1. Second, if μ^* is integral for some node s (i.e. $\mu^*_s(\alpha) = 0$), then there exists a global minimum z of the original function in which $z_s = \alpha$. In other words, by solving the LP relaxation we can obtain constraints on the global minima of the binary energy. These constraints can be expressed as

$$\mathrm{z}^{\min} \le \mathrm{z} \le \mathrm{z}^{\max}, \tag{9}$$

where $\mathrm{z}^{\min}, \mathrm{z}^{\max} \in \mathbb{B}^{\mathcal{V}}$ and inequalities are componentwise. For instance, $0 \le z_s \le 1$ implies no constraints on z_s, while $0 \le z_s \le 0$ implies that z_s is constrained to be 0.

If (9) holds for all optimal solutions z, then the pair $(\mathrm{z}^{\min}, \mathrm{z}^{\max})$ is said to define *strong* persistency; if (9) holds for some optimal solution z then $(\mathrm{z}^{\min}, \mathrm{z}^{\max})$ defines *weak* persistency (Boros & Hammer, 2002). It is important to note that the LP relaxation can be solved very efficiently by computing minimum cut/maximum flow in an appropriately constructed graph. The technique in (Boros et al., 1991) is perhaps the most efficient; we will refer to it as the Qudratic Pseudo-Boolean Optimization (QPBO) method[2]. It yields a pair $(\mathrm{z}^{\min}, \mathrm{z}^{\max})$ defining strong persistency.

Recently, two techniques were introduced which extend the QPBO method. The first one is *Probing* (Boros et al., 2006), or *QPBO-P*. It fixes a certain node s to a particular label α, runs QPBO thus obtaining some information about global minimizers of the energy. This information is incorporated into the energy (e.g. if we learn that $z_s = z_t$ for all optimal solutions then we contract nodes s and t), and the "probing" is performed for other nodes until convergence. An efficient implementation of QPBO-P was described in (Rother et al., 2007). The second method

[2]Following (Rother et al., 2007) we use the abbreviation QPBO to denote the max-flow algorithm for computing the roof-dual (Boros et al., 1991) rather then the optimization problem itself.

is *Improve*, or *QPBO-I* (Rother et al., 2007). It takes an input labeling z and tries to improve its energy by fixing a subset of nodes to labels specified by z and running QPBO. These operations guarantee not to increase the energy, and in practice do often decrease it.

3. Solving Multi-label Problems

We now address the problem of minimizing energy functions involving multi-label variables.

3.1. Converting Multi-label Problems Into Binary Ones

As was discussed in Sec. 2.2, there are simple transitions between energy minimization with two labels, MIN-CUT problem and the quadratic pseudo-Boolean optimization. Thus, it is not essentially important to which of those forms a multi-label problem will be reduced. The construction (Ishikawa, 2003; Kovtun, 2004; Schlesinger & Flach, 2006) adopted to our notation of binary energies is as follows.

We start the transformation procedure by obtaining a reparametrization $\hat{\theta} \equiv \theta$ which satisfies the following:

$$\begin{aligned} \hat{\theta}_{st}(1,j) = \hat{\theta}_{st}(i,1) &= 0 \quad st \in \mathcal{E},\ i,j \in \mathcal{L} \\ \hat{\theta}_{s}(1) &= 0 \quad s \in \mathcal{V}. \end{aligned} \tag{10}$$

This reparametrization is easy to construct. For example, to achieve $\hat{\theta}_{st}(i,1) = 0$ one needs to subtract the value $\gamma_c(i) = \theta_{st}(i,1)$ from $\theta_{st}(i,j)$ and add it to $\theta_s(i)$, which does not change the energy, and so on, see (Shekhovtsov et al., 2008) for details. Note that in the case of two labels, this reparametrization $\hat{\theta}$ immediately provides coefficients for writing a binary energy in the form (8).

Let the tuple $(\mathcal{L}, G, \theta)$ define a multi-label energy minimization problem. Let $\tilde{\mathcal{L}}$ to refer to the set of the last $K-1$ labels of $\mathcal{L}$, i.e. $\tilde{\mathcal{L}} = \{2 \dots K\}$. The components of the equivalent binary energy minimization problem are given as follows (Fig 1):

- Graph $N = (V, A)$, where $V = \mathcal{V} \times \tilde{\mathcal{L}}$ and $A = A_{\mathcal{E}} \cup A_{\mathcal{V}}$. $A_{\mathcal{E}} = \big\{((s,i),(t,j)) \,|\, st \in \mathcal{E},\ i,j \in \tilde{\mathcal{L}}\big\}$. $A_{\mathcal{V}} = \big\{((s,i),(s,i-1)) \,|\, s \in \mathcal{V},\ i = 3 \dots K\big\}$.
- Binary configuration $\mathrm{z} \in \mathbb{B}^V$. For a configuration $\mathrm{x} \in \mathcal{L}^{\mathcal{V}}$ the corresponding binary configuration z(x) is defined by

$$\mathrm{z}(\mathrm{x})_{(s,i)} = \delta_{\{i \le x_s\}}, \quad (s,i) \in V. \tag{11}$$

- For a binary configuration z, the corresponding multi-label configuration x (denoted as x(z)) is found as:

$$x_s = 1 + \sum_{i \in \tilde{\mathcal{L}}} z_{(s,i)}. \tag{12}$$

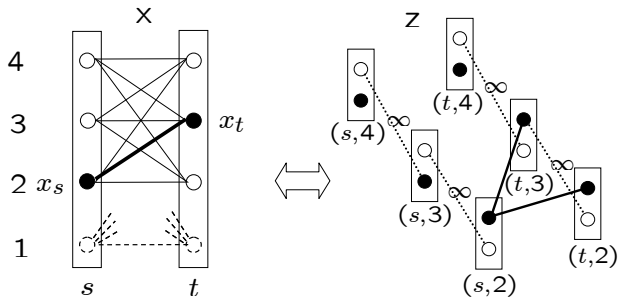


Figure 1. Converting multi-label problems into binary ones. Left: an interaction pair $st \in \mathcal{E}$ of the multi-label energy function; a labeling x is shown by black circles; lowest labels are dashed since all weights in $\hat{\theta}$ associated with them are 0. Right: binary variables $z_{(s,i)}, z_{(t,j)}$ used for encoding the multi-label problem; labeling z(x) is shown by black. Note that if the link $(x_s = 2, x_t = 3)$ is active (left) then two links $[(s,2),(t,2)]$ and $[(s,2),(t,3)]$ are active (right) .

- Binary energy function

$$E(\mathrm{z}|\eta) = H(\mathrm{z}) + \sum_{u \in V} \eta_u z_u + \sum_{uv \in A_{\mathcal{E}}} \eta_{uv} z_u z_v + \eta_{\text{const}}, \tag{13}$$

where weights η are set such that

$$E(\mathrm{z}(\mathrm{x})|\eta) = E(\mathrm{x}|\hat{\theta}) = E(\mathrm{x}|\theta),\ \forall \mathrm{x} \in \mathcal{L}^{\mathcal{V}}. \tag{14}$$

In particular pairwise terms η_{uv} are assigned as

$$\eta_{(s,i),(t,j)} = D_{ij}\hat{\theta}_{st} \quad st \in \mathcal{E},\ i,j \in \tilde{\mathcal{L}}, \tag{15}$$

where $D_{ij}\theta_{st} = \theta_{st}(i,j) + \theta_{st}(i-1,j-1) - \theta_{st}(i,j-1) - \theta_{st}(i-1,j)$. Hard constraints H are as follows:

$$H(\mathrm{z}) = \sum_{uv \in A_{\mathcal{V}}} h(z_u, z_v), \tag{16}$$

where $h(z_{(s,i)}, z_{(s,i-1)}) = 0$ if $z_{(s,i)} \le z_{(s,i-1)}$ and ∞ otherwise (see Fig. 1). Hard constraints ensure that any z with finite energy is in the form (11).

It is already known that the above defined transformation can be used together with st-mincut algorithms to efficiently and exactly solve lattice-submodular multi-label problems. In this paper, we broaden its applicability by showing how it can be used in conjunction with roof-duality to obtain partial optimal solutions for general problems. An important aspect of the transformation is that (11) depends on the ordering of $\mathcal{L}$. This will lead to certain limitations in the sequel. An interesting question raised by reviewers is whether it is possible to use a different reduction to binary variables which does not depend on the order. This does not look straightforward. For example, a rather natural reduction suggested by reviewers is to use binary indicator variables $z_{(s,i)} = \delta_{\{x_s = i\}}$

and enforce constraint $\sum_i z_{(s,i)} \leq 1$ via $K(K-1)/2$ hard pairwise terms and constraints $\sum_i z_{(s,i)} \geq 1$ by adding sufficiently large negative value to all unary terms. Unfortunately, the LP relaxation of the resulting binary problem can be shown to be degenerate (see (Shekhovtsov et al., 2008)).

3.2. Multi-label QPBO

Let (G, K, θ) define a multi-label energy minimization problem, and $(N, \mathbb{B}, \eta)$ define the corresponding binary energy minimization problem. The LP-relaxation of the multi-label problem is defined as:

$$\min_{\mu \in \Lambda_{G,\mathcal{L}}} \langle \mu, \theta \rangle \qquad \text{(LP-1)}$$

while that of the binary problem defined as:

$$\min_{\nu \in \Lambda_{N,\mathbb{B}}} \langle \nu, \eta \rangle. \qquad \text{(LP-2)}$$

We attempt minimization of $E(\mathrm{x}|\theta)$ by applying QPBO and its extensions -P, -I (Boros et al., 2006; Rother et al., 2007) to the binary energy $E(\cdot|\eta)$. We call the new methods *multi-label* QPBO (-P,-I), or short MQPBO (-P,-I). As the original QPBO methods efficiently solves LP-2, it is important to see how LP-2 is related to the original problem $\min_{\mathrm{x}} E(\mathrm{x}|\theta)$ and to its relaxation LP-1. The following results apply.

Statement 1. Let $(\mathrm{z}^{\min}, \mathrm{z}^{\max})$ define strong persistency for $E(\cdot|\eta)$ such that $E(\mathrm{z}^{\min}) < \infty$ and $E(\mathrm{z}^{\max}) < \infty$. Then any optimal configuration $\mathrm{x} \in \operatorname{argmin}_{\mathcal{L}^{\mathcal{V}}} E(\cdot|\theta)$ must satisfy

$$\mathrm{x}^{\min} \leq \mathrm{x} \leq \mathrm{x}^{\max}, \qquad (17)$$

where $\mathrm{x}^{\min} = \mathrm{x}(\mathrm{z}^{\min})$, $\mathrm{x}^{\max} = \mathrm{x}(\mathrm{z}^{\max})$.

The proof (Shekhovtsov et al., 2008) is simply by using the relation (12).

Thus for each variable s there is an interval of labels $[x_s^{\min}, x_s^{\max}]$ outside of which no label may be selected in an optimal solution. All labels outside the interval may therefore be safely discarded. As is seen, the ordering of $\mathcal{L}$ turns to be very important. While in many practical application there is a natural ordering defined, generally, constraints in the form of intervals derived from arbitrary ordering may be very weak.

A pairwise term θ_{st} is called submodular (resp. supermodular) if

$$D_{ij}\theta_{st} \leq 0 \text{ (resp. } \geq 0 \text{)}, \quad i, j = 2 \ldots K. \qquad (18)$$

Theorem 1. *If the term θ_{st} is either submodular or supermodular for each edge $st \in \mathcal{E}$, the relaxations LP-1 and LP-2 coincide, and there exist a mapping between their optimal solutions.*

Proof sketch. We have already used the mapping (11) to relate integral solutions of the two problems such that the energy is preserved. It is quite straightforward to extend it to an injective mapping Π of relaxed labelings μ to relaxed labelings ν, preserving the associated primal costs of LP-1 and LP-2. However, inverting this mapping is not always possible, so there might be a solution ν of LP-2, for which there is no corresponding solution μ of LP-1. Under conditions of the theorem a correction to a solution ν can be applied such that it remains optimal and has a preimage in the mapping Π feasible to LP-1. See details in (Shekhovtsov et al., 2008). □

Corollary 1. It is known that LP-2 can be solved using a network flow algorithm (Hammer et al., 1984). This implies that for a subclass of problems (defined by conditions of the above theorem) there exist **efficient** fully combinatorial algorithms to solve LP-1, which is an improvement over, *e.g.*, message passing algorithms such as TRW-S. But currently, we do not see applications for the case where a part of interactions is submodular and the other part is supermodular.

The next result shows that strong persistency constraints (17) can be also extended to relaxed labelings:

Theorem 2. *Let $\mathrm{x}^{\min} = \mathrm{x}(\mathrm{z}^{\min})$, $\mathrm{x}^{\max} = \mathrm{x}(\mathrm{z}^{\max})$ be the output of MQPBO, and let $\mu \in \Lambda_{G,\mathcal{L}}$ be an optimal solution of LP-1. Then $\mu_{s;i} = 0$ for labels i outside the interval $[x_s^{\min}, x_s^{\max}]$ for all $s \in \mathcal{V}$.*

Proof sketch. Assume μ violates constraints of the theorem. Let then $\nu = \Pi\mu$. For the binary problem it follows from the roof duality that a "truncated" solution, $\bar{\nu}$ may be constructed, which has non-zero weights $\bar{\nu}_u(a)$ only for labels $a \in \mathbb{B}$ such that $z_u^{\min} \leq a \leq z_u^{\max}$, $u \in V$ and such that the primal cost is decreased: $\langle \bar{\nu}, \eta \rangle < \langle \nu, \eta \rangle$. It can then be mapped back to a solution $\bar{\mu} = \Pi^{-1}\bar{\nu}$ of LP-1 of a strictly lower cost than μ. This contradicts to the optimality of μ. See details in (Shekhovtsov et al., 2008). □

The theorem shows that LP-1 never selects nodes which are rejected by LP-2, this may be useful in the analysis of the algorithms related to LP-1.

4. Enhanced Algorithms

In this section we discuss in more detail the "probing" and "improve" versions of MQPBO and show how they could be used in combination with other algorithms for minimization of multi-label energy functions.

MQPBO-P. This enhancement applies the probing technique (Boros et al., 2006; Rother et al., 2007) to binarized energy functions. It computes stronger constraints of the form (17) on the set of optimal config-

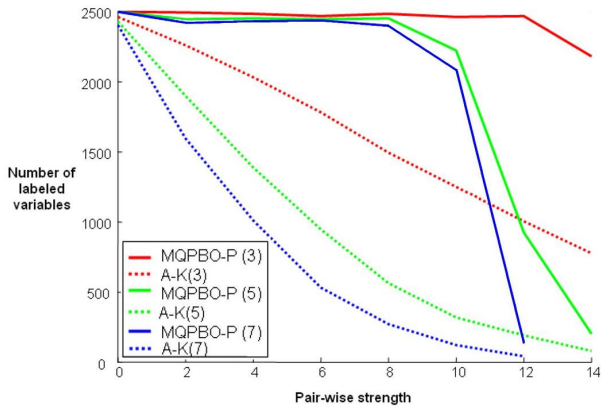


Figure 2. The number of labelled variables in the partially optimal solutions obtained by using MQPBO-P and the algorithm (Kovtun, 2003) (denoted by A-K). Results are shown for energy functions involving variables taking 3, 5 and 7 labels.

urations. Our results show that these constraints allow us to isolate the optimal labels for many random variables of the original energy function. In fact for certain energy functions we obtained the global minimum configuration. If latter is not the case then constraints (17) lead to a simplified minimization problem with a smaller or *restricted* solution space, which can be further approached by any other minimization algorithms.

As was mentioned above, MQPBO, and hence the probing extension, is not invariant to permutations of labels. While we are using a fixed ordering in all our experiments, the method could be potentially run under different orderings thus extracting multiple constraints.

MQPBO-P + X. Similar to QPBO+X (Rother et al., 2007), restriction of the energy function obtained by MQPBO(-P) is then minimized using any other minimization algorithm **X**. In our experiments we used max-product BP (Pearl, 1988), TRW-S (Kolmogorov, 2006) and α-expansion algorithm (Boykov et al., 2001).

MQPBO-I. By using QPBO-I (Rother et al., 2007) on the binarized problem any complete labelling of the multi-label MRF can be updated such that its energy never increases. This procedure can be seen as a local search algorithm.

5. Experimental Results

We now provide the results of using our method to minimize energy functions encountered in computer vision problems. To get a good understanding of the performance of our method we also tested it on synthetic energy functions.

	Pairwise Strength						
Truncation	λ=0	λ=2.4	λ=4.8	λ=7.2	λ=9.6	λ=12.0	λ=14.4
T = 6 (convex)	0	0	0	0	0	0	0
T = 5	0	0	0	0	0	0	0
T = 4	0	0	0	0	0	3	6
T = 3	0	3	4	14	22	26	30
T = 2	0	3	11	20	42	329	1971
T = 1 (Potts)	0	8	19	23	398	2336	2488

Figure 3. Effect of non-convexity: the number of unlabelled variables in the MQPBO-P solution for different values of pairwise strength λ and truncation T.

Synthetic Problems. The energy functions corresponding to the synthetic problems contained 50×50 multi-label variables which interacted under a 4-connected neighborhood. We used different numbers of labels and strengths of pairwise potentials in our experiments. Unary potentials $\theta_s(x_s)$ are sampled uniformly in $\{0,1\ldots100\}$. The pairwise potentials had the form of a linear truncated model $\theta_{st}(x_s,x_t) = \frac{\lambda}{T}\min(|x_s - x_t|, T)$, where T is the truncation and λ is the pairwise strength. Fig. 2 shows comparison of MQPBO-P to (Kovtun, 2003), truncation T is fixed to 1. It is seen that the proposed method labels more variables for a range of parameters. To study the effect of non-convexity, we varied truncation T and strength λ of the pairwise terms (Fig. 3). The number of labels in this case is fixed to 7. Our experiments show

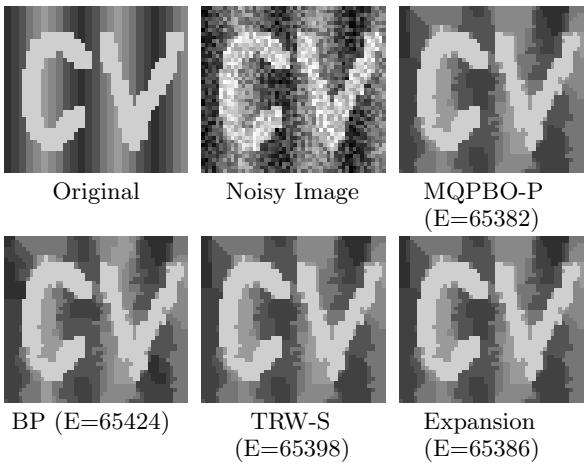


Figure 4. Image Denoising: results obtained by different energy minimization algorithms. The energy used for this experiment has $|\mathcal{L}_I| = 7$, $\gamma = 14$ and $T = 4$. Results are annotated with their respective energy costs. All algorithms were run untill convergence. Observe that MQPBO-P labels all variables and thus obtains the globally optimal solution for this energy.

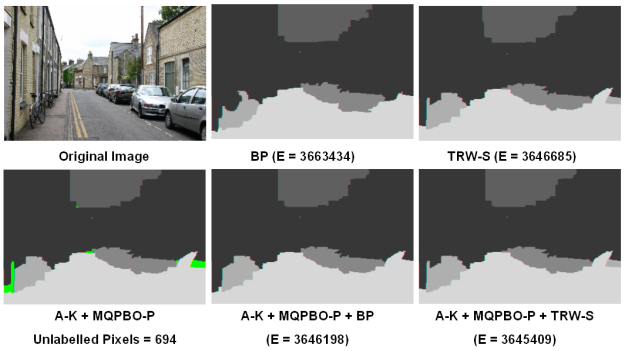


Figure 5. Object Segmentation and Recognition using (Shotton et al., 2006). The first row contains an image from the MSRC (Shotton et al., 2006) dataset and the results of running BP and TRW-S on the full energy. Different levels of gray denote different object classes such as "sky" and "road". The partially optimal solution obtained by running MQPBO-P and A-K (Kovtun, 2003) is shown in the bottom row, left image. Unlabeled pixels are shown in green. This solution is used to obtain a restricted energy. Running BP (bottom, middle) and TRW-S (bottom, right) on the restricted energy gives solutions with lower energy. Hence running A-K + MQPBO-P + "X" is better than running "X" alone. In all cases BP and TRW-S were run for 50 iterations.

that MQPBO-P finds the globally optimal solution of energy functions where the pairwise term is small in magnitude or is nearly *convex*. As expected, the number of variables labelled by the algorithm decrease with increase in the strength and non-convexity of the pairwise terms (Fig. 3).

Image Denoising. We now test the MQPBO-P algorithm on the problem of image denoising and restoration. There is a random variable for each pixel in the image. The label set for the problem is the set of intensities $\mathcal{L}_I$ the pixels can take. The unary cost for taking a particular label (intensity) is defined as $\theta_s(x_s) = |I_s - \phi(x_s)|$ where I_s is the intensity of the pixel s in the image, and the function ϕ maps the labels to their corresponding intensity levels. The pairwise terms of the energy are defined as: $\theta_{st}(x_s, x_t) = \gamma \min(|x_s - x_t|, T)$ where γ is a model parameter and T is the truncation used. Our experiments showed that the MQPBO-P algorithm performed quite well on the energy function and in some cases obtained the globally optimal solution which was not achieved by other energy minimization algorithms (see Fig. 4).

Object based Segmentation. We now show the results of using the MQPBO-P method for restricting the energy functions. Our results show that the restriction significantly reduces the number of variables and makes them amenable for minimization using other algorithms. We test the algorithm on the problem of object segmentation and recognition. We use the energy function formulated in (Shotton et al., 2006). The binarized energy function corresponding to this problem has around 10^7 variables and running MQPBO-P on it directly is quite time consuming. To reduce the size of the problem we first run the partial optimality algorithm described in (Kovtun, 2003), referred to as A-K. MQPBO-P is run on the restriction obtained using A-K. This combined procedure leaves 694 variables unlabelled. The results are shown in Fig. 5.

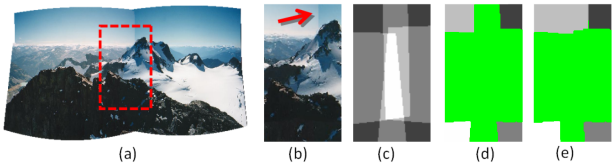


Figure 6. In this application we are stitching together four images (already rectified) to a panoramic image. Running alpha expansion gives result (a) and a zoom-in (dashed rectangle) is shown in (b). The red arrow indicates a visible cut. Image (c) shows the number of possible labels for this image area, where white means all four images overlap and very dark means only one image is possible for those pixels. When applying MQPBO and MQPBO-P to this image portion we obtain labelling (d) and (e) respectively, where green means unlabeled and the different gray scales represent different labels. We see that MQPBO-P is able to label more pixels than MQPBO. It is also worth noting that the visible cut in (b) is indeed the global minimum which can be seen from labelling (e).

Image Stitching. Finally, we investigated an application where MQPBO-P shows its limitations. For panoramic stitching the unary terms are either zero or infinity, depending on the presence or absence of an image, and consequently the pairwise terms dominate the energy. We use the panoramic stitching formulation from (Agarwala et al., 2004) where pairwise terms model the visibility of a transition, different for each pair of images. The results are discussed in fig 6.

6. Conclusions

This paper addressed the problem of minimizing non-submodular multi-label energy functions. These are used extensively in computer vision and are generally NP-hard to minimize. We present a method for obtaining partially optimal solutions of such energies derived from roof-dual based methods for binary energy functions. We give new theoretical insights in the underlying LP relaxation, being efficiently solved by the method. Although this work is mainly theoretical in nature we hope to inspire people to use these ideas to develop better optimization algorithms. We also believe that this approach is useful for other impor-

tant problems in computer vision such as MRF/CRF learning.

Acknowledgments

We would like to thank the anonymous reviewers for their useful comments. A. Shekhovtsov thanks the European Commission grant 215078 (DIPLECS) and Czech government grant MSM6840770038 for support. V. Kolmogorov thanks EPSRC for support. P. Torr thanks EPSRC and PASCAL, EU for support.

References

Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S. M., Colburn, A., Curless, B., Salesin, D., & Cohen, M. F. (2004). Interactive digital photomontage. *ACM Trans. Graph.*, *23*, 294–302.

Boros, E., & Hammer, P. (2002). Pseudo-boolean optimization. *Discrete Applied Mathematics*, 155–225.

Boros, E., Hammer, P. L., & Sun, X. (1991). *Network flows and minimization of quadratic pseudo-Boolean functions* (Technical Report RRR 17-1991). RUTCOR.

Boros, E., Hammer, P. L., & Tavares, G. (2006). *Preprocessing of unconstrained quadratic binary optimization* (Technical Report RRR 10-2006). RUTCOR.

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *PAMI*, *23*, 1222–1239.

Desmet, J., Maeyer, M. D., Hazes, B., & Lasters, I. (1992). The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, *356*, 539–542.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2000). Efficient matching of pictorial structures. *CVPR* (pp. 66–73).

Hammer, P., Hansen, P., & Simeone, B. (1984). Roof duality, complementation and persistency in quadratic 0-1 optimization. *Math. Programming*, 121–155.

Hammer, P. L. (1965). Some network flow problems solved with pseudo-Boolean programming. *Operation Research*, *13*, 388–399.

Ishikawa, H. (2003). Exact optimization for Markov random fields with convex priors. *PAMI*, *25*, 1333–1336.

Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *PAMI*, *28(10)*, 1568–1583.

Kolmogorov, V., & Rother, C. (2007). Minimizing non-submodular functions with graph cuts – a review. *PAMI*, *29*, 1274–1279.

Kolmogorov, V., & Zabih, R. (2004). What energy functions can be minimized via graph cuts? *PAMI*, *26*, 147–159.

Kolmogorov, V. N., & Wainwright, M. J. (2005). On optimality of tree-reweighted max-product message-passing. *Appeared in Uncertainty in Artificial Intelligence*.

Komodakis, N., Paragios, N., & Tziritas, G. (2007). MRF optimization via dual decomposition: Message-passing revisited. *ICCV* (pp. 1–8).

Komodakis, N., & Tziritas, G. (2005). A new framework for approximate labeling via graph cuts. *ICCV* (pp. 1018–1025).

Koval, V., & Schlesinger, M. (1976). Two-dimensional programming in image analysis problems. *Automatics and Telemechanics*, *2*, 149–168. In Russian.

Kovtun, I. (2003). Partial optimal labeling search for a NP-hard subclass of (max, +) problems. *DAGM-Symposium* (pp. 402–409).

Kovtun, I. (2004). *Image segmentation based on sufficient conditions of optimality in NP-complete classes of structural labelling problem*. Doctoral dissertation. In Ukrainian.

Lauritzen, S. L. (1998). *Graphical models*. No. 17 in Oxford Statistical Science Series. Oxford Science Publications.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Rother, C., Kolmogorov, V., Lempitsky, V., & Szummer, M. (2007). Optimizing binary MRFs via extended roof duality. *CVPR* (pp. 1–8).

Schlesinger, D. (2007). Exact solution of permuted submodular minsum problems. *EMMCVPR* (pp. 28–38). Springer.

Schlesinger, D., & Flach, B. (2006). *Transforming an arbitrary minsum problem into a binary one* (Technical Report TUD-FI06-01). Dresden University of Technology.

Schlesinger, M. (1976). Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika, Kiev*, *4*, 113–130. In Russian.

Schlesinger, M. I., & Flach, B. (2000). Some solvable subclasses of structural recognition problems. *Czech Pattern Recognition Workshop*.

Schlesinger, M. I., & Giginyak, V. V. (2007). Solution to structural recognition (MAX,+)-problems by their equivalent transformations. *Control Systems and Computers*, *1,2*.

Shekhovtsov, A., Kolmogorov, V., Kohli, P., Hlavac, V., Rother, C., & Torr, P. (2008). *LP-relaxation of binarized energy minimization*. Research Report CTU–CMP–2007–27). Czech Technical University.

Shotton, J., Winn, J. M., Rother, C., & Criminisi, A. (2006). *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. *ECCV (1)* (pp. 1–15).

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M. F., & Rother, C. (2006). A comparative study of energy minimization methods for Markov random fields. *ECCV (2)* (pp. 16–29).

Wainwright, M., Jaakkola, T., & Willsky, A. (2003). Exact MAP estimates by (hyper)tree agreement. In *Advances in neural information processing systems 15*, 809–816.

Werner, T. (2007). A linear programming approach to max-sum problem: A review. *PAMI*, *29*, 1165–1179.

Space-indexed Dynamic Programming: Learning to Follow Trajectories

J. Zico Kolter KOLTER@CS.STANFORD.EDU
Adam Coates ACOATES@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Yi Gu GUYINET@STANFORD.EDU
Charles DuHadway DUHADWAY@STANFORD.EDU
Computer Science Department, Stanford University, CA 94305

Abstract

We consider the task of learning to accurately follow a trajectory in a vehicle such as a car or helicopter. A number of dynamic programming algorithms such as Differential Dynamic Programming (DDP) and Policy Search by Dynamic Programming (PSDP), can efficiently compute non-stationary policies for these tasks — such policies in general are well-suited to trajectory following since they can easily generate different control actions at different times in order to follow the trajectory. However, a weakness of these algorithms is that their policies are *time-indexed*, in that they apply different policies depending on the current time. This is problematic since 1) the current time may not correspond well to where we are along the trajectory and 2) the uncertainty over states can prevent these algorithms from finding any good policies at all. In this paper we propose a method for *space-indexed* dynamic programming that overcomes both these difficulties. We begin by showing how a dynamical system can be rewritten in terms of a spatial index variable (i.e., how far along the trajectory we are) rather than as a function of time. We then use these space-indexed dynamical systems to derive space-indexed version of the DDP and PSDP algorithms. Finally, we show that these algorithms perform well on a variety of control tasks, both in simulation and on real systems.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

1. Introduction

We consider the task of learning to accurately follow a trajectory, for example in a car or helicopter. This is one of the most basic and fundamental problems in reinforcement learning and control. One class of approaches to this problem uses dynamic programming. These algorithms typically start at the last time-step T of a control task, and compute a simple (say, linear) controller for that time-step. Then, they use dynamic programming to compute controllers for time-steps $T-1$, $T-2$ and so on down to time-step 1. Some examples of algorithms in this family include (Jacobson & Mayne, 1970; Bagnell et al., 2004; Atkeson & Morimoto, 2003; Lagoudakis & Parr, 2003), and all of them output time-varying/non-stationary policies that choose the control action as a function of time. Given that following a trajectory requires one to choose very different control actions at different parts of the trajectory — for example, the controls while driving a car on a straight part of the trajectory are very different from the controls needed during a turn – these dynamic programming algorithms therefore initially seem well-suited for trajectory following.

However, a weakness in the naive dynamic programming approach is that the control policies are *time-indexed*. That is, these algorithms output a sequence of controllers $\pi_1, \pi_2, \ldots, \pi_T$ and execute controller π_t at time t. However, as time passes, the uncertainty over the state increases, and this can greatly degrade controller performance. For example, suppose we are driving a car around a track with both straight and curved portions, and suppose that the controller at time t assumed the car was on a curved portion. If, due to the natural stochasticity of the environment, the car was actually on a straight portion of the track at this time, the resulting controller would perform very poorly, and this problem increases over time. This

problem can be alleviated slightly be "re-indexing" the controllers by state during execution. That is, at time t we do not necessarily execute controller π_t, but instead we examine *all* the controllers $\pi_1, \ldots, \pi_T$, and execute the controller whose corresponding state is closest to the current state — several variations on this approach exist, and we will discuss them further in Section 5. However, there are two fundamental limitations of this general method. First, because we are executing a different policy from the one learned by dynamic programming, it is difficult to provide any of the performance guarantees that often accompany the purely time-indexed dynamic programming algorithms. Second, and more fundamentally, the uncertainty over states in the distant future often make it extremely difficult to learn *any* good policy using the time-indexed algorithms. This means that regardless of how we re-index the controllers during execution, we are unlikely to obtain good performance.

In this paper we propose a method for *space-indexed* dynamic programming that addresses both these concerns. More precisely, we will define a spatial index variable d that measures how far we have traveled along the target trajectory. Then, we will use policies π_d that depend on d — where we are along the trajectory — rather than the current time t. In order to learn such policies, we define the notion of a *space-indexed dynamical system*, and show how various dynamical systems can be rewritten such that their dynamics are indexed by d instead of by time t. This then allows us to extend various dynamic programming algorithms to produce space-indexed policies — in particular, we develop a space-indexed versions of the Differential Dynamic Programming (DDP) (Jacobson & Mayne, 1970) and Policy Search by Dynamic Programming (PSDP) algorithms (Bagnell et al., 2004). Finally, we successfully apply this method to several control tasks, both in simulation and in the real world.

The remainder of this paper is organized as follows. In Section 2 we show how to transform a standard (time-indexed) dynamical system into a space-indexed dynamical system. Using this transformation, in Section 3 we develop space-indexed versions of the DDP and PSDP algorithms. In Section 4 we present experimental results on several control tasks. Finally, in Sections 5 and 6 we discuss related work and conclude the paper.

2. Space-indexed Dynamical Systems

Standard dynamic programming algorithms are very efficient because they know in advance that the policies $\pi_1, \ldots, \pi_T$ will be executed in a certain sequence (and that each policy will be executed only once), and can thus solve for them in reverse order. The key difficulty of generalizing a dynamic programming algorithm to the space-indexed setting is that it is difficult to know in advance where in space (i.e., how far along the trajectory) the vehicle will be at each step, and thus which space-indexed policy will be executed when. For example, if the vehicle is currently at space-index d, then there is no guarantee that executing policy π_d for one time-step will put the vehicle in space-index $d+1$. But if π_d might be executed multiple times before switching to π_{d+1}, then in general its parameters cannot be solved for in closed form during the dynamic programming backup step, and require some complex policy search instead. In this section we discuss a method for addressing this problem. Specifically, we will rewrite the dynamics of a system so that the states and transitions are indexed by the spatial-index variable d rather than by the time t.

Suppose we are given a general non-linear dynamics model in the form of a (possibly stochastic) differential equation $\dot{s} = f(s, u)$, where $s \in \mathbb{R}^n$ denotes the state vector $u \in \mathbb{R}^m$ denotes the control input, and $\dot{s}$ denotes the derivative of the state vector with respect to time. While some classical control algorithms operate directly on this differential equation, a common technique in reinforcement learning and control is to create a discrete-time model of the system

$$s_{t+1} = F(s_t, u_t) + w_t$$

by numerical integration, where s_t and u_t denote the state and input at time t respectively, and w_t is a zero-mean IID noise term (typically taken to be Gaussian with some prespecified covariance, for example). A simple but very common method for achieving this discretization is by Euler integration. In this case the state evolves as

$$s_{t+\Delta t} = s_t + f(s_t, u_t)\Delta t + w_t$$

where Δt is the integration time constant (the variance of w_t will scale linearly with the time constant as well). Note that even though the system evolves in continuous time, by making the decision to model it as a discrete-time system, we have made a decision to explicitly represent the state only at certain *instants* in time ($t = \Delta t, t = 2\Delta t, \ldots$).

When transitioning to a space-indexed dynamical system, we instead will explicitly represent the state only when it is at certain *points* along the trajectory. We begin by representing the time-indexed state as $s_t = [x_t, \theta_t]^T$, where $x \in \mathbb{R}^p$ represents what we refer to as the *spatial portions* of the state (in this paper we typically consider the spatial portions of the state to be the 2D or 3D position). Now, assume we are

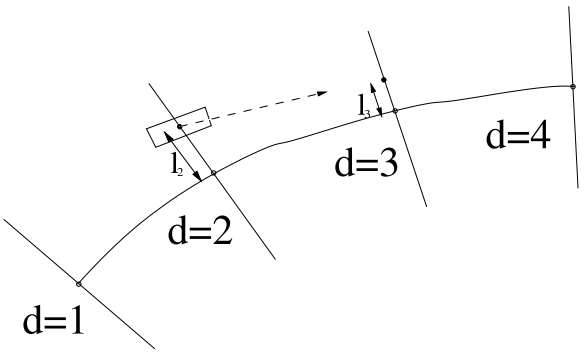


Figure 1. Figure illustrating space-indexed dynamics.

given a target trajectory in the space of x, such as the curved path shown in Figure 1. We choose a total of D discrete points along the trajectory, and designate the target state at these points as $x_1^\star, x_2^\star, \ldots, x_D^\star$. In the space-indexed dynamical system, we will explicitly represent the state only when the state lies on a hyperplane which is orthogonal to the target direction of travel and which passes through the one of the target points $x_d^\star$. More formally, we let $\dot{x}_d^\star$ be the instantaneous direction of motion (along the target trajectory) at point d. We will then explicitly represent the state only when $(x - x_d^\star)^T \dot{x}_d^\star = 0$. This situation is depicted in Figure 1.

Because we constrain the state in this manner, our space-indexed state will have a different set of variables as our time-indexed state. In particular, we represent the space-indexed state as $\tilde{s}_d = [t_d, \ell_d, \theta_d]^T$ where $t_d \in \mathbb{R}$ denotes the time of the system, $\ell_d \in \mathbb{R}^{p-1}$ denotes the lateral deviation from the target trajectory — for $x \in \mathbb{R}^p$ a point satisfying the constraint that $(x - x_d^\star)^T \dot{x}_d^\star = 0$ can be represented using $p-1$ dimensions and this gives the lateral deviation term — and θ_d denotes the non-spatial portions of the state as before. The time variable t_d is kept for completeness, but it can be ignored if the policies do not depend on time.

Now we can rewrite the dynamics so that they are indexed by space rather than time; this will give us an equation for computing $\tilde{s}_{d+1}$ from $\tilde{s}_d$. For simplicity, we develop an Euler integration-like method, but the technique could also be extended to higher-order numerical integration methods. Rather than simulating the system forward by a fixed time step, we *solve* for Δt such that the next state will lie exactly on the $d+1$ plane. Temporarily ignoring the noise term, we solve

$$(\dot{x}_{d+1}^\star)^T (x + \dot{x}\Delta t - x_{d+1}^\star) = 0$$

for Δt. Note that a positive solution for Δt may not always exist, but it typically does except in degenerate cases, when the vehicle starts moving perpendicular or backward with respect to the desired direction, and this is unlikely given any reasonable controller. When $\Delta t > 0$ does exist, we can compute $s_{t+\Delta t} = s_t + f(s_t, u_t)\Delta t$ and use this to find the next space-indexed state

$$\tilde{s}_{d+1} = [t_d + \Delta t, \ell_{d+1}, \theta_{t+\Delta t}]^T$$

where ℓ_{d+1} is $x_{t+\Delta t} - x_{d+1}^\star$ expressed in the $\mathbb{R}^{p-1}$ subspace defined by the plane through $x_{d+1}^\star$. This gives us our final space-indexed simulator in the form

$$\tilde{s}_{d+1} = \tilde{F}(\tilde{s}_d, u_d) + \tilde{w}_d \tag{1}$$

where $\tilde{w}_d$ is a noise term. Although the distribution of $\tilde{w}_d$ is in general quite complex, we can apply methods from stochastic calculus to efficiently draw samples from this distribution. However, in practice we find that a simpler approximate approach works just as well: we compute $s_{t+\Delta t}$ as above, assuming no noise, then add noise as in the time-indexed model. This will result in a point that may no longer lie exactly on the space-index plane, so lastly we form the line between the states s_t and $s_{t+\Delta t}$ and let $\tilde{s}_{d+1}$ be the point where this line intersects the $d+1$ plane.

3. Space-indexed Dynamic Programming

In this section we use the techniques presented in the previous section to develop space-indexed versions of the Differential Dynamic Programming (DDP) and Policy Search by Dynamic Programming (PSDP) algorithms. We begin by defining notation. Let S be the state space and A be the action space (so that in the context of the dynamical system above, $S = \mathbb{R}^n$ and $A = \mathbb{R}^m$). Since, as described above, there exists a one-to-one mapping from time-indexed states to space-indexed states, all the quantities below can be equivalently expressed in terms of the space-indexed state. A reward function is a mapping $R : S \to \mathbb{R}$ and a policy is a mapping $\pi : S \to A$. Given a non-stationary sequence of policies $(\pi_t, \ldots, \pi_T)$ we define the value function

$$V_{\pi_t, \ldots \pi_T}(s) = \tfrac{1}{T} E[\textstyle\sum_{i=t}^T R(s_i) | s_t = s; (\pi_t, \ldots, \pi_T)].$$

3.1. Space-indexed DDP

We first review (time-indexed) DDP briefly. DDP approximates the dynamics and cost function of a system along a specific sequence of states. Given an initial controller π_{init}, DDP simulates the system to generate a sequence of states $s_1, \ldots, s_T$. It then linearizes the dynamics around these points to obtain a time-varying linear dynamical system, and forms a second-order (quadratic) approximation to the reward function. This system can then be solved by the Linear

Quadratic Regulator (LQR) algorithm (Anderson & Moore, 1989), which results in a new controller and a new sequence of states. This process is repeated until convergence.

Space-indexed DDP proceeds in the same manner. Given some initial controller π_{init} and space-indexed dynamical system of the form (1)[1], we simulate the system forward for D space-indexes, resulting in a set of states $\tilde{s}_1, \ldots, \tilde{s}_D$. Using our dynamics model, we form the first order Taylor expansion of the dynamics at each point along the trajectory, which results in the (space-indexed) linearized dynamics:

$$\tilde{s}_{d+1} = A_d \tilde{s}_d + B_d u_d.$$

As in standard DDP, we form a quadratic approximation of the reward function $R_d(\tilde{s}) = -\tilde{s}^T Q_d \tilde{s}$, where Q_d is some (usually PSD) matrix. This reduces the problem to an LQR problem, which can be solved efficiently using a backward recursion.

3.2. Space-indexed PSDP

We now briefly review PSDP. As input, PSDP takes a time horizon T, a restricted policy class Π, and a sequence of baseline distributions over the states space $\mu_1, \ldots, \mu_T$, where we can informally think of μ_t as providing a distribution over which states would be visited at time t by a "good" policy. Given policies $\pi_{t+1}, \ldots, \pi_T$, PSDP computes (or approximates via Monte-Carlo sampling and parameter search)

$$\pi_t = \arg\max_{\pi \in \Pi} E_{s \sim \mu_t}[V_{\pi, \pi_{t+1} \ldots \pi_T}(s)]. \qquad (2)$$

By starting with $t = T$ and proceeding down to $t = 1$, the algorithm is able to generate a sequence of policies that can perform well on the desired task. The space-indexed version of PSDP proceeds exactly as above, replacing the time t with the space index d and using the space-indexed simulator to generate the Monte-Carlo samples.

Just as in the time-indexed version, the space-indexed version of PSDP comes with nontrivial performance guarantees, formalized by the theorem below. The theorem follows immediately from the equivalent theorem for the time-indexed version of PSDP, and from the fact that the space-indexed dynamics and reward function do not depend on time.

Theorem 3.1 [following (Bagnell et al., 2004)] *Suppose $\pi = (\pi_1, \ldots, \pi_D)$ is a policy returned by an ϵ-approximate version of state-indexed PSDP where on each step the algorithm obtains π_d such that*

$$E_{s \sim \mu_d}[V_{\pi_d, \pi_{d+1}, \ldots, \pi_D}(s)] \geq \arg\max_{\pi \in \Pi} E_{s \sim \mu_d}[V_{\pi, \pi_{d+1} \ldots \pi_D}(s)] - \epsilon$$

Then for all $\pi_{\text{ref}} \in \Pi^D$,

$$V_\pi(s_0) \geq V_{\pi_{\text{ref}}}(s_0) - D\epsilon - D d_{\text{var}}(\mu, \mu_{\pi_{\text{ref}}})$$

where μ is the baseline distribution over space-index states (without the time component) provided to SI-PSDP, d_{var} denotes the average variational distance, and $\mu_{\pi_{\text{ref}}}$ is the state distribution induced by π_{ref}.

This bound not only provides a performance guarantee for the space-indexed PSDP algorithm, it also helps to elucidate the advantage of space-indexing over time-indexing. The bound implies that to make PSDP and SI-PSDP perform as well as possible, it would be best to provide them with $\mu_{\pi^\star}$, the baseline distribution of the optimal controller, as the baseline distribution. But for time-indexed PSDP, the natural stochasticity of the environment can cause this distribution to be highly spread out over the state space, even for the optimal policy. Therefore, when performing the maximization (2), it is likely that *no* policy in the class will perform very well, since this would require a policy that could operate well over many different regions of the state space. Thus, regardless of whether or not we re-index the resulting controllers by state during execution, the time-indexed version of PSDP would fail to find a good policy. In contrast, if we are doing a good job following the trajectory, then we would expect that the distribution over states at each space-index would be much tighter, allowing the space-indexed PSDP to perform much better.

4. Experiments

4.1. Autonomous Driving

We begin by considering the problem of autonomously and accurately following a trajectory with a car, such as that shown in Figure 4. Our first set of experiments were carried out in a simulator of the vehicle, built following (Rossetter & Gerdes, 2002) (with model parameters such as the vehicle dimensions, total weight, etc). To follow the desired trajectory, we applied the space-indexed DDP algorithm described above.

Prior to the work presented in this paper, significant engineering effort went into a hand-designed trajectory following controller; this was an initial version of the controller described in (Hoffmann et al., 2007), which was a hand-optimized, linear, regulation controller that computes its actions as a function of state

[1] For the DDP algorithm, we ignore the noise term $\tilde{w}_d$ because by the principle of certainty equivalence, the optimal controller for LQR does not depend on the variance/magnitude of the noise (Anderson & Moore, 1989).

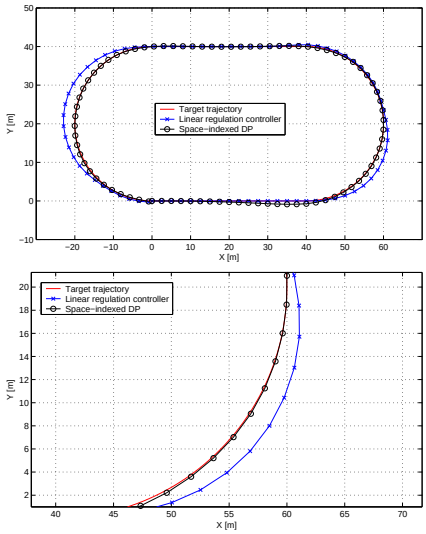


Figure 2. Comparison of the regulation controller and space-indexed controller. The figure below is a magnification of one of the turns.

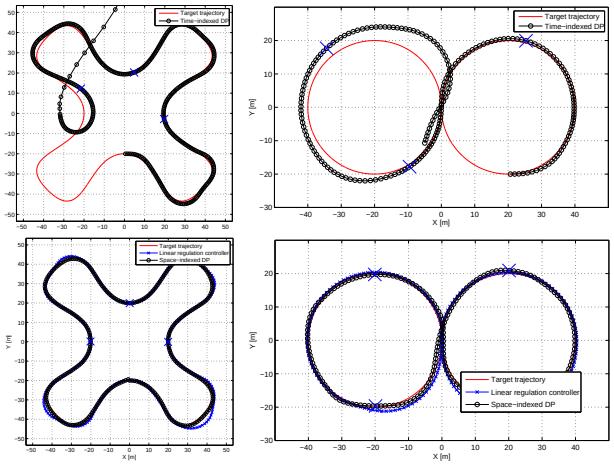


Figure 3. Two example trajectories where a time-indexed controller (above) performs significantly worse than the space-indexed controller (below).

features such as lateral error, orientation error, and so on. We used this controller to generate an initial trajectory for our space-indexed DDP algorithm; however, the results of space-indexed DDP were actually very insensitive to the choice of this initial controller.

Figure 2 shows the performance of the space-indexed DDP algorithm and the hand-tuned controller in simulation, when following an oval-like track at 30mph, along with a magnified view of the show the performance on one of the turns. We see that space-indexed DDP outperforms the hand-tuned controller; our controller has an RMS lateral error 0.26m, whereas the hand-tuned controller's RMS error is 1.18m.

Figure 3 shows a comparison between the performance of space-indexed and time-indexed dynamic programming. Due to the stochasticity of the simulator, the actual state s_t, for large t, is increasingly unlikely to be close to where the t-th step of the linearization occurred. Therefore the linearized approximation is less likely to be an accurate approximation of the "local" dynamics at time t. This is reflected in the figures: the time-indexed controllers initially perform well, but as time passes the controllers start to be executed at incorrect points along the trajectory, eventually leading the vehicle to veer off course. Using the space-indexed controller, however, the vehicle is able to accurately track the trajectory even for an arbitrarily long amount of time. For this relatively simple trajectory following task, re-indexing the time-indexed controllers by their spatial state, as described in the introduction, does perform well. However, as we will demonstrate in the next section, for more complex control tasks this is not the case.

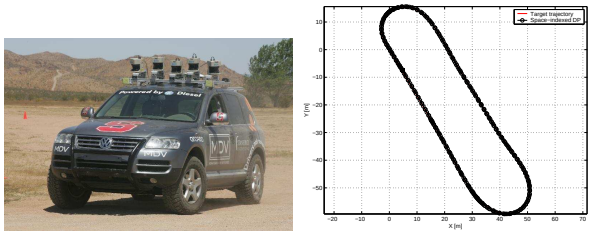


Figure 4. Picture of the vehicle used for experiments (left), and trajectory from a run on the actual car (right).

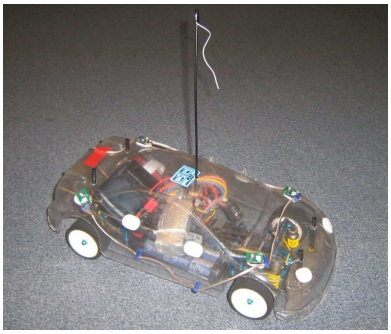


Figure 5. Autonomous RC car.

We also tested our method on the actual vehicle; the vehicle itself is described further in (Thrun & al., 2006). Figure 4 shown a typical result from an actual run on the vehicle moving at 10mph. The RMS error on the actual vehicle was about 0.17m, and the target and actual trajectories are indistinguishable in the figure.

4.2. Autonomous Driving with Obstacles

We next consider the more challenging control task of following a trajectory in the presence of obstacles. For this task we evaluated our methods on an RC car, shown in Figure 5. Since we want to learn a single controller that is capable of avoiding obsta-

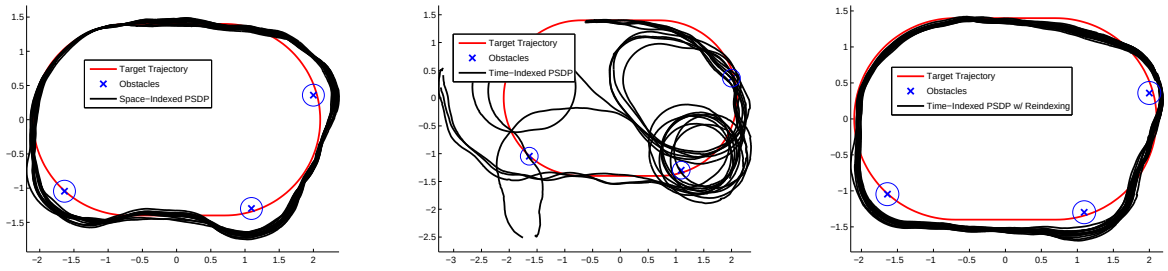


Figure 6. Trajectories taken by the real RC car on a course with obstacles, using space-indexed PSDP (left), time-indexed PSDP (middle), and time-indexed PSDP with re-indexing by space (right).

cles placed at arbitrary points along the trajectory, DDP is a poor algorithm (DDP learns a controller for a single fixed trajectory, but in this task we need to follow different trajectories depending on the location of the obstacles). Therefore, we apply the space-indexed version of PSDP to this task. Videos of the resulting controllers for this task are available at: `http://cs.stanford.edu/~kolter/icml08videos`.

In greater detail, we applied the space-indexed PSDP algorithm as follows. The native action space for the car domain is a two-dimensional input specifying the velocity and steering angle (between -28 degrees and 28 degrees), but for our task we kept the velocity fixed at 1.5 m/s and discretized the commanded steering angle into five equally spaced angles. To generate the initial distribution for PSDP, $\mu_1, \ldots, \mu_D$, we sampled 2000 different trials in a simulator of the car, using a PD controller. Then, for each space index from $D-1$ down to 1, we (approximately) solved the optimization problem (2) by first trying each possible action for each of the 2000 sampled states, then executing the learned controller for all subsequent space-indices to compute the resulting cost of the policy.[2] We then tried to learn the optimal action as a linear function of various state features; in particular, each controller was of the form: $u_{\text{steer}} = \arg\max_i w_i^T \phi(s)$ where $w_i \in \mathbb{R}^n$ is a weight vector learned by a multi-class SVM-like algorithm[3], and $\phi(s) \in \mathbb{R}^n$ is a feature vector. In our setting the features comprised of 1) the x and y location of the car, 2) the sine and cosine of the current car orientation, 3) 16 exponential RBF functions, spaced uniformly around the car, indicating the presence of an obstacle, and 4) a constant term. In addition to the space-indexed version, we also evaluated the performance of a pure time-indexed version, and a time-indexed version where we re-index the controllers as follows: at time t rather than execute the controller π_t, we examine all the controllers $\pi_1, \ldots, \pi_T$ and execute the controller $\pi_{t'}$ with minimum distance from the current state to the mean of the distribution $\mu_{t'}$.

Table 1. Average costs and collision counts for the simulated RC car with obstacles, averaged over 1000 runs.

Algorithm	Cost	Collisions
SI-PSDP	**56.17 ± 0.43**	**51**
TI-PSDP	58.39 ± 0.40	181
TI-PSDP w/ re-indexing	58.19 ± 0.37	212
Hand-tuned PD Controller	58.35 ± 0.34	231

Table 1 shows the average cost incurred and total number of collisions for the different controllers in 1000 simulated trials, where each trial had three randomly placed obstacles on the trajectory. As can be seen, the space-indexed version outperforms all the other variants of the algorithm as well as a hand-tuned PD controller that we previously spent a good deal of time trying to tune. The performance benefits of the space-indexed controller become even more pronounced on the real system. Figure 6 shows typical resulting trajectories from the space-indexed controller, the pure time-indexed controller, and the time-indexed controller with re-indexing. Due to the stochasticity of

[2] We used a cost function (i.e., negative reward), of

$$C(s_d) = \theta_1 \ell^2 + \theta_2 (1 - dist/\theta_3) \mathbf{1}\{dist < \theta_3\}$$

where $dist$ denotes the distance to the nearest obstacle, $\mathbf{1}$ is the indicator function, and $\theta_1, \ldots, \theta_3$ are parameters that trade off the relative cost of deviating from the trajectory and getting close to obstacles. For our experiments, we used $\theta_1 = 1000$, $\theta_2 = 500$ and $\theta_3 = 0.5$.

[3] Algorithmic details: the PSDP algorithm with discrete actions leads to a cost-sensitive, k-class learning problem with examples form $\{\phi(s^{(i)}) \in \mathbb{R}^n, c^{(i)} \in \mathbb{R}^k\}$, $i = 1, \ldots m$ where $\phi(s^{(i)})$ are the features, and $c_j^{(i)}$ represents the cost of classifying example i as belonging to class j. We approximately solve this problem with a support vector machine-like algorithm, which bears some similarly to previous work in cost-sensitive SVM earning (Geibel et al., 2004). The algorithm finds the solution to the optimization problem:

$$\begin{aligned} \min_{w,\xi \geq 0} \quad & \textstyle\sum_{j=1}^k \frac{1}{2}\|w_j\|^2 + C \sum_{i=1}^m \sum_{j,l=1}^k (c_j^{(i)} - c_l^{(i)})^+ \xi_{i,j,l} \\ \text{s.t.} \quad & (w_l - w_j)^T \phi(s^{(i)}) \geq 1 - \xi_{i,j,l} \quad \forall i, j, k. \end{aligned}$$

Figure 7. Tempest autonomous helicopter.

the real domain, the pure time-indexed approach performs very poorly. Re-indexing the controllers helps significantly, but the space-indexed version still performs substantially better (an incurred cost of 49.32 for space-indexed versus 59.74 for time-indexed with re-indexing, and the latter controller will nearly always hit at least one of the obstacles on the track). As seen in the figure, the space indexed version is able to track the trajectory well, while reliably avoiding obstacles.

4.3. Autonomous Helicopter Flight

We also apply these ideas to a simulated autonomous helicopter. This work used a stochastic simulator of the autonomous helicopter shown in Figure 7, and we considered the problem of making accurate, high-speed (5m/s) turns on this helicopter. We applied the space-indexed PSDP algorithm to this task due to the fact that the policy search setting allowed us to greatly restrict the class of control policies π_d under consideration (the space of all control policies for helicopter flight is very large, so we wanted to limit the risk of unexpected behavior). In particular, the "actions" of the controllers corresponded to picking a location of set point $x^\star$ which is then fed into a regulation controller, such as those described in (Bagnell & Schneider, 2001; Ng et al., 2004). Space constraints preclude a full description of the environment and algorithm, but the overall algorithm proceeds as in the previous section.

Figure 8 shows a typical result of applying the space-indexed PSDP algorithm to this task, along with the trajectory taken by a simple linear regulation controller. By varying the set point differently at different points along the trajectory, the space-indexed PSDP algorithm follows the trajectory much more accurately.

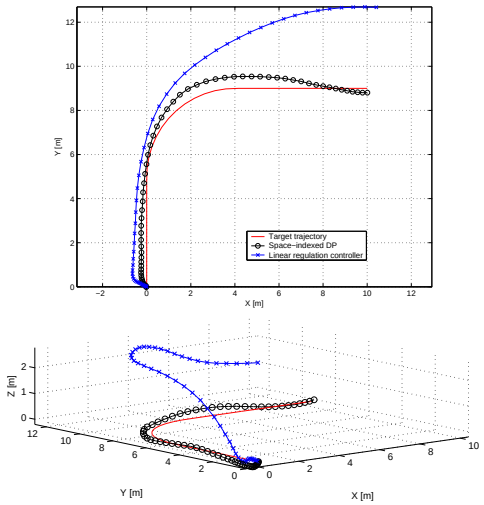


Figure 8. Comparison of the regulation controller and space-indexed controller in the helicopter simulation. The figure below shows the trajectories in three dimensions.

5. Related Work

The idea that a control policy should be dependent on the system's spatial state is by no means a new idea in the reinforcement learning and control literature. In the Markov Decision Process (MDP) formalism (Puterman, 1994), a policy is a mapping from states (which typically describe the *spatial* state of the system) to actions. In light of this observation, many classical dynamic programming algorithms such as value iteration or policy iteration can be viewed as performing dynamic programming on spatial states. However, in high-dimensional, continuous state spaces, the well-known "curse of dimensionality" renders a naive application of these algorithms intractable. Indeed, it is this reality that often motivates the jump to the trajectory following approach, where we want to find polices that perform well along the trajectory in particular.

There are also a number of methods for trajectory following. A common approach is to design a "regulation controller" that can keep the vehicle stable at a specified position $x^\star$. By smoothly moving $x^\star$ along the target trajectory, we can cause the vehicle to move along this path (Franklin et al., 1995; Dorf & Bishop, 2000). This approach works well when the regulation controller has very high bandwidth – i.e., if it can track $x^\star$ almost exactly as it varies — and is successful in application areas such as control of robot arms. But in more general settings in which the actual state x of the vehicle tends to lag well behind changes to $x^\star$, one often ends up manually and laboriously adjusting the regulation controller to try to obtain proper trajectory following performance. There are methods for compensating for this lag such as feedback linearization (Sastry, 1999), and there have also been many methods devised for trajectory following on specific systems (Egerstedt & Hu, 2000; Johnson & Calise, 2002). However, we know of no method for trajectory following in the general case of the nonholonomic, underaccuated vehicles that we consider.

The idea of partitioning the state space into regions, and using different controllers in the different regions, is a common practice in control, and often is referred to as gain-scheduling (Leith & Leithead, 2000). Taken

in the general sense, the algorithm we present in this paper can be viewed as a method for gain-scheduling, though more often the term is used for a particular application of this approach to the contexts of linear parameter varying systems. Such methods typically linearize the dynamical system around certain operating points, learn controllers at each point, and smoothly interpolate between controllers at various locations. However, the focus of this work is often to prove stability of such controllers using Lyapunov theory, and the overall approach is substantially different from what we consider here.

Model predictive control (MPC) (Garcia et al., 1989) (indirectly) addresses the issue of state uncertainty increasing over time, by explicitly computing new controllers at every time step in an online manner. However, MPC is generally orthogonal to the ideas we present here, since one could just as easily use a space-indexed dynamic programming method for the local controller in MPC. Furthermore, MPC can often times be computationally impractical to run real-time. An alternative approach is to use a local control method, such as DDP, in order to estimate the value function along several trajectories, and use these local estimates to build an approximate global model of the value function (Atkeson, 1994; Tassa et al., 2007). However, since these methods employ DDP, which is a time-indexed algorithm, they can potentially suffer the same problems as time-indexed methods in general.

6. Conclusions

In this paper we presented a space-indexed dynamic programming method for trajectory following. We showed how to convert standard time-indexed dynamical systems into equivalent space-indexed dynamical systems, and used this formulation to derive space-indexed versions of two well-known dynamic programming algorithms, DDP and PSDP. Finally, we successfully applied these methods to several control tasks, and demonstrated superior performance compared to their time-indexed counterparts.

Acknowledgments. This work was supported by the DARPA Learning Locomotion program under contract number FA8650-05-C-7261. We thank the anonymous reviewers for helpful comments, Sam Schreiber and Quan Gan for assistance with the RC car, Pieter Abbeel for assistance with the helicopter domain, Mark Woodward, Mike Montemerlo, Gabe Hoffmann, David Stavens and Sebastian Thrun for assistance with the DARPA Grand Challenge vehicle.

References

Anderson, B. D. O., & Moore, J. B. (1989). *Optimal control: Linear quadratic methods.*

Atkeson, C., & Morimoto, J. (2003). Nonparametric representation of policies and value functions: A trajectory-based approach. *NIPS 15.*

Atkeson, C. G. (1994). Using local trajectory optimizers to speed up global optimization in dynamic programming. *Neural Information Processing Systems 6.*

Bagnell, J., & Schneider, J. (2001). Autonomous helicopter control using reinforcement learning policy search methods. *Proceedings of the International Conference on Robotics and Automation.*

Bagnell, J. A., Kakade, S., Ng, A. Y., & Schneider, J. (2004). Policy search by dynamic programming. *Neural Information Processing Systems 16.*

Dorf, R., & Bishop, R. (2000). *Modern control systems, 9th edition.* Prentice-Hall.

Egerstedt, M., & Hu, X. (2000). Coordinated trajectory following for mobile manipulation. *Proceedings of the Internation Conference on Robotics and Automation.*

Franklin, G., Powell, J., & Emani-Naeini, A. (1995). *Feedback control of dynamic systems.* Addison-Wesley.

Garcia, C., Prett, D., & Morari, M. (1989). Model predictive control: theory and practice — a survey. *Automatica, 25,* 335–348.

Geibel, P., Brefeld, U., & Wysotzki, F. (2004). Perceptron and SVM learning with generalized cost models. *Intelligent Data Analysis, 8.*

Hoffmann, G., Tomlin, C., Montemerlo, M., & Thrun, S. (2007). Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. *Proc. 26th American Control Conf.*

Jacobson, D., & Mayne, D. (1970). *Differential dynamic programming.* Elsevier.

Johnson, E., & Calise, A. (2002). A six degree-of-freedom adaptive flight control architecture for trajectory following. *Proceedings of the AIAA Guidance, Navigation, and Control Conference.*

Lagoudakis, M., & Parr, R. (2003). Reinforcement learning as classification: Leveraging modern classifiers. *Proceedings of the Int'l Conf on Machine Learning.*

Leith, D., & Leithead, W. (2000). Survey of gain-scheduling analysis and design. *International Journal of Control, 73,* 1001–1025.

Ng, A. Y., Kim, H. J., Jordan, M., & Russell, S. (2004). Autonomous helicopter flight via reinforcement learning. *Neural Information Processing Systems 16.*

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming.* Wiley.

Rossetter, E., & Gerdes, J. (2002). Performance guarantees for hazard based lateral vehicle control. *Proceedings of the International Mechanical Engineering Conference and Exposition.*

Sastry, S. (1999). *Nonlinear systems.* Springer.

Tassa, Y., Erez, T., & Smart, W. (2007). Receding horizon differential dynamic programming. *NIPS 20.*

Thrun, S., & al. (2006). Winning the DARPA Grand Challenge. *J. of Field Robotics.* accepted for publication.

The Skew Spectrum of Graphs

Risi Kondor RISI@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, UCL, 17 Queen Square, London, WC1N 3AR, U.K.

Karsten M. Borgwardt KMB51@CAM.AC.UK
Department of Engineering, University of Cambridge, Trumpington Street, Cambridge, CB2 1PZ, U.K.

Abstract

The central issue in representing graph-structured data instances in learning algorithms is designing features which are invariant to permuting the numbering of the vertices. We present a new system of invariant graph features which we call the skew spectrum of graphs. The skew spectrum is based on mapping the adjacency matrix of any (weigted, directed, unlabeled) graph to a function on the symmetric group and computing bispectral invariants. The reduced form of the skew spectrum is computable in $O(n^3)$ time, and experiments show that on several benchmark datasets it can outperform state of the art graph kernels.

1. Introduction

After real valued vectors and strings, the third most fundamental type of data instance in machine learning are graphs. In addition to application domains such as bioinformatics (Sharan & Ideker, 2006), chemoinformatics (Bonchev & Rouvray, 1991), social networks (Kumar et al., 2006), etc., where information is presented as a graph from the start, graphs are also used to capture the relationships between the different parts of segmented images in computer vision (Harchaoui & Bach, 2007), and to capture grammatical structure in language (Collins & Duffy, 2002). Graphs may be directed or undirected, weighted or unweighted, and their vertices may be labeled, partially labeled or unlabeled. In each of these cases, the challenge is to represent graphs in a way that preserves their structure, but is insensitive to spurious transformations, such as changing the (arbitrary) numbering of their vertices.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Given a graph $\mathcal{G}$, the two main lines of research that have emerged to address the above problem focus respectively on (a) designing an explicit feature mapping $\mathcal{G} \mapsto (q_1, q_2, \ldots, q_k)$; and (b) designing a kernel $k(\mathcal{G}_1, \mathcal{G}_2)$. Proponents of the first approach exploit global invariant properties of $\mathcal{G}$, such as the eigenvalues of its graph Laplacian, or local invariant properties, such as the number of occurrences in $\mathcal{G}$ of a library of small subgraphs. In contrast, proponents of the kernel approach use various intuitions about simultaneous random walks and diffusion on product graphs (Gärtner, 2003).

The new method that we present in this paper belongs in the first of the above two categories, but is distinguished from prior work (with the exception of (Shawe-Taylor, 1993)) by its algebraic character. In this regard, it is related to the recent line of papers (Kondor et al., 2007; Huang et al., 2008; Kondor, 2007a) introducing concepts from non-commutative harmonic analysis to machine learning. The mathematical foundations of our work are Kakarala's seminal results on the bispectra of functions on compact groups (Kakarala, 1993; Kakarala, 1992), and the recent discovery of a unitarily equivalent, but computationally more attractive set of invariants called the skew spectrum (Kondor, 2007b). We show how these general theories can be harnessed to construct graph invariants, and examine in detail their computational properties.

Experiments on standard datasets of chemical compounds show that the skew spectrum of graphs is competitive with the state of the art in graph features, and in some cases outperforms all other methods. A major advantage of the skew spectrum is that since it is an explicit feature mapping, it can be applied as a preprocessing step, and hence scales linearly with the number of examples. The computational complexity of computing the (reduced) skew spectrum of a single graph of n nodes scales with n^3. Uniquely amongst the graph invariants used in machine learning, the skew spectrum has a fixed number of scalar components (85

for the complete skew spectrum and 49 for its reduced version), resulting in a very compact representation. This does not stop the skew spectrum form remaining competitive both in speed and representational accuracy up to about $n = 300$.

For those technical details of the skew spectrum which could not be squeezed into this conference paper we refer the reader to the accompanying report (Kondor, 2008).

2. Graph Invariants

In this paper $\mathcal{G}$ will be a directed weighted graph of n vertices. We represent $\mathcal{G}$ by its adjacency matrix $A \in \mathbb{R}^{n\times n}$, where $[A]_{i,j} \in \mathbb{R}$ is the weight of the edge from vertex i to vertex j. Unweighted graphs are special cases satisfying $[A]_{i,j} \in \{0, 1\}$, while undirected graphs are special cases satisfying $A^\top = A$. We assume that A has no self-loops, i.e., $[A]_{i,i} = 0$ for $i = 1, 2, \ldots, n$.

Recall that a **permutation** of n objects is a bijective map $\pi\colon \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$. Permuting the labels on the vertices of $\mathcal{G}$ by π results in a new adjacency matrix A^π with entries

$$[A^\pi]_{\pi(i),\pi(j)} = [A]_{i,j}, \tag{1}$$

but A and A^π both represent the same graph $\mathcal{G}$. A function $q(A)$ is called a **graph invariant** if it is invariant to relabelings of this kind, i.e., if $q(A) = q(A^\pi)$ for any permutation π. Our objective is to construct a system $(q_1, q_2, \ldots, q_k)$ of graph invariants which capture as much information about $\mathcal{G}$ as possible, yet can be computed economically.

2.1. Reduction to Left-translation Invariance

Our approach is based on the fact that permutations form a **group**. This means that if for a pair of permutations σ_1 and σ_2, we define their product $\sigma_3 = \sigma_2\sigma_1$ by composition of maps, i.e., $\sigma_3(i) = \sigma_2(\sigma_1(i))$, then the following axioms are satisfied:

1. for any two permutations σ_1 and σ_2, the product $\sigma_2\sigma_1$ is also a permutation;
2. for any three permutations σ_1, σ_2 and σ_3, $\sigma_1(\sigma_2\sigma_3) = (\sigma_1\sigma_2)\sigma_3$;
3. The identity $e(i) = i$ is a permutation;
4. For any permutation σ, there is an inverse permutation σ^{-1} satisfying $\sigma\sigma^{-1} = \sigma^{-1}\sigma = e$.

The group of permutations of n objects is called the **symmetric group** over n letters and is denoted $\mathbb{S}_n$.

To find graph invariants we begin by mapping A to a function $f\colon \mathbb{S}_n \to \mathbb{R}$, defined as

$$f(\sigma) = A_{\sigma(n),\sigma(n-1)}. \tag{2}$$

Note that this is a very special type of function on $\mathbb{S}_n$ in that it is constant on each block of permutations

$$S_{i,j} = \{\, \sigma \in \mathbb{S}_n \mid \sigma(n) = i,\ \sigma(n-1) = j \,\}. \tag{3}$$

For $k < n$, identifying $\mathbb{S}_k$ with the subgroup of permutations permuting $1, 2, \ldots, k$ amongst themselves and leaving $k + 1, \ldots, n$ fixed, the above blocks, of which there are $n(n-1)$ in total, each have the form $\sigma\,\mathbb{S}_{n-2} = \{\, \sigma\tau \mid \tau \in \mathbb{S}_{n-2} \,\}$, and are called **left** $\mathbb{S}_{n-2}$–**cosets**.

Defining f as in (2) ensures that under relabeling it transforms in a transparent fashion. Specifically, if f' is the function corresponding to A^π, then

$$f'(\pi\sigma) = A^\pi_{(\pi\sigma)(n),(\pi\sigma)(n-1)} = A_{\sigma(n),\sigma(n-1)} = f(\sigma). \tag{4}$$

In general, a function $g\colon \mathbb{S}_n \to \mathbb{R}$ related to f by $g(\sigma) = f(\pi^{-1}\sigma)$ is called the **left-translate** of f by π, and is denoted f^π. Equation 4 tells us that $f' = f^\pi$, reducing the problem of constructing graph invariants to finding left-translation invariant features of functions on $\mathbb{S}_n$.

2.2. Invariant Matrices

Now consider the weighted sum of matrices

$$\widehat{f}_\rho = \sum_{\sigma\in\mathbb{S}_n} f(\sigma)\,\rho(\sigma), \tag{5}$$

where $\rho(\sigma)$ is a system of complex valued matrices satisfying

$$\rho(\sigma_2\,\sigma_1) = \rho(\sigma_2)\,\rho(\sigma_1) \qquad \sigma_1, \sigma_2 \in \mathbb{S}_n,$$

as well as the unitarity condition $\rho(\sigma^{-1}) = (\rho(\sigma))^{-1} = \rho(\sigma)^\dagger$. Such systems of matrices are called **unitary matrix representations** of $\mathbb{S}_n$. Changing variables from σ to $\sigma' = \pi^{-1}\sigma$ shows that

$$\begin{aligned}\widehat{f}^\pi_\rho = \sum_{\sigma\in\mathbb{S}_n} f(\pi^{-1}\sigma)\,\rho(\sigma) &= \sum_{\sigma'\in\mathbb{S}_n} f(\sigma')\,\rho(\pi\sigma') \\ &= \sum_{\sigma'\in\mathbb{S}_n} f(\sigma')\,\rho(\pi)\,\rho(\sigma') = \rho(\pi)\,\widehat{f}_\rho,\end{aligned}$$

which suggests that (5) is a good starting point for constructing left-translation invariants of f. For example, the matrix $\widehat{a}_\rho = \widehat{f}^\dagger_\rho \cdot \widehat{f}_\rho$ is invariant because

$$\begin{aligned}\widehat{a}^\pi_\rho = \widehat{f}^{\pi\,\dagger}_\rho \cdot \widehat{f}^\pi_\rho = (\rho(\pi)\widehat{f}_\rho)^\dagger(\rho(\pi)\widehat{f}_\rho) = \\ \widehat{f}^\dagger_\rho\,\rho(\pi)^\dagger\,\widehat{\rho}(\pi)\,\widehat{f}_\rho = \widehat{f}^\dagger_\rho \cdot \widehat{f}_\rho = \widehat{a}_\rho.\end{aligned} \tag{6}$$

The question we face is how to construct such invariants in a systematic way with minimum redundancy, yet maximum representational power.

3. Irreps and the Fourier Transform

It is easy to see that if $\rho_1 : \mathbb{S}_n \to \mathbb{C}^{d\times d}$ is a unitary representation of $\mathbb{S}_n$, and T is any $d \times d$ unitary matrix, then $\rho_2(\sigma) = T\,\rho_1(\sigma)\,T^\dagger$ is also a unitary representation. Such pairs of representations are said to be **equivalent**. Once we have computed (5) with $\rho = \rho_1$, computing it again with $\rho = \rho_2$ will not lead to additional invariants, since $\widehat{f}_{\rho_2} = T\widehat{f}_{\rho_1}T^\dagger$.

Another potential source of redundancy is reducibility. A representation ρ is said to be **reducible** if for some unitary T it splits in the form

$$\rho(\sigma) = T \left(\begin{array}{c|c} \rho_1(\sigma) & \\ \hline & \rho_2(\sigma) \end{array} \right) T^\dagger \qquad \sigma \in \mathbb{S}_n$$

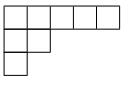
into a direct sum of smaller representations ρ_1 and ρ_2. Once again, $\widehat{f}_\rho$ does not supply any information on top of $\widehat{f}_{\rho_1}$ and $\widehat{f}_{\rho_2}$ because $\widehat{f}_\rho = T(\widehat{f}_{\rho_1} \oplus \widehat{f}_{\rho_2})T^\dagger$.

To avoid these redundancies we will use a *complete set of inequivalent irreducible unitary representations* (**irreps** for short). Such a set we denote by $\mathcal{R}$. The corresponding set of matrices

$$\widehat{f}_\rho = \sum_{\sigma\in\mathbb{S}_n} f(\sigma)\,\rho(\sigma), \qquad \rho \in \mathcal{R}, \tag{7}$$

is called the **Fourier transform** of f, and it provides the basis for generalizing harmonic analysis to non-commutative groups (Diaconis, 1988; Rockmore, 1997). Just as the classical Fourier transforms, $\mathcal{F}\colon f \to (\widehat{f}_\rho)_{\rho\in\mathcal{R}}$ satisfies a generalized form of the translation and convolution theorems. What is most crucial for our present purposes, however, is that (given the appropriate inner products) $\mathcal{F}$ is unitary, and therefore one–to–one: hence, no information is lost in going from f to the set of matrices $(\widehat{f}_\rho)_{\rho\in\mathcal{R}}$.

Several different systems of irreps for $\mathbb{S}_n$ are described in the literature (James & Kerber, 1981). In the interests of saving space, we only describe their general scheme, without going into the details of how to compute the actual representation matrices. In all the major representation schemes the individual irreps $\rho \in \mathcal{R}$ are indexed by **Young diagrams**, which are n boxes arranged in consecutive left-aligned rows satisfying the condition that no row overhangs the row above it. For example,



(8)

is a valid Young diagram for $n = 8$. We will use the letter λ to refer to Young diagrams and write $\lambda \vdash n$ to denote that λ is a Young diagram with n boxes. To simplify notation somewhat we write $\widehat{f}_\lambda$ for $\widehat{f}_{\rho_\lambda}$.

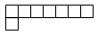
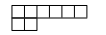
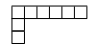
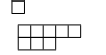
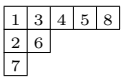
λ		d_λ
(n)		1
$(n-1,1)$		$n-1$
$(n-2,2)$		$\frac{n(n-3)}{2}$
$(n-2,1,1)$		$\frac{(n-1)(n-2)}{2}$
$(n-3,3)$		$\frac{n(n-1)(n-5)}{6}$
$(n-3,2,1)$		$\frac{n(n-2)(n-4)}{3}$

Table 1. The dimensionalities of some representations of $\mathbb{S}_n$. The diagrams are drawn as if $n = 8$, but the formulae hold for general n.

Young diagrams can also be described by listing the number of boxes in each row, for example, the above diagram is $\lambda = (5, 2, 1)$. For concreteness, when we need to draw Young diagrams we will always depict them as if $n = 8$.

Bijectively filling the boxes of a Young diagram with the numbers $1, 2, \ldots, n$ gives a **Young tableau**, and if a tableau satisfies the condition that in each row the numbers increase from left to right and in each column they increase from top to bottom it is called a **standard tableau**. For example,



is a standard tableau of shape $(5, 2, 1)$. The significance of standard tableaux is that they label the individual dimensions of the irrep of the same shape. Hence, we can find the dimensionality of ρ_λ by counting the number of possible standard tableaux of shape λ (Figure 1). An interesting special property of the symmetric group is that all the irreps can be chosen to be real valued. For generality, we retain the complex notation, but note that the actual system of irreps used in our experiments is real, so we could substitute "orthogonal" for "unitary" and $^\top$ for † throughout.

4. The Bispectrum and the Skew Spectrum

Armed with the irreps and non-commutative Fourier transforms, we can now undertake a more systematic study of left-translation invariant features of functions on the symmetric group. For example, (6) leads to the set of invariant matrices

$$\widehat{a}_\lambda = \widehat{f}_\lambda^\dagger \cdot \widehat{f}_\lambda, \qquad \lambda \vdash n,$$

which, by analogy with the analogous quantity in classical signal processing, is called the **power spectrum**

of f. The problem with the power spectrum is that it is very lossy. To see this, one need only consider $\widehat{f}'_\lambda = M_\lambda \widehat{f}_\lambda$ for any sequence of unitary matrices $(M_\lambda)_{\lambda \vdash n}$. The functions f and f' corresponding to these two Fourier transforms may be very different, yet their power spectrum will be the same.

4.1. The Bispectrum

Kakarala realized that the lossiness of the power spectrum can be addressed by forming tensor products of the various Fourier components, and proposed the alternative system of invariant matrices

$$\widehat{b}_{\lambda_1,\lambda_2} = (\widehat{f}_{\lambda_1} \otimes \widehat{f}_{\lambda_2})^\dagger \, C_{\lambda_1,\lambda_2} \Big[\bigoplus_\lambda \widehat{f}_\lambda\Big] \quad \lambda_1, \lambda_2 \vdash n \quad (9)$$

called the **bispectrum** (Kakarala, 1993)[1]. The bispectrum is based on the observation that $\widehat{f}_{\lambda_1} \otimes \widehat{f}_{\lambda_2}$ transforms according to

$$\widehat{f}^\pi_{\lambda_1} \otimes \widehat{f}^\pi_{\lambda_2} = (\rho_{\lambda_1}(\pi) \otimes \rho_{\lambda_2}(\pi)) \cdot (\widehat{f}_{\lambda_1} \otimes \widehat{f}_{\lambda_2}),$$

and that $\rho_{\lambda_1}(\pi) \otimes \rho_{\lambda_2}(\pi)$ is also a representation, although in general not irreducible. The general formula

$$\rho_{\lambda_1}(\sigma) \otimes \rho_{\lambda_2}(\sigma) = C_{\lambda_1,\lambda_2} \Big[\bigoplus_\lambda \rho_\lambda(\sigma)\Big] C^\dagger_{\lambda_1,\lambda_2} \quad (10)$$

telling us how to reduce it into a direct sum of irreps is called the Clebsch-Gordan decomposition, and the C_{λ_1,λ_2} unitary matrices appearing in (10) and (9) are called **Clebsch-Gordan matrices**.

By plugging (10) into (9) it is easy to see that the bispectrum is indeed invariant to left-translation. A much more remarkable fact, proved in (Kakarala, 1992), is that provided the technical condition that each $\widehat{f}_\lambda$ is invertible is satisfied, the bispectrum is also *complete* (or lossless) in the sense that the matrices $(\widehat{b}_{\lambda_1,\lambda_2})_{\lambda_1,\lambda_2 \vdash n}$ uniquely determine f up to translation.

4.2. The Skew Spectrum

Some of the drawbacks of using the bispectrum in practical applications are that (a) computing (9) may involve multiplying together very large matrices; (b) that the Clebsch-Gordan matrices, despite being universal constants, are not generally available in tabuated form; and (c) that for large n they are extremely difficult to compute. To address these concerns, Kondor (2007b) proposed an alternative set of invariants, called the **skew spectrum**, which are unitarily equivalent to the bispectrum, but much more straightforward to compute. The skew spectrum of $f\colon \mathbb{S}_n \to \mathbb{C}$ is defined as the collection of matrices

$$\widehat{q}_{\nu,\lambda} = \widehat{r}^\dagger_{\nu,\lambda} \cdot \widehat{f}_\lambda, \qquad \lambda \vdash n, \quad \nu \in \mathbb{S}_n, \qquad (11)$$

where $(\widehat{r}_{\nu,\lambda})_{\lambda \vdash n}$ is the Fourier transform of the function $r_\nu(\sigma) = f(\sigma\nu)\, f(\sigma)$. In (Kondor, 2007b) it is shown that if for some subgroup H, f is constant on left σH-cosets (as the function defined in (2) is constant on left $\mathbb{S}_{n-2}$-cosets), then it is sufficient to let ν take on just one value from each

$$H\sigma H = \{\, h_1 \sigma h_2 \mid h_1, h_2 \in H \,\}$$

double-coset, since every other component of $\widehat{q}$ will be linearly dependent on these.

5. The Skew Spectrum of Graphs

By the results of Sections 2 and 4, plugging (2) into (11) will give a relabeling invariant representation of any weighted graph $\mathcal{G}$. As it stands, however, this seems of only academic interest, since ν must extend over $n!$ different values for any one of which the combined size of the $(\widehat{q}_{\nu,\lambda})_{\lambda \vdash n}$ matrices is itself $n!$. Moreover, computing each $(\widehat{q}_{\nu,\lambda})_{\lambda \vdash n}$ requires a separate Fourier transform.

The first clue to how these problems may be remedied is provided by the comment at the end of the last section that if we are only interested in linearly independent invariants, then due to the special structure of f, we need only let ν take on one value from each $\mathbb{S}_{n-2}\, \sigma\, \mathbb{S}_{n-2}$ double coset. It is easy to see that there are only 7 such double cosets in $\mathbb{S}_n$, namely

$$\begin{aligned}
S^{n\mapsto n}_{n-1\mapsto n-1} &= \{\, \sigma \in \mathbb{S}_n \mid \sigma(n) = n,\ \ \sigma(n-1) = n-1 \,\} \\
S^{n\mapsto n-1}_{n-1\mapsto n} &= \{\, \sigma \in \mathbb{S}_n \mid \sigma(n) = n-1,\ \ \sigma(n-1) = n \,\} \\
S^{n\mapsto n}_{n-1\mapsto *} &= \{\, \sigma \in \mathbb{S}_n \mid \sigma(n) = n,\ \ \sigma(n-1) \in [n-2] \,\} \\
S^{n\mapsto n-1}_{n-1\mapsto *} &= \{\, \sigma \in \mathbb{S}_n \mid \sigma(n) = n-1,\ \ \sigma(n-1) \in [n-2] \,\} \\
S^{n\mapsto *}_{n-1\mapsto n-1} &= \{\, \sigma \in \mathbb{S}_n \mid \sigma(n) \in [n-2],\ \sigma(n-1) = n-1 \,\} \\
S^{n\mapsto *}_{n-1\mapsto n} &= \{\, \sigma \in \mathbb{S}_n \mid \sigma(n) \in [n-2],\ \ \sigma(n-1) = n \,\} \\
S^{n\mapsto *}_{n-1\mapsto *} &= \{\, \sigma \in \mathbb{S}_n \mid \sigma(n), \sigma(n-1) \in [n-2] \,\}\ ,
\end{aligned} \qquad (12)$$

where $[n-2] = \{1, 2, \ldots, n-2\}$.

Definition 1 *Given a graph $\mathcal{G}$ of n vertices and adjacency matrix A, the **skew spectrum** of $\mathcal{G}$ is defined as the collection of matrices*

$$\widehat{q}_{\nu,\lambda} = \widehat{r}^\dagger_{\nu,\lambda} \cdot \widehat{f}_\lambda, \qquad \lambda \vdash n, \qquad (13)$$

where $r_\nu(\sigma) = f(\sigma\nu)\, f(\sigma)$; f is defined as in (2); and ν takes on one value from each of the double cosets listed in (12).

The second important consequence of the form of (2) is that using the right system of irreps, $\widehat{f}$ becomes very

[1] The exact definition of the bispectrum varies somewhat between authors. However, the various definitions are all unitarily equivalent to each other.

sparse. To be specific, we use **Young's orthonormal representation** (YOR), which has the special property that if σ is restricted to $\mathbb{S}_{n-1}$, then the $\rho_\lambda(\sigma)$ matrices block-diagonalize in the form

$$\rho_\lambda(\sigma) = \bigoplus_{\lambda^-} \rho_{\lambda^-}(\sigma), \qquad \sigma \in \mathbb{S}_{n-1},$$

where λ^- extends over all valid Young diagrams derivable from λ by the removal of a single box. If the pair of standard tableaux t and t' feature n at the same box, then

$$[\rho_\lambda(\sigma)]_{t,t'} = [\rho_{\lambda^-}(\sigma)]_{t\downarrow_{n-1},t'\downarrow_{n-1}}$$

where $t\downarrow_{n-1}$ is the standard tableau that we get from t by removing the box containing n and λ^- is the corresponding Young diagram. If t and t' feature n at different locations, then $[\rho_\lambda(\sigma)]_{t,t'} = 0$. Applying this relation recursively gives that for $\sigma \in \mathbb{S}_k$,

$$[\rho_\lambda(\sigma)]_{t,t'} = \begin{cases} [\rho_{\lambda^-}(\sigma)]_{t\downarrow_k,t'\downarrow_k} & \text{or} \\ 0 \end{cases} \tag{14}$$

depending on whether $k+1,\ldots,n$ are each in the same boxes in t and t' or not.

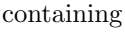
Now letting $\mathbb{S}_n/\mathbb{S}_{n-2}$ be a set of $n(n-1)$ permutations, one from each $\sigma\mathbb{S}_{n-2}$ coset, and defining $h_\sigma\colon \mathbb{S}_{n-2} \to \mathbb{C}$ as $h_\sigma(\tau) = f(\sigma\tau)$, the Fourier transform may be written as

$$\widehat{f}_\lambda = \sum_{\sigma\in\mathbb{S}_n/\mathbb{S}_{n-2}} \sum_{\tau\in\mathbb{S}_{n-2}} f(\sigma\tau)\,\rho_\lambda(\sigma)\,\rho_\lambda(\tau) = \sum_{\sigma\in\mathbb{S}_n/\mathbb{S}_{n-2}} \rho_\lambda(\sigma) \sum_{\tau\in\mathbb{S}_{n-2}} h_\sigma(\tau)\,\rho_\lambda(\tau).$$

Plugging in the appropriate decomposition of ρ_λ into a direct sum of irreps of $\mathbb{S}_{n-2}$ gives

$$\widehat{f}_\lambda = \sum_{\sigma\in\mathbb{S}_n/\mathbb{S}_{n-2}} \rho_\lambda(\sigma) \sum_{\tau\in\mathbb{S}_{n-2}} h_\sigma(\tau) \bigoplus_{\lambda^-} \rho_{\lambda^-}(\tau) = \sum_{\sigma\in\mathbb{S}_n/\mathbb{S}_{n-2}} \rho_\lambda(\sigma) \bigoplus_{\lambda^-} \left[\widehat{h}_\sigma\right]_{\lambda^-}, \tag{15}$$

showing that the Fourier transform over $\mathbb{S}_n$ may be broken down into $n(n-1)$ Fourier transforms over $\mathbb{S}_{n-2}$. This relationship is at the heart of the Clausen-type fast Fourier transforms for $\mathbb{S}_n$ (Clausen, 1989).

For f defined by (2), each h_σ is a constant function, and hence its Fourier transform has a very special form: since in YOR the irrep corresponding to $\lambda = (n)$ is the constant representation $\rho_{(n)}(\sigma) = (1)$, the corresponding $\left[\widehat{h}_\sigma\right]_\lambda$ component will be non-zero, but by unitarity all other components of $\widehat{h}_\sigma$ vanish. Plugging this result into (15) and using (14) shows that only those columns of $\widehat{f}$ may be non-zero which are indexed by standard tableau derivable from  by adding a box containing $n-1$ and another box containing n. Here and in the following, when drawing standard tableau, we only indicate the positions of those numbers in them that are not determined by the "numbers increase from left to right and top to bottom" rule. In addition, we use the symbol ■ to denote n and • to denote $n-1$. We summarize the above in the following theorem.

Theorem 1 *If f is defined as in (2), then the only non-zero entries of $\widehat{f}$ in YOR are:*

1. *the single scalar component $\widehat{f}_{(n)}$;*
2. *the*  *column of $\widehat{f}_{(n-1,1)}$;*
3. *the*  *column of $\widehat{f}_{(n-1,1)}$;*
4. *the*  *column of $\widehat{f}_{(n-2,2)}$;*
5. *the*  *column of $\widehat{f}_{(n-2,1,1)}$.*

This remarkable sparsity is the key to computing the skew spectrum of graphs efficiently. At the same time it is rather disappointing, since it manifestly destroys the invertibility of the $\widehat{f}_\lambda$ matrices required for Kakarala's completeness result. The $\widehat{r}_{\nu,\lambda}$ matrices are also column sparse, but their sparsity pattern is somewhat more complicated, so we leave describing it to (Kondor, 2008).

Equation (13) only yields non-zero elements in $\widehat{q}_{\nu,\lambda}$ where a non-zero row of $\widehat{r}^\dagger_{\nu,\lambda}$ meets a non-zero column of $\widehat{f}_\lambda$. By the above, this happens at only a constant number of row/column combinations. The exact result, derived in (Kondor, 2008), is the following.

Theorem 2 *Using YOR and an appropriate choice of $\{\nu\}$ double coset representatives, the skew spectrum of $\mathcal{G}$ has at most 85 non-zero scalar components.*

6. Computational Considerations

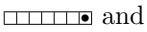
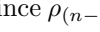
The computational properties of the skew spectrum are closely related to the structural results of the previous section. In particular, it is repeated applications of Clausen decompositions similar to (15) together with the sparsity of YOR that yields an efficient algorithm to compute $\widehat{q}$. In contrast to the previous section, we now employ a two-level factorization $\sigma = \sigma_1\sigma_2\tau$, where $\tau \in \mathbb{S}_{n-2}$, $\sigma_2 \in \mathbb{S}_{n-1}/\mathbb{S}_{n-2}$, and $\sigma_1 \in \mathbb{S}_n/\mathbb{S}_{n-1}$. As before, we have $n(n-1)$ functions $h_{\sigma_1\sigma_2}\colon \mathbb{S}_{n-2} \to \mathbb{C}$ defined $h_{\sigma_1\sigma_2}(\tau) = f(\sigma_1\sigma_2\tau)$, and by (2) each of these

is a constant function equal to $[A]_{\sigma_1\sigma_2(n),\,\sigma_1\sigma_2(n-1)}$. However, now we will also have intermediate functions $g_{\sigma_1}\colon \mathbb{S}_{n-1} \to \mathbb{C}$ defined $g_{\sigma_1}(\tau) = f(\sigma_1\tau)$. We then have the following results.

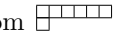
Lemma 1 *Each $\widehat{g}_{\sigma_1}$ can be computed from A in $O(n^2)$ scalar operations.*

Proof. Similarly to (15), we can relate the Fourier transform of g_{σ_1} to the Fourier transforms of $(h_{\sigma_1\sigma_2})_{\sigma_2}$ by

$$[\widehat{g}_{\sigma_1}]_\lambda = \sum_{\sigma_2 \in \mathbb{S}_{n-1}/\mathbb{S}_{n-2}} \rho_\lambda(\sigma_2) \bigoplus_{\lambda^-} \left[\widehat{h}_{\sigma_1\sigma_2}\right]_{\lambda^-}.$$

Since each $\widehat{h}_{\lambda_1\lambda_2}$ is confined to the one dimensional component $[\widehat{h}_{\sigma_1\sigma_2}]_{(n-2)}$, the only non-zero columns of $\widehat{g}_{\sigma_1}$ will be the ones indexed by standard tableaux derivable from  by addition of the single box , namely and  . The first one of these is trivial to compute, since $\rho_{(n-1)}(\sigma_2) \equiv (1)$, collapsing the above sum to

$$[\widehat{g}_{\sigma_1}]_{(n-1)} = \sum_{\sigma_2 \in \mathbb{S}_{n-1}/\mathbb{S}_{n-2}} \left[\widehat{h}_{\sigma_1\sigma_2}\right]_{(n-2)}.$$

This is a sum of $n-1$ scalars, so it can be computed in $O(n)$ time. Computing the second component involves taking the direct sum $M_{\sigma_1\sigma_2} = \bigoplus_{\lambda^-} \left[\widehat{h}_{\sigma_1\sigma_2}\right]_{\lambda^-}$, where λ^- extends over the two diagrams $(n-2)$ and $(n-3,1)$ derivable from  by removing a box. However, $\left[\widehat{h}_{\sigma_1\sigma_2}\right]_{(n-3,1)} = 0$, so $M_{\sigma_1\sigma_2}$ has only one non-zero entry. For given σ_2, multiplying $\rho_{(n-2,1)}(\sigma_2)$ with $M_{\sigma_1\sigma_2}$ thus requires $n-2$ operations. We are summing over $(n-1)$ possible values of σ_2, so the total time complexity is $(n-1)(n-2)$. ■

Lemma 2 *$\widehat{f}$ can be computed from the intermediate transforms $(\widehat{g}_{\sigma_1})_{\sigma_1 \in \mathbb{S}_n/\mathbb{S}_{n-1}}$ in $O(n^3)$ operations.*

The proof of Lemma 2 is similar to that of Lemma 1, but also involves considerations of the sparsity of the YOR matrices. Unfortunately, space limitations prevent us from providing a proof of this result. Putting the two lemmas together gives the following theorem.

Theorem 3 *The Fourier transform of f as defined in (2) can be computed in $O(n^3)$ operations.*

Proof. Each of the n different $\widehat{g}$ transforms can be computed in $O(n^2)$ operations, followed by the single $O(n^3)$ step of computing $\widehat{f}$ from the $\widehat{g}$'s. ■

Computing $\widehat{r}_\nu$ is unfortunately more costly than computing $\widehat{f}$. An extended version of this paper, which is in preparation, will show that the time complexity of this is $O(n^6)$. While for n less than about 20 this might still be feasible, for the type of experiments on which we wish to validate the skew spectrum it is not a viable option. The following subsection shows that most of the components of $\widehat{q}$ can still be computed in $O(n^3)$ operations.

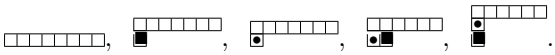
6.1. The Reduced Skew Spectrum

The expensive part of computing $\widehat{r}_\nu$ is computing those columns outside the five listed in Theorem 1. This leads to the idea of simply forcing these columns to be zero.

Definition 2 *Given a graph $\mathcal{G}$ of n vertices and adjacency matrix A, the **reduced skew spectrum** of $\mathcal{G}$ is the collection of matrices*

$$\widehat{q}^*_{\nu,\lambda} = \widehat{r}^*_{\nu,\lambda}{}^\dagger \cdot \widehat{f}_\lambda, \qquad \lambda \vdash n, \tag{16}$$

where f,r, and ν are as in Definition 1, and $\widehat{r}^*_\nu$ denotes the projection of $\widehat{r}_\nu$ to its columns labeled by

 . (17)

Since $\widehat{r}^*_\nu$ is identical to $\widehat{r}_\nu$ except for zeroing out certain columns, $(\widehat{q}^*_\nu)_\nu$ will yield a subset of the 85 scalar invariants in $(\widehat{q}_\nu)_\nu$. For each value of ν, for $\lambda = (n)$ we have one row of $\widehat{r^*}_\nu^\dagger$ meeting one column of $\widehat{f}_\nu$ giving one component; for $\lambda = (n-1,1)$ we have two rows meeting two columns, giving four components, etc. In total the reduced skew spectrum has $7\,(1+4+1+1) = 49$ non-zero scalar components.

The space of functions the Fourier transform of which has the sparsity pattern (16) is exactly the space of functions which are invariant on $\sigma\mathbb{S}_{n-2}$ cosets. This means that for each $\widehat{r}^*_\nu$ there must be a corresponding matrix B_ν related to it the same way that f is related to the adjacency matrix A. These matrices are given by the following theorem, the proof of which we again relegate to a longer publication.

Theorem 4 *For $\widehat{r}^*_\nu$ as defined in Definition 2,*

$$r^*_\nu(\sigma) = [B_\nu]_{\sigma(n),\sigma(n-1)},$$

where the seven possible B_ν matrices corresponding to the seven double cosets listed in (12) are

$$\begin{aligned}
[B_1]_{i,j} &= A_{i,j}\, A_{i,j} \\
[B_2]_{i,j} &= A_{i,j}\, A_{j,i} \\
[B_3]_{i,j} &= \tfrac{1}{n} A_{i,j} \textstyle\sum_{i'=1}^n A_{i',j} \\
[B_4]_{i,j} &= \tfrac{1}{n} A_{i,j} \textstyle\sum_{j'=1}^n A_{i,j'}
\end{aligned}$$

$$[B_5]_{i,j} = \frac{1}{n} A_{j,i} \sum_{i'=1}^{n} A_{i',j}$$
$$[B_6]_{i,j} = \frac{1}{n} A_{j,i} \sum_{j'=1}^{n} A_{i,j'}$$
$$[B_7]_{i,j} = \frac{1}{n(n-1)} A_{i,j} \sum_{i'=1}^{n} \sum_{j'=1}^{n} A_{i',j'}$$

Theorem 4 tells us that the reduced skew spectrum is very simple to compute: simply form the matrices $B_1, \ldots, B_7$, compute the corresponding $\widehat{r}^*_\nu$ the same way as $\widehat{f}$ is computed from A and form the products (16). In total this takes 8 partial Fourier transforms, each of which takes $O(n^3)$ time.

7. Experiments

In our experiments we evaluate the performance of the skew spectrum features on four benchmark datasets of chemical structures of molecules: MUTAG, ENZYMES, NCI1, and NCI109. MUTAG (Debnath et al., 1991) is a dataset of 188 mutagenic aromatic and heteroaromatic nitro compounds. The classification task is to predict for each molecule whether it exerts a mutagenic effect on the Gram-negative bacterium *Salmonella typhimurium*. ENZYMES is a dataset which we obtained from (Borgwardt et al., 2005), and which consists of 600 enzymes from the BRENDA enzyme database (Schomburg et al., 2004). In this case the task is to correctly assign each enzyme to one of the 6 EC top level classes. The average number of nodes of the graphs in this dataset is 32.6 and the average number of edges is 124.3. Finally, we also conducted experiments on two balanced subsets of NCI1 and NCI109, which classify compounds based on whether or not they are active in an anti-cancer screen ((Wale & Karypis, 2006) and `http://pubchem.ncbi.nlm.nih.gov`).

Since in these datasets the number of vertices varies from graph to graph, we set n to be the maximum over the entire dataset and augment each of the smaller graphs with the appropriate number of unconnected "phantom" nodes. The experiments consisted of running SVMs on the above data using the reduced skew spectrum features (linear kernel on these features), the random walk kernel (Gärtner et al., 2003), (with λ set to 10^{-3} on MUTAG/ENZYMES, and 10^{-4} on the NCI datasets for optimal performance), and an equal length shortest-path kernel (Borgwardt & Kriegel, 2005).

Our experimental procedure was as follows. We split each dataset into 10 folds of identical sizes. We then split 9 of these folds again into 10 parts, trained a C-SVM (implemented by LIBSVM (Chang & Lin, 2001)) on 9 parts, and predicted on the 10th part. We repeated this training and prediction procedure for $C \in \{10^{-7}, 10^{-6}, \ldots, 10^{7}\}$, and determined the C reaching maximum prediction accuracy on the 10th part. We then trained an SVM with this best C on all 9 folds (= 10 parts), and predicted on the 10th fold, which acts as an independent evaluation set. We repeated the whole procedure 10 times so that each fold acts as independent evaluation set exactly once. For each dataset and each method, we repeat the whole experiment 10 times and report mean accuracy levels and standard errors in Table 2. In three out of four experiments the skew spectrum beats the other methods, including the shortest-path kernel, which is considered state of the art for graphs of this type. Using a Gaussian RBF kernel instead of the linear kernel yields very similar results.

8. Conclusions

We have presented a new system of graph invariants, called the skew spectrum of graphs, based on a purely algebraic technique. From a mathematical point of view the skew spectrum is interesting because it brings a fundamentally new technique to constructing graph invariants. From a practical machine learning point of view the skew spectrum is interesting because it provides a powerful, yet efficiently computable representation for graph structured data instances.

Acknowledgments

We would like to thank Ramakrishna Kakarala for providing us with a hard copy of his thesis. We would also like to thank Dan Rockmore, Tony Jebara, Rocco Servedio, Maria Chudnovsky and Balázs Szendrői for discussions and the anonymous reviewers for helpful comments.

References

Bonchev, D., & Rouvray, D. H. (Eds.). (1991). *Chemical graph theory: Introduction and fundamentals*, vol. 1. London, UK: Gordon and Breach Science Publishers.

Borgwardt, K. M., & Kriegel, H.-P. (2005). Shortest-path kernels on graphs. *Proc. Intl. Conf. Data Mining* (pp. 74–81).

Borgwardt, K. M., Ong, C. S., Schonauer, S., Vishwanathan, S. V. N., Smola, A. J., & Kriegel, H. P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, *21*, i47–i56.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

	MUTAG	ENZYME	NCI1	NCI109
Number of instances/classes	188/2	600/6	4110/2	4127/2
Max. number of nodes	28	126	111	111
Reduced skew spectrum	**88.61** (0.21)	25.83 (0.34)	**62.72** (0.05)	**62.62** (0.03)
Random walk kernel	71.89 (0.66)	14.97 (0.28)	51.30 (0.23)	53.11 (0.11)
Shortest-path kernel	81.28 (0.45)	**27.53** (0.29)	61.66 (0.10)	62.35 (0.13)

Table 2. Prediction accuracy in percent of the (reduced) skew spectrum features and state of the art graph kernels on four classification benchmarks in 10 repetitions of 10-fold cross-validation. Standard errors are indicated in parentheses. Best results for each datasets are in bold.

Clausen, M. (1989). Fast generalized Fourier transforms. *Theor. Comput. Sci.*, 55–63.

Collins, M., & Duffy, N. (2002). Convolution kernels for natural language. *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., & Hansch, C. (1991). Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *J Med Chem*, *34*, 786–797.

Diaconis, P. (1988). *Group representation in probability and statistics*, vol. 11 of *IMS Lecture Series*. Institute of Mathematical Statistics.

Gärtner, T. (2003). A survey of kernels for structured data. *SIGKDD Explorations*, *5*, 49–58.

Gärtner, T., Flach, P., & Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. *COLT03* (pp. 129–143). Springer.

Harchaoui, Z., & Bach, F. (2007). Image classification with segmentation graph kernels. *Proceedings of CVPR07*.

Huang, J., Guestrin, C., & Guibas, L. (2008). Efficient inference for distributions on permutations. *Proceedings of NIPS07*.

James, G., & Kerber, A. (1981). *The representation theory of the symmetric group*. Addison-Wesley.

Kakarala, R. (1992). *Triple corelation on groups*. Doctoral dissertation, Department of Mathematics, UC Irvine.

Kakarala, R. (1993). A group theoretic approach to the triple correlation. *IEEE Workshop on higher order statistics* (pp. 28–32).

Kondor, R. (2007a). A novel set of rotationally and translationally invariant features for images based on the non-commutative bispectrum. `http://arxiv.org/abs/cs.CV/0701127`.

Kondor, R. (2007b). The skew spectrum of functions on finite groups and their homogeneous spaces. `http://arxiv.org/abs/0712.4259`.

Kondor, R. (2008). The skew spectrum of graphs. To appear at `http://arxiv.org/`.

Kondor, R., Howard, A., & Jebara, T. (2007). Multi-object tracking with representations of the symmetric group. *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*.

Kumar, R., Novak, J., & Tomkins, A. (2006). Structure and evolution of online social networks. *KDD* (pp. 611–617).

Rockmore, D. N. (1997). Some applications of generalized FFTs. *Proceedings of the DIMACS workshop on groups and computation*.

Schomburg, I., Chang, A., Ebeling, C., Gremse, M., Heldt, C., Huhn, G., & Schomburg, D. (2004). Brenda, the enzyme database: updates and major new developments. *Nucleic Acids Research*, *32D*, 431–433.

Sharan, R., & Ideker, T. (2006). Modeling cellular machinery through biological network comparison. *Nature Biotechnology*, *24*, 427–433.

Shawe-Taylor, J. (1993). Symmetries and discriminability in feedforward network architectures. *IEEE Transactions on Neural Networks*, *4*, 816–826.

Wale, N., & Karypis, G. (2006). Comparison of descriptor spaces for chemical compound retrieval and classification. *Proc. of ICDM* (pp. 678–689). Hong Kong.

Fast Estimation of First-Order Clause Coverage through Randomization and Maximum Likelihood

Ondřej Kuželka KUZELO1@FEL.CVUT.CZ
Filip Železný ZELEZNY@FEL.CVUT.CZ
Czech Technical University in Prague, Technická 2, 166 27 Prague 6, Czech Republic

Abstract

In inductive logic programming, θ-subsumption is a widely used coverage test. Unfortunately, testing θ-subsumption is NP-complete, which represents a crucial efficiency bottleneck for many relational learners. In this paper, we present a probabilistic estimator of clause coverage, based on a randomized restarted search strategy. Under a distribution assumption, our algorithm can estimate clause coverage without having to decide subsumption for all examples. We implement this algorithm in program ReCovEr. On generated graph data and real-world datasets, we show that ReCovEr provides reasonably accurate estimates while achieving dramatic runtimes improvements compared to a state-of-the-art algorithm.

1. Introduction

In most inductive logic programming (ILP) algorithms, learned hypothesis are (sets of) first-order clauses. Usually, θ-subsumption is used to test whether a clause entails an example. Since ILP systems need to evaluate large numbers of clauses during hypothesis search, efficiency of the subsumption procedure is one of the crucial factors for performance of learning. Unfortunately, deciding θ-subsumption is an NP-complete problem.

One line of research has focused on developing algorithms for this problem using sophisticated heuristics from the field of constraint satisfaction problems (CSP). Maloberti et Sebag (2004) exploited the correspondence of θ-subsumption with CSP to develop the algorithm Django. Django is currently considered the fastest subsumption checker, outperforming traditional techniques (based on the Prolog unification mechanism) by orders of magnitude. Therefore we employ Django in comparative experiments later in this paper. Another stream of research dealt with incomplete heuristic algorithms for θ-subsumption. Sebag et al. (1997) presented a tractable approximation of θ-subsumption called stochastic matching. Arias et al. (2007) implemented a randomized table-based method.

Unlike the mentioned incomplete heuristic algorithms, our approach uses a complete, albeit randomized, subsumption procedure that correctly decides both subsumption and non-subsumption if given sufficient finite time. Our ultimate estimation of the clause coverage (i.e. the number of subsumed examples) is however an approximation, rapidly achieved by restarting the subsumption procedure each time with a bounded runtime. Subsequent restarts generate an integer sequence, from which the coverage is estimated by maximum likelihood.

Randomized restarted strategies, exploited in our work, have been extensively studied in the past decade (Gomes et al., 2000). They have been demonstrated to be extremely useful for solving many hard combinatorial problems such as satisfiability of boolean formulas or for solving constraint satisfaction problems. Reported reduction in runtimes are often in orders of magnitude. Randomized restarted strategies have been also used in inductive logic programming (Železný et al., 2006), however, not for subsumption checking. Rather, restarts were applied on the clause-search procedure.

This paper is organized as follows. In Section 2 we formalize subsumption and expose the basic algorithms employed as building blocks in our estimation approach. In Section 3 we conduct a preliminary motivating study of runtime distribution. The estimation

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Algorithm 1 *SubsumptionCheck*(C, e): A simple subsumption check algorithm

```
Input: Clause C, example e;
if C ⊆ e then
  return YES
else
  Choose variable V from C using a heuristic function
  for ∀S ∈ PossibleSubs(V, C, e) do
    C' ← Substitute V with S
    if ∀W ∈ Adjacency(V) : PossibleSubs(W, C', e) ≠ ∅ then
      SearchedNodes ← SearchedNodes + 1
      if SubsumptionCheck(C', e) = YES then
        return YES
      end if
    end if
  end for
  return NO
end if
```

algorithm is then developed in Section 4. In Section 5, we compare our algorithm with Django on synthetic and on real-life data. Section 6 concludes the paper.

2. Preliminaries

2.1. Language

In the rest of the paper we assume for simplicity that hypotheses C are clauses without function and constant symbols and examples e are ground clauses. When needed, clauses will be treated as atom sets, e.g. for two clauses C and D, $C \subseteq D$ will denote that C contains all literals contained by D. θ-subsumption is defined as follows

Definition We say that clause C θ-subsumes clause D (denote $C \preceq_\theta D$) iff there exists a substitution θ such that $C\theta \subseteq D$.

2.2. Subsumption Algorithm

We consider a simple heuristic algorithm (Algorithm 1) for verifying whether a clause C subsumes an example e. Similarly to Django (Maloberti & Sebag, 2004) this algorithm is inspired by the CSP framework. It is a backtracking search algorithm with forward checking, a variable selection heuristic and randomization. The heuristic function aims at choosing variables whose substitution makes it likely that an inconsistency, if one exists, is detected soon. For a variable V, the function computes the sum of occurrences of variables in clause C that have already been grounded and that share at least one literal with V. This sum is then multiplied by $1 + \frac{1}{D}$, where D is an upper bound on the size of the domain of V computed in the initialisation phase of the algorithm's run. The variable which maximizes this function is selected; in case of a tie, a random choice is made with uniform probability among the highest scoring variables. Function *PossibleSubs*(V, C, e) returns all terms S (in a random order), which can be substituted for V satisfying that all literals $l \in C$ remain consistent with e. The function prunes away a subset of possible groundings for V whose inclusion in θ would imply $C\theta \not\subseteq e$. In general though, not all such groundings are detected by the function.

3. Subsumption Test Runtimes

We first aimed at obtaining a domain-independent runtime distribution of the subsumption algorithm and thus conducted preliminary experiments with randomly generated hypotheses and examples from the domain of oriented colored graphs. In the clausal representation, each graph acquires the form of a definite clause

$$h \leftarrow l_1 \wedge l_2 \wedge \ldots$$

where h is a fixed head and l_i are first-order atoms, each being one of $edge(t_1, t_2)$, $black(t_3)$, $red(t_4)$. In hypotheses, t_i are variables, in examples these are constants.

For generality, we devised two different graph generators. The first generator generates Erdos-Rényi random graphs where any two vertices are connected with a pre-set probability c (by an edge of a random orientation). The second produces scale-free ("small world") graphs. Here, the graph grows until some desired size is reached; at any step a vertex is added and connected to k vertices already present in the graph. An edge is attached to a vertex with probability increasing with the number of edges already connected to the vertex. In both algorithms, all vertices are colored as black with probability 0.5 and red otherwise. We will refer to the parameter c (k, respectively) of a random uniform (scale-free, respectively) graph as the *connectivity* of the graph.

We subjected Algorithm 1 to experiments with random sets of hypotheses and examples, under various settings of n and c (n and k, respectively), where n denotes number of vertices in the underlying graph and c and k are parameters of the random graphs explained above. Here, we review our principal findings about the respective runtime distributions, since they motivate the design of the estimation algorithm in the next section.

Our first objective was to verify the presence of heavy tails in the runtime distributions $F(t)$. For $t > 0$, the number $F(t)$ is the probability that the tested algorithm resolves a random subsumption instance in no more than t units of time, corresponding to the number of explored search nodes. Informally, a heavy-

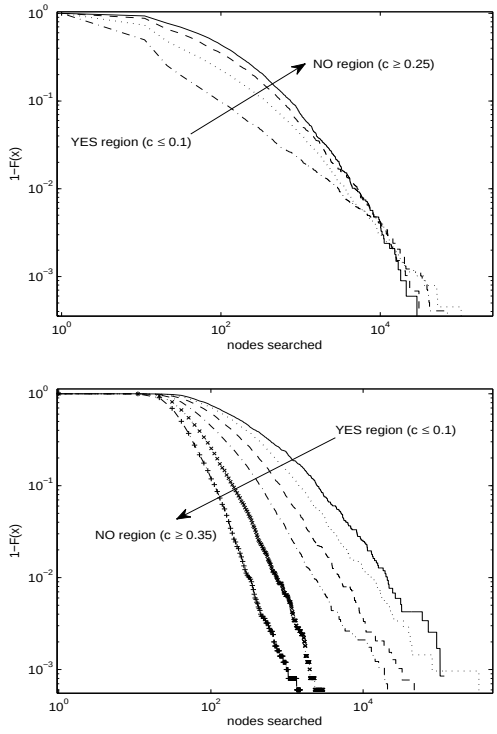


Figure 1. **Top:** The runtime distributions for satisfiable instances with hypotheses built using the Erdos-Rényi random graph generator with $n = 15$ vertices and connectivity consecutively $c \in \{0.1, 0.15, 0.2, 0.25\}$. The graphs corresponding to examples had $n = 50$ vertices and connectivity $p = 0.3$. **Bottom:** The subsumption test runtime distributions for unsatisfiable instances with hypotheses with connectivity consecutively $c \in \{0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$.

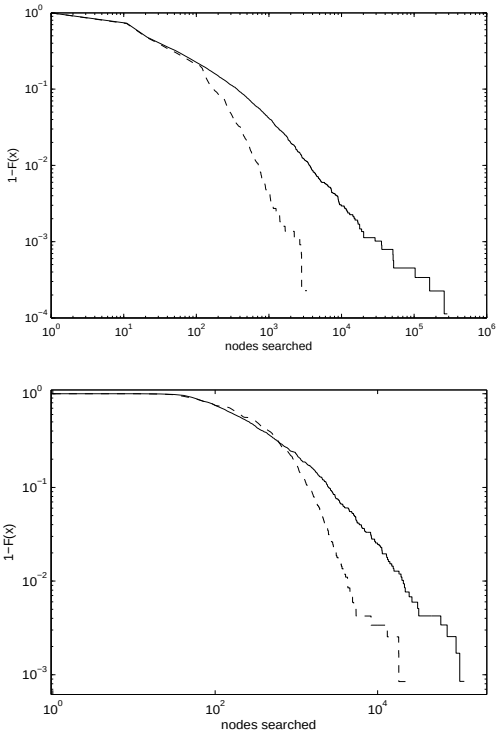


Figure 2. Effect of the restarted strategy for satisfiable (top) and unsatisfiable instances (bottom). The random graphs corresponding to hypotheses had $n = 15$ vertices and connectivity $c = 0.15$. In both cases, the random graphs corresponding to examples had $n = 50$ vertices and connectivity $c = 0.3$. Both hypotheses and examples were randomly generated by the Erdos-Rényi generator.

tailed distribution indicates the non-negligible probability of subsumption instances on which the checking algorithm gets stuck for an extremely long runtime. For example, a heavy tail is exhibited if $1-F(t)$ decays at a power-law rate, i.e. slower than standard distributions which decay exponentially. The presence of a heavy tail in an empirically obtained runtime distribution $F(t)$ is usually checked graphically, by plotting $1 - F(t)$ against t on a log-log scale. In the case of a power-law distribution, this plot then acquires a linear shape (Gomes et al., 2000).

A series of experiments in the phase transition framework (Giordana & Saitta, 2000), which we have performed, revealed a systematic progression from heavy-tailed regimes corresponding to configurations located in the YES region of the phase transition spectrum to non-heavy-tailed regimes corresponding to configurations located in the NO region. This observation agrees with the previous study (Gomes et al., 2005). This progression is shown in Fig. 1 for the Erdos-Rényi graph data. The same trends were observed for the small-world graph data. The plotted runtime distributions refer to subsumption checks between hypotheses with fixed numbers of vertices and connectivity changing among particular distributions, and examples with fixed numbers of vertices and with fixed connectivity. The runtime distributions plotted in the top panel of Fig. 1 refer to satisfiable problem instances, i.e. those where the hypotheses θ-subsume the examples. The distributions in the bottom panel of Fig. 1 refer to unsatisfiable problem instances.

Due to the observed presence of heavy tails in a range of parameters, we next assessed the impact of restarts. For this sake we designed a complete restarted randomized subsumption algorithm, which repeatedly executes Algorithm 1. At each execution $n = 1, 2, \ldots,$ the number of search nodes in the Algorithm 1 is bounded by some pre-defined number $R(n)$. This loop is terminated once answer YES or NO is obtained from

Algorithm 1. Completeness of this restarted approach is guaranteed by the assumption that $R(n) \to \infty$ as $n \to \infty$. Recall that randomization is facilitated by tie-breaking in the heuristic function used in Algorithm 1 and by randomization of the value ordering.

The basic trends we observed for all tested parameter values are represented by Fig. 2: (i) restarts significantly reduce runtime expectation for both satisfiable and unsatisfiable instances, (ii) unsatisfiable instances take much longer to prove in the restarted approach. Observation (i) alone motivates to use the restarted variant of Algorithm 1 as a fast complete method for subsumption testing. We explore this idea elsewhere (Kuželka & Železný, 2009), whereas this paper addresses observation (ii). This observation is easily explained: while satisfiability can in principle be shown in any single restart, unsatisfiability can only be shown after n restarts making $R(n)$ sufficiently high. We would like to avoid the runtime components corresponding to $R(n)$ series growing to excessive values.

4. ReCovEr: A Restarted Coverage Estimator

We first explain the intuition underlying ReCovEr. We are given a clause C, and example set E and we would like to estimate the coverage $cov(C, E) = |\{e \in E \mid C \preceq_\theta e\}|$. Let us run Algorithm 1 on C and e, successively for all $e \in E$. For each e, we however stop the algorithm if no decision has been made in R steps. Let $\mathcal{E} \subseteq E$ be the subset of examples proven to be subsumed by C in this experiment. Denote $s_1 = |\mathcal{E}|$. We now remove all examples in $\mathcal{E}$ from E and repeat this experiment, obtaining analogical number s_2. Further such iterations generate numbers s_3, s_4, etc. Clearly, for the desired value $cov(C, E)$, we have that $cov(C, E) = \lim_{j\to\infty} S_j$ where $S_j = \sum_{i=1}^{j} s_i$. Under a certain assumption, the series S_j is geometrical rather than arbitrary. The main idea of ReCovEr is that the limit of S_j for $j \to \infty$ can thus be estimated by extrapolating the series from its first few elements $S_1, S_2, \ldots$. Thus we achieve a coverage estimate without excessive effort to refute subsumption for the examples not subsumed by C.

In order to precisely derive an estimation algorithm following the above idea, we first need to make the following assumption.

Assumption 4.1 *Given a clause C and a set of examples E, the probability p that Algorithm 1 finds a solution (i.e. returns YES as its answer) before it explores more than R nodes of the search tree, is the same for all $e \in E$ such that C subsumes e.*

Algorithm 2 *ReCovEr*(C, E, R, M, Δ): Algorithm for coverage estimation

Input: Clause C and set of examples E, Integers R ('cutoff'), M, Δ;

```
tries ← 0
Unknown ← Examples
CoveredInIthTry ← []
repeat
  tries ← tries + 1
  CoveredInThisTry ← 0
  for ∀E ∈ Unknown do
    Answer ← Run SubsumptionCheck(C, E) with number of
    searched nodes limited to R
    if Answer = PositiveMatching then
      CoveredInThisTry ← CoveredInThisTry + 1
      Unknown ← Unknown\E
    end if
  end for
  CoveredInIthTry[tries] ← CoveredInThisTry
until TerminationCondition
return LikelihoodEstimate(tries)
```

In other words, we assume that properties of particular examples such as their size are not dramatically different. The assumption will be empirically validated in the next section.

We assume a given clause C and we fix a constant cutoff value R. In the first step, for each $e \in E$ we run $SubsumptionCheck(P, e)$ (Algorithm 1), stopping it as soon as the number of searched nodes has reached R. Then, after $|E|$ restarts (each time with a different $e \in E$), we can derive the probability that the algorithm has produced exactly m_1 'YES' responses in this first step. In particular, this probability $P(m_1)$ is

$$P(m_1) = \binom{A}{m_1} p^{m_1}(1-p)^{A-m_1} \tag{1}$$

where $A = |\{e \in E | C \preceq_\theta e\}|$. In the next step, all m_1 examples shown to be subsumed in the first step are removed from E and the procedure is repeated with the remaining examples. In general, we can derive the probability that exactly m_i YES answers are generated in the i-th step. Thus for $i = 2$, we obtain

$$P(m_2|m_1) = \binom{A-m_1}{m_2} p^{m_2}(1-p)^{A-m_1-m_2} \tag{2}$$

and similarly for an arbitrary $i \geq 1$, we have

$$P(m_i|m_{i-1}, \ldots, m_1) = \binom{A - \sum_{j=1}^{i-1} m_j}{m_i} p^{m_i}(1-p)^{A-\sum_{j=1}^{i} m_j} \tag{3}$$

The probability of a sequence $(m_1, \ldots, m_k)$, where m_i is the number of examples for which YES was produced in the i-th step, is given by

$$P(m_1, \ldots, m_k) = \prod_{i=1}^{k} P(m_i | m_{i-1}, \ldots, m_1) \quad (4)$$

Substituting for $P(m_i|m_{i-1}, \ldots, m_1)$ from Eq. 3 and taking the logarithm Eq. 4 results in

$$\ln\left(P(m_1, \ldots, m_k)\right) = \sum_{i=1}^{k} (\alpha + m_i \ln p + \beta) \quad (5)$$

where

$$\alpha = \ln \begin{pmatrix} A - \sum_{j=1}^{i-1} m_j \\ m_i \end{pmatrix}$$

and

$$\beta = \left(A - \sum_{j=1}^{i} m_j\right) \ln(1-p)$$

To find the parameters A and p for which $P(m_1, \ldots, m_k)$ is maximized, we take the partial derivative of Eq. 5 with respect to p and then find its roots, yielding

$$p = \frac{\sum_{i=1}^{k} m_i}{\sum_{i=1}^{k} m_i + \sum_{i=1}^{k} \left(A - \sum_{j=1}^{i} m_j\right)} \quad (6)$$

Finding the global maximum of $P(m_1, \ldots, m_k)$ from Eq. 4 on the set

$$D = \{(A, p) | A \in \{1, 2, \ldots, |E|\} \wedge p \in [0; 1]\} \quad (7)$$

is now straightforward, since using (6) we can find the maximum on every line

$$L_i = \{(i, p) | p \in [0; 1]\} \quad (8)$$

The maximum on line L_i is located either at the value of p given by (6) or at one of the borders of L_i. It then suffices to evaluate (4) at these three points of L_i for every i ($1 \leq i \leq |E|$). The estimate of A then equals the index i of the L_i on which the maximum is located.

The described estimator is used in ReCovEr (Algorithm 2). The question how to choose k, i.e. how long a sequence $(m_1, \ldots, m_k)$ should be generated as the input to the estimator, is tackled iteratively: the sequence is being extended until a termination condition is met. We have considered several termination conditions, of which two turned out to be quite useful. The first termination condition stops generating the sequence when two subsequent estimates differ by less than some Δ_e, specified as a parameter. The second termination condition stops generating the sequence when estimate and number of examples already shown to be covered by the clause differ again by less than some Δ_c, which ensures that the estimator will never overestimate the actual coverage by more than Δ_c. A minimum length M of the sequence is however imposed in both previous cases, to avoid premature estimates coinciding by chance.

Another degree of freedom in Algorithm 2 is the cutoff R, which may significantly affect the performance of the restarted algorithm. A heuristic method suggests itself that first tries to find a suitable cutoff. Unlike Algorithm 2 it starts with a base cutoff value, and then doubles it after every single restart. If at any restart Algorithm 1 with cutoff set to R covers fewer examples than the same algorithm at previous restart with cutoff set to $\frac{R}{2}$, then we can accept cutoff $\frac{R}{2}$.

5. Experiments

In this section, we first investigate the sensitivity of ReCovEr to a violation of Assumption 4.1. Then we evaluate its performance and precision on graph data generated by the two random graph generators described in Section 2 and on real-world data from organic chemistry and from engineering. We compare performance of ReCovEr with that of the state-of-the-art θ-subsumption algorithm Django.

5.1. Sensitivity Analysis

Here we address Assumption 4.1. Informally, we first want to verify (i) how the assumption deviates from the empirical 'truth', and subsequently, (ii) how much these deviations influence ReCovEr's precision.

(i) According to Assumption 4.1, probability $p \in [0; 1]$ would be a constant. Dismissing this assumption, we treat p as a random variable with some distribution on $[0; 1]$, which we would like to estimate. A standard approach to this task is to parameterize a Beta distribution on $[0; 1]$ from empirical data. To obtain the data, we experimented with the settings from Section 3 with parameters of generated hypotheses $c = 0.35$, $n = 10$, parameters of generated examples $c = 0.3$, $n = 50$ and ReCovEr's cutoff $R = 75$. This resulted in Beta distributions with standard deviation extending up to about 0.25 (i.e. 25% of the p's range). These distributions are plotted in dashed lines in Fig. 3.

(ii) Now we investigate ReCovEr's sensitivity to the modeled deviations. We assume to have $n = 100$ ex-

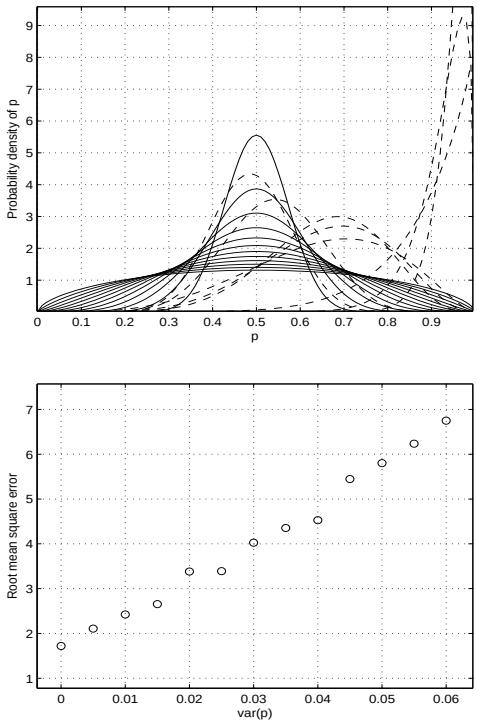


Figure 3. **Top:** Beta distributions with mean $\mu = 0.5$ and variance consecutively $0, 0.005, \ldots, 0.06$, which model the distribution of p (solid lines). Beta distributions fitted to actual probabilities are shown in dashed lines. **Bottom:** Dependence of root mean square error of ReCovEr's estimates on the variance of p.

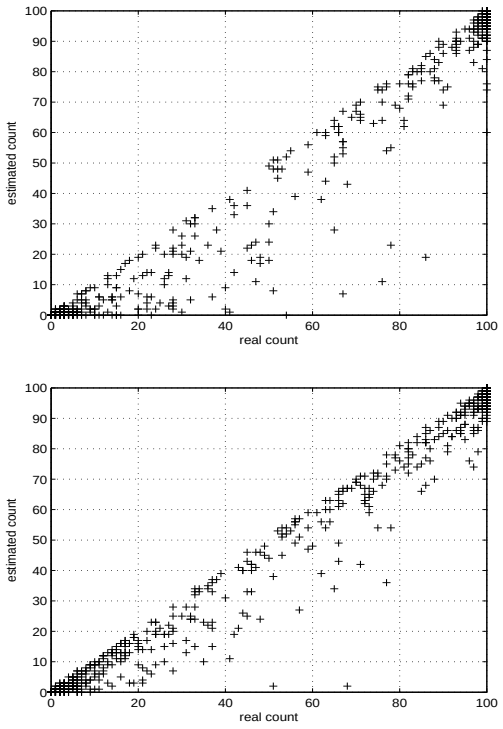


Figure 4. Precision of ReCovEr (Algorithm 2) presented as 1000 points with coordinates (estimated coverage, actual coverage). Hypotheses and examples were generated by the Erdos-Rényi random graph generator with $c = 0.3$, $n = 15$ for hypotheses and $c = 0.3$, $n = 100$ for examples. The 1000 estimates correspond to 1000 different hypotheses tested on a pre-fixed set of 100 examples. **Top:** Base value for cutoff is $R = 100$. **Bottom:** Base value for cutoff is $R = 200$.

amples, of which 50 were covered by a clause C. Further, probabilities p_i that Algorithm 1 finds a solution for a covered example e_i in time less than R were sampled from the Beta distribution with given mean $\mu = 0.5$ and variance consecutively $0, 0.005, \ldots, 0.06$ (i.e. growing up to the 25% standard deviation). Then, we simulated ReCovEr's estimation procedure on these data. The top panel of Fig. 3 displays the beta distributions (solid lines) from which probabilities p_i were sampled. We used the stopping condition based on difference of estimate and lower bound, the parameters were $M = 3$, $\Delta = 1$.

The bottom panel of Fig. 3 displays the dependence of root mean square error on the variance of the beta distributions in the top panel. It is encouraging to see that the root mean square error grows roughly *linearly* with growing variance in p's distribution, indicating ReCovEr's robustness towards this variance. Of course, the ultimate judge of whether this dependence is acceptable is the extent to which a learning algorithm based on ReCovEr would be affected by the estimation imprecision caused by the estimation. This is studied further.

5.2. Experiments with Generated Graph Data

Figure 4 demonstrates the precision of ReCovEr on the graph data generated by the Erdos-Rényi generator by showing 1000 pairs (estimated coverage, actual coverage). Hypotheses and examples were generated with $c = 0.3$ for hypotheses and $c = 0.3$ for examples. The graphs corresponding to hypotheses had 15 vertices, and the graphs corresponding to examples had 100 vertices. The top panel refers to estimates obtained by Algorithm 2 enhanced by cutoff selection with base cutoff $R = 100$, while the bottom panel refers to estimates obtained by the same algorithm with base cutoff $R = 200$. A bias towards coverage under-estimation can be observed, as well as a positive effect of the higher base cutoff on estimation precision.

Table 1 shows average runtimes of ReCovEr and Django. Table 2 shows average runtimes of Django and ReCovEr on artificial graph data with small-world topology generated by Algorithm 5. In this case, the graphs underlying the hypotheses had 15 vertices and their connectivity was $k = 4$. The graphs underlying the examples had 100 vertices and connectivity $k = 20$. A dramatic speedup from Django's runtime is exhibited in both cases.

Note that there is no immediate reason to avoid the under-estimation bias because coverage is usually tested on two example sets (positive and negative). The two results are usually subtracted thus (mostly) canceling the bias. Whether the observed estimation variance is tolerable for the task of clause *ranking* usual in inductive logic programming is the subject of the experiments in the next section.

Algorithm	Avg. Time [s]
ReCovEr, $R = 100$	6.7
ReCovEr, $R = 200$	12.5
Django	483.2

Table 1. Average coverage test runtimes for the configuration from Fig. 4.

Algorithm	Avg. Time [s]
ReCovEr, $R = 100$	8.9
ReCovEr, $R = 200$	16.3
Django	519.8

Table 2. Average coverage test runtimes for the configuration with small world graph data.

5.3. Experiments with Real-World Data

In order to assess performance in conditions of a real-life learning setting, we decided not to generate clauses entirely randomly. Our intention was to simulate general principles of clause production in an inductive logic programming system, while avoiding an overfit to a specific clause search strategy (which would e.g. be a result of adhering to a specific heuristic function for selecting literals). Thus we developed a simple relational learner, which we use for further experiments with ReCovEr. The learner (Algorithm 3) is a randomized variation of a specific-to-general beam search. It starts with the most specific clause $\perp$ and at each search step, it generates at least $n \cdot |Beam|$ new hypotheses by removing random subsets of literals from the hypotheses already present in $Beam$. The output of the algorithm is one best clause, which is why we assess its quality through precision and recall.

Algorithm 3 $Learner(\perp, p, BeamSize, Tries)$: A Clause Learner

Input: Most specific clause $\perp$, Real numbers p, Integers $BeamSize$, $MaxSearched$

$Beam \leftarrow \{\perp\}$
$BestClause \leftarrow \perp$
repeat
  $Candidates \leftarrow Beam$
  for $\forall h_i \in Beam$ **do**
    for $i = 1 \ldots BeamSize$ **do**
      $GenerateClause(h_i)$
      $C \leftarrow$ connected components of c
      Evaluate each $c_i \in C$
      $Candidates \leftarrow candidates \cup C$
    end for
  end for
  for $\forall h \in Candidates$ such that h is estimated to be better than $BestClause$ **do**
    if h is shown to be better than $BestClause$ by a deterministic subsumption algorithm **then**
      $BestClause \leftarrow h$
    end if
  end for
  Choose $BeamSize$ best hypotheses from $Candidates$ and add them to $Beam$
  $Explored \leftarrow Explored + 1$
until $Beam = \{\}$ or $Explored = Tries$

The first set of experiments, which we have conducted with Algorithm 3, deals with the Mutagenesis dataset (Srinivasan et al., 1996). This dataset consists of descriptions of 188 organic molecules, which are marked according to their mutagenicity. In our experiments, we used only the information about atom-bond relationships and about types of atoms. We did not consider numerical parameters such as *lumo* or *logp*. Our relational-logic representation of these molecules consisted of ternary literals for atomic bonds $bond(at1, at2, bondType)$, unary literals representing types of particular bonds and unary literals for atom types. We have considered three variants of relational logic description of the molecules, with growing complexity (size of examples). The first version **Muta-v1** uses a naive representation. Here, each molecular bond is represented by a single literal $bond(at1, at2, bondType)$, thus imposing a bond orientation (atom order) chosen at random. The second source of imprecision of this representation is that two variables in a clause may represent the same (chemical) atom, which does not make intuitive sense. The second version **Muta-v2** deals with the first source of imprecision, as it represents every atomic bond with a pair of literals $bond(at1, at2, bondType)$ and $bond(at2, at1, bondType)$. The third version **Muta-v3** solves the second source of imprecision by adding literals $different(a, b)$ for all pairs of atom-representing constants a, b.

The second set of experiments pertains to class-labeled CAD data (product structures) described in (Žáková

et al., 2007), consisting of 96 CAD examples each containing several hundreds of first-order literals.

The main observation provided by the experiments is that ReCovEr becomes quickly superior to Django as the example size grows, whereas the two algorithms do not significantly differ in terms of the training-set[1] accuracy of the discovered clauses. It is interesting to note that Django's poor runtime performance on the learning tasks with large examples (**CAD** data and **Muta-v2**) was often due to occasional subsumption cases. Clearly, this is a manifestation of heavy tails present in Django's runtime distribution. Unlike Django, ReCovEr was exhibiting steady performance.

Dataset	ReCovEr [s]	**Django** [s]
Muta-v1	42	29
Muta-v2	513	1627
Muta-v3	1695	>5h
CAD	121	>2h

Table 3. Average runtimes of the learner (Algorithm 3, $p = 0.75$, $Tries = 10$) for real-world datasets.

Dataset	**Avg. Precision**	**Avg. Recall**
Muta-v1	0.84	0.61
Muta-v2	0.81	0.65
Muta-v3	0.83	0.84
CAD	0.92	0.7

Table 4. Quality of learned hypotheses for ReCovEr

Dataset	**Avg. Precision**	**Avg. Recall**
Muta-v1	0.86	0.6
Muta-v2	0.82	0.65
Muta-v3	n.a.	n.a.
CAD	n.a.	n.a.

Table 5. Quality of learned hypotheses for Django

6. Conclusions

In this paper, we have introduced ReCovEr, an algorithm exploiting restarts for a maximum-likelihood based estimation of clause coverage. ReCovEr avoids heavy tails as well as laborious proving of certain unsatisfiable subsumption instances. We have shown that ReCovEr provides favorable runtimes while achieving reasonable precision, which is illustrated by experiments on synthetic graph data and on real-life data from organic chemistry and engineering. In future work we mainly want to develop theoretical bounds for ReCovEr's estimation precision.

Acknowledgements. We are grateful to ICML 2008 reviewers for their insightful comments and to the area chair for his extensive effort to understand and improve our initially unclear presentation. The authors are supported by the Grant Agency of the Czech Republic through the project 201/08/0486 Merging Machine Learning with Constraint Satisfaction.

[1] As this paper is not concerned with improving generalization performance, we did not measure accuracies on hold-out test sets.

References

Arias, M., Khardon, R., & Maloberti, J. (2007). Learning Horn expressions with Logan-H. *Journal of Machine Learning Research*, *8*, 549–587.

Giordana, A., & Saitta, L. (2000). Phase transitions in relational learning. *Machine Learning*, *41*, 217–251.

Gomes, C. P., Fernández, C., Selman, B., & Bessière, C. (2005). Statistical regimes across constrainedness regions. *Constraints*, *10*, 317–337.

Gomes, C. P., Selman, B., Crato, N., & Kautz, H. A. (2000). Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, *24*, 67–100.

Kuželka, O., & Železný, F. (2009). A restarted strategy for efficient subsumption testing. *Fundamenta Informaticae, spec. issue on multi-relational data mining. (Accepted).*

Maloberti, J., & Sebag, M. (2004). Fast theta-subsumption with constraint satisfaction algorithms. *Machine Learning*, *55*, 137–174.

Sebag, M., & Rouveirol, C. (1997). Tractable induction and classification in first-order logic via stochastic matching. *IJCAI97* (pp. 888–893). Morgan Kaufmann.

Srinivasan, A., Muggleton, S., Sternberg, M. J. E., & King, R. D. (1996). Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, *85*, 277–299.

Železný, F., Srinivasan, A., & Page, D. (2006). Randomised restarted search in ILP. *Machine Learning*, *64*, 183–208.

Žáková, M., Železný, F., Garcia-Sedano, J., Tissot, C. M., Lavrač, N., Křemen, P., & Molina, J. (2007). Relational data mining applied to virtual engineering of product designs. *ILP06* (pp. 439–453). Springer.

Query-Level Stability and Generalization in Learning to Rank

Yanyan Lan* LANYANYAN@AMSS.AC.CN
Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, 100190, P. R. China.

Tie-Yan Liu TYLIU@MICROSOFT.COM
Microsoft Research Asia, Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing, 100190, P. R. China.

Tao Qin* QINSHITAO99@MAILS.THU.EDU.CN
Department of Electronic Engineering, Tsinghua University, Beijing, 100084, P. R. China.

Zhiming Ma MAZM@AMT.AC.CN
Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, 100190, P. R. China.

Hang Li HANGLI@MICROSOFT.COM
Microsoft Research Asia, Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing, 100190, P. R. China.

Abstract

This paper is concerned with the generalization ability of learning to rank algorithms for information retrieval (IR). We point out that the key for addressing the learning problem is to look at it from the viewpoint of *query*. We define a number of new concepts, including query-level loss, query-level risk, and query-level stability. We then analyze the generalization ability of learning to rank algorithms by giving query-level generalization bounds to them using query-level stability as a tool. Such an analysis is very helpful for us to derive more advanced algorithms for IR. We apply the proposed theory to the existing algorithms of Ranking SVM and IRSVM. Experimental results on the two algorithms verify the correctness of the theoretical analysis.

1. Introduction

Recently, learning to rank has gained increasing attention in machine learning and information retrieval (IR). When applied to IR, learning to rank is a task as follows. Given a set of training queries, their associated documents, and the corresponding relevance judgments, a ranking model is created which best represents the relevance of documents with respect to queries. When a user submits a query to the IR system, the trained model assigns a score to each document associated with the query, sorts the documents based on their scores, and presents the top ranked documents to the user. Average ranking accuracy over a large number of queries is usually used to evaluate the effectiveness of a ranking model. Therefore, from the application's perspective, both training and evaluation should be conducted at query level.

Many learning to rank algorithms have been proposed in recent years. Examples include the pointwise ranking algorithms like MCRank (Li et al., 2007), the pairwise ranking algorithms like Ranking SVM (Herbrich et al., 1999) and RankBoost (Freund et al., 2003), and the listwise ranking algorithms like ListNet (Cao et al., 2007). Analysis on the algorithms in the light of statistical learning theory, however, was not sufficient, particularly that on the generalization ability of the proposed algorithms. The pointwise and pairwise approaches transform the ranking problem to classification or regression, and thus existing theory on classification and regression can be applied. However, it deviates from the direction of enhancing ranking accuracy at query level. Furthermore, the listwise approach lacks of analysis on generalization ability.

In this paper, we investigate the generalization ability of learning to rank algorithms, in particular from the viewpoint of query-level training and evaluation.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.
*The work was performed when the first and the third authors were interns at Microsoft Research Asia.

We propose a new probabilistic formulation of learning to rank for IR. The formulation can naturally represent the pointwise, pairwise and listwise approaches in a unified framework. Within the framework, we introduce the concepts of query-level loss, query-level risk, and particularly query-level stability. Query-level stability measures whether the output of a learning algorithm changes largely with small changes in the training queries. With query-level stability as a tool we can conduct analysis on query-level generalization bounds of learning algorithms. A query-level generalization bound indicates how well one can enhance the expected ranking accuracy (corresponding to the expected risk) by enhancing the average ranking accuracy in training (corresponding to the empirical risk).

We take the algorithms of Ranking SVM (Joachims, 2002; Herbrich et al., 1999) and IRSVM (Cao et al., 2006; Qin et al., 2007) as examples, and apply the proposed theory to them. Our theoretical result shows that the query-level generalization bound of Ranking SVM is not reasonably good, mainly because Ranking SVM is trained at document pair level, not query level. Furthermore, IRSVM does have a better generalization bound than Ranking SVM, due to its stronger query-level stability. We also conducted experiments and our experimental results agree with the theoretical findings.

The contributions of this paper are listed as follows. (1) A proposal on conducting analysis on learning to rank algorithms at query level is made. (2) A new probabilistic formulation of learning to rank is proposed. (3) A new methodology for analyzing generalization ability of learning to rank algorithms on the basis of query-level stability is proposed. (4) The proposed theory is applied to learning to rank algorithms of Ranking SVM and IRSVM. The correctness of the theory has been verified by experiments.

2. Previous Work

2.1. Ranking in IR

Ranking is a central issue for IR. Many methods for creating ranking models have been proposed, including heuristics and learning based methods, (Baeza-Yates & Ribeiro-Neto, 1999; Herbrich et al., 1999; Joachims, 2002; Freund et al., 2003; Burges et al., 2005; Cao et al., 2007). Typically a ranking model is defined as a function of features based on query-document pair, and is learned with training data containing a number of queries, associated documents, and corresponding relevance judgments. Measures for evaluating the performance of a ranking model, such as Precision, MAP (Baeza-Yates & Ribeiro-Neto, 1999), and NDCG (Järvelin & Kekäläinen, 2002) have been defined and used. All the measures are query-based; if the evaluation measure for a query q is $EV(q)$, then the averaged $EV(q)$ on a number of queries is used. From the application's perspective, both training and testing in learning to rank should be conducted at query level.

2.2. Learning to Rank

So far learning to rank has been addressed by the pointwise, pairwise, and listwise approaches. In the pointwise approach (Li et al., 2007), ranking is transformed to regression or classification, and the loss function in learning is defined as a function of a single document. In the pairwise approach (Herbrich et al., 1999; Joachims, 2002; Freund et al., 2003; Cao et al., 2006), ranking is transformed to pairwise classification, and the loss function is defined on a document pair. In the listwise approach (Cao et al., 2007; Qin et al., 2007), document lists are viewed as learning instances and the loss function is defined on that basis.

Although many learning methods have been proposed, theoretical investigations on them were not sufficient. Since training and testing should be conducted at query level, studies on query-level generalization ability of learning algorithms are really needed. Unfortunately, it was missing in the previous work.

2.3. Stability Theory

The notion of stability (Devroye & Wagner, 1979) has been proposed for analyzing the generalization bounds of learning algorithms.

Bousquet et al. (Bousquet & Elisseeff, 2002) propose the theory of uniform leave-one-out stability. Based on it, the generalization bounds of classification algorithms such as Support Vector Machines (SVM) can be derived. Agarwal et al. (Agarwal & Niyogi, 2005) apply the stability tool to bipartite ranking.

We can apply the existing stability theory to get document level and document pair level generalization bounds. However, they may be not suitable for the task of IR. In this paper, we propose query-level stability and reveal the relation between query-level stability and query-level generalization bound.

3. Probabilistic Formulation for Ranking

As explained in Section 2, ranking in IR is evaluated at query level. Therefore, to design and evaluate a learning to rank algorithm, we should also look at it from

the query perspective. To this end, we give a novel probabilistic formulation of ranking for IR, which contains queries and their *associates* (documents, document pairs, or document sets) in two layers. We then introduce the notions of query-level loss and query-level risk.

Assume that query q is a random sample from the query space $\mathcal{Q}$ according to a probability distribution $P_{\mathcal{Q}}$. For query q, an associate $\omega^{(q)}$ and its ground-truth $g(\omega^{(q)})$ are sampled from space $\Omega \times \mathcal{G}$ according to a joint probability distribution D_q, where Ω is the space of associates and $\mathcal{G}$ is the space of ground truth. Here the associate $\omega^{(q)}$ can be a single document, a pair of documents, or a set of documents, and correspondingly the ground truth $g(\omega^{(q)})$ can be a relevance score (or class label), an order on a pair of documents, or a permutation (list) of documents. Let $l(f; \omega^{(q)}, g(\omega^{(q)}))$ denote a loss (referred to as *associate-level loss*) defined on $(\omega^{(q)}, g(\omega^{(q)}))$ and a ranking function f.

Expected query-level loss is defined as:

$$L(f;q) = \int_{\Omega \times \mathcal{G}} l(f; \omega^{(q)}, g(\omega^{(q)}))\, D_q(d\omega^{(q)}, dg(\omega^{(q)})).$$

Empirical query-level loss is defined as:

$$\hat{L}(f;q) = \frac{1}{n_q} \sum_{j=1}^{n_q} l(f; \omega_j^{(q)}, g(\omega_j^{(q)})),$$

where $(\omega_j^{(q)}, g(\omega_j^{(q)})), j = 1 \cdots, n_q$ stands for n_q associates of q, which are sampled i.i.d. according to D_q. The empirical query-level loss can be an estimate of the expected query-level loss. It can be proven that the estimation is consistent.

The goal of learning to rank is to select the ranking function f which can minimize the *expected query-level risk* defined as:

$$R_l(f) = E_{\mathcal{Q}} L(f;q) = \int_{\mathcal{Q}} L(f;q)\, P_{\mathcal{Q}}(dq). \tag{1}$$

In practice, $P_{\mathcal{Q}}$ is unknown. What we have are the training samples $(q_1, S_1), \cdots, (q_r, S_r)$, where $S_i = \{(\omega_1^{(i)}, g(\omega_1^{(i)})), \cdots, (\omega_{n_i}^{(i)}, g(\omega_{n_i}^{(i)}))\}, i = 1, \cdots, r$, and n_i is the number of associates for query q_i. Here $q_1, \cdots, q_r$ can be viewed as data sampled *i.i.d.* according to $P_{\mathcal{Q}}$, and $(\omega_j^{(i)}, g(\omega_j^{(i)}))$ as data sampled *i.i.d.* according to $D_{q_i}, j = 1, \cdots, n_i, i = 1, \cdots, r$.

Empirical query-level risk is defined as:

$$\widehat{R_l}(f) = \frac{1}{r} \sum_{i=1}^{r} \hat{L}(f; q_i). \tag{2}$$

The empirical query-level risk is an estimate of the expected query-level risk. It can be proven that the estimation is consistent.

This probabilistic formulation can cover most of existing learning to rank algorithms. If we let the associate to be a single document, a document pair, or a document set, we can respectively define pointwise, pairwise, or listwise losses, and develop pointwise, pairwise, or listwise approaches to learning to rank.

(a) *Pointwise Case*

Let $\mathcal{D}$ denote the document space. We use a feature mapping function $\phi : \mathcal{Q} \times \mathcal{D} \to \mathcal{X}(= R^d)$ to create a d-dimensional feature vector for each query-document pair. For each query q, suppose that the feature vector of a document is $x^{(q)}$ and its relevance score (or class label) is $y^{(q)}$, then $(x^{(q)}, y^{(q)})$ can be viewed as a random sample from $\mathcal{X} \times R$ according to a probability distribution D_q. If $l(f; x^{(q)}, y^{(q)})$ is a pointwise loss (square loss for example), then the expected query-level loss becomes:

$$L(f;q) = \int_{\mathcal{X} \times R} l\left(f; x^{(q)}, y^{(q)}\right)\, D_q\left(dx^{(q)}, dy^{(q)}\right).$$

Given training samples $(q_1, S_1), \cdots, (q_r, S_r)$, where $S_i = \{(x_1^{(i)}, y_1^{(i)}), \cdots, (x_{n_i}^{(i)}, y_{n_i}^{(i)})\}, i = 1, \cdots, r$, the empirical query-level loss of query $q_i, (i = 1, \cdots, r)$ turns out to be:

$$\hat{L}(f;q_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} l(f; x_j^{(i)}, y_j^{(i)}).$$

(b) *Pairwise Case*

For each query q, $z^{(q)} = (x_1^{(q)}, x_2^{(q)})$ stands for a document pair associated with it. Moreover, $y^{(q)} = 1$ if $x_1^{(q)}$ is ranked above $x_2^{(q)}$, $y^{(q)} = -1$ otherwise. Let $\mathcal{Y} = \{1, -1\}$. $(x_1^{(q)}, x_2^{(q)}, y^{(q)})$ can be viewed as a random sample from $\mathcal{X}^2 \times \mathcal{Y}$ according to a probability distribution D_q. If $l(f; z^{(q)}, y^{(q)})$ is a pairwise loss (hinge loss for example, (Herbrich et al., 1999)), then the expected query-level loss becomes:

$$L(q) = \int_{\mathcal{X}^2 \times \mathcal{Y}} l\left(f; z^{(q)}, y^{(q)}\right)\, D_q\left(dz^{(q)}, dy^{(q)}\right).$$

Given training samples $(q_1, S_1), \cdots, (q_r, S_r)$, where $S_i = \{(z_1^{(i)}, y_1^{(i)}), \cdots, (z_{n_i}^{(i)}, y_{n_i}^{(i)})\}, i = 1, \cdots, r$, the empirical query-level loss of query $q_i, (i = 1, \cdots, r)$ turns out to be:

$$\hat{L}(f;q_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} l(f; z_j^{(i)}, y_j^{(i)}).$$

(c) *Listwise Case*

For each query q, let $s^{(q)}$ denote a set of m documents associated with it, $\pi(s^{(q)}) \in \Pi$ denote a permutation of documents in $s^{(q)}$ according to their relevance degrees to the query, where Π is the space of all permutations

on m documents. $(s^{(q)}, \pi(s^{(q)}))$ can be viewed as a random sample from $\mathcal{X}^m \times \Pi$ according to a probability distribution D_q. If $l(f; s^{(q)}, \pi(s^{(q)}))$ is a listwise loss (cross entropy loss for example, (Cao et al., 2007)), then the expected query-level loss becomes:

$$L(q) = \int_{\mathcal{X}^m \times \Pi} l\left(f; s^{(q)}, \pi\left(s^{(q)}\right)\right) D_q\left(ds^{(q)}, d\pi\left(s^{(q)}\right)\right).$$

Given training samples $(q_1, S_1), \cdots, (q_r, S_r)$, where $S_i = \{(s_1^{(i)}, \pi(s_1^{(i)})), \cdots, (s_{n_i}^{(i)}, \pi(s_{n_i}^{(i)}))\}, i = 1, \cdots, r$, the empirical query-level loss of query $q_i, (i = 1, \cdots, r)$ turns out to be:

$$\hat{L}(f, q_i) = \frac{1}{n_i} \sum_{j=1}^{n_i} l(f; s_j^{(i)}, \pi(s_j^{(i)})).$$

4. Stability Theory For Query-level Generalization Bound Analysis

Based on the probabilistic formulation, we propose a novel concept named query-level stability. We further discuss how to use query-level stability to analyze the generalization ability of a learning to rank algorithm.

First, we give a definition to uniform leave-one-query-out associate-level loss stability. The stability of a learning algorithm represents the degree of change in the loss of prediction when randomly removing a query and its associates from the training data.

Definition 1. *Let $\mathcal{A}$ be a learning to rank algorithm, $\{(q_i, S_i), i = 1, \cdots, r\}$ be the training set, l be the associate-level loss function, and τ be a function mapping an integer to a real number. We say that $\mathcal{A}$ has uniform leave-one-query-out associate-level loss stability with coefficient τ with respect to l, if $\forall q_j \in \mathcal{Q}, S_j \in (\Omega \times \mathcal{G})^{n_j}, j = 1, \cdots, r, q \in \mathcal{Q}, (\omega^{(q)}, g(\omega^{(q)})) \in \Omega \times \mathcal{G}$, the following inequality holds:*

$$\left| l(f_{\{(q_i,S_i)\}_{i=1}^r}, \omega^{(q)}, g(\omega^{(q)})) - l(f_{\{(q_i,S_i)\}_{i=1,i\neq j}^r}, \omega^{(q)}, g(\omega^{(q)})) \right| \leq \tau(r).$$

Here $\{(q_i, S_i)\}_{i=1,i\neq j}^r$ stands for the samples $(q_1, S_1), \cdots, (q_{j-1}, S_{j-1}), (q_{j+1}, S_{j+1}), \cdots, (q_r, S_r)$, where (q_j, S_j) is deleted. $f_{\{(q_i,S_i)\}_{i=1}^r}$ stands for the ranking function learned from $\{(q_i, S_i)\}_{i=1}^r$. We will use the notations hereafter.

With the definition, we can obtain the following lemma. It states that, if an algorithm has uniform leave-one-query-out associate-level loss stability, it will be stable in terms of expected query-level loss and empirical query-level loss. For ease of explanation, we simply call the uniform leave-one-query-out associate-level loss stability *query-level stability*.

Lemma 1. *Let $\mathcal{A}$ be a learning to rank algorithm, $\{(q_i, S_i), i = 1, \cdots, r\}$ be the training set, and l be the associate-level loss function. If $\mathcal{A}$ has leave-one-query-out associate-level loss stability with coefficient τ with respect to l, then the following inequalities hold:*

$$\left| L(f_{\{(q_i,S_i)\}_{i=1}^r}, q) - L(f_{\{(q_i,S_i)\}_{i=1,i\neq j}^r}, q) \right| \leq \tau(r),$$
$$\left| \hat{L}(f_{\{(q_i,S_i)\}_{i=1}^r}, q) - \hat{L}(f_{\{(q_i,S_i)\}_{i=1,i\neq j}^r}, q) \right| \leq \tau(r).$$

Based on the concept of query-level stability, we can derive a query-level generalization bound, as shown in Theorem 1. The theorem states that if an algorithm has query-level stability, then with high probability over the samples, the expected query-level risk can be bounded by the empirical risk and a term which depends on the query number and parameters of the algorithm. Furthermore, the theorem quantifies the expected loss on new queries, which is exactly what we mean by query-level generalization.

Theorem 1. *Let $\mathcal{A}$ be a learning to rank algorithm, $(q_1, S_1), \cdots, (q_r, S_r)$ be r training samples, and let l be the associate-level loss function. If (1) $\forall (q_1, S_1), \cdots, (q_r, S_r), q \in \mathcal{Q}, (\omega^{(q)}, g\left(\omega^{(q)}\right) \in \Omega \times \mathcal{G}$, $\left| l\left(f_{(q_i,S_i)_{i=1}^r}, \omega^{(q)}, g\left(\omega^{(q)}\right)\right)\right| \leq B$, (2) $\mathcal{A}$ has query-level stability with coefficient τ, then $\forall \delta \in (0, 1)$ with probability at least $1 - \delta$ over the samples of $\{(q_i, S_i)\}_{i=1}^r$ in the product space $\prod_{i=1}^r \{\mathcal{Q} \times (\Omega \times \mathcal{G})^\infty\}$, the following inequality holds:*

$$R_l\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right) \leq \widehat{R_l}\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right) + 2\tau(r) + (4r\tau(r) + B)\sqrt{\frac{\ln\frac{1}{\delta}}{2r}}.$$

Proof. For clarity of the proof, we first give the following definitions:

$$\rho(\{(q_i, S_i)\}_{i=1}^r) \triangleq R_l\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right) - \widehat{R_l}\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right),$$
$$\int_{\Omega_1} \triangleq \int_{\mathcal{Q}} \int_{(\Omega\times\mathcal{G})^{n_1}} \cdots \int_{\mathcal{Q}} \int_{(\Omega\times\mathcal{G})^{n_r}}, \int_{\Omega_2} \triangleq \int_{\mathcal{Q}} \int_{\Omega\times\mathcal{G}},$$

$$P_1(d\omega) \triangleq D_{q^r}^{n_r}(dS_r) P_{\mathcal{Q}}(dq_r) \cdots D_{q^1}^{n_1}(dS_1) P_{\mathcal{Q}}(dq_1),$$
$$P_2(d\omega') \triangleq D_q(d\omega^{(q)}, dg(w^{(q)})) P_{\mathcal{Q}}(dq).$$

We then prove the theorem in two steps.

1) *Get the bound of*

$$\left| \rho(\{(q_i, S_i)\}_{i=1}^r) - \int_{\Omega_1} \rho(\{(q_i, S_i)\}_{i=1}^r) P_1(d\omega) \right|.$$

For this purpose, we get the upper bound of the following term first:

$$\left| \rho(\{(q_i, S_i)\}_{i=1}^r) - \rho(\{(q_i, S_i)\}_{i=1}^{r,j,q_j'}) \right|$$

where $\{(q_i,S_i)\}_{i=1}^{r,j,q_j'}$ means that query (q_j,S_j) is changed for another query (q_j',S_j'), where S_j' refers to $(w_1^{(j')},g(w_1^{(j')})),\cdots,(w_{n_j'},g(w_{n_j'}^{(j')}))$.

To utilize the query-level stability, we divide ρ into two terms: $\rho=\rho_1-\rho_2$, and discuss either of them separately, as follows.

$$\begin{aligned}&\rho_1(\{(q_i,S_i)\}_{i=1}^r)\triangleq R_l\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right)\\&=\int_{\Omega_2}l(f_{\{(q_i,S_i)\}_{i=1}^r};\omega^{(q)},g(\omega^{(q)}))P_2(d\omega^{'}).\\&\rho_2(\{(q_i,S_i)\}_{i=1}^r)\triangleq \widehat{R_l}\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right)\\&=\frac{1}{r}\sum_{i=1}^{r}\frac{1}{n_i}\sum_{j=1}^{n_i}l(f_{\{(q_i,S_i)\}_{i=1}^r};\omega_j^{(i)},g(\omega_j^{(i)})).\end{aligned}$$

Based on query-level stability, we can obtain that $\forall q_j\in\mathcal{Q},S_j\in(\Omega\times\mathcal{G})^{n_j},j=1,\cdots,r,q,q_j'\in\mathcal{Q},S_j'\in\{\mathcal{Q}\times(\Pi\times\mathcal{G})^{n_j'}\},(\omega^{(q)},g(\omega^{(q)}))\in\Omega\times\mathcal{G}$, the following inequality holds:

$$\begin{aligned}&\Big|l(f_{\{(q_i,S_i)\}_{i=1}^r},\omega^{(q)},g(\omega^{(q)}))\\&-l(f_{\{(q_i,S_i)\}_{i=1}^{r,j,q_j'}},\omega^{(q)},g(\omega^{(q)}))\Big|\le 2\tau(r).\end{aligned}\tag{3}$$

With (3), as ρ_1 is an integral function, the following inequality holds:

$$|\rho_1(\{(q_i,S_i)\}_{i=1}^r)-\rho_1(\{(q_i,S_i)\}_{i=1}^{r,j,q_j'})|\le 2\tau(r).\tag{4}$$

As for ρ_2, we have

$$\begin{aligned}&|\rho_2(\{(q_i,S_i)\}_{i=1}^r)-\rho_2(\{(q_i,S_i)\}_{i=1}^{r,j,q_j'})|\\&\le\frac{1}{r}\sum_{i=1,i\neq j}^{r}\frac{1}{n_i}\sum_{j=1}^{n_i}|l(f_{\{(q_i,S_i)\}_{i=1}^r};\omega_j^{(i)},g(\omega_j^{(i)}))\\&-l(f_{\{(q_i,S_i)\}_{i=1}^{r,j,q_j'}};\omega_j^{(i)},g(\omega_j^{(i)}))|\\&+\frac{1}{r}|\frac{1}{n_j}\sum_{s=1}^{n_j}l(f_{\{(q_i,S_i)\}_{i=1}^{n_j}};\omega_s^{(j)},g(\omega_s^{(j)}))\\&-\frac{1}{n_j'}\sum_{s=1}^{n_j'}l(f_{\{(q_i,S_i)\}_{i=1}^{r,j,q_j'}};\omega_s^{(j')},g(\omega_s^{(j')}))|\\&\le 2\tau(r)+\frac{B}{r}.\end{aligned}\tag{5}$$

By jointly considering (4) and (5), we obtain:

$$|\rho(\{(q_i,S_i)\}_{i=1}^r)-\rho(\{(q_i,S_i)\}_{i=1}^{r,j,q_j'})|\le 4\tau(r)+\frac{B}{r}.$$

Based on McDiarmid's inequality(McDiarmid, 1989), with probability at least $1-\delta$ over the samples of $\{(q_i,S_i)\}_{i=1}^r$ in the product space $\prod_{i=1}^r\{\mathcal{Q}\times(\Omega\times\mathcal{G})^\infty\}$, we have

$$\begin{aligned}\rho(\{(q_i,S_i)\}_{i=1}^r)\quad&\le\quad\int_{\Omega_1}\rho(\{(q_i,S_i)\}_{i=1}^r)\,P_1(d\omega).\\&+\quad(4r\tau(r)+B)\sqrt{\frac{\ln\frac{1}{\delta}}{2r}}.\end{aligned}\tag{6}$$

2) *Get the bound of* $\left|\int_{\Omega_1}\rho(\{(q_i,S_i)\}_{i=1}^r)\,P_1(d\omega)\right|$

$$\begin{aligned}&\int_{\Omega_1}\rho[\{(q_i,S_i)\}_{i=1}^r]P_1(d\omega)\\&=\int_{\Omega_1}\int_{\Omega_2}[l(f_{\{(q_i,S_i)\}_{i=1}^r};\omega^{(q)},g(\omega^{(q)}))]\,P_2(d\omega^{'})\,P_1(d\omega)\\&-\int_{\Omega_1}l(f_{\{(q_i,S_i)\}_{i=1}^r};\omega_j^{(i)},g(\omega_j^{(i)}))\,P_1(d\omega)\\&=\int_{\Omega}\int_{\Omega_2}[l(f_{\{(q_i,S_i)\}_{i=1}^r};\omega^{(q)},g(\omega^{(q)}))\\&-l(f_{\{(q_i,S_i)\}_{i=1}^r};\omega_j^{(i)},g(\omega_j^{(i)}))]\,P_2(d\omega^{'})\,P_1(d\omega).\\&=\int_{\Omega_1}\int_{\Omega_2}[l(f_{\{(q_i,S_i)\}_{i=1}^r};\omega^{(q)},g(\omega^{(q)}))\\&-l(f_{\{(q_i,S_i)\}_{i=1}^{r,i,q}};\omega_j^{(q)},g(\omega_j^{(q)}))]\,P_2(d\omega^{'})\,P_1(d\omega).\end{aligned}$$

The reason that the last equality holds is as follows. Because the integral is conducted over all of the samples, and the samples are *i.i.d.*, we can change the ith query in the training set for $(q,\omega^{(q)},g(\omega^{(q)}))$. Then by further using (3), we have:

$$\left|\int_{\Omega_1}\rho[\{(q_i,S_i)\}_{i=1}^r]P_1(d\omega)\right|\le 2\tau(r).\tag{7}$$

Merging Eq. (6) and (7) yields the inequality in the theorem. □

5. Case Study

Without loss of generality, we take existing algorithms of Ranking SVM (Joachims, 2002; Herbrich et al., 1999) and IRSVM (Cao et al., 2006; Qin et al., 2007) as examples to show how to analyze the query-level generalization bound of an algorithm, using the tool of query-level stability. Both of the two algorithms belong to the pariwise case of our probabilistic formulation. It should be noted that the framework is neither limited to these two algorithms nor to the pair-wise case, we leave the discussions on other algorithms or other approaches to our future work.

5.1. Generalization Bound of Ranking SVM

Ranking SVM is widely used in ranking for IR, which views document pair as associate of the query and minimizes:

$$\min_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^{n}l_h(f;z_i,y_i)+\lambda\|f\|_K^2,\tag{8}$$

where $l_h(f;z_i,y_i)$ is the hinge loss, and K is a kernel function in the Reproducing Kernel Hilbert Space (RKHS).

Using the conventional stability theory (Bousquet & Elisseeff, 2002), we can get the following lemma which shows the query-level stability of Ranking SVM.

Lemma 2. *If* $\forall x \in \mathcal{X}, K(x,x) \leq \kappa^2 < \infty$, *then Ranking SVM has query-level stability with coefficient* $\tau(r) = \frac{4\kappa^2}{\lambda r} \times \max_{\forall n_i, S_i} \frac{n_i}{\frac{1}{r}\sum_{i=1}^{r} n_i}$.

As for this lemma, we have the following discussions. (1) When r approaches infinity, suppose the mean and variance of the distribution of n_q are μ and σ^2 respectively. Then by the Law of Large Numbers and Chebyshev's inequality, $\forall 0 < \delta < 1, \forall \epsilon > 0, \exists R(\epsilon)$, if $r > R(\epsilon)$, with probability at least $1-\delta$, the following inequality holds:

$$max_{\forall n_i, S_i} \frac{n_i}{\frac{1}{r}\sum_{i=1}^{r} n_i} \leq \frac{1 + \frac{\sigma}{\mu\sqrt{\frac{\delta}{r}}}}{1 - \frac{\varepsilon}{\mu}}.$$

Therefore, $\tau(r) \leq \frac{4\kappa^2}{\lambda r} \frac{1+\frac{\sigma}{\mu\sqrt{\frac{\delta}{r}}}}{1-\frac{\varepsilon}{\mu}}$. That is, $\tau(r)$ will approach zero, with a convergence rate of $O(\frac{1}{\sqrt{r}})$, when r goes to infinity.

(2) When r is finite (which is the case in practice), we have no reasonable statistical estimation of the term $\max_{\forall n_i, S_i} \frac{n_i}{\frac{1}{r}\sum_{i=1}^{r} n_i}$. As a result, we can only get a loose bound for $\tau(r)$ as $\frac{4\kappa^2}{\lambda}$. That is, when r increases but is still finite, $\tau(r)$ does not necessarily decrease.

Based on the above lemma, we can further derive the generalization bound of Ranking SVM. In particular, as the function $f_{\{(q_i,S_i)\}_{i=1}^r}$ is learned from the training samples $(q_1, S_1), \cdots, (q_r, S_r)$, there is a constant C, such that, $\forall (q_1, S_1), \cdots, (q_r, S_r)$, $\left\| f_{\{(q_i,S_i)\}_{i=1}^r} \right\|_K \leq C$. Then, $\forall (q_1, S_1), \cdots, (q_r, S_r), z \in \mathcal{Z}, y \in \mathcal{Y}$, $l_h\left(f_{\{(q_i,S_i)\}_{i=1}^r}, z, y\right) \leq 1 + 2C\kappa$. By further considering Theorem 1, we obtain the following theorems.

Theorem 2. *If* $\forall x \in \mathcal{X}, K(x,x) \leq \kappa^2 < \infty$, *then for Ranking SVM,* $\forall \delta \in (0,1), \forall \epsilon > 0, \exists R(\epsilon)$, *if* $r > R(\epsilon)$, *then with probability at least* $1-2\delta$ *over the samples of* $\{(q_i, S_i)\}_{i=1}^r$ *in the product space* $\prod_{i=1}^r \{\mathcal{Q} \times (\mathcal{X} \times \mathcal{X} \times \mathcal{Y})^\infty\}$, *we have:*

$$R_l\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right) \leq \widehat{R_l}\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right) + \frac{8\kappa^2}{\lambda r} \frac{1+\frac{\sigma}{\mu\sqrt{\frac{\delta}{r}}}}{1-\frac{\varepsilon}{\mu}} + \left(\frac{16\kappa^2 \frac{1+\frac{\sigma}{\mu\sqrt{\frac{\delta}{r}}}}{1-\frac{\varepsilon}{\mu}} + \lambda(1+2C\kappa)}{\lambda}\right)\sqrt{\frac{\ln\frac{1}{\delta}}{2r}}.$$

Theorem 3. *If* $\forall x \in \mathcal{X}, K(x,x) \leq \kappa^2 < \infty$ *and we have no constraint on* r, *then for Ranking SVM,* $\forall \delta \in (0,1)$, *with probability at least* $1-\delta$ *over the samples of* $\{(q_i, S_i)\}_{i=1}^r$ *in the product space* $\prod_{i=1}^r \{\mathcal{Q} \times (\mathcal{X} \times \mathcal{X} \times \mathcal{Y})^\infty\}$, *we only have:*

$$R_l\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right) \leq \widehat{R_l}\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right) + \frac{8\kappa^2}{\lambda} + \left(\frac{16r\kappa^2 + \lambda(1+2C\kappa)}{\lambda}\right)\sqrt{\frac{\ln\frac{1}{\delta}}{2r}}.$$

Theorem 2 states that when the number of training queries tends to be infinity, with high probability the empirical query-level risk of Ranking SVM will converge to its expected query-level risk. However, when the number of training queries is finite, the expected query-level risk and empirical query-level risk are not necessarily close to each other, and the bound in Theorem 3 quantifies the difference, which is an increasing function of the number of training queries.

5.2. Generalization Bound of IRSVM

In IR application, the numbers of document pairs associated with different queries vary largely (See LETOR or other public dataset). In consideration of this, IRSVM, studied in (Cao et al., 2006) and (Qin et al., 2007), is an adaptive version of Ranking SVM to the IR applications, which minimizes:

$$\min_{f \in \mathcal{F}} \frac{1}{r}\sum_{i=1}^{r} \frac{1}{n_i}\sum_{j=1}^{n_i} l_h(f; z_j^{(i)}, y_j^{(i)}) + \| f \|_K^2 . \qquad (9)$$

We can prove the query-level stability of IRSVM as shown in Lemma 3. Due to space limitations, we omit the proof.

Lemma 3. *If* $\forall x \in \mathcal{X}, K(x,x) \leq \kappa^2 < \infty$, *then IRSVM has query-level stability* $\tau(r) = \frac{4\kappa^2}{\lambda r}$.

With a similar analysis to that for Ranking SVM, we obtain the following theorem.

Theorem 4. *If* $\forall x \in \mathcal{X}, K(x,x) \leq \kappa^2 < \infty$, *then for IRSVM,* $\forall \delta \in (0,1)$, *with probability at least* $1-\delta$ *over the samples of* $\{(q_i, S_i)\}_{i=1}^r$ *in the product space* $\prod_{i=1}^r \{\mathcal{Q} \times (\mathcal{X} \times \mathcal{X} \times \mathcal{Y})^\infty\}$, *we have:*

$$R_l\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right) \leq \widehat{R_{l_h}}\left(f_{\{(q_i,S_i)\}_{i=1}^r}\right) + \frac{8\kappa^2}{\lambda r} + \frac{16\kappa^2 + \lambda(1+2C\kappa)}{\lambda}\sqrt{\frac{\ln\frac{1}{\delta}}{2r}}.$$

The theorem states that when the number of training queries tends to be infinity, with high probability the empirical query-level risk of IRSVM will converge to its expected query-level risk. When the number of queries is finite, the bound in the theorem quantifies the difference between the two risks, which is a decreasing function of the number of training queries.

Remark 1. *By comparing Theorem 2 and Theorem 4, we can find that the convergence rates of the empirical query-level risk to the expected query-level risk for Ranking SVM and IRSVM are the same, i.e.* $O(\frac{1}{\sqrt{r}})$. *However, by comparing Theorem 3 to Theorem 4, we can see that for the case of finite* r, *the bound of IRSVM is much tighter than that of Ranking SVM.*

6. Experiments and Discussion

We conducted experiments on Ranking SVM and IRSVM to verify our theoretical results.

6.1. Query-level Stability

First, we conducted an experiment to compare the stabilities of Ranking SVM and IRSVM. We randomly sampled 1,200 queries from a search engine's data repository, each query associated with hundreds of documents and their relevance labels. There are five labels: "perfect", "excellent", "good", "fair", and "bad". We split the queries into three sets: a training set with 200 queries, a validation set with 500 queries, and a test set with 500 queries (we denote the test set as $\mathcal{T}$). The validation set was used to select the regularization parameter λ for Ranking SVM and IRSVM.

We first trained two ranking models with Ranking SVM and IRSVM, denoted as f_0 and $f_0^{'}$ respectively. Then we randomly deleted one query from the training set, and trained two new models with Ranking SVM and IRSVM, denoted as f_1 and $f_1^{'}$ respectively. We repeated this process 30 times, and created the models $f_1, f_2, \cdots, f_{30}$ an $f_1^{'}, f_2^{'}, \cdots, f_{30}^{'}$. Then on the test set, we compared the associate-level loss for f_0 with that for f_i, and obtained the difference Δ_i for Ranking SVM. Similarly, we computed $\Delta_i^{'}$ for IRSVM.

$$\Delta_i = \max_{q \in \mathcal{T}} \max_{z \in S_q} |l_h(f_0, z^{(q)}, y^{(q)}) - l_h(f_i, z^{(q)}, y^{(q)})|,$$

$$\Delta_i^{'} = \max_{q \in \mathcal{T}} \max_{z \in S_q} |l_h(f_0^{'}, z^{(q)}, y^{(q)}) - l_h(f_i^{'}, z^{(q)}, y^{(q)})|.$$

According to Definition 1, Δ_i can bound from below the query-level stability $\tau(r)(r = 200)$ of Ranking SVM. Similarly, $\Delta_i^{'}$ can bound from below the query-level stability $\tau(r)(r = 200)$ of IRSVM. In this regard, we can compare stabilities of Ranking SVM and IRSVM by comparing Δ_i and $\Delta_i^{'}$.

We list all the 30 values of Δ_i and $\Delta_i^{'}$ in Table 1. From it, we can see that Δ_i is always much larger than $\Delta_i^{'}$. The mean (or maximum) value of Δ_i over the 30 trials is 1.23 (or 4.53). It is about more than ten times higher than the mean (or maximum) value of $\Delta_i^{'}$, which is only 0.12 (or 0.27). Furthermore, the variance of Δ_i (i.e. 0.72) is also larger than that of $\Delta_i^{'}$ (i.e. 0.003). These results indicate that the query-level stability of RankSVM is not so good as that of IRSVM. (Note that Lemmas 2 and 3 hold for any r, the number of training queries. We simply set $r = 200$.)

6.2. Query-level Generalization Bounds

Next, we compared the performances of Ranking SVM and IRSVM, to verify the theoretical results on their query-level generalization bounds.

From Theorems 3 and 4 we can see that the bound for Ranking SVM is much looser than that for IRSVM, especially when the number of training queries r is large but finite. We interpret the result as follow.

The actual empirical risk and expected risk with respect to Ranking SVM are as follows.

$$\widehat{R_{l_h}}(f) = \frac{1}{n} \sum_{i=1}^{n} l_h(f; z^{(i)}, y^{(i)})), n = \sum_{i=1}^{r} n_i.$$

$$R_{l_h}(f) = \int_{\mathcal{X}^2 \times \mathcal{Y}} l_h(f; z, y) P(dz, dy).$$

In the definitions, only document pair but no query appears, and thus we call them the *pair-level risks*. For comparison, we also list the *query-level risks* for the learning to rank problem (See also Section 3) where hinge loss is used as associate-level loss.

$$\widehat{R_{l_h}}(f) = \frac{1}{r} \sum_{i=1}^{r} \frac{1}{n_i} \sum_{j=1}^{n_i} l_h(f; z^{(i)}, y^{(i)}).$$

$$R_{l_h}(f) = \int_{\mathcal{Q}} \int_{\mathcal{X}^2 \times \mathcal{Y}} l_h(f; z^{(q)}, y^{(q)})\, D_q(dz^{(q)}, dy^{(q)})\, P_{\mathcal{Q}}(dq).$$

By comparing the above formulas, we can clearly see that what is optimized in Ranking SVM (i.e. the pair-level risk) is not equal to what should be optimized (i.e. the query-level risks), unless every training query has the same number of document pairs, which is not true in practice. In contrast, it is easy to verify that what is optimized in IRSVM is exactly the query-level risk. Therefore, no surprisingly IRSVM has a better query-level generalization bound.

In summary, the theoretical results indicate that the performance of Ranking SVM on the test set in terms of a query-level measure should not be so good as that of IRSVM. We have verified this through experiments. We tested the ranking performances of Ranking SVM (RankSVM for short) and IRSVM on the test set, in terms of Precision and NDCG. The results are shown in Figure 1. Furthermore, MAP [1] for Ranking SVM is 0.39 and MAP for IRSVM is 0.41. From the results, we can see that IRSVM achieves better ranking performance than RankSVM, in terms of all the query-level measures. This is also consistent with the results reported in (Cao et al., 2006) and (Qin et al., 2007).

7. Conclusions

In this paper, we have studied the generalization ability of learning to rank algorithms for IR. A probabilistic formulation for ranking has been proposed, which covers ranking algorithms belonging to the pointwise,

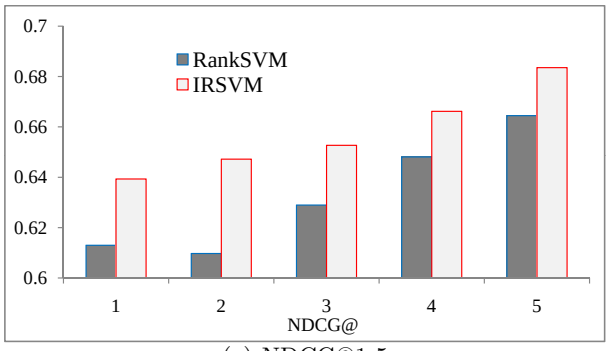
[1] To compute MAP, we treated "perfect", "excellent" and "good" as relevant, and "fair" and "bad" as irrelevant.

Table 1. Comparison of Query-level Stability

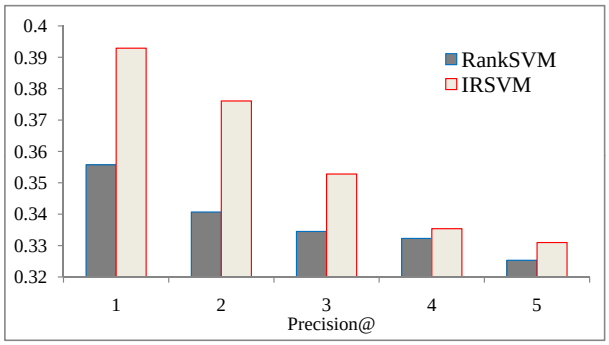
i	1	2	3	4	5	6
Δ_i	3.59	1.14	0.88	0.81	1.84	1.15
Δ_i'	0.07	0.07	0.06	0.06	0.05	0.24
	7	8	9	10	11	12
	0.89	1.30	0.90	1.42	1.38	1.39
	0.18	0.06	0.09	0.08	0.11	0.15
	13	14	15	16	17	18
	0.56	1.43	1.42	1.01	1.13	1.34
	0.11	0.13	0.14	0.11	0.06	0.11
	19	20	21	22	23	24
	1.04	0.86	0.43	0.51	0.64	0.92
	0.08	0.05	0.09	0.20	**0.27**	0.14
	25	26	27	28	29	30
	0.50	0.88	**4.53**	0.99	1.13	0.62
	0.18	0.08	0.12	0.09	0.21	0.14

pairwise and listwise approaches. The tool of query-level stability has been developed, which has been further used to analyze the generalization bound of a ranking algorithm. We have applied the tool to two existing ranking algorithms (Ranking SVM and IRSVM) and obtained theoretical results. We have also verified the correctness of the results by experiments.

As far as we know, this is the first work on query-level generalization bound of learning to rank algorithms. There are still many issues to investigate. (1) We have taken SVM based ranking algorithms as examples. We will try to obtain similar results for other algorithms, such as RankBoost. (2) We have focused on the pairwise approach. The proposed formulation for ranking and the tool of query-level stability can also be used to analyze other approaches. (3) It is worth checking whether new learning to rank algorithms can be derived under the guide of the theoretical study.



(a) NDCG@1-5



(b) Precision@1-5

Figure 1. Accuracies of Ranking SVM and IRSVM

References

Agarwal, S., & Niyogi, P. (2005). Stability and generalization of bipartite ranking algorithms. *Proc. of COLT'05* (pp. 32–47).

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval.* Addison Wesley.

Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research, 2*, 499–526.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *ICML '05* (pp. 89–96).

Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., & Hon, H.-W. (2006). Adapting ranking svm to document retrieval. *SIGIR '06* (pp. 186–193).

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. *ICML '07* (pp. 129–136).

Devroye, L., & Wagner, T. (1979). Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory, 25*, 601–604.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res., 4*, 933–969.

Herbrich, R., Obermayer, K., & Graepel, T. (1999). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers.* (pp. 115–132).

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst., 20*, 422–446.

Joachims, T. (2002). Optimizing search engines using clickthrough data. *KDD '02* (pp. 133–142).

Li, P., Burges, C., & Wu, Q. (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. *NIPS2007.*

McDiarmid, C. (1989). *On the method of bounded differences.* Cambridge University Press.

Qin, T., Zhang, X.-D., Tsai, M.-F., Wang, D.-S., Liu, T.-Y., & Li, H. (2007). Query-level loss functions for information retrieval. *Information Processing & Management.*

Modeling Interleaved Hidden Processes

Niels Landwehr NIELS.LANDWEHR@CS.KULEUVEN.BE
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A 3001 Heverlee, Belgium

Abstract

Hidden Markov models assume that observations in time series data stem from some hidden process that can be compactly represented as a Markov chain. We generalize this model by assuming that the observed data stems from *multiple* hidden processes, whose outputs interleave to form the sequence of observations. Exact inference in this model is NP-hard. However, a tractable and effective inference algorithm is obtained by extending structured approximate inference methods used in factorial hidden Markov models. The proposed model is evaluated in an activity recognition domain, where multiple activities interleave and together generate a stream of sensor observations. It is shown to be more accurate than a standard hidden Markov model in this domain.

1. Introduction

Hidden Markov models (HMMs) are among the most popular approaches for modeling time series data, and have seen widespread application in areas such as speech recognition, bioinformatics, or robotics. They assume that observed data stems from a hidden process which is stationary and Markov. However, in some application domains this single-process model is not appropriate. Consider for instance a log of web server requests, and assume we have no definite knowledge about which request has been issued by which user (e.g. because of proxy use). Clearly, there is no single hidden Markov process that accounts for the sequence of observed requests. Instead, there are multiple processes, one per user, which interleave to generate the sequence of observations. Another example, and the main motivation for the work presented in this paper, is *activity recognition*: the task of inferring a user's current activity from a stream of dense sensor data. In many situations, users switch back and forth between multiple activities, which causes sensor observations associated with the individual activities to interleave in time. The specific scenario considered in this paper is that objects used in activities of daily living are equipped with small RFID tags, which are picked up by a wearable RFID reader whenever a user interacts with the object. The task is to infer the sequence of activities carried out given the observed object interactions. In the light of recent advances in RFID technology, which allow tags to be cheaply mass-produced and readers to be made wearable, such application scenarios are attracting increasing research interest from both academia and industry (Wang et al., 2007).

HMMs have been widely used in activity recognition: activities are modeled as hidden states that emit the object tags observed by the RFID reader (Patterson et al., 2005). This is an appropriate model if activities are atomic and carried out sequentially. In many domains, however, activities are hierarchically structured, as sets of *basic* activities can be grouped into *high-level* activities. High-level activities typically interleave in time as a user is switching between them, as illustrated in Figure 1. In this example, a user is having breakfast, which consists of high-level activities makeToast, makeJuice, and getNews with corresponding basic activities. The domain could be modeled with a standard HMM by "flattening" the three activities into one process with 7 states, but in this case part of the problem structure would be lost. Alternatively, we can model the activities as three *different* processes which interleave in time. This has the advantage of decoupling transition dynamics within one high-level activity from the interleaving behavior, yielding a more concise representation with fewer parameters.

In this paper, we present a probabilistic model in which observations are generated by multiple, interleaved hidden processes. The hidden processes are stationary Markov chains, and the switching mechanism by which they interleave is again Markov. Although there exists a large body of related work, to the best

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

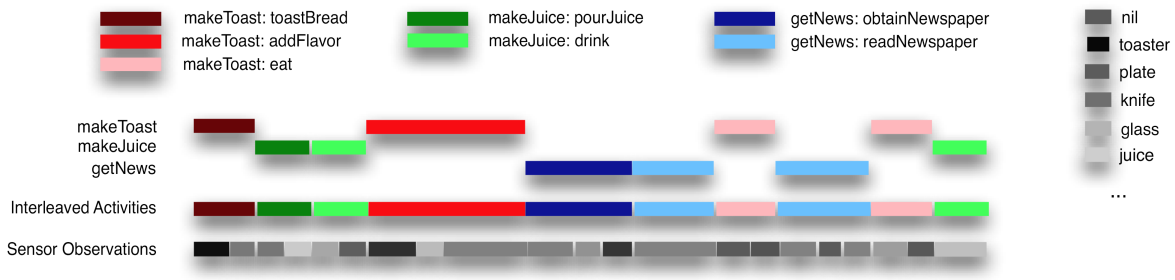


Figure 1. Interleaving in an activity recognition domain. Three high-level activities (makeToast, makeJuice, getNews) with corresponding basic activities are interleaved in time as a user switches between them. Different activities can produce identical sensor observations, and therefore neither the interleaving nor the actual activities are directly observable from the sensor data.

of our knowledge this interleaved setting has not been addressed before. A simplified version has been discussed in (Batu et al., 2004); however, tractable inference algorithms are not explored in this work. Simplicial mixtures of Markov chains, which employ a generative semantics similar to latent Dirichlet allocation, also address a similar problem (Girolami & Kabán, 2003). However, they restrict the constituent processes to be Markov rather than hidden Markov. A further class of models assumes several hidden processes that run in parallel, and that observations stem from their joint state. Examples include *factorial hidden Markov models* (Ghahramani & Jordan, 1997), *hidden Markov decision trees* (Jordan et al., 1996), *coupled hidden Markov models* (Brand, 1997) and *mixed hidden Markov models* (Altman, 2007). In contrast to our approach, these models focus on factorizing complex state spaces into cross-products of simpler components, rather than modeling interleaved processes. Another related technique are *switching state-space models* (SSSMs) (Ghahramani & Hinton, 1998), in which several processes run in parallel and an additional *switch* variable selects one *active* process from which the current observation is generated. SSSMs are different in that processes run concurrently, while an interleaving of processes is characterized by the fact that an inactive process is stopped and only resumes when it becomes active again. This creates additional dependencies between processes which cannot be modeled in a SSSM. Finally, *hierarchical hidden Markov models* model hierarchical structure within the hidden process that generates the observations (Fine et al., 1998). However, the component processes cannot interleave, and thus the model is not appropriate in our domain.

The next section introduces the proposed model more formally. Afterwards, we discuss the key problem of hidden state inference: given a sequence of observations, find the most likely configuration of hidden processes to have generated the data. Unfortunately, exact inference can be shown to be NP-hard; however, efficient structured approximate inference techniques can be applied (Section 3). Finally, the proposed technique is evaluated in an activity recognition domain, and shown to outperform standard HMM-based approaches (Section 4).

2. The Model

Let $Y_1, ..., Y_T$ denote a sequence of observations, where the Y_t take on one of D discrete values. A hidden Markov model μ (Rabiner, 1989) defines a sequence $X_1, ..., X_T$ of hidden state variables, with $X_t \in \{1, ..., K\}$ and K the number of different states the hidden process can take on. To simplify notation, assume that there is a special start state 0 the process is in at time $t = 0$, that is, $X_0 = 0$. The first transition is from X_0 to $X_1 \in \{1, ..., K\}$, and afterwards the first output Y_1 is emitted. The HMM is characterized by initial state probabilities $a_{0i} = P(X_1 = i \mid X_0 = 0)$, state transition probabilities $a_{ij} = P(X_t = j \mid X_{t-1} = i)$ for $t \geq 2$ and emission probabilities $b_{il} = P(Y_t = l \mid X_t = i)$ for $t \geq 1$. The joint distribution of observations $\mathbf{Y} = Y_1, ...Y_T$ and hidden states $\mathbf{X} = X_1, ..., X_T$ is given by

$$P(\mathbf{X}, \mathbf{Y}) = \prod_{t=1}^{T} P(X_t \mid X_{t-1}) P(Y_t \mid X_t).$$

We will also refer to $\mathbf{X}$ as the hidden process that generated the observations $\mathbf{Y}$.

We propose a model for multiple, interleaved hidden processes. Intuitively, an additional *switching process* controls a token that is handed from process to process, and determines which of the processes is active at a particular point t in time. The active process tran-

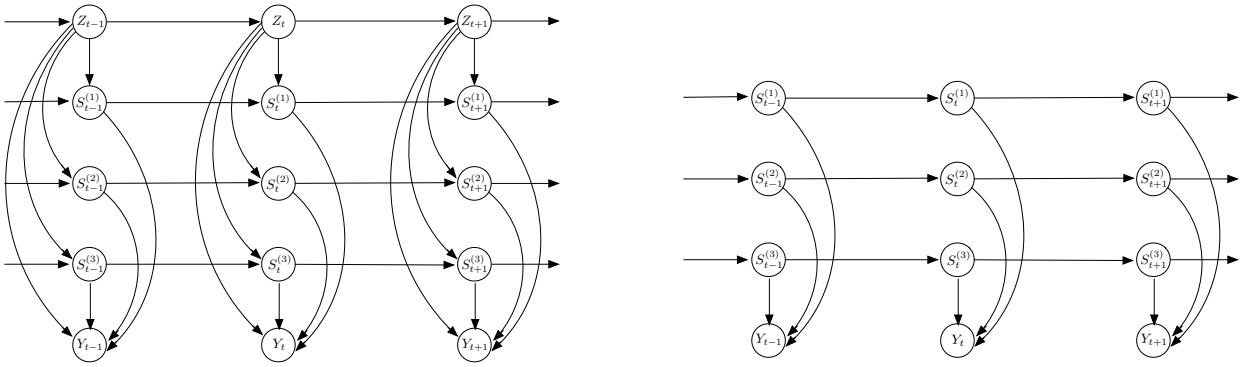


Figure 2. Interleaved mixture of hidden Markov models (left) and factorial hidden Markov model (right) in dynamic Bayesian network notation (for $M = 3$).

sitions to a new state and outputs the observation Y_t, while all other processes remain "frozen" in time.

More formally, let $\mu_1, ..., \mu_M$ be hidden Markov models with initial state probabilities $a_{0i}^{(m)}$, transition probabilities $a_{ij}^{(m)}$ and emission probabilities $b_{il}^{(m)}$. For ease of notation, we assume the number of states K is identical for all μ_m, but the model trivially generalizes to processes with state spaces of different size. Let furthermore $\bar{\mu}$ be a Markov process with states $\{1, ..., M\}$, initial state probabilities d_{0i} and transition probabilities d_{ij}. Let Z_t denote a random variable representing the state of $\bar{\mu}$ at time t, and $S_t^{(m)}$ denote random variables representing the state of process μ_m at time t for $1 \leq m \leq M$. $Z_t \in \{1, ..., M\}$ determines the *active* process at time t, and we will refer to $\bar{\mu}$ as the *switching process*. At every step t in time, a new active process is sampled from $\bar{\mu}$ with probability $P(Z_t = j \mid Z_{t-1} = i) = d_{ij}$. Afterwards, the states of $\mu_1, ..., \mu_M$ are updated according to

$$P(S_t^{(m)} = j \mid S_{t-1}^{(m)} = i, Z_t = k) = \begin{cases} a_{ij}^{(m)} & k = m; \\ \delta_{ij} & k \neq m, \end{cases} \tag{1}$$

where $\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $i \neq j$. In other words, a process μ_m transitions to a new state with probability given by its transition matrix if it is active at time t, and stays in its old state otherwise.

Finally, the probability of emitting symbol Y_t is

$$P(Y_t = l \mid S_t^{(1)} = i_1, ..., S_t^{(M)} = i_M, Z_t = k) = b_{i_k l}^{(k)} \tag{2}$$

That is, it is given by the emission probability of the process that is active at time t. Let $\mathbf{S}_t = S_t^{(1)}, ..., S_t^{(M)}$, $\mathbf{Z} = Z_1, ..., Z_T$ and $\mathbf{S} = \mathbf{S}_1, ..., \mathbf{S}_T$. Then

$$P(\mathbf{Z}, \mathbf{S}, \mathbf{Y}) = \prod_{t=1}^{T} P(Z_t \mid Z_{t-1}) P(Y_t \mid \mathbf{S}_t, Z_t) \prod_{m=1}^{M} P(S_t^{(m)} \mid S_{t-1}^{(m)}, Z_t) \tag{3}$$

We will refer to this model as an *interleaved mixture of hidden Markov models*. It is represented by the dynamic Bayesian network structure given in Figure 2 (left). The model is structurally related to a factorial hidden Markov model (Ghahramani & Jordan, 1997), shown in Figure 2 (right). However, the structure is extended by the additional chain of Z_t nodes that determine the currently active process. Although the structure is densely connected, the set of parameters is simply the union of the parameter sets of the constituent HMMs $\mu_1, ..., \mu_M$ and the switching process $\bar{\mu}$.

The following alternative interpretation of the model can be given. Let $\mathbf{z}$ denote an interleaving[1] and let $t_1^m, ..., t_{T^m}^m$ denote the sequence positions for which $z_t = m$. That is, $\mathbf{Y}_{\downarrow \mu_m} = Y_{t_1^m}, ..., Y_{t_{T^m}^m}$ is the projection of $\mathbf{Y}$ to elements generated by μ_m, and $\mathbf{S}_{\downarrow \mu_m} = S_{t_1^m}^{(m)}, ..., S_{t_{T^m}^m}^{(m)}$ the corresponding hidden state variables. It is easily verified that

$$P(\mathbf{Z}, \mathbf{S}, \mathbf{Y}) = P(\mathbf{Z}) \prod_{m=1}^{M} P_{\mu_m}(\mathbf{Y}_{\downarrow \mu_m}, \mathbf{S}_{\downarrow \mu_m}),$$

where $P_{\mu_m}(\mathbf{Y}_{\downarrow \mu_m}, \mathbf{S}_{\downarrow \mu_m})$ is the joint distribution of hidden states $\mathbf{S}_{\downarrow \mu_m}$ and observations $\mathbf{Y}_{\downarrow \mu_m}$ in the original HMM μ_m. This reformulation gives rise to an intuitive approach for sampling from $\mathbf{Y}$: first sample an interleaving pattern $\mathbf{z}$ from $\bar{\mu}$, and afterwards $\mathbf{Y}_{\downarrow \mu_m}$ from μ_m for $1 \leq m \leq M$.

[1] In general, we denote random variables with uppercase letters, and their instantiations with lower-case letters.

3. Inference and Parameter Estimation

A key task in the activity recognition domains we have in mind is *hidden state inference*: find

$$(\mathbf{z}^*, \mathbf{s}^*) = \operatorname*{argmax}_{\mathbf{z},\mathbf{s}} P(\mathbf{z}, \mathbf{s} \mid \mathbf{y}) \tag{4}$$

for a given sequence $\mathbf{y}$ of observations. This involves simultaneously finding a segmentation of $\mathbf{y}$ into subsequences $\mathbf{y}_{\downarrow\mu_m}$ generated by μ_m (the $\mathbf{z}^*$), and most likely hidden states for $\mathbf{y}_{\downarrow\mu_m}$ in μ_m (the $\mathbf{s}^*$).

3.1. Exact Inference

Two special cases of the problem are trivial. For $M = 1$, the model coincides with a hidden Markov model, and the Viterbi algorithm returns the most likely hidden states in time $O(K^2T)$. Moreover, if the output symbol sets of $\mu_1, ..., \mu_M$ are disjoint, the interleaving $\mathbf{z}$ is directly observable, and $\mathbf{s}$ can be obtained by running M instances of Viterbi in time $O(MK^2T)$.

The more interesting case of $M \geq 2$ and non-disjoint output symbol sets is inherently more difficult due to its combinatorial nature—the M constituent chains are coupled via the switching process and observations, and thus cannot be handled independently. Accordingly, exact graphical model inference (e.g. with the junction tree algorithm) applied to the model in Figure 2 (left) has costs exponential in M, because the cliques at Y_t are of size $O(M)$. In fact, for *general* graphical model structures of this form there is no tractable inference algorithm available. However, the conditional distributions $P(Y_t \mid \mathbf{S}_t, Z_t)$ have a particularly simple form, which could make the problem easier. Unfortunately, this is not the case:

Theorem. *Exact inference for interleaved mixtures of hidden Markov models is NP-hard.*

The theorem is proved by reduction from the strongly NP-hard *3-partition problem* (Garey & Johnson, 1975):

Problem (3-partition problem)**.** *Let S be a multiset of $M = 3N$ positive integers. Is there a partition of S into subsets $S_1, ..., S_N$ of size 3 each such that the sum over the integers in each subset is the same?*

A detailed proof is omitted for lack of space. Intuitively, the relationship is that an interleaving of $\mu_1, ..., \mu_M$ "partitions" a given sequence into the parts generated by the different processes (cf. Figure 1). Note that a key issue is the strong NP-hardness of 3-partition: the problem is NP-hard even if numbers in the input are given in unary notation (or, equivalently, if integers in S are polynomially bounded in M).

3.2. Approximate Inference

Approximate inference in graphical models has received much attention, and a variety of techniques are available. The most simple class of methods are Markov chain Monte Carlo (MCMC) approaches. In Gibbs sampling, for instance, iterative conditional resampling of random variables defines a Markov process whose stationary distribution—under certain conditions—will be the conditional distribution in Equation (4). However, MCMC is not an effective inference method in our case, because the Markov process defined by the Gibbs sampler is not ergodic. There can be two state configurations with positive probability that cannot be transformed into each other by single-variable changes without passing through an invalid (probability zero) configuration, such as any configuration with $S_{t-1}^{(m)} \neq S_t^{(m)}$ but $Z_t \neq m$. This effectively traps the Gibbs sampler in a subspace of all configurations and prevents MCMC convergence.

The problem is that Gibbs sampling, by updating only one variable at a time, ignores the specific model structure. Instead, we have to resort to approximate inference methods that better exploit model structure. Examples include structured variational approximations (Ghahramani & Jordan, 1997) and an iterative approximate inference method known as the chainwise Viterbi algorithm (Saul & Jordan, 1999). These algorithms are used in factorial HMMs for computing EM statistics and hidden state inference. In the rest of the Section, we present an extension of chainwise Viterbi for solving the problem given by Equation (4).

The idea behind chainwise Viterbi is to repeatedly solve tractable sub-problems of the (intractable) global optimization problem. For factorial hidden Markov models, the natural sub-problem to solve is to optimize hidden states in one chain $\mathbf{S}^{(m)} = S_1^{(m)}, ..., S_T^{(m)}$ conditioned on the current states of the other chains:

$$\begin{aligned}\mathbf{s}_{new}^{(m)} &= \operatorname*{argmax}_{\mathbf{s}^{(m)}} P(\mathbf{s}^{(m)} \mid \{\mathbf{s}^{(l)} : l \neq m\}, \mathbf{y}) \\ &= \operatorname*{argmax}_{\mathbf{s}^{(m)}} P(\mathbf{s}^{(1)}, ..., \mathbf{s}^{(M)}, \mathbf{y}).\end{aligned}$$

In the dynamic Bayesian network representing an interleaved mixture of HMMs (Figure 2, left), there are two different types of hidden chains: the chains $\mathbf{S}^{(1)}, ..., \mathbf{S}^{(M)}$ representing the constituent processes $\mu_1, ..., \mu_M$ and the chain $\mathbf{Z}$ representing the switching process $\bar{\mu}$. Assume first that $\mathbf{Z}$ is kept fix, and the goal is to conditionally optimize a chain $\mathbf{S}^{(m)}$. This is straightforward: for a given interleaving pattern, the chains $\mathbf{S}^{(1)}, ..., \mathbf{S}^{(M)}$ become independent given $\mathbf{Z}$ and $\mathbf{Y}$ due to the special form of the conditional distribu-

Algorithm 1 Chainwise Viterbi for interleaved mixtures of hidden Markov models

Input: model $\mathcal{M}$, observations $\mathbf{Y}$
$(\mathbf{S}, \mathbf{Z}) :=$ consistent-configuration($\mathcal{M}$)
while not converged **do**
  choose $m, n \in \{1, ..., M\}, m \neq n$
  let $(\mathbf{S}^{(m)}, \mathbf{S}^{(n)}, \mathbf{Z}) := \underset{\mathbf{S}^{(m)}, \mathbf{S}^{(n)}, \mathbf{Z}}{\operatorname{argmax}} P(\mathbf{S}, \mathbf{Z}, \mathbf{Y})$
end while
return $\mathbf{S}, \mathbf{Z}$

tions $P(Y_t \mid \mathbf{S}_t, Z_t)$, cf. Equation (2). They can thus be optimized independently with standard Viterbi.

We therefore focus on the task of optimizing $\mathbf{Z}$ given $\mathbf{S}^{(1)}, ..., \mathbf{S}^{(M)}$. A straightforward update

$$\mathbf{z}_{new} = \underset{\mathbf{z}}{\operatorname{argmax}} P(\mathbf{z}, \mathbf{s}^{(1)}, ..., \mathbf{s}^{(M)}, \mathbf{y})$$

is not very effective: as a process μ_m can only change state at time t if it is active, we know from $S_t^{(m)} \neq S_{t-1}^{(m)}$ that $Z_t = m$. Thus, the joint state of $\mathbf{S}^{(1)}, ..., \mathbf{S}^{(M)}$ essentially determines $\mathbf{Z}$. To change the state of Z_t from m to n, it is necessary to also update $S_t^{(m)}$ and $S_t^{(n)}$ to reflect that μ_n is now active at time t. The solution is to jointly optimize *two* constituent chains $\mathbf{S}^{(m)}$, $\mathbf{S}^{(n)}$ and the switching chain $\mathbf{Z}$ by

$$(\mathbf{s}_{new}^{(m)}, \mathbf{s}_{new}^{(n)}, \mathbf{z}_{new}) = \underset{\mathbf{s}^{(m)}, \mathbf{s}^{(n)}, \mathbf{z}}{\operatorname{argmax}} P(\mathbf{s}, \mathbf{y}, \mathbf{z}). \qquad (5)$$

Intuitively speaking, this update allows to re-assign observations that have so far been attributed to process μ_m to process μ_n, by changing some Z_t from m to n and updating $\mathbf{S}^{(m)}$ and $\mathbf{S}^{(n)}$ accordingly. If it is repeatedly applied with different process indices m, n, the interleaving can be arbitrarily revised. Algorithm 1 describes this chainwise update scheme in pseudocode. The method consistent-configuration($\mathcal{M}$) initializes the states of the hidden variables to some positive-probability configuration[2]. When choosing $m, n \in \{1, ..., M\}$ different strategies are possible; we assume the algorithm repeatedly cycles through all pairs $n \neq m$. If the update step (5) is implemented exactly, $P(\mathbf{s}, \mathbf{z}, \mathbf{y})$ will increase unless the hidden state configuration is left unchanged. Thus, the algorithm will always converge (though not necessarily to the true global optimum).

An efficient implementation of the update step (5) is crucial for fast inference. This can be achieved by dynamic programming in the spirit of the Viterbi algorithm (Rabiner, 1989). Moreover, the particularly restrictive form of the model (basically, that only the active chain changes state at any point in time) can be exploited. This allows much faster inference than for *general* graphical models with the DAG structure given in Figure 2 (left), as will be briefly outlined now.

To simplify notation, assume that $n = 1$ and $m = 2$. In analogy to the Viterbi algorithm, define

$$\delta_{ijk}[t] = \max_{\mathbf{D}} P(\mathbf{D}, S_t^{(1)} = i, S_t^{(2)} = j, Z_t = k, \mathbf{Y}, \mathbf{S}^{(3)}, ..., \mathbf{S}^{(M)})$$

with

$$\mathbf{D} = \{S_1^{(1)}, ..., S_{t-1}^{(1)}, S_1^{(2)}, ..., S_{t-1}^{(2)}, Z_1, ..., Z_{t-1}\}.$$

Initialization of $\delta_{ijk}[1]$ is straightforward. For the recursive definition of $\delta_{ijk}[t]$, let

$$C[k] = \prod_{m=3}^{M} P(S_t^{(m)} = s_t^{(m)} \mid S_{t-1}^{(m)} = s_{t-1}^{(m)}, Z_t = k),$$

where $s_{t-1}^{(m)}$, $s_t^{(m)}$ for $m \geq 3$ denote the current values of the fixed chains $\mu_3, ..., \mu_M$. Now two cases have to be considered. If $k \geq 3$, chains $1, 2$ cannot have changed state, and

$$\delta_{ijk}[t] = \max_{k'=1,...,M} \delta_{ijk'}[t-1] d_{k'k} b_{sy}^{(k)} C[k]$$

with $s = S_t^{(k)}$ and $y = Y_t$. This quantity can be computed in time $O(M)$. If $k \in \{1, 2\}$, we have to take into account state changes on the chains being optimized. Assume without loss of generality that $k = 1$. Now

$$\delta_{ij1}[t] = \max_{k'=1,...,M} \max_{i'=1,...,K} \delta_{i'jk'}[t-1] d_{k'1} a_{i'i}^{(1)} b_{iy}^{(1)} C[1],$$

with $y = Y_t$. This quantity can be computed in time $O(KM)$. There are $O(K^2MT)$ values of the form $\delta_{ijk}[t]$ to compute. However, time for computing all values is bounded by $O(K^2M(M+K)T)$, as the case $k \in \{1, 2\}$ only appears $O(K^2T)$ times.

The maximum probability of a hidden state configuration is

$$\max_{\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \mathbf{z}} P(\mathbf{s}, \mathbf{z}, \mathbf{y}) = \max_{ijk} \delta_{ijk}[T],$$

and a maximizing configuration is found by keeping track of where maxima occur in backtrace variables.

It is instructive to compare the complexity of the outlined chainwise Viterbi algorithm to inference in an HMM where hidden states are "flattened" into a single process. This HMM has a state space of size KM, and standard Viterbi has thus complexity $O(K^2M^2T)$, similar to the $O(K^2M(M+K)T)$ for a single update step in chainwise Viterbi. However, several such update steps will be needed before convergence.

[2] This is trivial if observation probabilities are always non-zero, as e.g. in Laplace-smoothed models.

3.3. Parameter Estimation

There are different possible settings for learning the proposed model from data. In the activity recognition setting discussed in Section 4, both sensor observations and activities are given for the training set. In this fully observable case maximum-likelihood model parameters can essentially be determined by counting. More generally, if the interleaving is known for the training data (that is, we know which part of each sequence has been generated by which process), the problem reduces to independently estimating the parameters of $\mu_1, ..., \mu_M$ with the standard Baum-Welch algorithm (Rabiner, 1989). In an unsupervised learning setting, expectation-maximization including the unknown interleaving $\mathbf{Z}$ is a natural choice. However, for the same reasons as discussed in Section 3, exact computation of the expectation step will be infeasible. In factorial hidden Markov models, this problem is solved elegantly by a structured variational approximation, and exploring variational inference methods for the interleaved mixture model presented in this paper is an interesting direction for future work. A simple alternative is to employ *hard EM*: instead of computing exact expectations, hidden states are set to their max-likelihood values given the observations, and expectations determined by counting. Together with the chainwise Viterbi algorithm discussed in Section 3.2 this yields a tractable method which is straightforward to implement.

4. Experimental Evaluation

The proposed model has been evaluated in an activity of daily living (ADL) recognition domain, where the goal is to infer a user's activity from a stream of dense RFID sensor data. The dataset has been collected in a real RFID environment at Intel Research Seattle (Landwehr et al., 2007). Objects are equipped with small RFID tags, and the user is wearing a lightweight RFID reader in a bracelet around the wrist. Whenever the reader comes close (10–15 centimeters) to a tagged object, the object tag is recorded. The sequence of observed tags thus indicates the objects a user has been interacting with while performing the activity.

We recorded activities involved in making breakfast at home, as this domain showcases the kind of interleaving behavior we are interested in (cf. Figure 1). The dataset consists of 20 sequences of RFID tag observations collected from 5 different persons having breakfast. Sequences are hand-labeled with the true current activity based on a human observer. There are 18 basic activities organized into 6 high-level activities, 24 different classes of tagged objects (including *nil* if no object was observed), and a total of 4597 timepoints to be classified. Timepoints at which no activity is taking place and activities with a coverage of less than 1% were removed, leaving 14 activities and 3545 timepoints in the dataset. The average number of segments into which a high-level activity is broken up because of interleaving is 3.95. There is significant overlap between observations associated with different activities, either because the same object is used in different activities or noise in the sensor data. More specifically, the average overlap in the set of observations associated with two different activities is 40.6%.

A standard approach in ADL recognition is based on HMMs: each basic activity corresponds to a hidden state, and sensor data to observations. In the described domain this means that all activities are "flattened" into one hidden process, and their hierarchical structure is lost. This approach will serve as a baseline, denoted by HMM. Alternatively, high-level activities can be modeled as separate hidden processes using the model described in Section 2. Here we consider a slight extension of this model: state transition probabilities in the active hidden process μ_{Z_t} depend not only on the previous state but also on whether or not the process has just become active; that is, $Z_t \neq Z_{t-1}$. The motivation for this extension is that high-level activities are typically interrupted at a point where the basic activity changes as well. It is straightforward to generalize the model and algorithms discussed in Section 2 and Section 3 to include this dependency.

Each high-level activity A is represented as a process μ_A, and the state space of μ_A are the basic activities associated with A. Note that the method, when applied to a given observation sequence, will automatically chose the (approximately) most likely subset of high-level activities that explains the observations. This model, together with the approximate inference technique discussed in Section 3.2 will be denoted as HMMmix. In the chainwise Viterbi algorithm, hidden states are initialized to the most likely activity given the current sensor observation (as observed in the training data). Furthermore, a version with exact inference (denoted HMMmix*) is run for comparison.

The experimental study seeks to answer the following two questions:

(Q1) Does reconstruction accuracy increase if high-level activities are modeled as separate processes?

(Q2) Does the approximate inference algorithm for HMMmix yield results similar to exact inference?

The rationale behind (**Q1**) is that modeling high-level

Table 1. Average cross-validated accuracy for MAJORITY, MAJORITY/OBSERVATION, HMM, HMMMIX and HMMMIX* on the ADL dataset. • indicates that result for HMMMIX is significantly better than result for other method (paired two-sided t-test, $p = 0.05$).

Method	Accuracy
MAJORITY	21.2 ± 25.4•
MAJORITY/OBSERVATION	71.4 ± 10.3•
HMM	84.0 ± 9.8•
HMMMIX	$\mathbf{86.0} \pm 8.6$
HMMMIX*	$\mathbf{86.0} \pm 8.6$

activities as separate processes will capture transition dynamics more concisely, as it decouples dynamics within a high-level activity from the switching dynamics. This is reflected in the number of model parameters: The "flattened" HMM representation requires $O((MK)^2) = O(M^2K^2)$ parameters to specify transition dynamics, while HMMMIX only requires $O(M^2 + MK^2)$ parameters.

To evaluate the different approaches, we performed a leave-one-sequence-out cross-validation. On the respective training set, models are estimated from fully observable training data, i.e., information on both sensor observations and activities is available. Given a test sequence, the most likely joint state of hidden variables in the model is determined, yielding a prediction of the current basic activity at every point in time. This is compared against the known true activity, and average prediction accuracy is computed. Table 1 shows reconstruction accuracy for HMM, HMMMIX and HMMMIX*. Additionally, accuracy for always predicting the most frequent activity (MAJORITY), and the most frequent activity given a particular sensor observation (MAJORITY/OBSERVATION) are shown. HMMMIX significantly outperforms HMM (paired two-sided t-test, $p = 0.05$), and predictions made by HMMMIX and HMMMIX* are identical in this experiment. This affirmatively answers questions **Q1** and **Q2**. Figure 3 shows the convergence behavior of chainwise Viterbi. The normalized log-likelihood of the current configuration of hidden states and the reconstruction accuracy given by this configuration are plotted as a function of the algorithm iteration. As expected, both likelihood and accuracy increase as the algorithm repeatedly revises the current interleaving. Furthermore, convergence occurs after a small number of iterations.

There are two sources of information for predicting the activity at a point t in time: the current sensor observation, and transition dynamics for activities

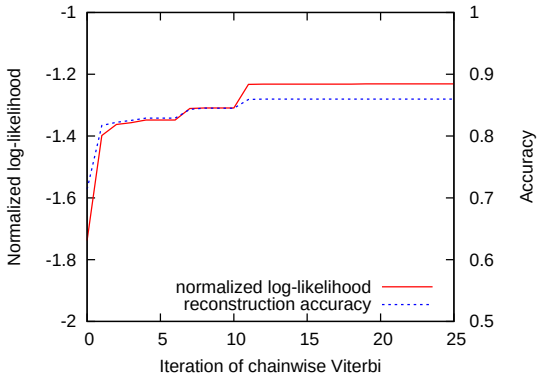


Figure 3. Normalized log-likelihood and reconstruction accuracy of the current hidden state configuration as a function of the number of iterations in chainwise Viterbi. Results are averaged over all test sequences.

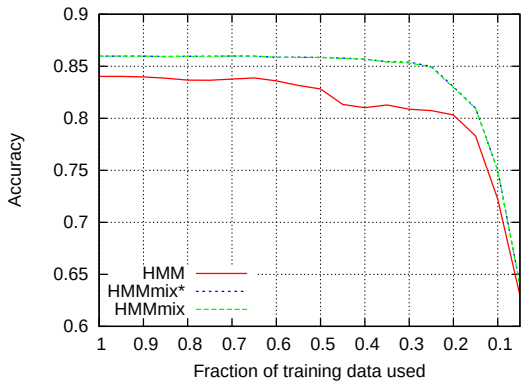


Figure 4. Reconstruction accuracy as a function of the fraction γ of training sequences used to estimate model transition parameters, while emission probabilities are estimated from all available training data. Results are averaged over 5 runs of cross-validation.

(which capture the influence of past and future observations on the current prediction). The MAJORITY/OBSERVATION approach already performs well; this indicates that much information is obtained simply from the current sensor observation. To further investigate the influence of transition dynamics on reconstruction accuracy, the following experiment was carried out. When estimating a model from data, only a randomly selected fraction γ of the training sequences is used to estimate transition probabilities, while all available data is used to estimate emission probabilities. Figure 4 shows reconstruction accuracy as a function of γ. The experiment confirms that HMMMIX outperforms HMM, and that approximate inference gives solutions very close to those of exact inference (solutions differ slightly, but the curves for HMMMIX and HMMMIX* in Figure 4 are indistin-

guishable). Moreover, the difference between HMM and HMMmix is most pronounced if only 20% to 40% of training sequences are used to estimate transition parameters. This supports the hypothesis that the more concise representation of transition dynamics in HMMmix (with fewer model parameters) explains its superior performance, as a concise representation matters most if training data is sparse.

5. Conclusions and Related Work

We have introduced a model for interleaved mixtures of hidden processes, which was shown to be superior to a single-process model in an activity recognition domain. The model should be generally applicable in situations where only the interleaved output of several independent processes can be observed. Related work includes several extensions of hidden Markov models (as discussed in Section 1), and activity recognition approaches based on HMMs such as (Patterson et al., 2005) and (Zhang et al., 2007) or dynamic Bayesian networks (Wang et al., 2007). The proposed method not only labels sequence positions but returns a structured parse of the sequence in terms of a set of hidden processes. Thus, it is also related to segmentation models, grammar-based approaches, and more generally models for predicting structured data (see (Bakir et al., 2007) for an overview). Directions for future work include semi- and unsupervised learning settings, and testing the model in different domains and on larger activity recognition datasets.

Acknowledgments The author would like to thank Matthai Philipose and Intel Research Seattle for making the activity recognition dataset available, and Luc De Raedt, Ingo Thon, Bernd Gutmann and Siegfried Nijssen for helpful discussions. The comments from the anonymous reviewers also helped to improve the paper. This work was supported by the Research Foundation-Flanders (FWO-Vlaanderen), and GOA/08/008 project "Probabilistic Logic Learning".

References

Altman, R. M. (2007). Mixed hidden Markov models: An extension of the hidden Markov model to the longitudinal data setting. *Journal of the American Statistical Association*, *102*, 201–210.

Bakir, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (2007). *Predicting Structured Data (Neural Information Processing)*. The MIT Press.

Batu, T., Guha, S., & Kannan, S. (2004). Inferring Mixtures of Markov Chains. *Proceedings of the 17th Annual Conference on Learning Theory*.

Brand, M. (1997). *Coupled hidden Markov models for modeling interactive processes* (Technical Report 405). MIT Media Lab.

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, *32*, 41–62.

Garey, M. R., & Johnson, D. S. (1975). Complexity Results for Multiprocessor Scheduling under Resource Constraints. *SIAM Jour. Comp.*, *4*, 397–411.

Ghahramani, Z., & Hinton, G. E. (1998). *Switching State-Space Models* (Technical Report). Department of Computer Science, University of Toronto.

Ghahramani, Z., & Jordan, M. I. (1997). Factorial Hidden Markov Models. *Mach. Lear.*, *29*, 245–273.

Girolami, M., & Kabán, A. (2003). Simplicial Mixtures of Markov Chains: Distributed Modelling of Dynamic User Profiles. *Proc. of the 17th Ann. Conference on Neural Information Processing Systems*.

Jordan, M. I., Ghahramani, Z., & Saul, L. K. (1996). Hidden Markov Decision Trees. *Proceedings of the 9th Conference on Advances in Neural Information Processing Systems*.

Landwehr, N., Gutmann, B., Thon, I., Philipose, M., & De Raedt, L. (2007). Relational Transformation-based Tagging for Human Activity Recognition. *Proc. of the Intern. Workshop on Knowledge Discovery from Ubiquitous Data Streams*.

Patterson, D., Fox, D., Kautz, H., & Philipose, M. (2005). Fine-Grained Activity Recognition by Aggregating Abstract Object Usage. *Proc. of the 9th IEEE Intern. Symp. on Wearable Computers*.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–286.

Saul, L. K., & Jordan, M. I. (1999). Mixed Memory Markov Models: Decomposing Complex Stochastic Processes as Mixtures of Simpler Ones. *Machine Learning*, *37*, 75–87.

Wang, S., Pentney, W., Popescu, A.-M., Choudhury, T., & Philipose, M. (2007). Common Sense Based Joint Training of Human Activity Recognizers. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

Zhang, W., Chen, F., Xu, W., & Cao, Z. (2007). Decomposition in hidden Markov models for activity recognition. *Multimedia Content Anal. and Mining*.

Exploration Scavenging

John Langford JL@YAHOO-INC.COM
Alexander Strehl STREHL@YAHOO-INC.COM
Yahoo! Research, 111 W. 40th Street, New York, New York 10018

Jennifer Wortman WORTMANJ@SEAS.UPENN.EDU
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104

Abstract

We examine the problem of evaluating a policy in the contextual bandit setting using only observations collected during the execution of another policy. We show that policy evaluation can be impossible if the exploration policy chooses actions based on the side information provided at each time step. We then propose and prove the correctness of a principled method for policy evaluation which works when this is not the case, even when the exploration policy is deterministic, as long as each action is explored sufficiently often. We apply this general technique to the problem of offline evaluation of internet advertising policies. Although our theoretical results hold only when the exploration policy chooses ads independent of side information, an assumption that is typically violated by commercial systems, we show how clever uses of the theory provide non-trivial and realistic applications. We also provide an empirical demonstration of the effectiveness of our techniques on real ad placement data.

1. Introduction

The k-armed bandit problem (Lai & Robbins, 1985; Berry & Fristedt, 1985; Auer et al., 2002; Even-Dar et al., 2006) has been studied in great detail, primarily because it can be viewed as a minimal formalization of the exploration problem faced by any autonomous agent. Unfortunately, while its minimalism admits tractability and insight, it misses details that are necessary for application to many realistic problems. For instance, the problem of internet advertising can be viewed as a type of bandit problem in which choosing an ad or set of ads to display corresponds to choosing an arm to pull. However, this formalization is inadequate in practice, as vital information is ignored. In particular, a successful ad placement policy might choose ads based on the content of the web page on which the ads are displayed. The standard k-armed bandit formulation ignores this useful information.

This shortcoming can be rectified by modeling the problem as an instance of the *contextual bandit problem* (Langford & Zhang, 2007), a generalization of the k-armed bandit problem that allows an agent to first observe an *input* or *side information* before choosing an arm. This problem has been studied under different names, including associative reinforcement learning (Kaelbling, 1994), bandits with side information (Wang et al., 2005), and bandits with experts (Auer et al., 1995), yet its analysis is far from complete.

In this paper, we study *policy evaluation* in the contextual bandit setting. Policy evaluation is the problem of evaluating a new strategy for behavior, or *policy*, using only observations collected during the execution of another policy. The difficulty of this problem stems from the lack of control over available data. Given complete freedom, an algorithm could evaluate a policy simply by executing it for a sufficient number of trials. However, in real-world applications, we often do not have the luxury of executing arbitrary policies, or we may want to distinguish or search among many more policies than we could evaluate independently.

We begin by providing impossibility results characterizing situations in which policy evaluation is *not* possible. In particular, we show that policy evaluation can be impossible when the exploration policy depends on the current input. We then provide and prove the correctness of a principled method for policy evaluation when this is not the case. This technique, which we

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

call "exploration scavenging," can be used to accurately estimate the value of any new policy as long as the exploration policy does not depend on the current input and chooses each action sufficiently often, even if the exploration policy is deterministic. The ability to depend on deterministic policies makes this approach more applicable than previous techniques based upon known and controlled randomization in the exploring policy. We also show that exploration scavenging can be applied if we wish to choose between multiple policies, even when these policies depend on the input, which is a property shared by most real ad-serving policies. This trick allows exploration scavenging to be applied to a broader set of real-life problems.

The motivating application for our work is *internet advertising*. Each time a user visits a web page, an advertising engine places a limited number of ads in a *slate* on the page. The ad company receives a payment for every ad clicked by the user. Exploration scavenging is well-suited for this application for a few reasons. First, an advertising company may want to evaluate a new method for placing ads without incurring the risk and cost of actually using the new method. Second, there exist logs containing huge amounts of historical click data resulting from the execution of existing ad-serving policies. It is economically sensible to use this data, if possible, when evaluating new policies.

In Section 4, we discuss the application of our methods to the ad display problem, and present empirical results on data provided by Yahoo!, a web search company. Although this application actually violates the requirement that the exploration policy be independent of the current input, the techniques show promise, leading us to believe that exploration scavenging can be useful in practice even when the strong assumptions necessary for the theoretical results do not hold.

To our knowledge, the only similar application work that has been published is that of Dupret et al. (2007) who tackle a similar problem from a Bayesian perspective using different assumptions which lead to different solution techniques. Our approach has the advantage that the estimated value is the output of a simple function rather than an EM optimization, facilitating interpretation of the evaluation method.

2. The Contextual Bandit Setting

Let $\mathcal{X}$ be an arbitrary input space, and $\mathcal{A} = \{1, \cdots, k\}$ be a set of actions. An instance of the contextual bandit problem is specified by a distribution D over tuples $(x, \vec{r})$ where $x \in \mathcal{X}$ is an input and $\vec{r} \in [0,1]^k$ is a vector of rewards. Events occur on a round by round basis where on each round t:

1. The world draws $(x_t, \vec{r}_t) \sim D$ and announces x_t.
2. The algorithm chooses an action $a_t \in \mathcal{A}$, possibly as a function of x_t and historical information.
3. The world announces the reward r_{t,a_t} of action a_t.

The algorithm does not learn what reward it would have received if it had chosen an action $a \neq a_t$.

The standard goal in this setting is to maximize the sum of rewards r_{t,a_t} over the rounds of interaction. An important subgoal, which is the focus of this paper, is policy evaluation. Here, we assume that we are given a data set $S \in (\mathcal{X} \times \mathcal{A} \times [0,1])^T$, which is generated by following some fixed policy π for T steps. Now, given a different policy $h : \mathcal{X} \to \mathcal{A}$, we would like to estimate the *value* of policy h, that is,

$$V_D(h) := E_{(x,\vec{r})\sim D}[r_{h(x)}].$$

The standard k-armed bandit is a special case of the contextual bandit setting in which $|\mathcal{X}| = 1$.

3. Evaluating Policies

In this section, we characterize situations in which policy evaluation may not be possible, and provide techniques for estimating the value of a policy when it is. To start, we show that when the exploration policy π depends on the input x, policy evaluation can be impossible. Later, we show that when the exploration policy π has no dependence on the current input, there exists a technique for accurately estimating the value of a new policy h as long as the exploration policy chooses each action sufficiently often. Finally, we show that exploration scavenging can be applied in the situation in which we are choosing between multiple exploration policies, even when the exploration policies themselves depend on the current input.

3.1. Impossibility Results

First, note that policy evaluation is not possible when the exploration policy π chooses some action a with zero probability. This is true even in the standard k-armed bandit setting. If the exploration policy always chooses action 1, and the policy to evaluate always chooses action 2, then policy evaluation is hopeless.

It is natural to ask if it is possible to build a policy evaluation procedure that is guaranteed to accurately evaluate a new policy given data collected using an arbitrary exploration policy π as long as π chooses each action sufficiently often. The following theorem shows that this goal is unachievable. In particular, it shows

that if the exploration policy π depends on the current input, then there are cases in which new policies h cannot be evaluated using observations gathered under π, even if π chooses each action frequently. Specifically, there can exist two contextual bandit distributions D and D' that result in indistinguishable observation sequences even though $V_D(h)$ and $V_{D'}(h)$ are far apart. Later we show that in the same context, if we disallow input-dependent exploration policies, policy evaluation becomes possible

Theorem 1 *There exist contextual bandit problems D and D' with $k = 2$ actions, a hypothesis h, and a policy π dependent on the current observation x_t with each action visited with probability $1/2$, such that observations of π on D are statistically indistinguishable from observations of π on D', yet $|V_D(h) - V_{D'}(h)| = 1$.*

Proof: The proof is by construction. Suppose x_t takes on the values 0 and 1, each with probability 0.5 under both D and D'. Let $\pi(x) = x$ be the exploration policy, and let $h(x) = 1 - x$ be the policy we wish to evaluate. Suppose that rewards are deterministic given x_t, as summarized in the following table.

	Under D		Under D'	
	$r_{t,0}$	$r_{t,1}$	$r_{t,0}$	$r_{t,1}$
$x_t = 0$	0	0	0	1
$x_t = 1$	0	1	1	1

Then $V_D(h) = 0$, while $V_{D'}(h) = 1$, but observations collected using exploration policy π are indistinguishable for D and D'. ■

3.2. Techniques for Policy Evaluation

We have seen that policy evaluation can be impossible in general if the exploration policy π depends on the current input or fails to choose each action sufficiently often. We now discuss techniques for policy evaluation when this is not the case. Theorem 2 shows that in some very special circumstances, it is possible to create an unbiased estimator for the value of a policy h using exploration data from another policy. The main result of this section, Theorem 3, shows that this estimator is often close to the value of the policy, even when the stringent conditions in the Theorem 2 are not satisfied.

Theorem 2 *For any contextual bandit distribution D over $(x, \vec{r})$, any policy h, any exploration policy π such that (1) for each action a, there is a constant $T_a > 0$ for which $|\{t : a_t = a\}| = T_a$ with probability 1, and (2) π chooses a_t independent of x_t,*

$$V_D(h) = E_{\{x_t,\vec{r}_t\}\sim D^T}\left[\sum_{t=1}^{T}\frac{r_{t,a_t}I(h(x_t)=a_t)}{T_{a_t}}\right].$$

Proof:

$$\begin{aligned}
&E_{\{x_t,\vec{r}_t\}\sim D^T}\left[\sum_{t=1}^{T}\frac{r_{t,a_t}I(h(x_t)=a_t)}{T_{a_t}}\right] \\
&= E_{\{x_t,\vec{r}_t\}\sim D^T}\left[\sum_{a=1}^{k}\sum_{\{t:a_t=a\}}\frac{r_{t,a}I(h(x_t)=a)}{T_a}\right] \\
&= \sum_{a=1}^{k}E_{\{x_t,\vec{r}_t\}\sim D^T}\left[\sum_{\{t:a_t=a\}}\frac{r_{t,a}I(h(x_t)=a)}{T_a}\right] \\
&= \sum_{a=1}^{k}E_{(x,\vec{r})\sim D}\left[T_a\frac{r_aI(h(x)=a)}{T_a}\right] \\
&= E_{x,\vec{r}\sim D}\left[\sum_{a=1}^{k}r_aI(h(x)=a)\right] = V_D(h).
\end{aligned}$$

The first equality is a reordering of the sum. The second and fourth follow from linearity of expectation.

The third equality is more subtle. Consider a fixed action a. The term $\sum_{\{t:a_t=a\}} r_{t,a}I(h(x_t) = a)/T_a$ involves drawing T bandit samples $(x_t, \vec{r}_t)$ and summing the term $r_{t,a}I(h(x_t) = a)/T_a$ over only the times t for which the exploration policy chose action a. There are precisely T_a such trials. The equality then follows from the fact that the quantity $r_{t,a}I(h(x_t) = a)/T_a$ is identically distributed for all t such that $a_t = a$. It is critical that the exploration policy π chooses a_t independent of x_t (to make the numerator identical) and that T_a is fixed (to make the denominator identical). If a_t depends on x_t, then these values are no longer identically distributed and the equality does not hold. This is important, as we have seen that evaluation is not possible in general if a_t can depend on x_t. ■

Conditions (1) and (2) in the theorem are satisfied, for example, by any policy which visits each action and chooses actions independent of observations. This theorem represents the limit of what we know how to achieve with a strict equality. It can replace the sample selection bias (Heckman, 1979) lemma used in the analysis of the Epoch-Greedy algorithm (Langford & Zhang, 2007), but cannot replace the analysis used in EXP4 (Auer et al., 1995) without weakening their theorem statement to hold only in IID settings.

The next theorem, which is the main theoretical result of this paper, shows that in a much broader set of circumstances, the estimator in the previous lemma is useful for estimating $V_D(h)$. Specifically, as long as the exploration does not depend on the current input and chooses each action sufficiently frequently, the estimator can be used for policy evaluation.

Theorem 3 *For every contextual bandits distribution D over $(x, \vec{r})$ with rewards $r_a \in [0,1]$, for every sequence of T actions a_t chosen by an exploration policy π that may be a function of history but does not depend on x_t, for every hypothesis h, then for any $\delta \in (0,1)$, with probability $1-\delta$,*

$$\left| V_D(h) - \sum_{t=1}^{T} \frac{r_{t,a_t} I(h(x_t) = a_t)}{T_{a_t}} \right| \leq \sum_{a=1}^{k} \sqrt{\frac{2\ln(2kT/\delta)}{T_a}}$$

where $T_a = |\{t : a_t = a\}|$.

Proof: First notice that

$$V_D(h) = \sum_{a=1}^{k} E_{x,\vec{r}\sim D}\left[r_a I(h(x) = a)\right]. \qquad (1)$$

Fix an action a. Let t_i denote the ith time step that action a was chosen, with $t_i = 0$ if $i > T_a$. Note that t_i is a random variable. For $i = 1, \ldots, T$ define

$$Z_i = \begin{cases} r_{t_i,a} I(h(x_{t_i}) = a) \\ \qquad -E_{x,\vec{r}\sim D}[r_a I(h(x) = a)] & \text{if } i \leq T_a, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $Z_i \in [-1,1]$ and $E[Z_i] = 0$ for all i. Now fix a positive integer $t \in \{1, \ldots, T\}$. We apply Azuma's inequality (see, for example, Alon and Spencer (2000)) to show that for any $\delta' \in (0,1)$, with probability $1-\delta'$,

$$\frac{1}{t}\left|\sum_{i=1}^{t} Z_i\right| \leq \sqrt{\frac{2\ln(2/\delta')}{t}}\ , \qquad (2)$$

and so if $t \leq T_a$,

$$\left| E_{x,\vec{r}\sim D}[r_a I(h(x) = a)] - \frac{1}{t}\sum_{i=1}^{t} r_{t_i,a} I(h(x_{t_i}) = a) \right|$$

is upper bounded by $\sqrt{2\ln(2/\delta')/t}$. Applying the union bound with $\delta' = \delta/(Tk)$, we see that Equation 2 holds for *all* $t \in \{1, \ldots, T\}$ and thus for $t = T_a$ with probability δ/k. Applying the union bound again yields a bound that holds for all actions. Summing over actions and applying Equation 1 yields the lemma. ∎

Note that the counter-example given in Theorem 1 satisfies all conditions of Theorem 3 except for the assumption on π. Thus, we cannot solve the policy exploration problem, in general, unless we make assumptions that limit π's dependence on input.

Corollary 4 *For every contextual bandit distribution D over $(x, \vec{r})$, for every exploration policy π choosing action a_t independent of the current input, for every policy h, if every action $a \in \{1, \cdots, k\}$ is guaranteed to be chosen by π at least a constant fraction of the time, then as $T \to \infty$, the estimator*

$$\hat{V}_D(h) = \sum_{t=1}^{T} \frac{r_{t,a_t} I(h(x_t) = a_t)}{T_{a_t}}$$

grows arbitrarily close to $V_D(h)$ with probability 1.

These observations can be utilized in practice in a simple way. Given a data set S of observations (x_t, a_t, r_{a_t}) for $t = \{1, \cdots, T\}$, we can calculate $\hat{V}_D(h)$ as above and use this as an estimator of $V_D(h)$. For sufficiently large data sets S, as long as each action is chosen sufficiently often, this estimator is accurate.

3.3. Tighter Bounds in Special Cases

In some special cases when there is sufficient randomization in the exploration policy or the policy h, it is possible to achieve tighter bounds using a slightly modified estimator. Theorem 5 shows that the dependence on the number of actions k can be improved in the special case in which $\Pr(h(x) = a_t) = 1/k$ independent of x. This is true, for instance, when either the exploration policy π or the policy h chooses actions uniformly at random. We suspect that tighter bounds can be achieved in other special cases as well.

Theorem 5 *For every contextual bandits distribution D over $x, \vec{r}$ with rewards $r_a \in [0,1]$, for every sequence of actions a_t chosen by an exploration policy π that may be a function of history but does not depend on x_t and every hypothesis h, if $\Pr(h(x) = a_t) = 1/k$ independent of x and if $|\{t : a_t = a\}| > 0$ for all a, then for any $\delta \in (0,1)$, with probability $1-\delta$,*

$$\left| V_D(h) - \sum_{t=1}^{T} \frac{k r_{t,a_t} I(h(x_t) = a_t)}{T} \right| \leq k\sqrt{\frac{2\ln(2k/\delta)}{T}}\ .$$

Proof: Since we have assumed that $\Pr(h(x_t) = a_t) = 1/k$ independent of x_t,

$$\begin{aligned} V_D(h) &= E_{x,\vec{r}\sim D}[k r_{h(x)} I(h(x) = a_t)] \\ &= E_{x,\vec{r}\sim D}[k r_{a_t} I(h(x) = a_t)]\ . \end{aligned}$$

For all t, define

$$Z_t = k r_{t,a_t} I(h(x_t) = a_t) - E_{x,\vec{r}\sim D}[k r_{a_t} I(h(x) = a_t)]\ .$$

$Z_t \in [-k, k]$ and $E[Z_{t,a}] = 0$. Applying Azuma's inequality and the union bound yields the lemma. ∎

3.4. Multiple Exploration Policies

So far, all of our positive theoretical results have required that the exploration policy choose actions independent of the current input. There do exist special cases in which exploration data is provably useful for policy evaluation even when the exploration policy depends on context. We briefly describe one such case.

Suppose we have collected data from a system that has rotated through K known exploration policies $\pi_1, \pi_2, \cdots, \pi_K$ over time. For example, we may have logs of historical ad display data from a company that has used K different ad display policies. Each individual exploration policy π_i may depend on context, but we assume that the choice of which policy was used by the system at any given time does not.

We can redefine the action of the bandit problem as a choice of one of the K base policies to follow; action a_i now corresponds to choosing the ad chosen by policy π_i. Since historically the decision about which policy to use was made independent of the context x, we can view the exploration policy as oblivious with respect to x. Theorem 3 then implies that we can accurately estimate the value of any policy π which chooses from the set of actions chosen by the K base policies. This can be more powerful than competing with each historic policy, because π can make context-dependent choices about which policy to follow, potentially achieving better performance than any single policy.

4. Application to Internet Advertising

Technology companies are interested in finding better ways to search both over the myriad pages of the internet and over the increasingly large selection of potential ads to display. However, given a candidate algorithm (or *ad-serving policy* in the case of online advertising), a company faces a real-life "exploration-exploitation" dilemma. The new algorithm could be better than existing ones, but it could be worse. To evaluate the performance of an algorithm, the company might decide to adopt it for a short time on a subset of web traffic. This method produces accurate estimates of performance, but the evaluation phase can be costly in terms of lost revenue if the candidate algorithm performs poorly, and this cost grows linearly in the number of candidate algorithms that the company would like to evaluate. Clearly, a method of determining the strengths or weaknesses of an algorithm without adopting it would be highly useful.

In this section, we tackle the problem of evaluating a new ad-serving policy using data logged from an existing system. We state our results in terms of the online advertising problem, but everything we discuss can be applied to web search with little or no modification.

We begin by showing how to directly apply exploration scavenging techniques to the problem, and discuss the primary drawbacks of this simple approach. Instead, we consider a standard simplifying assumption whose adoption leads to a more realistic method for policy evaluation. This assumption, that click-through rates are factorable, leads to another interesting theoretical problem, estimating the attention decay coefficients of the click-through rates, which can also be accomplished using techniques from Section 3.

4.1. The Direct Approach

The online advertising problem can be directly mapped to the contextual bandit problem, allowing us to apply results from Section 3. Here the input space is the universe of all possible web pages and the action space contains all slates of ads. The reward is a bit vector that identifies whether or not each returned ad was clicked.[1] This bit vector can be converted to a single real-valued reward r in a number of ways, for instance, by simply summing the components, yielding the total number of clicks received, and normalizing. The example would then be used to compute a number $r \cdot I(h(x) = s)/\text{Count}(s)$, where $\text{Count}(s)$ is the number of times the slate s was displayed during all trials. According to Theorem 3, summing this quantity over all trials yields a good estimator of the value of the new policy h.

There is a significant drawback to this approach. Due to the indicator variable, the contribution to the sum for a single example is zero unless $h(x) = s$, which means that the slate chosen by the candidate algorithm h is *exactly* the same as the slate produced by the current system π. With a large set of ads and a large slate size, it is very unlikely that the same slate is chosen many times, and thus the resulting estimator for the value of h has an extremely high variance and may not exist for most slates. In the next section, we show how a standard assumption in the online advertising community can be used to reduce the variance.

4.2. The Factoring Assumption

The problem described above can be avoided by making a *factoring assumption*. Specifically, we assume that the probability of clicking an ad can be decomposed or factored into two terms, an intrinsic *click-through rate* (CTR) that depends only on the web

[1] The reward function can be modified easily to reflect the actual revenue generated by each click.

page x and the ad a, and a position-dependent multiplier C_i for position i, called the *attention decay coefficient* (ADC). This assumption is commonly used in the sponsored search literature (Borgs et al., 2007; Lahaie & Pennock, 2007).

Formally, let $\mathcal{P}(x, a, i)$ be the probability that ad a is clicked when placed in position i on web page x. We assume that $\mathcal{P}(x, a, i) = C_i \cdot \mathcal{P}(x, a)$, where $\mathcal{P}(x, a)$ is the intrinsic (position independent) click-through rate for ad a given input x, and C_i is a position-dependent constant. Here $C_1 = 1$, so $\mathcal{P}(x, a) = \mathcal{P}(x, a, 1)$.

A key observation is that this assumption allows us to transition from dealing directly with slates of ads to focusing on single ads. Let ℓ be the number of ads shown in a slate. Given an example $(x, s, \vec{r})$, we can form ℓ transformed examples of the form (x, a_i, r'_i) where a_i is the ith ad in the slate and $r'_i = r_i/C_i$. In other words, r'_i is $1/C_i$ if the ith ad was clicked, and 0 otherwise; the division by the ADC puts the rewards on the same scale, so the expected value of the reward for a fixed pair (x, a_i) is $\mathcal{P}(x, a_i)$.

Let $\sigma(a, x)$ be the slot in which the evaluation policy h places ad a on input x; if h does not display a on input x, then $\sigma(a, x) = 0$. For convenience, define $C_0 = 0$. We define a new estimator of the value of h as

$$\hat{V}_D(h) = \sum_{t=1}^{T} \sum_{i=1}^{\ell} \frac{r'_i C_{\sigma(a_i,x)}}{T_{a_i}} , \qquad (3)$$

where T_a is the total number of impressions received by a (i.e., the total number of times add a is displayed). Here C_i takes the place of the indicator function used in the estimates in Section 3, giving higher weights to the rewards of ads that h places in better slots.

Using the results from Section 3, it is straight-forward to show that this estimator is consistent as long as the current ad-serving policy does not depend on the input webpage x and every ad is displayed. However, to apply this transformation, we require knowledge of the ADCs. In the next section we show how to estimate them, again using nonrandom exploration.

4.3. Estimating Attention Decay Coefficients

Assume that a data set S is available from the execution of an ad-serving policy π that chooses the tth slate of ads to display independent of the input x_t (though possibly dependent on history). As before, S includes observations $(x_t, \vec{a}_t, \vec{r}_{t,a_t})$ for $t = \{1, \cdots, T\}$, where $\vec{a}_t$ is the slate of ads displayed at time t and $\vec{r}_{t,a_t}$ is the reward vector. Our goal is to use this data to estimate the attention decay coefficients $C_2, \ldots, C_\ell$.

We first discuss a naive ADC estimator, and then go on to show how it can be improved. In the following sections, let $C(a, i)$ be the number of clicks on ad a observed during rounds in which ad a is displayed in position i. Let $M(a, i)$ be the number of impressions of ad a in slot i, i.e., the number of times that the exploration policy chooses to place ad a in slot i. Finally, let $\mathrm{CTR}(a, i) = C(a, i)/M(a, i)$ be the observed click-through rate of ad a in slot i, with $\mathrm{CTR}(a, i)$ defined to be 0 when $M(a, i) = 0$.

4.3.1. The Naive Estimator

Initially, one might think that the ADCs can be calculated by taking the ratio between the global empirical click-through rate for each position i and the global empirical click-through rate for position 1. Formally,

$$\mathrm{Est}_{\mathrm{naive}}(\mathrm{i}) := \frac{\sum_a C(a, i)/\sum_a M(a, i)}{\sum_a C(a, 1)/\sum_a M(a, 1)}.$$

Unfortunately, as we will see in Section 4.4, this method has a bias which is often quite large in practice. In particular, it often underestimates the ratios C_i due to the fact that existing ad-serving policies generally already place better ads (with higher $\mathcal{P}(x, a)$) in the better slots. To overcome this bias, we must design a new estimator.

4.3.2. A New Estimator

Consider a fixed ad a and a fixed position $i > 1$. Clearly if a is placed in position i sufficiently many times, it is possible to estimate the probability of a being clicked in position i fairly accurately. If we also estimate the corresponding click-through rate for ad a in position 1, we may estimate C_i using a ratio of these two click-through rates, since $C_i = E_{x\sim D}[\mathcal{P}(x, a, i)]/E_{x\sim D}[\mathcal{P}(x, a, 1)]$. If we perform this procedure for all ads, we can average the resulting estimates to form a single, typically very accurate, estimate. Formally, we propose an estimator of the form

$$\mathrm{Est}_{\vec{\alpha}}(i) = \frac{\sum_a \alpha_a \mathrm{CTR}(a, i)}{\sum_a \alpha_a \mathrm{CTR}(a, 1)}, \qquad (4)$$

where $\vec{\alpha}$ is a vector of nonnegative constants α_a for each ad $a \in \mathcal{A}$.

Theorem 6 *If the ad-display policy chooses slates independent of input and $\vec{\alpha}$ has all positive entries, then the estimator $\mathrm{Est}_{\vec{\alpha}}$ in Equation 4 is consistent.*

Proof: Consider any fixed ad a and position i, and suppose that we are only interested in revenue generated by position i. Let h be the constant hypothesis that always places ad a in position i. $V_D(h)$ is then

$E_{x\sim D}\mathcal{P}(x,a,i)$. From Corollary 4, it is clear that

$$\hat{V}_D(h) = \sum_{t=1}^{T} \frac{(r_t I(h(x_t) = a_t))}{|\{t' : a_{t'} = a_t\}|}$$

converges to $V_D(h)$. Here $\hat{V}_D(h)$ is precisely $\mathrm{CTR}(a,i)$, so $\mathrm{CTR}(a,i)$ converges to $E_{x\sim D}\mathcal{P}(x,a,i)$ for all a and i. This implies that $\mathrm{Est}_{\vec{\alpha}}(p)$ converges to

$$\frac{\sum_a \alpha_a E_{x\sim D}\mathcal{P}(x,a,i)}{\sum_a \alpha_a E_{x\sim D}\mathcal{P}(x,a,1)} = \frac{\sum_a \alpha_a E_{x\sim D} C_i \mathcal{P}(x,a)}{\sum_a \alpha_a E_{x\sim D} C_1 \mathcal{P}(x,a)} = \frac{C_i}{C_1} = C_i \ .$$

■

Theorem 6 leaves open the question of how to choose $\vec{\alpha}$. If every component of $\vec{\alpha}$ is set to the same value, then the estimate for C_i can be viewed as the mean of all estimates of C_i for each ad a. However, it may be the case that the estimates for certain ads are more accurate than others, in which case we'd like to weight those more heavily. In particular, we may want to pick $\vec{\alpha}$ to minimize the variance of our final estimator. Since it is difficult to analytically compute the variance of a quotient, we approximate it by the variance of the sum of the numerator and denominator, as this tends to reduce the variance of the quotient. The proof of the following theorem is omitted due to lack of space.

Theorem 7 *The variance of the expression*

$$\sum_a \alpha_a \mathrm{CTR}(a,i) + \sum_a \alpha_a \mathrm{CTR}(a,1)$$

subject to $\sum_a \alpha_a = 1$ *is minimized when*

$$\alpha_a := \frac{2M(a,i)\cdot M(a,1)}{M(a,i)\sigma^2_{a,1} + M(a,1)\sigma^2_{a,i}},$$

where $\sigma^2_{a,i}$ *is the variance of the indicator random variable that is 1 when ad* a *is clicked given that ad* a *is placed in position* i*.*

It is undesirable that π is required to have no dependence on the current web page x_t when choosing the slate of ads to display, since most current ad-serving algorithms violate this assumption. However, as we have seen in Section 3.1, when this assumption is violated, exploration scavenging is no longer guaranteed to work. In the worst case, we cannot trust our estimated ADCs from data generated by an x-dependent π. Luckily, in practice, it is generally not the case that extreme scenarios like the counterexample in the proof of Theorem 1 arise. It is more likely that the existing ad-serving algorithm and the new algorithm choose among the same small set of ads to display for any given context (for example, the set of ads for which advertisers have placed bids for the current search term in the sponsored search setting) and the primary difference between policies is the order in which these ads are displayed. In such settings it is also the case that additional opportunities for exploration arise naturally. For example, sometimes ads run out of budget, removing them from consideration and forcing the ad-serving algorithm to display an alternate slate of ads.

4.4. Empirical comparison

We are interested in comparing the methods developed in this work to standard methods used in practice. A common technique for estimating ADCs borrowed from the information retrieval literature is discounted cumulative gain (Järvelin & Kekäläinen, 2002). In relation to our work, discounted cumulative gain (DCG) can be viewed as a particular way to specify the ADCs that is *not* data-dependent. In particular, given a parameter b, DCG would suggest defining $C_i = 1/log_b(b+i)$ for all i. As shown next, when we estimated the ADCs using our new method on a large set of data we get values that are very close to those calculated using DCG with $b = 2$.

We present coefficients that were computed from training on about 20 million examples obtained from the logs of "Content Match", Yahoo!'s online advertisement engine. Since we don't know the true variances $\sigma^2_{a,p}$ for the distributions over clicks, we heuristically assume they are all equal and use the estimator defined by $\alpha_a = M(a,p)\cdot M(a,1)/(M(a,p)+M(a,1))$. The following table summarizes the coefficients computed for the first four slots using the naive estimator and the new estimator, along with the DCG coefficients. As suspected, the coefficients for the new estimator are larger than the old, suggesting a reduction in bias.

	C_1	C_2	C_3	C_4
Naive	1.0	0.512090	0.369638	0.271847
New	1.0	0.613387	0.527310	0.432521
DCG	1.0	0.630930	0.5	0.430677

4.5. Toward A Realistic Application

To reduce the high variance of the direct application of exploration scavenging to internet advertising, we made use of the factoring assumption and derived the estimator given in Equation 3. Unfortunately this new estimator may still have an unacceptably large variance. By examining Equation 3, we observe that the method only benefits from examples in which the exploration policy and the new policy h choose overlapping sets of ads to display. When ads are drawn from

a large database, this may be too rare of an event.

Instead of considering policies which rank from the set of all ads, we can consider policies h_π reordering the ads which π chooses to display. A good reordering policy plausibly provides useful information to guide the choice of a new ranking policy.

We define an alternate estimator

$$\hat{V}_D(h_\pi) = \sum_{t=1}^{T} \sum_{i=1}^{\ell} r'_i C_{\sigma'(a_i,x)} \ ,$$

where $\sigma'(a_i, x)$ is the slot that h_π would assign to ad a_i in this new model. This method gives us an (unnormalized) estimate of the value of first using π to choose k ads to display in a slate and then using h_π to reorder them. This approach has small variance and quickly converges.

To illustrate our method we used a training set of 20 million examples gathered using Yahoo!'s current ad-serving algorithm π. We let the policy h_π be the policy that reorders ads to display those with the highest empirical click-through rate first, ignoring the context x. We used $r = C_{j'}/C_i$, (with coefficients given by the new unbiased method) to compute the number of clicks we expect the new policy (using h_π to reorder π's slate) to receive per click of the old policy π. Here j' is the relative position of ad a_i when the ads in the slate shown by π are reordered (in descending order) by h_π. This number, which was computed using a test set of about two million examples, turned out to be 1.086. When we computed the same quantity for the policy h'_π that reorders ads at random, we obtained 1.016. Thus, exploration scavenging strongly suggests using policy h_π over h'_π, matching our intuition.

5. Conclusion

We study the process of "exploration scavenging," reusing information from one policy to evaluate a new policy, and provide procedures that work *without* randomized exploration, as is commonly required. This new ability opens up the possibility of using machine learning techniques in new domains which were previously inaccessible.

Using the derandomized exploration techniques described here, we show how to estimate the value of a policy reordering displayed ads on logged data *without* any information about random choices made in the past. There are several caveats to this approach, but the results appear to be quite reasonable.

Note that this methodology is simply *impossible* without considering methods for derandomized exploration, so the techniques discussed here open up new possibilities for solving problem.

Acknowledgments

We are grateful to Michael Kearns for a useful discussion of the theoretical results, and to our anonymous reviewers for their thought-provoking questions.

References

Alon, N., & Spencer, J. (2000). *The probabilistic method*. Interscience Series in Discrete Mathematics and Optimization. John Wiley. Second edition.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite time analysis of the multi-armed bandit problem. *Machine Learning*, *47*, 235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-armed bandit problem. *36th Annual IEEE Symposium on Foundations of Computer Science*.

Berry, D. A., & Fristedt, B. (1985). *Bandit problems: Sequential allocation of experiments*. London, UK: Chapman and Hall.

Borgs, C., Chayes, J., Etesami, O., Immorlica, N., Jain, K., & Mahdian, M. (2007). Dynamics of bid optimization in online advertisement auctions. *16th International World Wide Web Conference*.

Dupret, G., Murdock, V., & Piwowarski, B. (2007). Web search engine evaluation using clickthrough data and a user model. *16th Intl. World Wide Web Conference*.

Even-Dar, E., Mannor, S., & Mansour, Y. (2006). Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, *7*, 1079–1105.

Heckman, J. (1979). Sample selection bias as a specification error. *Econometrica*, *47*, 153–161.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, *20*, 422–446.

Kaelbling, L. P. (1994). Associative reinforcement learning: A generate and test algorithm. *Machine Learning*, *15*.

Lahaie, S., & Pennock, D. (2007). Revenue analysis of a family of ranking rules for keyword auctions. *8th ACM Conference on Electronic Commerce*.

Lai, T., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, *6*, 4–22.

Langford, J., & Zhang, T. (2007). The epoch greedy algorithm for contextual multi-armed bandits. *Advances in Neural Information Processing Systems*.

Wang, C.-C., Kulkarni, S. R., & Poor, H. V. (2005). Bandit problems with side observations. *IEEE Transactions on Automatic Control*, *50*, 338–355.

Classification using Discriminative Restricted Boltzmann Machines

Hugo Larochelle LAROCHEH@IRO.UMONTREAL.CA
Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Dept. IRO, Université de Montréal C.P. 6128, Montreal, Qc, H3C 3J7, Canada

Abstract

Recently, many applications for Restricted Boltzmann Machines (RBMs) have been developed for a large variety of learning problems. However, RBMs are usually used as feature extractors for another learning algorithm or to provide a good initialization for deep feed-forward neural network classifiers, and are not considered as a stand-alone solution to classification problems. In this paper, we argue that RBMs provide a self-contained framework for deriving competitive non-linear classifiers. We present an evaluation of different learning algorithms for RBMs which aim at introducing a discriminative component to RBM training and improve their performance as classifiers. This approach is simple in that RBMs are used directly to build a classifier, rather than as a stepping stone. Finally, we demonstrate how discriminative RBMs can also be successfully employed in a semi-supervised setting.

1. Introduction

Restricted Boltzmann Machines (RBMs) (Smolensky, 1986) are generative models based on latent (usually binary) variables to model an input distribution, and have seen their applicability grow to a large variety of problems and settings in the past few years. From binary inputs, they have been extended to model various types of input distributions (Welling et al., 2005; Hinton et al., 2006). Conditional versions of RBMs have also been developed for collaborative filtering (Salakhutdinov et al., 2007) and to model motion capture data (Taylor et al., 2006) and video sequences (Sutskever & Hinton, 2007).

RBMs have been particularly successful in classification problems either as feature extractors for text and image data (Gehler et al., 2006) or as a good initial training phase for deep neural network classifiers (Hinton, 2007). However, in both cases, the RBMs are merely the first step of another learning algorithm, either providing a preprocessing of the data or an initialization for the parameters of a neural network. When trained in an unsupervised fashion, RBMs provide no guarantees that the features implemented by their hidden layer will ultimately be useful for the supervised task that needs to be solved. More practically, model selection can also become problematic, as we need to explore jointly the space of hyper-parameters of both the RBM (size of the hidden layer, learning rate, number of training iterations) and the supervised learning algorithm that is fed the learned features. In particular, having two separate learning phases (feature extraction, followed by classifier training) can be problematic in an online learning setting.

In this paper, we argue that RBMs can be used successfully as stand-alone non-linear classifiers alongside other standard classifiers like neural networks and Support Vector Machines, and not only as feature extractors. We investigate training objectives for RBMs that are more appropriate for training classifiers than the common generative objective. We describe Discriminative Restricted Boltzmann Machines (DRBMs), i.e. RBMs that are trained more specifically to be good classification models, and Hybrid Discriminative Restricted Boltzmann Machines (HDRBMs) which explore the space between discriminative and generative learning and can combine their advantages. We also demonstrate that RBMs can be successfully adapted to the common semi-supervised learning setting (Chapelle et al., 2006) for classification problems. Finally, the algorithms investigated in this paper are well suited for online learning on large datasets.

2. Restricted Boltzmann Machines

Restricted Boltzmann Machines are undirected generative models that use a layer of hidden variables to model a distribution over visible variables. Though they are most often trained to only model the inputs

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

of a classification task, they can also model the joint distribution of the inputs and associated target classes (e.g. in the last layer of a Deep Belief Network in Hinton et al. (2006)). In this section, we will focus on such joint models.

We assume given a training set $\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}$, comprising for the i-th example an input vector $\mathbf{x}_i$ and a target class $y_i \in \{1, \ldots, C\}$. To train a generative model on such data we consider minimization of the negative log-likelihood

$$\mathcal{L}_{gen}(\mathcal{D}_{train}) = -\sum_{i=1}^{|\mathcal{D}_{train}|} \log p(y_i, \mathbf{x}_i). \quad (1)$$

An RBM with n hidden units is a parametric model of the joint distribution between a layer of hidden variables (referred to as neurons or features) $\mathbf{h} = (h_1, \ldots, h_n)$ and the observed variables made of $\mathbf{x} = (x_1, \ldots, x_d)$ and y, that takes the form

$$p(y, \mathbf{x}, \mathbf{h}) \quad \propto \quad e^{-E(y,\mathbf{x},\mathbf{h})}$$

where

$$E(y, \mathbf{x}, \mathbf{h}) \quad = \quad -\mathbf{h}^T\mathbf{W}\mathbf{x} - \mathbf{b}^T\mathbf{x} - \mathbf{c}^T\mathbf{h} - \mathbf{d}^T\vec{y} - \mathbf{h}^T\mathbf{U}\vec{y}$$

with parameters $\Theta = (\mathbf{W}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{U})$ and $\vec{y} = (1_{y=i})_{i=1}^{C}$ for C classes. This model is illustrated in Figure 2. For now, we consider for simplicity binary input variables, but the model can be easily generalized to non-binary categories, integer-valued, and continuous-valued inputs (Welling et al., 2005; Hinton et al., 2006). It is straightforward to show that

$$p(\mathbf{x}|\mathbf{h}) \quad = \quad \prod_i p(x_i|\mathbf{h})$$

$$p(x_i = 1|\mathbf{h}) \quad = \quad \text{sigm}(b_i + \sum_j W_{ji} h_j) \quad (2)$$

$$p(y|\mathbf{h}) \quad = \quad \frac{e^{d_y + \sum_j U_{jy} h_j}}{\sum_{y^*} e^{d_{y^*} + \sum_j U_{jy^*} h_j}} \quad (3)$$

where sigm is the logistic sigmoid. Equations 2 and 3 illustrate that the hidden units are meant to capture predictive information about the input vector as well as the target class. $p(\mathbf{h}|y, \mathbf{x})$ also has a similar form:

$$p(\mathbf{h}|y, \mathbf{x}) \quad = \quad \prod_j p(h_j|y, \mathbf{x})$$

$$p(h_j = 1|y, \mathbf{x}) \quad = \quad \text{sigm}(c_j + U_{jy} + \sum_i W_{ji} x_i).$$

When the number of hidden variables is fixed, an RBM can be considered a parametric model, but when it is allowed to vary with the data, it becomes a non-parametric model. In particular, Freund and Haussler (1994); Le Roux and Bengio (2008) showed that

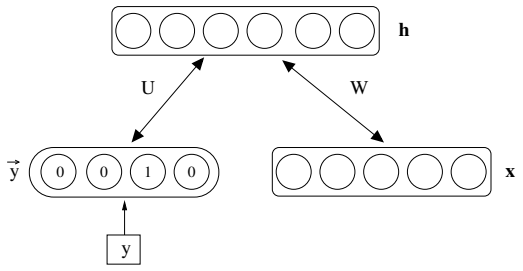


Figure 1. Restricted Boltzmann Machine modeling the joint distribution of inputs and target classes

an RBM with enough hidden units can represent any distribution over binary vectors, and that adding hidden units guarantees that a better likelihood can be achieved, unless the generated distribution already equals the training distribution.

In order to minimize the negative log-likelihood (eq. 1), we would like an estimator of its gradient with respect to the model parameters. The exact gradient, for any parameter $\theta \in \Theta$ can be written as follows:

$$\frac{\partial \log p(y_i, \mathbf{x}_i)}{\partial \theta} \quad = \quad -\mathbf{E}_{\mathbf{h}|y_i,\mathbf{x}_i}\left[\frac{\partial}{\partial \theta} E(y_i, \mathbf{x}_i, \mathbf{h})\right] + \mathbf{E}_{y,\mathbf{x},\mathbf{h}}\left[\frac{\partial}{\partial \theta} E(y, \mathbf{x}, \mathbf{h})\right].$$

Though the first expectation is tractable, the second one is not. Fortunately, there exists a good stochastic approximation of this gradient, called the contrastive divergence gradient (Hinton, 2002). This approximation replaces the expectation by a sample generated after a limited number of Gibbs sampling iterations, with the sampler's initial state for the visible variables initialized at the training sample $(y_i, \mathbf{x}_i)$. Even when using only one Gibbs sampling iteration, contrastive divergence has been shown to produce only a small bias for a large speed-up in training time (Carreira-Perpiñan & Hinton, 2005).

Online training of an RBM thus consists in cycling through the training examples and updating the RBM's parameters according to Algorithm 1, where the learning rate is controlled by λ.

Computing $p(y, \mathbf{x})$ is intractable, but it is possible to compute $p(y|\mathbf{x})$, sample from it, or choose the most probable class under this model. As shown in Salakhutdinov et al. (2007), for reasonable numbers of classes C (over which we must sum), this conditional distribution can be computed exactly and efficiently, by writing it as follows:

$$p(y|\mathbf{x}) \quad = \quad \frac{e^{d_y} \prod_{j=1}^{n} \left(1 + e^{c_j + U_{jy} + \sum_i W_{ji} x_i}\right)}{\sum_{y^*} e^{d_{y^*}} \prod_{j=1}^{n} \left(1 + e^{c_j + U_{jy^*} + \sum_i W_{ji} x_i}\right)}.$$

Algorithm 1 Training update for RBM over $(y, \mathbf{x})$ using Contrastive Divergence

Input: training pair $(y_i, \mathbf{x}_i)$ and learning rate λ
% Notation: $a \leftarrow b$ means a is set to value b
% $a \sim p$ means a is sampled from p

% Positive phase
$y^0 \leftarrow y_i$, $\mathbf{x}^0 \leftarrow \mathbf{x}_i$, $\widehat{\mathbf{h}}^0 \leftarrow \text{sigm}(\mathbf{c} + W\mathbf{x}^0 + \mathbf{U}\vec{y^0})$

% Negative phase
$\mathbf{h}^0 \sim p(\mathbf{h}|y^0, \mathbf{x}^0)$, $y^1 \sim p(y|\mathbf{h}^0)$, $\mathbf{x}^1 \sim p(\mathbf{x}|\mathbf{h}^0)$
$\widehat{\mathbf{h}}^1 \leftarrow \text{sigm}(\mathbf{c} + W\mathbf{x}^1 + \mathbf{U}\vec{y^1})$

% Update
for $\theta \in \Theta$ **do**
$\theta \leftarrow \theta - \lambda \left(\frac{\partial}{\partial \theta} E(y^0, \mathbf{x}^0, \widehat{\mathbf{h}}^0) - \frac{\partial}{\partial \theta} E(y^1, \mathbf{x}^1, \widehat{\mathbf{h}}^1) \right)$
end for

Precomputing the terms $c_j + \sum_i W_{ji} x_i$ and reusing them when computing $\prod_{j=1}^{n} \left(1 + e^{c_j + U_{jy^*} + \sum_i W_{ji} x_i}\right)$ for all classes y^* permits to compute this conditional distribution in time $O(nd + nC)$.

3. Discriminative Restricted Boltzmann Machines

In a classification setting, one is ultimately only interested in correct classification, not necessarily to have a good $p(\mathbf{x})$. Because our model's $p(\mathbf{x})$ can be inappropriate, it can then be advantageous to optimize directly $p(y|\mathbf{x})$ instead of $p(y, \mathbf{x})$:

$$\mathcal{L}_{disc}(\mathcal{D}_{train}) = - \sum_{i=1}^{|\mathcal{D}_{train}|} \log p(y_i|\mathbf{x}_i). \quad (4)$$

We refer to RBMs trained according to $\mathcal{L}_{disc}$ as Discriminative RBMs (DRBMs). Since RBMs (with enough hidden units) are universal approximators for binary inputs, it follows also that DRBMs are universal approximators of conditional distributions with binary inputs.

A DRBM can be trained by contrastive divergence, as has been done in conditional RBMs (Taylor et al., 2006), but since $p(y|\mathbf{x})$ can be computed exactly, we can compute the exact gradient:

$$\begin{aligned} \frac{\partial \log p(y_i|\mathbf{x}_i)}{\partial \theta} &= \sum_j \text{sigm}(o_{yj}(\mathbf{x}_i)) \frac{\partial o_{yj}(\mathbf{x}_i)}{\partial \theta} \\ &- \sum_{j, y^*} \text{sigm}(o_{y^* j}(\mathbf{x}_i)) p(y^*|\mathbf{x}_i) \frac{\partial o_{y^* j}(\mathbf{x}_i)}{\partial \theta} \end{aligned}$$

where $o_{yj}(\mathbf{x}) = c_j + \sum_k W_{jk} x_k + U_{jy}$. This gradient can be computed efficiently and then used in a stochastic gradient descent optimization. This discriminative approach has been used previously for fine-tuning the top RBM of a Deep Belief Network (Hinton, 2007).

4. Hybrid Discriminative Restricted Boltzmann Machines

The advantage brought by discriminative training usually depends on the amount of available training data. Smaller training sets tend to favor generative learning and bigger ones favor discriminative learning (Ng & Jordan, 2001). However, instead of solely relying on one or the other perspective, one can adopt a hybrid discriminative/generative approach simply by combining the respective training criteria. Though this method cannot be interpreted as a maximum likelihood approach for a particular generative model as in Lasserre et al. (2006), it proved useful here and elsewhere (Bouchard & Triggs, 2004). In this paper, we used the following criterion:

$$\mathcal{L}_{hybrid}(\mathcal{D}_{train}) = \mathcal{L}_{disc}(\mathcal{D}_{train}) + \alpha \mathcal{L}_{gen}(\mathcal{D}_{train}) \quad (5)$$

where the weight α of the generative criterion can be optimized (e.g., based on the validation set classification error). Here, the generative criterion can also be seen as a data-dependent regularizer for a DRBM. We will refer to RBMs trained using the criterion of equation 5 as Hybrid DRBMs (HDRBMs).

To train an HDRBM, we can use stochastic gradient descent and add for each example the gradient contribution due to $\mathcal{L}_{disc}$ with α times the stochastic gradient estimator associated with $\mathcal{L}_{gen}$ for that example.

5. Semi-supervised Learning

A frequent classification setting is where there are few labeled training data but many unlabeled examples of inputs. Semi-supervised learning algorithms (Chapelle et al., 2006) address this situation by using the unlabeled data to introduce constraints on the trained model. For example, for purely discriminative models, these constraints are often imposed on the decision surface of the model. In the RBM framework, a natural constraint is to ask that the model be a good generative model of the unlabeled data, which corresponds to the following objective:

$$\mathcal{L}_{unsup}(\mathcal{D}_{unlab}) = - \sum_{i=1}^{|\mathcal{D}_{unlab}|} \log p(\mathbf{x}_i) \quad (6)$$

where $\mathcal{D}_{unlab} = \{(\mathbf{x}_i)\}_{i=1}^{|\mathcal{D}_{unlab}|}$ contains unlabeled examples of inputs. To train on this objective, we can once again use a contrastive divergence approximation

of the log-likelihood gradient:

$$\frac{\partial \log p(\mathbf{x}_i)}{\partial \theta} = -\mathbb{E}_{y,\mathbf{h}|\mathbf{x}_i}\left[\frac{\partial}{\partial \theta} E(y_i, \mathbf{x}_i, \mathbf{h})\right] + \mathbb{E}_{y,\mathbf{x},\mathbf{h}}\left[\frac{\partial}{\partial \theta} E(y, \mathbf{x}, \mathbf{h})\right]$$

The contrastive divergence approximation is slightly different here. The first term can be computed in time $O(Cn + nd)$, by noticing that it is equal to

$$\mathbb{E}_{y|\mathbf{x}_i}\left[\mathbb{E}_{\mathbf{h}|y,\mathbf{x}_i}\left[\frac{\partial}{\partial \theta} E(y_i, \mathbf{x}_i, \mathbf{h})\right]\right]$$

One could either average the usual RBM gradient $\frac{\partial}{\partial \theta} E(y_i, \mathbf{x}_i, \mathbf{h})$ for each class y (weighted by $p(y|\mathbf{x}_i)$), or sample a y from $p(y|\mathbf{x}_i)$ and only collect the gradient for that value of y. In the sampling version, the online training update for this objective can be described by replacing the statement $y^0 \leftarrow y_i$ with $y^0 \sim p(y|\mathbf{x}_i)$ in Algorithm 1. We used this version in our experiments.

In order to perform semi-supervised learning, we can weight and combine the objective of equation 6 with those of equations 1, 4 or 5

$$\mathcal{L}_{semi-sup}(\mathcal{D}_{train}, \mathcal{D}_{unlab}) = \mathcal{L}_{\text{TYPE}}(\mathcal{D}_{train}) + \beta \mathcal{L}_{unsup}(\mathcal{D}_{unlab}) \quad (7)$$

where TYPE $\in \{gen, disc, hybrid\}$. Online training according to this objective simply consists in applying the appropriate update for each training example, based on whether it is labeled or not.

6. Related Work

As mentioned earlier, RBMs (sometimes also referred to as harmoniums (Welling et al., 2005)) have already been used successfully in the past to extract useful features for another supervised learning algorithm. One of the main contributions of this paper lies in the demonstration that RBMs can be used on their own without relying on another learning algorithm, and provide a self-contained framework for deriving competitive classifiers. In addition to ensuring that the features learned by the RBM's hidden layer are discriminative, this approach facilitates model selection since the discriminative power of the hidden layer units (or features) can be tracked during learning by observing the progression of classification error on a validation set. It also makes it easier to tackle online learning problems relatively to approaches where learning features (hidden representation) and learning to classify are done in two separate phases (Hinton et al., 2006; Bengio et al., 2007).

Gehler et al. (2006); Xing et al. (2005) have shown that the features learned by an RBM trained by ignoring the labeled targets can be useful for retrieving documents or classifying images of objects. However, in both these cases, the extracted features were linear in the input, were not trained discriminatively and had to be fed to another learning algorithm which ultimately performed classification. McCallum et al. (2006) presented Multi-Conditional Learning (MCL)[1] for harmoniums in order to introduce a discriminative component to harmoniums' training, but the learned features still had to be fed to another learning algorithm.

RBMs can also provide a good initialization for the parameters of neural network classifiers (Hinton, 2007), however model selection issues arise, for instance when considering the appropriate number of learning updates and the magnitude of learning rates of each training phase. It has also been argued that the generative learning aspect of RBM training was a key element to their success as good starting points for neural network training (Bengio et al., 2007), but the extent to which the final solution for the parameters of the neural network is influenced by generative learning is not well controlled. HDRBMs can be seen as a way of addressing this issue.

Finally, though semi-supervised learning was never reported for RBMs before, Druck et al. (2007) introduced semi-supervised learning in hybrid generative/discriminative models using a similar approach to the one presented in section 5. However, they worked with log-linear models, whereas the RBMs used here can perform non-linear classification. Log-linear models depend much more on the discriminative quality of the features that are fed as input, whereas an RBM can learn useful features using their hidden variables, at the price of non-convex optimization.

7. Experiments

We present experiments on two classification problems: character recognition and text classification. In all experiments, we performed model selection on a validation set before testing. For the different RBM models, model selection[2] consisted in finding good values for

[1] We experimented with a version of MCL for the RBMs considered in this paper, however the results did not improve on those of HDRBMs.

[2] Model selection was done with a grid-like search over λ (between 0.0005 and 0.1, on a log scale), n (50 to 6000), α for HDRBMs (0 to 0.5, on a log scale) and β for semi-supervised learning (0, 0.01 or 0.1). In general, bigger values of n were found to be more appropriate with more generative learning. If no local minima was apparent, the

the learning rate λ, the size of the hidden layer n and good weights for the different types of learning (generative and semi-supervised weights). Also, the number of iterations over the training set was determined using early stopping according to the validation set classification error, with a look ahead of 15 iterations.

7.1. Character Recognition

We evaluated the different RBM models on the problem of classifying images of digits. The images were taken from the MNIST dataset, where we separated the original training set into training and validation sets of 50000 and 10000 examples and used the standard test set of 10000 examples. The results are given in Table 1. The ordinary RBM model is trained generatively (to model (x, y)), whereas RBM+NNet is an unsupervised RBM used to initialize a one-hidden layer supervised neural net (as in (Bengio et al., 2007)). We give as a comparison the results of a Gaussian kernel SVM and of a regular neural network (random initialization, one hidden layer, hyperbolic tangent hidden activation functions).

First, we observe that a DRBM outperforms a generative RBM. However, an HDRBM appears able to make the best out of discriminative and generative learning and outperforms the other models.

We also experimented with a sparse version of the HDRBM model, since sparsity is known to be a good characteristic for features of images. Sparse RBMs were developed by Lee et al. (2008) in the context of deep neural networks. To introduce sparsity in the hidden layer of an RBM in Lee et al. (2008), after each iteration through the whole training set, the biases $\mathbf{c}$ in the hidden layer are set to a value that maintains the average of the conditional expected value of these neurons to an arbitrarily small value. This procedure tends to make the biases negative and large. We follow a different approach by simply subtracting a small constant δ value, considered as an hyper-parameter[3], from the biases after each update, which is more appropriate in an online setting or for large datasets.

This sparse version of HDRBMs outperforms all the other RBM models, and yields significantly lower classification error then the SVM and the standard neural network classifiers. The performance achieved by the sparse HDRBM is particularly impressive when compared to reported performances for Deep Belief Networks (1.25% in Hinton et al. (2006)) or of a deep neural network initialized using RBMs (around 1.2% in Bengio et al. (2007) and Hinton (2007)) for the MNIST dataset with 50000 training examples.

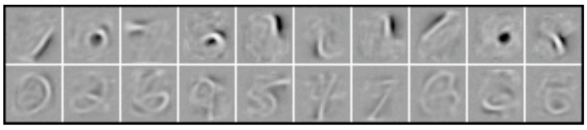


Figure 2. Filters learned by the HDRBM on the MNIST dataset. The top row shows filters that act as spatially localized stroke detectors, and the bottom shows filters more specific to a particular shape of digit.

Table 1. Comparison of the classification performances on the MNIST dataset. SVM results for MNIST were taken from http://yann.lecun.com/exdb/mnist/. On this dataset, differences of 0.2% in classification error is statistically significant.

Model	Error
RBM ($\lambda = 0.005$, $n = 6000$)	3.39%
DRBM ($\lambda = 0.05$, $n = 500$)	1.81%
RBM+NNet	1.41%
HDRBM ($\alpha = 0.01$, $\lambda = 0.05$, $n = 1500$)	1.28%
Sparse HDRBM (idem + $n = 3000$, $\delta = 10^{-4}$)	**1.16%**
SVM	1.40%
NNet	1.93%

The discriminative power of the HDRBM can be better understood by looking a the rows of the weight matrix $\mathbf{W}$, which act as filter features. Figure 2 displays some of these learned filters. Some of them are spatially localized stroke detectors which can possibly be active for a wide variety of digit images, and others are much more specific to a particular shape of digit.

7.2. Document Classification

We also evaluated the RBM models on the problem of classifying documents into their corresponding newsgroup topic. We used a version of the 20-newsgroup dataset[4] for which the training and test sets contain documents collected at different times, a setting that is more reflective of a practical application. The original training set was divided into a smaller training

grid was extended. The biases $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ were initialized to 0 and the initial values for the elements of the weight matrices $\mathbf{U}$ and $\mathbf{W}$ were each taken from uniform samples in $\left[-m^{-0.5}, m^{-0.5}\right]$, where m is the maximum between the number of rows and columns of the matrix.

[3]To chose δ, given the selected values for λ and α for the "non sparse" HDRBM, we performed a second grid-search over δ (10^{-5} and 0.1, on a log scale) and the hidden layer size, testing bigger hidden layer sizes then previously selected.

[4]This dataset is available in Matlab format here: http://people.csail.mit.edu/jrennie/20Newsgroups/20news-bydate-matlab.tgz

set and a validation set, with 9578 and 1691 examples respectively. The test set contains 7505 examples. We used the 5000 most frequent words for the binary input features. The results are given in Figure 3(a). We also provide the results of a Gaussian kernel SVM[5] and of a regular neural network for comparison.

Once again, HDRBM outperforms the other RBM models. However, here the generatively trained RBM performs better then the DRBMs. The HDRBM also outperforms the SVM and neural network classifiers.

In order to get a better understanding of how the HDRBM solves this classification problem, we first looked at the weights connecting each of the classes to the hidden neurons. This corresponds to the columns $\mathbf{U}_{\cdot y}$ of the weight matrix $\mathbf{U}$. Figure 3(b) shows a similarity matrix $\mathbf{M}(\mathbf{U})$ for the weights of the different newsgroups, where $\mathbf{M}(\mathbf{U})_{y_1 y_2} = \text{sigm}(\mathbf{U}_{\cdot y_1}^T \mathbf{U}_{\cdot y_2})$. We see that the HDRBM does not use different neurons for different newsgroups, but shares some of those neurons for newsgroups that are semantically related. Another interesting visualization of this characteristic is given in Figure 3(c), where the columns of $\mathbf{U}$ were projected on their two principal components. In both cases, we see that the HDRBM tends to share neurons for similar topics, such as computer (`comp.*`), science (`sci.*`) and politics (`talk.politics.*`), or secondary topics such as sports (`rec.sports.*`) and other recreational activities (`rec.autos` and `rec.motorcycles`).

Table 2 also gives the set of words used by the HDRBM to recognize some of the newsgroups. To obtain this table we proceeded as follows: for each newsgroup y, we looked at the 20 neurons with the largest weight among $\mathbf{U}_{\cdot y}$, aggregated (by summing) the associated input-to-hidden weight vectors, sorted the words in decreasing order of their associated aggregated weights and picked the first words according to that order. This procedure attempts to approximate the positive contribution of the words to the conditional probability of each newsgroup.

7.3. Semi-supervised Learning

We evaluated our semi-supervised learning algorithm for the HDRBM on both the digit recognition and document classification problems. We also experimented with a version (noted MNIST-BI) of the MNIST dataset proposed by Larochelle et al. (2007) where background images have been added to MNIST digit images. This version corresponds to a much harder problem, but it will help to illustrate the advantage brought by semi-supervised learning in HDRBMs. The HDRBM trained on this data used truncated exponential input units (see (Bengio et al., 2007)).

In this semi-supervised setting, we reduced the size of the labeled training set to 800 examples, and used some of the remaining data to form an unlabeled dataset $\mathcal{D}_{unlab}$. The validation set was also reduced to 200 labeled examples. Model selection[6] covered all the parameters of the HDRBM as well as the unsupervised objective weight β of equation 7. For comparison purposes, we also provide the performance of a standard non-parametric semi-supervised learning algorithm based on function induction (Bengio et al., 2006b), which includes as a particular case or is very similar to other non-parametric semi-supervised learning algorithms such as Zhu et al. (2003). We provide results for the use of a Gaussian kernel (NP-Gauss) and a data-dependent truncated Gaussian kernel (NP-Trunc-Gauss) used in Bengio et al. (2006b), which essentially outputs zero for pairs of inputs that are not near neighbors. The experiments on the MNIST and MNIST-BI (with background images) datasets used 5000 unlabeled examples and the experiment on 20-newsgroup used 8778. The results are given in Table 3, where we observe that semi-supervised learning consistently improves the performance of the HDRBM.

The usefulness of non-parametric semi-supervised learning algorithms has been demonstrated many times in the past, but usually so on problems where the dimensionality of the inputs is low or the data lies on a much lower dimensional manifold. This is reflected in the result on MNIST for the non-parametric methods. However, for high dimensional data with many factors of variation, these methods can quickly suffer from the curse of dimensionality, as argued by Bengio et al. (2006a). This is also reflected in the results for the MNIST-BI dataset which contains many factors of variation, and for the 20-newsgroup dataset where the input is very high dimensional.

Finally, it is important to notice that semi-supervised learning in HDRBMs proceeds in an online fashion and hence could scale to very large datasets, unlike more standard non-parametric methods.

7.4. Relationship with Feed-forward Neural Networks

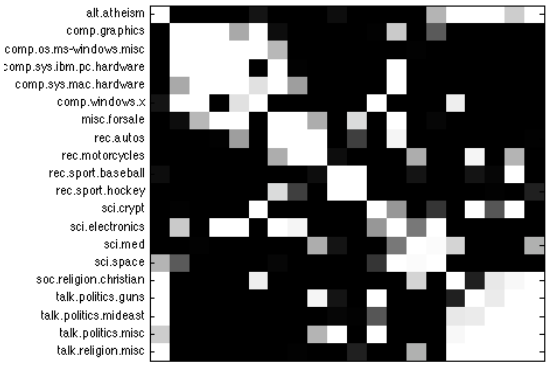
There are several similarities between discriminative RBMs and neural networks. In particular, the computation of $p(y|\mathbf{x})$ could be implemented by a single layer neural network with softplus and softmax acti-

[5] We used `libSVM` v2.85 to train the SVM model

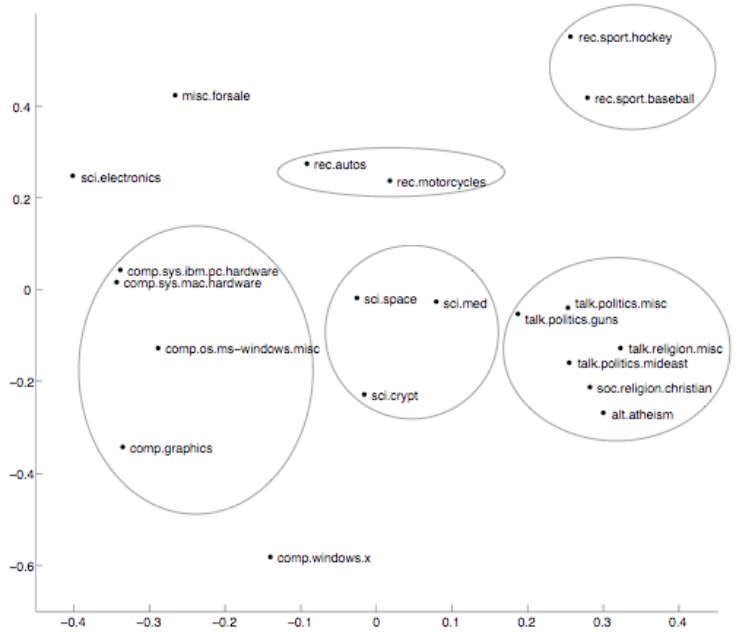
[6] $\beta = 0.1$ for MNIST and 20-newsgroup and $\beta = 0.01$ for MNIST-BI was found to perform best.

Model	Error
RBM ($\lambda = 0.0005$, $n = 1000$)	24.9%
DRBM ($\lambda = 0.0005$, $n = 50$)	27.6%
RBM+NNet	26.8%
HDRBM ($\alpha = 0.005$, $\lambda = 0.1$, $n = 1000$)	**23.8%**
SVM	32.8%
NNet	28.2%

(a) Classification performances



(b) Similarity matrix



(c) PCA embedding

Figure 3. Experiment on 20-newsgroup dataset. (Top left) Classification performance for the different models. The error differences between HDRBM and other models is statistically significant. (Bottom left) Similarity matrix of the newsgroup weights vectors $U_{\cdot y}$. (Right) Two dimensional PCA embedding of the newsgroup weights.

Table 2. Most influential words in the HDRBM for predicting some of the document classes

Class	**Words**
alt.atheism	bible, atheists, benedikt, atheism, religion
misc.forsale	sell, condition, floppy, week, am, obo
sci.crypt	sternlight, bontchev, nsa, escrow, hamburg
comp.graphics	tiff, ftp, window, gif, images, pixel
rec.autos	cars, ford, autos, sho, toyota, roads
talk.politics.guns	firearms, handgun, firearm, gun, rkba

Table 3. Comparison of the classification errors in semi-supervised learning setting. The errors in bold are statistically significantly better.

Model	MNIST	MNIST-BI	20-news
HDRBM	9.73%	42.4%	40.5%
Semi-sup HDRBM	8.04%	**37.5%**	**31.8%**
NP-Gauss	10.60%	66.5%	85.0%
NP-Trunc-Gauss	**7.49%**	61.3%	82.6%

vation functions in its hidden and output layers respectively, with a special structure in the output and hidden weights where the value of the output weights is fixed and many of the hidden layer weights are shared.

The advantage of working in the framework of RBMs is that it provides a natural way to introduce generative learning, which we used here to derive a semi-supervised learning algorithm. As mentioned earlier, a form of generative learning can be introduced in standard neural networks simply by using RBMs to initialize the hidden layer weights. However the extent to which the final solution for the parameters of the neural network is influenced by generative learning is not well controlled. This might explain the superior performance obtained by a HDRBM compared to a single hidden layer neural network initialized with an RBM (RBM+NNet in the tables).

8. Conclusion and Future Work

We argued that RBMs can and should be used as stand-alone non-linear classifiers alongside other standard and more popular classifiers, instead of merely being considered as simple feature extractors. We evaluated different training objectives that are more appropriate to train an RBM in a classification setting. These discriminative versions of RBMs integrate the process of discovering features of inputs with their use in classification, without relying on a separate classi-

fier. This insures that the learned features are discriminative and facilitates model selection. We also presented a novel but straightforward semi-supervised learning algorithm for RBMs and demonstrated its usefulness for complex or high dimensional data.

For future work, we would like to investigate the use of discriminative versions of RBMs in more challenging settings such as in multi-task or structured output problems. The analysis of the target weights for the 20-newsgroup dataset seem to indicate that RBMs would be good at capturing the conditional statistical relationship between multiple tasks or in the components in a complex target space. Exact computation of the conditional distribution for the target is not tractable anymore, but there exists promising techniques such as mean-field approximations that could estimate that distribution. Moreover, in the 20-newsgroup experiment, we only used 5000 words in input because generative training using Algorithm 1 does not exploit the sparsity of the input, unlike an SVM or a DRBM (since in that case the sparsity of the input makes the discriminative gradient sparse too). Motivated by this observation, we intend to explore ways to introduce generative learning in RBMs and HDRBMs which would be less computationally expensive when the input vectors are large but sparse.

Acknowledgments

We thank Dumitru Erhan for discussions about sparse RBMs and anonymous reviewers for helpful comments.

References

Bengio, Y., Delalleau, O., & Le Roux, N. (2006a). The curse of highly variable functions for local kernel machines. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*, 107–114. Cambridge, MA: MIT Press.

Bengio, Y., Delalleau, O., & Le Roux, N. (2006b). Label propagation and quadratic criterion. In O. Chapelle, B. Schölkopf and A. Zien (Eds.), *Semi-supervised learning*, 193–216. MIT Press.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems 19* (pp. 153–160). MIT Press.

Bouchard, G., & Triggs, B. (2004). The tradeoff between generative and discriminative classifiers. *IASC International Symposium on Computational Statistics (COMPSTAT)* (pp. 721–728). Prague.

Carreira-Perpiñan, M., & Hinton, G. (2005). On contrastive divergence learning. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, Jan 6-8, 2005, Savannah Hotel, Barbados* (pp. 33–40). Society for Artificial Intelligence and Statistics.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Cambridge, MA: MIT Press.

Druck, G., Pal, C., Mccallum, A., & Zhu, X. (2007). Semi-supervised classification with hybrid generative/discriminative methods. *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 280–289). New York, NY, USA: ACM.

Freund, Y., & Haussler, D. (1994). *Unsupervised learning of distributions on binary vectors using two layer networks* (Technical Report UCSC-CRL-94-25). University of California, Santa Cruz.

Gehler, P. V., Holub, A. D., & Welling, M. (2006). The rate adapting poisson model for information retrieval and object recognition. *ICML '06: Proceedings of the 23rd international conference on Machine learning* (pp. 337–344). New York, NY, USA: ACM.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*, 1771–1800.

Hinton, G. (2007). To recognize shapes, first learn to generate images. In P. Cisek, T. Drew and J. Kalaska (Eds.), *Computational neuroscience: Theoretical insights into brain function*. Elsevier.

Hinton, G. E., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. *Twenty-fourth International Conference on Machine Learning (ICML'2007)*.

Lasserre, J. A., Bishop, C. M., & Minka, T. P. (2006). Principled hybrids of generative and discriminative models. *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 87–94). Washington, DC, USA: IEEE Computer Society.

Le Roux, N., & Bengio, Y. (2008). Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, *to appear*.

Lee, H., Ekanadham, C., & Ng, A. (2008). Sparse deep belief net model for visual area v2. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*. Cambridge, MA: MIT Press.

McCallum, A., Pal, C., Druck, G., & Wang, X. (2006). Multi-conditional learning: Generative/discriminative training for clustering and classification. *Twenty-first National Conference on Artificial Intelligence (AAAI-06)*. AAAI Press.

Ng, A. Y., & Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS* (pp. 841–848).

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. *ICML '07: Proceedings of the 24th international conference on Machine learning* (pp. 791–798). New York, NY, USA: ACM.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. Rumelhart and J. McClelland (Eds.), *Parallel distributed processing*, vol. 1, chapter 6, 194–281. Cambridge: MIT Press.

Sutskever, I., & Hinton, G. (2007). Learning multilevel distributed representations for high-dimensional sequences. *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, March 21-24, 2007, Porto-Rico*.

Taylor, G., Hinton, G., & Roweis, S. (2006). Modeling human motion using binary latent variables. *Advances in Neural Information Processing Systems 20*. MIT Press.

Welling, M., Rosen-Zvi, M., & Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press.

Xing, E. P., Yan, R., & Hauptmann, A. G. (2005). Mining associated text and images with dual-wing harmoniums. *UAI* (pp. 633–641). AUAI Press.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. *ICML'2003*.

Transfer of Samples in Batch Reinforcement Learning

Alessandro Lazaric LAZARIC@ELET.POLIMI.IT
Marcello Restelli RESTELLI@ELET.POLIMI.IT
Andrea Bonarini BONARINI@ELET.POLIMI.IT
DEI, IIT-Lab, Politecnico di Milano, P.za Leonardo da Vinci, 32, I-20133, Milan, Italy

Abstract

The main objective of transfer in reinforcement learning is to reduce the complexity of learning the solution of a target task by effectively reusing the knowledge retained from solving a set of source tasks. In this paper, we introduce a novel algorithm that transfers samples (i.e., tuples $\langle s, a, s', r \rangle$) from source to target tasks. Under the assumption that tasks have similar transition models and reward functions, we propose a method to select samples from the source tasks that are mostly similar to the target task, and, then, to use them as input for batch reinforcement-learning algorithms. As a result, the number of samples an agent needs to collect from the target task to learn its solution is reduced. We empirically show that, following the proposed approach, the transfer of samples is effective in reducing the learning complexity, even when some source tasks are significantly different from the target task.

1. Introduction

The main objective of transfer in Reinforcement Learning (RL) is to reduce the learning time. In fact, the solution of a set of *source* tasks can provide useful information about how to solve a related *target* task, thus reducing the amount of experience needed to solve it. In order to design an effective transfer algorithm, two aspects must be taken into account: *what* to transfer, that is the knowledge retained from the source tasks, and *when* to transfer, that is the identification of tasks from which transfer is likely to be effective.

There exists a much empirical evidence about the effectiveness of techniques such as *task decomposition*, *options*, *shaping rewards*, *exploration strategies*, in improving the learning speed of RL algorithms in single-task problems. Many studies focus on extending such techniques to the transfer scenario. In particular, hierarchical solutions are often used (Şimşek et al., 2005; Mehta et al., 2005) to augment the action space with policies suitable for the solution of a wide range of tasks sharing the same dynamics, but with different goals. In (Konidaris & Barto, 2007), a set of options is learned in an *agent space* defined by a set of features shared across the tasks, thus making the options reusable even in tasks with different state spaces. The improvement of learning speed can also be obtained through direct transfer of solutions from source to target task. In this scenario, the main issue is to map the solution learned in a source task to the state-action space of the target task, thus initializing the learning algorithm to a convenient solution. Different aspects of a learning algorithm can be initialized, such as value functions, policies, and approximator structure (Taylor et al., 2007, and references therein).

Although these approaches study how the transfer of different elements from source to target tasks can impact on the performance of an RL algorithm, they often rely on the assumption that the tasks are strictly related and they do not address the problem of *negative transfer* (Rosenstein et al., 2005). In fact, transfer may bias the learning process towards solutions that are completely different from the optimal one, thus worsening the learning performance. Some works focus on the definition of measures of relatedness between tasks that can be used to select from which source tasks transfer is actually convenient. An experimental analysis of measures that estimate the expected speed-up on the basis of information such as policy overlapping, Q-values, and reward structure is reported in (Carroll & Seppi, 2005). Unfortunately, it is often difficult to compute these measures before actually solving the target task, and, thus, they can be used only to analyze the effectiveness of a transfer

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

method. In (Ferns et al., 2004), different metrics for the distance between tasks are proposed and theoretical bounds on the difference between the corresponding optimal value functions are derived.

In this paper, we focus on a perspective that received little attention so far, the transfer of samples. We propose a mechanism that selectively transfers samples from source to target tasks on the basis of the similarity of source tasks with the samples collected in the target task. We introduce a criterion to select from which sources transfer should occur, and, within each task, which samples are more likely to speed-up the learning process. As a result, through selective transfer of samples, it is possible to reduce the number of samples needed to solve the target task.

The paper is organized as follows. In Section 2 we introduce notation and we briefly review batch RL. In Section 3 we propose a novel mechanism for transfer of samples in batch RL algorithms. In Section 4 we report the experimental results of sample transfer. In Section 5 we relate our work with other transfer-learning approaches. Finally, in Section 6 we draw conclusions, and we propose directions for future works.

2. Batch Reinforcement Learning

In RL, the interaction between the agent and the environment is modeled as a discrete-time Markov Decision Process (MDP). An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$ is the transition model that assigns to each state-action pair a probability distribution over $\mathcal{S}$, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathbb{R})$ is the reward function that assigns to each state-action pair a probability distribution over $\mathbb{R}$, $\gamma \in [0, 1)$ is the discount factor. At each time step, the agent chooses an action according to its current *policy* $\pi : \mathcal{S} \to \Pi(\mathcal{A})$, which maps each state to a probability distribution over actions. The goal of an RL agent is to maximize the expected sum of discounted rewards, that is to learn an optimal policy π^* that leads to the maximization of the value function in each state. The optimal action-value function $Q^*(s, a)$ is defined by the Bellman equations

$$Q^*(s,a) = \sum_{s'} \mathcal{P}(s'|s,a) \left[R(s,a) + \gamma \max_{a'} Q^*(s',a') \right],$$

where $R(s, a) = E[\mathcal{R}(s, a)]$ is the expected reward.

One of the main drawbacks of online RL algorithms (e.g., Q-learning) when applied to real-world problems is the large amount of experience needed to solve a task. In order to overcome this drawback, batch approaches have been proposed. The main idea is to distinguish between the exploration strategy that collects samples of the form $\langle s, a, s', r \rangle$ (*sampling* phase), and the offline learning algorithm that, on the basis of the samples, computes the approximation of the action-value function (*learning* phase). In this paper, we focus on fitted algorithms, although the proposed transfer mechanism can be applied to any batch RL algorithm. The idea underlying fitted solutions (Ernst et al., 2005, and references therein) is to reformulate the learning of the value function as a sequence of regression problems. Given a set of samples, Fitted Q-Iteration (FQI)(Ernst et al., 2005) estimates the optimal action-value function by iteratively extending the optimization horizon. At the first iteration, the algorithm defines a regression problem for a 1-step problem, in which the action-value function is equal to the reward function. An approximation is computed running a chosen regression algorithm on the available samples. Thereafter, at each iteration k, corresponding to a k-step horizon, a new regression problem is stated, in which the training samples are computed exploiting the approximation of the action-value function at the previous iteration.

3. Transfer of Samples in Batch Reinforcement Learning

We formulate the transfer problem as the problem of solving a target task given a set of source tasks drawn according to a given probability distribution defined on a set of tasks which differ in either the transition model or the reward function, or both, but share the same state-action space.

Definition 1 *A* task *T is an MDP defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_T, \mathcal{R}_T, \gamma \rangle$, in which the transition model $\mathcal{P}_T$ defines the* dynamics*, and the reward function $\mathcal{R}_T$ defines the* goal*.*

Definition 2 *An* environment *E is defined by the tuple $\langle \mathcal{T}, \Omega \rangle$, where $\mathcal{T}$ is the* task space *and Ω is the* task distribution *that provides the probability of a task $T \in \mathcal{T}$ to occur.*

In batch RL algorithms, the element that mainly affects the learning performance is the set of samples used to feed the learning algorithm, the more informative the samples the better the approximation. We focus on the way this set of samples can be augmented by the inclusion of samples drawn from a set of source tasks. The basic intuition underlying this idea is that, since tasks are related through the task distribution Ω, some of the source tasks are likely to contain samples similar to those in the target task. Therefore, we expect the transfer of samples to improve performance of

batch RL algorithms even when a very limited number of samples have been actually collected from the target task. This improvement is particularly important in domains where sampling is slow and expensive (e.g., robotic applications).

More formally, we consider the scenario in which a set of n source tasks $\{S_k\}$, with $S_k \in \mathcal{T}$ and $k \in \mathbb{N}_n$, drawn from Ω are available. From each source task m samples have been collected, while only $t \ll m$ samples are available from the target task T. Let $\{\widehat{S}_k\}$ and $\widehat{T}$ be the sample sets for the source and target tasks respectively. The transfer algorithm selects a set of samples from the source tasks that are used to augment $\widehat{T}$, thus building a new set of samples $\widetilde{T}$. Finally, samples in $\widetilde{T}$ are used as input for the learning algorithm.

3.1. Task Compliance

The main problem of transferring samples across tasks is to avoid negative transfer, that is the transfer of samples from source tasks that are significantly different from the target task. Therefore, we need to identify which source tasks are more likely to have samples similar to those in the target task. Alternatively, this problem can be stated as a *model identification problem*. Let us consider the following scenario: The task space $\mathcal{T}$ contains n tasks, and m samples have been already collected from each task. Let T be a new task drawn according to Ω and $\widehat{T}$ the set of samples collected from it, with $|\widehat{T}| = t \ll m$. Since the transfer of samples from all the tasks in $\mathcal{T}$ may worsen the performance in T, we need to identify which of the previously solved tasks is actually T according to the available samples. Starting from a uniform prior over the tasks in $\mathcal{T}$, we compute the posterior distribution as the probability of a task to be the model from which samples in $\widehat{T}$ are drawn. As the number of samples t increases, the posterior distribution is updated accordingly until the total probability mass concentrates on the task equal to T. Then, the m samples previously collected in the task equal to T can be added to $\widehat{T}$ and used to feed the batch RL algorithm, thus improving its learning performance.

In the general case in which $\mathcal{T}$ is infinite or contains many tasks, the probability to have one source task identical to the target task is negligible. Thus, instead of the probability of a source task to generate *all* the samples collected in the target task, we compute its *compliance* with T as the *average* probability of generating the samples in $\widehat{T}$. Then, we transfer samples from source tasks proportionally to their compliance with the target task.

Let us consider a source task S and the set of target samples $\widehat{T}$. Given a state-action pair $\langle s, a\rangle$, the probability of S to be the model from which the target samples in $\langle s, a\rangle$ are extracted, that is the likelihood of the model in $\langle s, a\rangle$, can be simply computed by applying the Bayes theorem as [1]

$$\begin{aligned} P\left(S|\widehat{T}_{\langle s,a\rangle}\right) &\propto P\left(\widehat{T}_{\langle s,a\rangle}|S\right) P(S) \\ &= \prod_{\tau_i \in \widehat{T}_{\langle s,a\rangle}} P(\tau_i|S)\, P(S) \\ &= \prod_{\tau_i \in \widehat{T}_{\langle s,a\rangle}} \mathcal{P}_S(s'_i|s_i,a_i)\mathcal{R}_S(r_i|s_i,a_i) P(S)\,, \end{aligned} \quad (1)$$

where $\widehat{T}_{\langle s,a\rangle} = \{\tau_i \in \widehat{T} | s_i = s, a_i = a\}$, $P(S)$ is the *prior* on the source task S, and $P(S|\widehat{T}_{\langle s,a\rangle})$ is the *posterior* distribution over the source tasks in $\langle s, a\rangle$.

Unfortunately, the posterior probability cannot be immediately computed without the exact model of S. On the other hand, we have a set of m samples $\widehat{S}$ previously collected in S, from which an approximation of the continuous model can be computed. In the following, with an abuse of notation, with $\widehat{T}$ and $\widehat{S}$ we denote both the sets of samples and the model approximations built on them. Let $\tau_i = \langle s_i, a_i, s'_i, r_i\rangle$ be a sample in $\widehat{T}$, the probability of this sample to be generated by S given the set of source samples $\widehat{S}$ is

$$P(\tau_i|\widehat{S}) = \mathcal{P}_{\widehat{S}}(s'_i|s_i,a_i)\mathcal{R}_{\widehat{S}}(r_i|s_i,a_i),$$

where $\mathcal{P}_{\widehat{S}}$ and $\mathcal{R}_{\widehat{S}}$ are the approximated transition and reward models respectively. Since in continuous spaces the probability to have samples in the same state-action pair is negligible, it is necessary to use an approximation that generalizes over all the samples close to $\langle s_i, a_i\rangle$. In particular, we follow the kernel-based approximation proposed in (Jong & Stone, 2007).

Let $\varphi(\cdot)$ be a kernel function (e.g., a Gaussian kernel $\varphi(x) = \exp(-x^2/\delta)$ with bandwidth δ) applied to a given distance metric d (e.g., Euclidean or Mahalanobis distance). First of all, we define the similarity (*compliance* in the following) between the experience tuple τ_i and the experience tuples $\sigma_j \in \widehat{S}$ in terms of dynamics and reward. We define the compliance of τ_i with respect to σ_j for the transition model as

$$\lambda^{\mathcal{P}}_{ij} = w_{ij} \cdot \varphi\left(\frac{d(s'_i, s_i + (s'_j - s_j))}{\delta_{s'}}\right),$$

where

$$w_{ij} = \frac{\varphi\left(\frac{d(\langle s_i,a_i\rangle,\langle s_j,a_j\rangle)}{\delta_{sa}}\right)}{\sum_{l=1}^{m} \varphi\left(\frac{d(\langle s_i,a_i\rangle,\langle s_l,a_l\rangle)}{\delta_{sa}}\right)}.$$

[1] We assume that samples are mutually independent.

While the first term (w_{ij}) of $\lambda^{\mathcal{P}}_{ij}$ is a weight that takes into consideration the relative closeness of the two samples in the state-action space, the second term measures the similarity of the outcome. In particular, under the assumption that the transition model is continuous in the state-action space, it measures the distance between s'_i and the state obtained by applying the state transition $(s'_j - s_j)$ of σ_j to state s_i (see Jong & Stone, 2007). Therefore, the dynamics of τ_i is highly compliant with that of σ_j when they are close and their state transitions are similar.
Similarly, the compliance of the reward in τ_i with respect to that of σ_j is defined as

$$\lambda^{\mathcal{R}}_{ij} = w_{ij}\varphi\left(\frac{|r_i - r_j|}{\delta_r}\right).$$

The approximated transition and reward models are the average of the compliance between τ_i and all the samples in $\widehat{S}$

$$\mathcal{P}_{\widehat{S}}(s'_i|s_i,a_i) = \frac{1}{Z^{\mathcal{P}}}\sum_{j=1}^{m}\lambda^{\mathcal{P}}_{ij};\ \ \mathcal{R}_{\widehat{S}}(r_i|s_i,a_i) = \frac{1}{Z^{\mathcal{R}}}\sum_{j=1}^{m}\lambda^{\mathcal{R}}_{ij},$$

where $Z^{\mathcal{P}}$ and $Z^{\mathcal{R}}$ are normalization terms. Finally, we define the *compliance* of τ_i to S approximated using samples in $\widehat{S}$ as

$$\lambda_i = P(\tau_i|\widehat{S}) = \frac{1}{Z^{\mathcal{P}}Z^{\mathcal{R}}}\left(\sum_{j=1}^{m}\lambda^{\mathcal{P}}_{ij}\right)\left(\sum_{j=1}^{m}\lambda^{\mathcal{R}}_{ij}\right).$$

Recalling Equation 1, given the compliance of samples in $\langle s,a\rangle$, the probability of the model in $\langle s,a\rangle$ becomes

$$P\left(S|\widehat{T}_{\langle s,a\rangle}\right) \propto \prod_{\tau_i\in\widehat{T}_{\langle s,a\rangle}}\lambda_i P(S). \tag{2}$$

Starting from the probability in each state-action pair, we compute a global measure of the probability for the task to contain samples similar to target samples. We define the compliance of a task S as the average likelihood computed over each state-action pair experienced in the target task.

Definition 3 *Given the target samples $\widehat{T}$ and the source samples $\widehat{S}$, the* task compliance *of S is*

$$\Lambda = \frac{1}{|\widehat{U}|}\sum_{\langle s,a\rangle\in\widehat{U}} P\left(S|\widehat{T}_{\langle s,a\rangle}\right), \tag{3}$$

where $\widehat{U}$ contains all the distinct state-action pairs in the samples of $\widehat{T}$.

Since the probability to have two samples in the very same state-action pair is negligible, it follows that $|\widehat{U}| = |\widehat{T}| = t$ and the previous definition reduces to

$$\Lambda = \frac{1}{t}\sum_{i=1}^{t}\lambda_i P(S), \tag{4}$$

where $P(S)$ is a prior on the source task. When n source tasks with m samples each are available, and t samples are collected from T, the computation of the task compliance has a time complexity of $\Theta(nmt)$.

3.2. Sample Relevance

Although the measure of compliance is effective in identifying which sources, in average, are more convenient to transfer samples from, it does not provide any suggestion about which samples in $\widehat{S}$ are actually better to transfer. In the following, we introduce the concept of *relevance* of each sample $\sigma_j \in \widehat{S}$. The idea is to use the compliance of σ_j with the target task. Unfortunately, in this case, the measure of compliance is often unreliable because of a poor approximation of the target task. In fact, while each source task contains m samples, only $t \ll m$ samples are available for the target task. As a result, it may happen that the compliance of σ_j is computed according to samples τ_i that are significantly far in the state-action space. Therefore, we need a formulation of relevance strictly related to the compliance whenever the number of samples in $\widehat{T}$ close to σ_j is sufficient, while tending to a default value when the compliance is not reliable. Given the definition of compliance $\lambda^{\mathcal{P}}_{ji}$ and $\lambda^{\mathcal{R}}_{ji}$ of σ_j with a sample τ_i, the compliance of σ_j with the approximated model of the target task $\widehat{T}$ is

$$\lambda_j = P(\sigma_j|\widehat{T}) = \frac{1}{Z^{\mathcal{P}}Z^{\mathcal{R}}}\left(\sum_{i=1}^{t}\lambda^{\mathcal{P}}_{ji}\right)\left(\sum_{i=1}^{t}\lambda^{\mathcal{R}}_{ji}\right). \tag{5}$$

Let the samples τ_i be sorted in ascending order according to w_{ji}. We compute the average distance between σ_j and the samples $\tau_i \in \widehat{T}$ as

$$d_j = \frac{1}{h_j}\sum_{i=1}^{h_j} d(\langle s_j,a_j\rangle, \langle s_i,a_i\rangle), \tag{6}$$

where h_j is such that $\sum_{i=1}^{h_j} w_{ji} < \mu$, where $\mu \in (0;1]$ determines the fraction of the total number of samples considered in the computation of the average distance.

Definition 4 *Given the compliance λ_j and the average distance d_j, the* relevance *of σ_j is defined as*

$$\rho_j = \rho(\overline{\lambda}_j, d_j) = e^{-\left(\frac{\overline{\lambda}_j - 1}{d_j}\right)^2}, \tag{7}$$

where $\overline{\lambda}_j$ is the compliance normalized over all the samples in $\widehat{S}$.

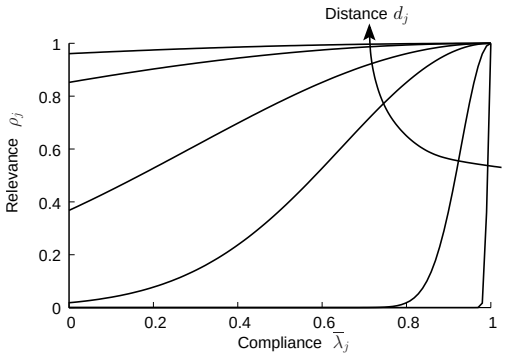


Figure 1. Relevance function for different values of d_j.

Algorithm 1 The sample transfer algorithm

Input: source tasks $\{S_k\}_{k\in\mathbb{N}_n}$, target task T
Parameters: δ_{sa}, $\delta_{s'}$, δ_r, t, m
Output: transferred sample set $\widetilde{T}$
for $k = 1$ to n **do**
  $\widehat{S}_k \leftarrow$ sampling(S_k, m)
end for
$\widehat{T} \leftarrow$ sampling(T, t)
for $k = 1$ to n **do**
  $\Lambda_k \leftarrow$ compliance$(\widehat{S}_k, \widehat{T})$
  for $\sigma_j \in \widehat{S}_k$ **do**
    $\rho_j \leftarrow$ relevance$(\sigma_j, \widehat{T})$
  end for
  Draw $(m-t)\overline{\Lambda}_k$ samples from $\widehat{S}_k$ proportionally to ρ_j
end for
Put the additional samples in $\widehat{T}$ and form the sample set $\widetilde{T}$

The relevance function is shown in Figure 1 for different values of distance d_j. As it can be noticed, sample σ_j may have high relevance in two distinct cases: *(i)* where there is a number of close samples τ_i which it is compliant with, *(ii)* where there are no close samples and, independently from the compliance, we assume a high relevance value. The assumption underlying this definition is that, whenever there is no evidence against the transfer of a sample, it is convenient to transfer it. In fact, in transfer problems the learner often needs to infer knowledge about unexplored regions of the target task. In these regions, the algorithm selects samples from the most compliant source tasks. The assumption is that samples far from target samples, but drawn from highly compliant tasks, are worth transferring, since they provide information in regions that have not been actually experienced.

3.3. Transfer of Samples

The actual transfer process is based on the compliance of the source tasks with the target samples and on the relevance of samples within each source task. For sake of simplicity, we bound the number of samples used by the learning algorithm to m. Since $|\widehat{T}| = t$ samples are already available, $m - t$ samples need to be extracted and transferred from the source tasks. For each source task S_k, the number of samples transferred to the sample set $\widetilde{T}$ of the new target task is proportional to its normalized compliance $\overline{\Lambda}_k = \frac{\Lambda_k}{\sum_{l=1}^{n} \Lambda_l}$. Then, for each source task, samples are drawn according to their relevance, thus avoiding to transfer samples that are quite dissimilar from those in the target task. The whole sample-transfer process is summarized in Algorithm 1.

4. Experiments

In order to evaluate the performance of the sample-transfer algorithm we consider a variant of the boat problem proposed in (Lazaric et al., 2007). The problem is to learn a controller to drive a boat from the left bank to the right-bank quay of a river, in presence of a non-linear current. The boat's bow coordinates, x and y, are defined in the range $[0, 200]$ and the controller sets the desired direction $a \in [-90°, -45°, 0°, 45°, 90°]$. The chosen action is perturbed by a uniform noise in the range $[-5°; 5°]$. The control frequency is set to 1Hz. For the lack of space, we refer the reader to (Lazaric et al., 2007) for the equations of the dynamics. In addition, we introduce sandbanks, i.e., regions of the river in which the speed is reduced by 20%. The reward function is defined as:

$$\mathcal{R}(x,y) = \begin{cases} +10 & x = 200 \text{ and } y \in \mathcal{Z}_s \\ \text{D(x,y)} & x = 200 \text{ and } y \in \mathcal{Z}_v \\ \text{-10} & x = 200 \text{ and } y \in \mathcal{Z}_f \\ \text{-2} & (x,y) \in \text{sandbank} \\ \text{-2} & y \le 0 \text{ or } y \ge 200 \\ 0 & \text{elsewhere} \end{cases} \quad (8)$$

where D is a function that gives a reward decreasing linearly from 10 to -10 relative to the distance from the quay, $\mathcal{Z}_s$ is the quay zone, $\mathcal{Z}_v$ is the viability zone around the quay, and $\mathcal{Z}_f$ is the failure zone in all the other bank points. The dynamics and learning parameters are summarized in Tab. 1. In the following experiments, we use Gaussian kernels and Mahalanobis distance (see Section 3.1). The results are obtained by averaging 100 runs. In FQI, we use extra-randomized trees (Ernst et al., 2005) with 50 trees, 2 random splits, and 2 minimum sample size for each node, trained on 25 iterations. Samples are obtained through random sampling run on independent episodes of maximum 50 steps each. Each episode restarts the boat at the left bank in a random position. Testing is performed on 1,000 episodes with the initial position drawn at random from 20 evenly spaced positions at the left bank.

The first experiment is meant to illustrate the effectiveness of the relevance in identifying which samples are worth transferring. We consider a transfer prob-

Parameter	Value
I/p	0.1 / 0.9
s_{MAX}/s_D	2.5 / 1.75
Z_s / Z_v width	10.0 / 10.0

Parameter	Value
m	2000
μ	0.8
δ_{sa}	0.1
δ_r	0.5
$\delta_{s'}$	0.1

Table 1. The dynamics and transfer parameters.

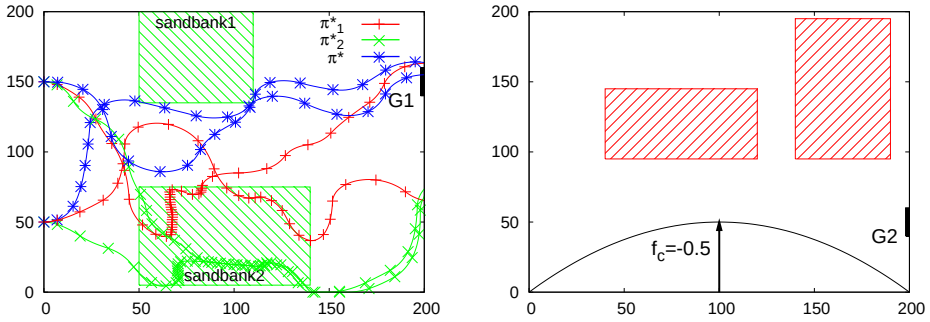


Figure 2. (*left*) Sandbanks and goal of the target task and trajectories of the optimal policies of S_1, S_2, and T tested in T. (*right*) Sandbanks and goal of S_2.

lem with three tasks in which S_1 and S_2 are the source tasks and T is the target task. In T the quay is G_1 and there are two sandbanks as illustrated in Figure 2-(*left*). In task S_1 there are two quays G_1 and G_2, and there is only one sandbank corresponding to the region labeled as *sandbank1* in Figure 2-(*left*). Task S_2 has the quay G_2 and the sandbanks illustrated in Figure 2-(*left*). While T and S_1 have the same current force ($f_c = 0.5$), the current in S_2 is in the opposite direction ($f_c = -0.5$). The source task S_2 has a completely different dynamics and reward function from those in T because of different sandbanks and current. Therefore, samples transferred from S_2 are likely to induce negative effects on the learning performance of T. Furthermore, as it can be noticed from the trajectories shown in Figure 2-(*left*), the optimal policy π^*_2 of S_2 obtains very poor performance when tested on T. On the other hand, S_1 has the same dynamics as T in large regions of the state-action space and shares one goal with T. Although its optimal policy π^*_1 is significantly different from π^*, it is possible to choose samples from $\widehat{S}_1$ to improve the performance in T.

Figure 3-(*left*) shows the performance obtained by FQI with four different configurations: *No Transfer*, *Random*, *Compliance*, and *Relevance Transfer*. In the first configuration FQI is run with samples directly collected from T. The other three configurations are run on the sample set $\widetilde{T}$ obtained by transferring samples chosen at random, according to the compliance, and according to the relevance respectively. Furthermore, we also report the performance obtained by transferring policies π^*_1 and π^*_2 as baselines. The augmentation of $\widehat{T}$ with samples drawn from S_1 and S_2 at random does not lead to any significant improvement of the performance with respect to learning directly on samples in $\widehat{T}$. In fact, the only advantage achieved with the transferred samples is that the agent avoids to go outside of the boundaries, but she learns neither to avoid sandbanks nor to achieve the goal. The main reason for this poor performance is that samples drawn from S_2 do not provide any information about the actual dynamics and rewards of T and, thus, may lead to learning very bad policies. On the other hand, the compliance-based transfer successfully excludes samples of S_2 from the transfer process (the normalized compliance of S_1 for $t = 200$ is $\overline{\Lambda}_1 = 0.93 \pm 0.09$), thus augmenting $\widehat{T}$ with samples mainly coming from S_1. Since S_1 shares with T the dynamics and the rewards in all the state space but at *sandbank2* and in the quay G_2, the transfer is positive and leads to a significant improvement in the performance of the learning process. Nonetheless, there are still many trajectories leading to the quay G_2 and crossing the sandbank because of the negative effect of transferring samples from regions with dynamics and reward different from T. In Figure 3-(*right*) we report the relevance of the samples in $\widehat{S}_1$ (averaged on all the actions). As it can be noticed, the relevance identifies the regions where samples are actually similar in source and target tasks, excluding samples coming from *sandbank2* and the lower quay G_2. As a result, the performance of the relevance-based transfer is further improved.

In order to evaluate the relative improvement of transfer, we compute the area ratio (Taylor et al., 2007) of the three transfer configurations, defined as the difference between the accumulated reward with and without transfer divided by the reward accumulated without transfer. Figure 3-(*center*) shows the area ratio for the three transfer configurations. As it can be noticed, random transfer does not lead to any significant improvement, while relevance-based transfer improves the performance by $75.3\% \pm 13.2$. All the differences are statistically significant ($p < 0.01$).

In the previous experiment, source and target tasks have been designed to show how the algorithm works. Now, we consider the general case in which tasks are

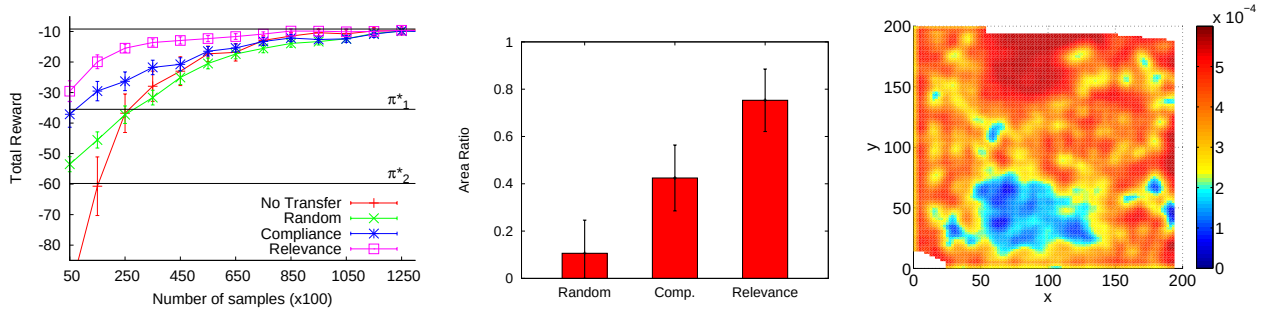


Figure 3. (*left-center*) Total reward and area ratio. (*right*) Relevance of samples in $\widehat{S}_1$ at convergence.

drawn from an infinite task space $\mathcal{T}$. For sake of simplicity, we consider the same target task of the previous experiment, while source tasks have current f_c=0.5 and one sandbank. Source tasks are drawn from a distribution Ω such that the coordinates of the center, height, and width of the sandbank are uniformly drawn from the space $[20.0; 180.0] \times [20.0; 180.0] \times [40.0; 100.0] \times [40.0; 100.0]$, while the quay position is drawn uniformly in $[20.0; 180.0]$. In Figure 4, we report the results of relevance-based transfer obtained by averaging the result with 10 different sets of five source tasks. Although the source tasks are different from the target in large regions, the transfer algorithm is able to identify which samples are worth transferring from the source tasks and it successfully improves the learning performance with an area ratio of $59.5\% \pm 15.4$.

Since the algorithms of transfer in RL proposed so far rely on temporal-difference or model-based learning algorithms, an empirical comparison with the performance of sample transfer would not be fair. Nonetheless, in the next section, we discuss its similarities and differences with other transfer approaches.

5. Related Works

In (Sunmola & Wyatt, 2006) a Bayesian approach is used for transfer of MDPs, where source task models are pre-posteriors for the distributions of the parameters of the target model and model-based RL is used to compute the solution. Although we similarly adopt a Bayesian argument in the compliance, we directly transfer samples and we use a model-free learning algorithm. Furthermore, instead of a parametric approximation of the model of the source tasks, we follow a non-parametric solution.

The task compliance can be interpreted as a sort of distance metric between tasks. In (Ferns et al., 2004), distance metrics for MDP similarity are introduced in the context of bisimulation to aggregate states with similar dynamics and reward. Under a transfer perspective, these metrics can be used to measure the difference between states in distinct tasks and to bound the performance loss of using the optimal policy of a source task in the target task. Unfortunately, this technique cannot be directly applied to our scenario for different reasons. The computation of the Kantorovich distance between different states is very expensive, because it requires the solution of a complex optimization problem. Furthermore, the proposed algorithm needs either the exact models of tasks or accurate approximations. On the other hand, we adopt a solution with low computational complexity, linearly depending on the number of samples of the source tasks. Finally, empirical analysis (Phillips, 2006) showed that the theoretical bounds on the performance loss are too loose and they do not provide useful directions about the actual performance of the transferred policy.

The transfer of samples is also related to works about transfer of solutions in the RL context (Taylor et al., 2007). Although the transfer of samples or solutions (e.g., policies) from only one source task obtains similar results, there are situations in which sample transfer can obtain better results than solution transfer. Even when the difference between source and target tasks is limited to few state-action pairs, the optimal policies of the two tasks can be significantly different and the transfer may achieve very poor performance. On the other hand, the transfer of samples can still be effective. In fact, since most of the samples in the two tasks are identical, the learning algorithm can benefit from samples coming from the source task independently from the actual difference of their optimal policies. Furthermore, the transfer of samples does not require to actually solve the source tasks, and it can be used even when the samples are not enough to solve source tasks. In (Mehta et al., 2005) a solution in which the model-based hierarchical task decomposition allows for transfer at multiple levels of the hierarchy is proposed. This approach relies on the assumption that rewards are a linear combination of *basis* re-

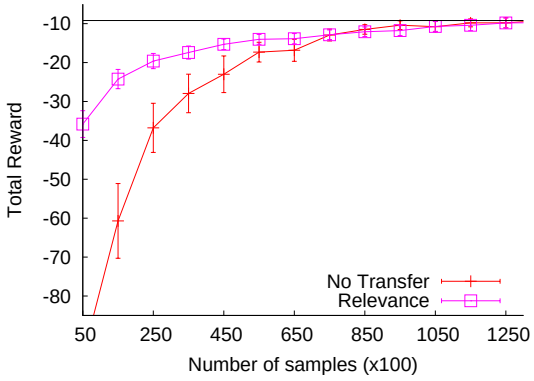


Figure 4. Performance with five random source tasks.

ward functions and it can be applied only to problems of goal transfer, with a fixed transition model. On the other hand, sample transfer can be applied to any transfer scenario. Finally, a method for mapping samples from a source to a target task with different state-action space is proposed in (Taylor et al., 2008).

6. Conclusions

In this paper, we introduced a mechanism for the transfer of samples with the aim of improving the learning performance. The main advantages of the proposed solution are: *(i)* it is independent from the similarity of the policies and action-value functions of the tasks at hand and, thus, can be applied to a wide range of problems, *(ii)* it is independent from the batch RL algorithm, *(iii)* it can be applied to any transfer problem in which either reward or transition or both models change. Experimental results show the effectiveness of the method in improving the learning performance and in avoiding negative transfer when the source tasks are significantly different from the target.

Some aspects of the algorithm can be improved in future works. In case of tasks that either share exactly the same transition or reward model, it is possible to transfer only the part of the samples common to all the tasks. For instance, if two tasks share the same transition model, but have different goals, it is possible to transfer the $\langle s, a, s' \rangle$ part of the samples and to "complete" the sample using an approximation of the reward function of the target task (e.g., using the first iteration of FQI). Furthermore, the sample-transfer algorithm could be integrated with the model proposed in (Taylor et al., 2008) in order to deal with problems with tasks defined on different state-action spaces.

References

Carroll, J. L., & Seppi, K. (2005). Task similarity measures for transfer in reinforcement learning task libraries. *Proceddings of IJCNN* (pp. 803–808).

Şimşek, O., Wolfe, A. P., & Barto, A. G. (2005). Identifying useful subgoals in reinforcement learning by local graph partitioning. *Proceedings of ICML* (pp. 816–823).

Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, *6*, 503–556.

Ferns, N., Panangaden, P., & Precup, D. (2004). Metrics for finite markov decision processes. *Proceedings of UAI* (pp. 162–169).

Jong, N. K., & Stone, P. (2007). Model-based function approximation for reinforcement learning. *Proceedings of AAMAS* (pp. 1–8).

Konidaris, G., & Barto, A. G. (2007). Building portable options: Skill transfer in reinforcement learning. *Proceedings of IJCAI* (pp. 895–900).

Lazaric, A., Restelli, M., & Bonarini, A. (2007). Reinforcement learning in continuous action spaces through sequential monte carlo methods. *Advances in Neural Information Processing Systems*.

Mehta, N., Natarajan, S., Tadepalli, P., & Fern, A. (2005). Transfer in variable-reward hierarchical reinforcement learning. *NIPS Workshop on Inductive Transfer*.

Phillips, C. (2006). *Knowledge transfer in markov decision processes* (Technical Report). McGill School of Computer Science. (http://www.cs.mcgill.ca/˜martin/usrs/phillips.pdf).

Rosenstein, M. T., Marx, Z., Kaelbling, L. P., & Dietterich, T. G. (2005). To transfer or not to transfer. *NIPS Workshop on Inductive Transfer*.

Sunmola, F. T., & Wyatt, J. L. (2006). Model transfer for markov decision tasks via parameter matching. *Workshop of the UK Planning and Scheduling Special Interest Group*.

Taylor, M. E., Jong, N. K., & Stone, P. (2008). Transferring instances for model-based reinforcement learning. *AAMAS 2008 Workshop on Adaptive Learning Agents and Multi-Agent Systems*.

Taylor, M. E., Stone, P., & Liu, Y. (2007). Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, *8*, 2125–2167.

Local Likelihood Modeling of Temporal Text Streams

Guy Lebanon LEBANON@STAT.PURDUE.EDU
Yang Zhao ZHAO18@STAT.PURDUE.EDU
Department of Statistics, Purdue University – West Lafayette, IN 47907

Abstract

Temporal text data is often generated by a time-changing process or distribution. Such a drift in the underlying distribution cannot be captured by stationary likelihood techniques. We consider the application of local likelihood methods to generative and conditional modeling of temporal document sequences. We examine the asymptotic bias and variance and present an experimental study using the RCV1 dataset containing a temporal sequence of Reuters news stories.

1. Introduction

Time stamped documents such as news stories often cannot be accurately modeled by a single time invariant distribution. An alternative is to assume that the concepts underlying the distribution generating the data drift with time. In other words, the data is generated by a time dependent process $z^{(t)} \sim p_t(z), t \in I \subset \mathbb{R}$ whose approximation $\{\hat{p}_t : t \in I\}$ becomes the main objective of the learning task. We assume that the time t is a continuous quantity, even in cases where the realized time points form a discrete sample. For example, assuming that the time stamps represent the days of the year when the documents were authored, we assume that the set $\{1, \ldots, 365\}$ is a discrete sample from a underlying continuous interval $[1, 365]$. We further assume that the data samples $z^{(t)}$, sampled from p_t, correspond to pairs $z^{(t)} = (x, y)$ constituting a document x and a categorial-valued label y. Such pairs (x, y) appear often in practice, for example with y corresponding to the document topic (Lewis et al., 2004), sentiment (Pang & Lee, 2005), author (Mosteller & Wallace, 1964) or Email spam/no-spam (Mulligan, 1999).

Assuming that our data is a set of time stamped documents and labels $(t, (x, y))$, the drift $p_t(x, y)$ can be characterized by considering the temporal transition of the joint distribution $p_t(x, y)$, the conditionals $p_t(y|x)$, $p_t(x|y)$, or the marginals $p_t(x), p_t(y)$. The choice of which of the distributions above to model depends on the application at hand. For example, modeling $p_t(y|x)$ is usually sufficient for document classification purposes while modeling $p_t(x|y)$ is necessary for language modeling which is an important component in speech recognition, machine translation, and IR.

We demonstrate the presence of concept drift in practice by considering the Reuters RCV1 dataset (Lewis et al., 2004) which contains over 800,000 news stories gathered in a period spanning 365 consecutive days and categorized according to topic. Figure 1 displays the temporal change in the relative frequency (number of appearance in a document divided by document length) of three words: `million`, `common`, and `Handelsgesellschaft` (German trade unions) for documents in the most popular RCV1 category titled `CCAT`. It is obvious from these plots that the relative frequency of these words vary substantially in time. For example, the word `Handelsgesellschaft` appear in 8 distinct time regions, representing time points in which German trade unions were featured in the Reuters news archive.

The temporal variation in relative frequencies illustrated by Figure 1 corresponds to a drift in the distribution generating the data. Since the drift is rather pronounced, standard estimation methods based on maximum likelihood are not likely to accurately model the data. In this paper, we consider instead estimating $\{p_t(x, y) : t \in I\}$ based on the the local likelihood principle. Local likelihood is a locally weighted version of the loglikelihood with the weights determined by the difference between the time points associated with the sampled data and a the time at which the inference takes place.

After presenting a more formal discussion of concept drift in Section 3 and the definition of local likelihood in Section 4 we turn to examine in detail the case of

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

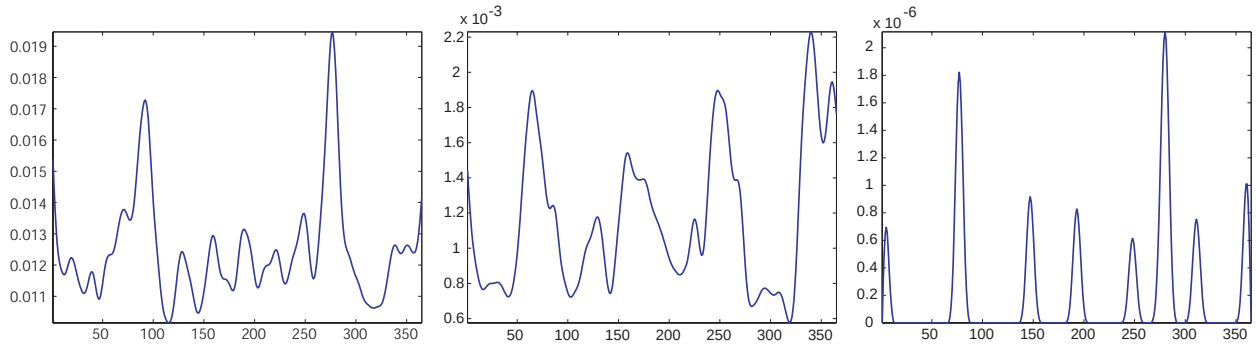


Figure 1. Estimated relative frequency (number of appearances in a document divided by document length) of three words from the most popular category in RCV1 as a function of time. The three panels correspond to the words `million`, `common`, and `Handelsgesellschaft` (German trade unions). The displayed curves were smoothed to remove sampling noise.

modeling $p_t(x|y)$ with local likelihood for n-grams and modeling $p_t(y|x)$ with local likelihood for logistic regression. In the case of 1-grams or the naive Bayes model, we provide a precise as well as asymptotic description of the bias and variance which illuminates certain facts concerning the selection of weights and the difference between the online and offline scenarios. Experiments conducted on the RCV1 dataset demonstrates the local likelihood estimation in practice and contrasts it with more standard non-local alternatives.

2. Related Work

Concept drift or similar phenomena under different names have been studied in a number of communities. It has recently gained interest primarily due to an increase in the need to model large scale temporal data streams.

Early machine learning literature on the concept drift problem involved mostly computational learning theory tools (Helmbold & Long, 1994; Kuh et al., 1990). Hulten et al. (2001) studied the problem in the context of datamining large scale streams whose distribution change in time. More recently, Forman (2006) studied the concept drift phenomenon in the context of information retrieval in large textual databases. Sharan and Neville (2007) consider the modeling of temporal changes in relational databases and its application to text classification.

Overall, the prevailing techniques have been to train standard methods on examples obtained by filtering the data through a sliding window. Tibshirani and Hastie (1987) developed the local likelihood idea in the statistics community within the context of non-parametric smoothing and regression. More details on local likelihood can be found in (Loader, 1999).

3. The Concept Drift Phenomenon and its Estimation

Formally, the concept drift phenomenon may be thought of as a smooth flow or transition of the joint distribution of a random vector. We will focus on the case of a joint distribution of a random vector X and a random variable Y representing predictor and response variables. We will also restrict our attention to temporal or one dimensional drifts.

Definition 1. *Let X and Y be two discrete random vectors taking values in $\mathcal{X}$ and $\mathcal{Y}$. A smooth temporal drift of X, Y is a smooth mapping from $I \subset \mathbb{R}$ to a family of joint distributions*

$$t \mapsto p_t(x, y) \stackrel{\text{def}}{=} p_t(X = x, Y = y).$$

By restricting ourselves to discrete random variables we can obtain a simple geometrical interpretation of concept drift. Denoting the simplex of all distributions over the set S by

$$\mathbb{P}_S \stackrel{\text{def}}{=} \left\{ r \in \mathbb{R}^{|S|} \;:\; \forall i \, r_i \geq 0, \; \sum_{i=1}^{|S|} r_i = 1 \right\} \quad (1)$$

we have that Definition 1 is equivalent to a smooth parameterized curve in the simplex $\mathbb{P}_{\mathcal{X} \times \mathcal{Y}}$.

The drift in the joint distribution can be decomposed in several ways. The first decomposition $p_t(x, y) = p_t(x|y)p_t(y)$ is useful for generative modeling and the second decomposition $p_t(x, y) = p_t(y|x)p_t(x)$ is useful for conditional modeling. In the generative case we will focus on modeling $p_t(x|y)$ since modeling $p_t(y)$ is typically an easier problem due to its lower dimensionality (in most cases involving text documents $|\mathcal{Y}| \ll |\mathcal{X}|$). In the case of conditional modeling, we focus on modeling $p_t(y|x)$ and we ignore the drift in the marginal $p_t(x)$ since it is irrelevant for discriminative tasks.

In both cases we assume that our data is a set of time-stamped labeled documents sampled from $p_t(x, y)$ where the time points t are sampled from a distribution $g(t)$. If g is a continuous density, the number of samples at time t, denoted by N_t, is no greater than 1 with probability 1. In practice, however, we allow N_t to be larger than 1 in order to account for the discretization of time. We thus have the data

$$D = \{(x_{tj}, y_{tj}) : t \in I \subset \mathbb{R}, j = 1, \ldots, N_t\} \quad (2)$$

where the time points are sampled from $g(t)$ and $(x_{tj}, y_{tj}) \sim p_t(x, y)$.

To illustrate these concepts in the context of the RCV1 dataset, we display in Figure 2 the total number of words per day (left) and the total number of documents per day (right) corresponding to the most popular category in RCV1. As is evident from the right panel, $g(t)$ is a highly non-uniform density corresponding to varying amount of news content in different dates.

It is easy to come up with two simple solutions to the problem of concept drift modeling. The first solution, called the extreme global model, is to simply ignore the temporal drift and use all of the samples in D regardless of their time stamp. This approach results in a single global model $\hat{p}$ which serves as an estimate for the entire flow $\{p_t, t \in I\}$ effectively modeling the concept drift as a degenerate curve equivalent to a stationary point in the simplex. The second simple alternative, called the extreme local model, is to model p_t using only data sampled from time t i.e. $\{(x_{tj}, y_{tj}) : j = 1, \ldots, N_t\}$. This alternative decomposes the concept drift estimation into a sequence of disconnected estimation problems.

The extreme local model has the benefit that if the individual estimation problems are unbiased, the estimation of the concept drift is unbiased as well. The main drawback of this method is the high estimation variance resulting from the relatively small number of daily samples N_t used to estimate the individual models. Furthermore, assuming D is finite we can only estimate the drift in the finite number of time points appearing in the dataset D (since we have no training data for the remaining time points). On the other hand, the extreme global model enjoys low variance since it uses all data points to estimate p_t. Its main drawback is that it is almost always heavily biased due to the fact that samples from one distribution p_{t_1} are used to estimate a different distribution p_{t_2}.

It is a well known fact that the optimal solution in terms of minimizing the mean squared estimation error usually lies between the extreme local and extreme global models. An intermediate solution can trade-off increased bias for reduced variance and can significantly improve the estimation accuracy. Motivated by this principle, we employ local smoothing in forming a local version of the maximum likelihood principle which includes as special cases the two extreme models mentioned above. The intuition behind local smoothing in the present context is that due to the similarity between p_t and $p_{t+\epsilon}$, it makes sense to estimate p_t using samples from neighboring time points $t + \epsilon$. However, in contrast to the global model the contribution of points sampled from $p_{t+\epsilon}$ towards estimating p_t should decrease as ϵ increases.

4. Local Likelihood and Concept Drift

The local likelihood principle extends the ideas of non-parametric regression smoothing and density estimation to likelihood-based inference. We concentrate on using the local likelihood principle for estimating $p_t(x|y)$ and $p_t(y|x)$ which are described next.

4.1. Local Likelihood for n-Gram Estimation

We apply local likelihood to the problem of estimating $p_t(x|y)$ by assuming the naive Bayes assumption i.e. that $x|y$ is generated by a multinomial distribution or its n-gram extensions. Assuming documents contain words belonging to a finite dictionary of size V, the naive Bayes assumption may be stated as

$$p_t(x|y) \propto \prod_{w \in V} \theta_w^{c(w,x)}, \qquad \theta \in \mathbb{P}_V \quad (3)$$

where $c(w, x)$ represents the number of times word w appears in document x. Similarly, the n-gram model extends naive Bayes (3) by considering n-order Markov dependency. The naive Bayes and n-gram are a mainstay of statistical text processing (Manning & Schutze, 1999) and usually lead to accurate language modeling, especially when appropriate smoothing is used (Chen & Goodman, 1998). For notational simplicity we consider the problem of estimating $p_t(x)$ rather than the equivalent $p_t(x|y)$ and we concentrate on naive Bayes i.e. 1-gram. Extending the discussion to n-grams with $n > 1$ is relatively straightforward and is omitted due to lack of space.

Applied to the concept drift problem, the local log-likelihood at time t is a smoothed or weighted version of the loglikelihood of the data D in (2) with the amount of smoothing determined by a non-negative smoothing kernel $K_h : \mathbb{R} \to \mathbb{R}$

$$\ell_t(\eta|D) \stackrel{\text{def}}{=} \sum_{\tau \in I'} K_h(t - \tau) \sum_{j=1}^{N_\tau} \log p(x_{\tau j}\,; \eta). \quad (4)$$

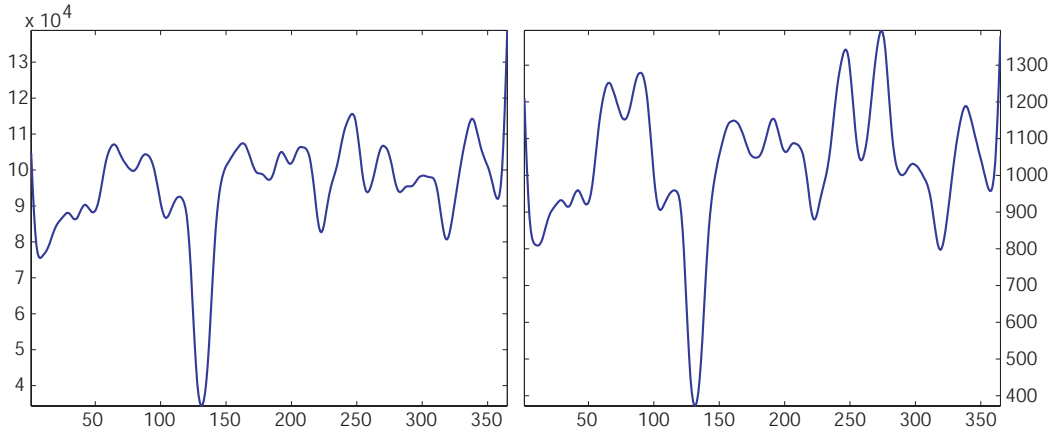


Figure 2. Total number of words per day (left) and documents per day (right) for the most popular category in RCV1. The displayed curves were smoothed to remove sampling noise.

We assume that the kernel function is a normalized density concentrated around 0 and parameterized by a scale parameter $h > 0$ reflecting its spread and satisfying the relation $K_h(r) = h^{-1}K(r/h)$ for some $K : \mathbb{R} \to \mathbb{R}$ referred to as the base kernel form. We further assume that K has bounded support and $\int u^r K(u)\, du < \infty$ for $r \leq 2$. Wand and Jones (1995) provide more details on the formal requirements of a smoothing kernel.

Three popular kernel choices are the tricube, triangular and uniform kernels, defined as $K_h(r) = h^{-1}K(r/h)$ where the $K(\cdot)$ functions are respectively

$$K(r) = (1 - |r|^3)^3 \cdot 1_{\{|r|<1\}} \quad (5)$$

$$K(r) = (1 - |r|) \cdot 1_{\{|r|<1\}} \quad (6)$$

$$K(r) = 2^{-1} \cdot 1_{\{|r|<1\}}. \quad (7)$$

The uniform kernel is the simplest choice and leads to a local likelihood (4) equivalent to filtering the data by a sliding window i.e. $\hat{\theta}_t$ is computed based on data from adjacent time points with uniform weights. Unfortunately, it can be shown that the uniform kernel is suboptimal in terms of its statistical efficiency or rate of convergence to the underlying distribution (Wand & Jones, 1995). Surprisingly, the triangular kernel has a higher statistical efficiency than the Gaussian kernel and is the focus of our experiments in this subsection. We use the tricube kernel in the next subsection.

The scale parameter h is central to the bias-variance tradeoff. Large h represents more uniform kernels achieving higher bias and lower variance. Small h represents a higher degree of locality or lower bias but higher variance. Since $\lim_{h\to 0} K_h$ approaches Dirac's delta function and $\lim_{h\to\infty} K_h$ approaches a constant function the local log-likelihood (4) interpolates between the loglikelihoods of the extreme local model and the extreme global model mentioned in Section 3 as h ranges from 0 to $+\infty$.

Solving the maximum local likelihood problem for each t provides an estimation of the entire drift $\{\hat{\theta}_t : t \in \mathbb{R}\}$ with $\hat{\theta}_t = \arg\max_{\eta\in\Theta} \ell_t(\eta|D)$. In the case of the naive Bayes or n-gram model we obtain a closed form expression for the local likelihood maximizer $\hat{\theta}_t$ as well as convenient expressions for its bias and variance. In general, however, there is no closed form maximizer and iterative optimization algorithms are needed in order to obtain $\hat{\theta}_t = \arg\max_{\eta\in\Theta} \ell_t(\eta|D)$ for all t.

We denote the length of a document in (2) by $|x_{tj}| \stackrel{\text{def}}{=} \sum_{v\in V} c(x_{tj}, v)$ and the total number of words in day t in (2) by $|x_t| \stackrel{\text{def}}{=} \sum_{j=1}^{N_t} |x_{tj}| = \sum_{v\in V}\sum_{j=1}^{N_t} c(v, x_{tj})$. We assume that the length of documents x_{tj} is independent of t and is drawn from a distribution with expectation λ.

Under the above assumptions, the local likelihood (4) of the naive Bayes model becomes

$$\ell_t(\eta|D) = \sum_{\tau\in I'} K_h(t-\tau) \sum_{j=1}^{N_\tau} \sum_{w\in V} c(w, x_{\tau j}) \log \eta_w$$

where $\eta \in \mathbb{P}_V$. The local likelihood has a single global maximum whose closed form is obtained by setting to 0 the gradient of the Lagrangian

$$0 = \frac{1}{[\hat{\theta}_t]_w} \sum_{\tau\in I} K_h(t-\tau) \sum_{j=1}^{N_\tau} c(w, x_{\tau j}) + \lambda_w$$

to obtain

$$[\hat{\theta}_t]_w = \frac{\sum_{\tau \in I} K_h(t-\tau) \sum_{j=1}^{N_\tau} c(w, x_{\tau j})}{\sum_{\tau \in I} K_h(t-\tau)|x_\tau|}. \quad (8)$$

The estimator $\hat{\theta}_t$ is a normalized linear combination of word counts where the combination coefficients are determined by the kernel function and normalized by the number of words in different days. We note that $\hat{\theta}_t$ in (8) is different from a weighted averaging of the relative frequencies $c(w, x_{\tau j}) / \sum_{w'} c(w', x_{\tau j})$.

We distinguish between two fundamental scenarios for predicting the drift θ_t.

Offline scenario: The goal is to estimate the drift $\{\theta_t : t \in \mathbb{R}\}$ given the entire dataset D. In this case we will consider symmetric kernels $K(r) = K(-r)$ which will achieve an increased convergence rate of $\hat{\theta}_t \to \theta_t$ as indicated by Proposition 2.

Online scenario: The goal is estimate a model for the present distribution θ_t using training data from the past i.e. a dataset whose time stamps are strictly smaller than t. This corresponds to situations where the data arrives sequentially as a temporal stream and at each time point a model for the present is estimated using the available stream at that time. We realize this restriction by constraining K to satisfy $K(r) = 0, r \leq 0$.

As with other statistical estimators, the accuracy of $\hat{\theta}_t$ may be measured in terms of its mean squared error $\mathsf{E}\,(\hat{\theta}_t - \theta_t)^2$ which decomposes as the sum of the squared bias and variance of $\hat{\theta}_t$. Examining these quantities allow us to study the convergence rate of $\hat{\theta}_t \to \theta$ and its leading coefficient .

Proposition 1. *The bias vector* $\mathsf{bias}(\hat{\theta}_t) \stackrel{\text{def}}{=} \mathsf{E}\hat{\theta}_t - \theta_t$ *and variance matrix of* $\hat{\theta}_t$ *in* (8) *are*

$$\mathsf{bias}(\hat{\theta}_t) = \frac{\sum_{\tau \in I} K_h(t-\tau)|x_\tau|\,(\theta_\tau - \theta_t)}{\sum_{\tau \in I} K_h(t-\tau)|x_\tau|} \quad (9)$$

$$\mathsf{Var}(\hat{\theta}_t) = \frac{\sum_{\tau \in I} K_h^2(t-\tau)|x_\tau|\,(diag(\theta_\tau) - \theta_\tau \theta_\tau^\top)}{\left(\sum_{\tau \in I} K_h(t-\tau)|x_\tau|\right)^2} \quad (10)$$

where $diag(z)$ *is the diagonal matrix* $[diag(z)]_{ij} = \delta_{ij} z_i$.

Proof. The random variable (RV) $c(w, x_{\tau j})$ is distributed as a sum of multivariate Bernoulli RVs, or single draws from multinomial distribution. The expectation and variance of the estimator are that of a linear combination of iid multinomial RVs. To conclude the proof we note that for $Y \sim \mathrm{Mult}(1, \theta)$, $\mathsf{E}\,Y = \theta$, $\mathsf{Var}(\theta) = \mathrm{diag}(\theta) - \theta\theta^\top$. □

Examining Equations (9)-(10) reveals the expected dependency of the bias on h and θ_t. The contribution to the bias of the terms $(\theta_\tau - \theta_t)$, for large $|\tau - t|$, will decrease as h decreases since the kernel becomes more localized and will reduce to 0 as $h \to 0$. Similarly, for slower drifts, $\|\theta_\tau - \theta_t\|, t \approx \tau$ will decrease and reduce the bias.

Despite the relative simplicity of Equations (9)-(10), it is difficult to quantitatively capture the relationship between the bias and variance, the sample size, h, λ, and the smoothness of θ_t, g. Towards this goal we derive the following asymptotic expansions.

Proposition 2. *Assuming (i)* θ, g *are smooth in* t, *(ii)* $h \to 0, hn \to \infty$, *(iii)* $g > 0$ *in a neighborhood of* t, *and (iv) document lengths do not depend on* t *and have expectation* λ, *we have in the offline case*

$$\mathsf{bias}(\hat{\theta}_t|I) = h^2 \mu_{21}(K) \left(\dot{\theta}_t \frac{g'(t)}{g(t)} + \frac{1}{2}\ddot{\theta}_t \right) + o_P(h^2) \quad (11)$$

$$\mathsf{Var}(\hat{\theta}_t|I) = \frac{\mu_{02}(K)}{(nh)g(t)\lambda}(diag(\theta_t) - \theta_t\theta_t^\top) + o_P((nh)^{-1})$$

and in the online case

$$\mathsf{bias}(\hat{\theta}_t|I) = h\mu_{11}(K)\dot{\theta}_t + o_P(h) \quad (12)$$

$$\mathsf{Var}(\hat{\theta}_t|I) = \left(\frac{\mu_{02}(K)}{nhg(t)\lambda} + \frac{\mu_{12}(K)g'(t)}{ng^2(t)\lambda} \right)(diag(\theta_t) - \theta_t\theta_t^\top) + \frac{\mu_{12}(K)}{n\lambda g(t)}(diag(\dot{\theta}_t) - \dot{\theta}_t\theta_t^\top - \theta_t\dot{\theta}^\top) + o_P((nh)^{-1})$$

where $\mu_{kl}(K) \stackrel{\text{def}}{=} \int u^k K^l(u)\,du$ *is assumed to be finite and* $\dot{\theta}_t$ *is the vector* $[\dot{\theta}_t]_i = \frac{d}{dt}[\theta_t]_i$.

The proof is somewhat similar to the derivation of the asymptotic bias and variance of the Nadaraya-Watson local regression (Wand & Jones, 1995) and is omitted due to space limitations. The notation $g_n \xrightarrow{p} f$ represents convergence in probability of g_n to f i.e. $\forall \epsilon > 0, P(|g_n - f| > \epsilon) \to 0$, and $g_n = o_P(f_n)$ represents $g_n / f_n \xrightarrow{p} 0$.

Corollary 1. *Under the assumptions in Proposition 2, and in particular* $h \to 0, nh \to \infty$, *the estimator* $\hat{\theta}_t$ *is consistent i.e.* $\hat{\theta}_t \xrightarrow{p} \theta_t$ *in both the offline and online settings.*

Proposition 2 specifies the conditions for consistency as well as the rate of convergence. In particular, the bias of online kernels converges at a linear rather than quadratic rate. In either cases, the estimator is biased and inconsistent unless $h \to 0, n \to \infty$ and $nh^{-1} \to$

∞. Expressions (11)-(12) reveal the performance gain associated with a slower drift and sampling density g indicated by $\dot{\theta}_t$ and $g'(t)$ and with more (represented by n) and longer (represented by λ) documents.

Figure 3 displays the RCV1 per-word test set loglikelihood for the online and offline scenarios as a function of the (triangular) kernel's bandwidth. As expected, offline kernels performs better than online kernels with both achieving the best performance for a bandwidth approximately 25 which corresponds to a support of 25 days in the online scenario and 50 days in the offline scenario. Note that in addition to obtaining higher accuracy than the global model corresponding to $h \to \infty$, the local model enjoys computational efficiency as it ignores a large portion of the training data.

A central issue in local likelihood modeling is selecting the appropriate bandwidth h. A practical solution is to use cross validation or some other automatic bandwidth selection mechanism. On RCV1 data, the performance of such cross validation schemes is very good and the estimated bandwidth possesses test set loglikelihood that is almost identical to the optimal bandwidth (see Figure 4, left).

Allowing the kernel scale to vary over time results in a higher modeling accuracy than using fixed bandwidth for all dates (see Figure 4, right). A time-dependent cross validation procedure may be used to approximate the time-dependent optimal bandwidth which performs slightly better than the fixed-date cross validation estimator. Note that the accuracy with which the cross validation estimator approximates the optimal bandwidth is lower in the time-dependent or varying bandwidth situation due the fact that much less data is available in each of the daily cross validation problems.

From a theoretical perspective, the asymptotic bias and variance can be used to characterize the optimal bandwidth and study its properties. Minimizing the (offline) leading term of sum of component-wise MSE with respect to h we obtain the bandwidth estimator

$$\hat{h}_t^5 = \frac{\mu_{02}(K)\mathrm{tr}(\mathrm{diag}(\theta_t) - \theta_t\theta_t^\top)}{4n\lambda\mu_{21}^2(K)\sum_j \left([\dot{\theta}_t]_j\, g'(t)/\sqrt{g(t)} + \sqrt{g(t)}[\ddot{\theta}_t]_j\,/2\right)^2}. \quad (13)$$

As expected, the optimal bandwidth decreases as $n, \lambda, \|\dot{\theta}_t\|, \|\ddot{\theta}\|$ increases. Intuitively this makes sense since in these cases the variance decreases and bias either increases or stays constant. In practice, $\dot{\theta}_t, \ddot{\theta}_t$ may vary significantly with time which leads to the conclusion that a single bandwidth selection for all t may not perform adequately. These changes are illustrated in Figure 5 (left) which demonstrates the temporal change in the gradient norm.

Perhaps more interesting than the dependency of the optimal bandwidth on $n, \lambda, \dot{\theta}_t, \ddot{\theta}_t$ is its dependency on the time sampling distribution $g(t)$. Equation (13) reveals an un-expected non-monotonic dependency of the optimal bandwidth in $g(t)$. The dependency, expressed by $\hat{h}_t \propto (\sum_{j=1}^V (c_{1j}/\sqrt{g(t)} + c_{2j}\sqrt{g(t)})^2)^{-1/5}$ is illustrated in Figure 6 (left) where we assume for simplicity that c_{1j}, c_{2j} do not change with j resulting in $(\hat{h}_t)^{-1} \propto (c_1/\sqrt{g(t)} + c_2\sqrt{g(t)})^{2/5}$. The key to understanding this relationship is the increased asymptotic bias due to the presence of the term $g'(t)/g(t)$ in Equation (11). Intuitively, the variations in $g(t)$ expressed by $g'(t)$ introduce a bias component which alters the otherwise monotonic role of the optimal bandwidth and bias-variance tradeoff. Since $g(t)$ is highly non-uniform (as illustrated in Figure 2), this dependency of $\hat{h}_t$ on $g(t)$ is likely to play a significant role.

We finally point out that different words w have different parameters $[\theta_t]_w$ and parameter derivatives $[\dot{\theta}_t]_w$ which indicates that it is unlikely that a single bandwidth will work best for all words. Frequent words are likely to benefit more from narrow kernel smoothing than rare words which almost never appear. As a result, a lower bandwidth should be used for frequent words while a high bandwidth should be used for rare words. A systematic investigation of these topics is beyond the scope of this paper.

4.2. Local Likelihood for Logistic Regression

Often, the primary goal behind modeling the drift is conditional modeling i.e. predicting the value of y given x. In this case, drift modeling should focus on estimating the conditional $p_t(y|x)$ since modeling the marginal $p_t(x)$ becomes irrelevant. In contrast to the modeling of the conditional by Bayes rule $p_t(y|x) \propto p_t(x|y)p_t(y)$ described in the previous section, we explore here direct modeling of $\{p_t(y|x) : t \in I\}$ using local likelihood for logistic regression.

By direct analogy to Equation (4) the conditional local likelihood estimator $p_t(y|x)$ is the maximizer of the locally weighted conditional loglikelihood

$$\ell_t(\eta|D) = \sum_{\tau\in I}^{n} K_h(t-\tau) \sum_{j=1}^{N_\tau} \log p(y_{\tau j}|x_{\tau j};\eta) \quad \eta \in \Theta.$$

As in the generative case, the kernel parameter h balances the degree of the kernel's locality and controls the bias-variance tradeoff.

Denoting by $f(x)$ the vector of relative frequencies in the document x, the logistic regression model

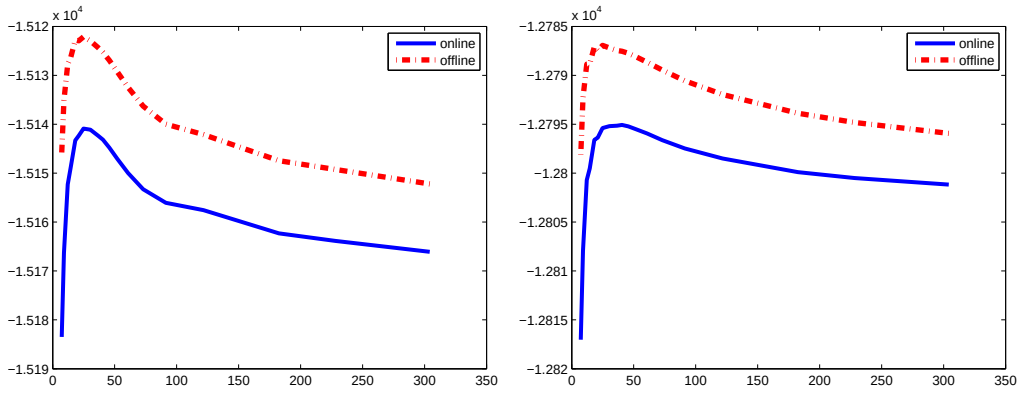


Figure 3. Per-word log-likelihood of held out test set as a function of the triangular kernel's bandwidth for the two largest RCV1 categories (CCAT (left) and GCAT (right)). In all four cases, the optimal bandwidth seems to be approximately 25 which indicates a support of 25 days for the online kernels and 50 days for the offline kernels.

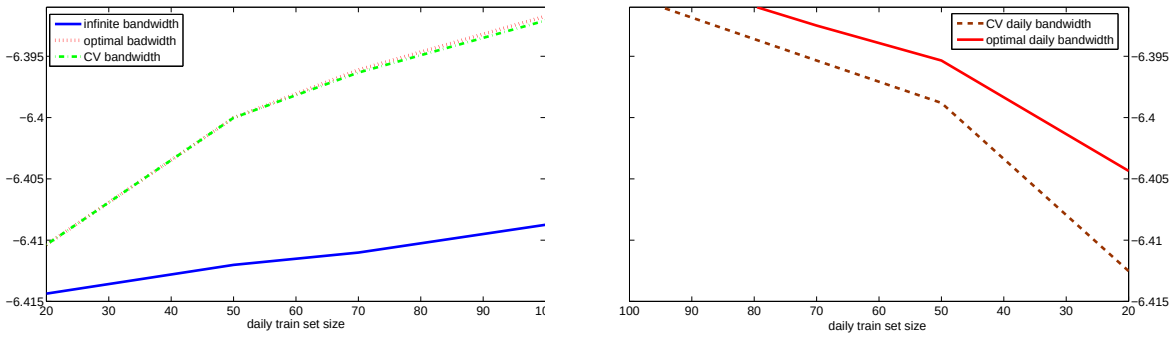


Figure 4. Per-word log-likelihood over held-out test set for various bandwidths as a function of the daily training set size. Left: The extreme global model corresponding to $h \to \infty$ performs worst. Selecting the bandwidth by cross validation results in an accurate estimate and test-set loglikelihood almost identical to that of the optimal slope. Right: Allowing the kernel scale to vary over time results in a higher modeling accuracy than using fixed bandwidth for all dates.

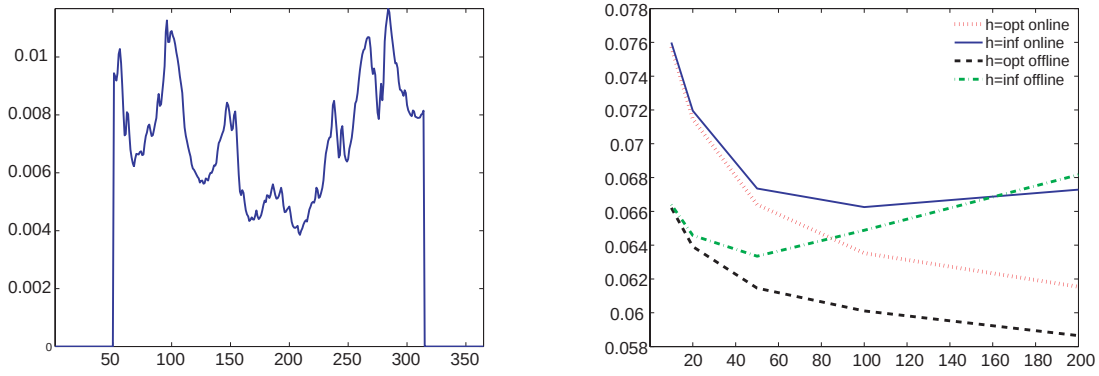


Figure 5. Left: Estimated gradient norm for the most popular category in RCV1 as a function of t. The derivatives were estimated using local smoothing. To avoid running into boundary effects we ignore the first and last 50 days. Right: Classification error rate over a held-out test set for the local logistic regression model as a function of the train set size.

$\log \frac{p(1|x;\theta_t)}{1-p(1|x;\theta_t)} = \langle \theta_t, f(x) \rangle$, $\theta \in \mathbb{R}^V$ leads to the following local conditional likelihood

$$\ell_t(\eta|D) = -\sum_{\tau \in I}^{n} K_h(t-\tau) \sum_{j=1}^{N_\tau} \log\left(1 + e^{-y_{\tau j}\langle x_{\tau j}, \eta \rangle}\right).$$

In contrast to the naive Bayes model in the previous section, the local likelihood does not have a close form maximizer. However, it can be shown that under mild conditions it is a concave problem exhibiting a single global maximum (for each t) (Loader, 1999). Most of the standard iterative algorithms for training logistic regression can be modified to account for the local weighting introduced by the smoothing kernel. Moreover, recently popularized regularization techniques such as the penalty $c\|\eta\|^q, q = 1, 2$ may be added to the local likelihood to obtain a local regularized version equivalent to maximum posterior estimation.

Figure 5 (right) displays classification error rate over a held-out test set for local logistic regression as a function of the train set size. The classification task was predicting the most popular class vs the second most popular class in RCV1. The plots in the figure contrast the performance of the online and offline tricube kernels with optimal and infinite bandwidths, using L_2 regularization. The local model achieved a relative reduction of error rate over the global model by about 8%. As expected, the online kernel generally achieve worse error rates than the offline kernels. In all the experiments mentioned above we averaged over multiple random samplings of the training set to remove sampling noise.

5. Discussion

A large number of textual datasets such as emails, webpages, news stories, etc. contain time stamped documents. For such datasets, considering a drifting rather than a stationary distribution is often appropriate. The local likelihood framework provides a natural extension for many standard likelihood models to the concept drift scenario. As the drift becomes more noticeable and the data size increases the potential benefits of local likelihood methods over their extreme global or local counterparts increase.

In this paper we illustrate the drift phenomenon and examine the properties of the local likelihood estimator including the asymptotic bias and variance tradeoff and optimal bandwidth. Experiments conducted on the RCV1 dataset demonstrate the validity of the local likelihood estimators in practice and contrast them with more standard non-local alternatives.

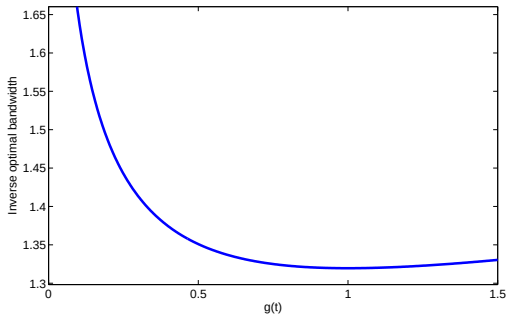


Figure 6. Inverse of the optimal bandwidth derived from Equation (13) as a function of $g(t)$: $(\hat{h}_t)^{-1} \propto (c_1/\sqrt{g(t)} + c_2\sqrt{g(t)})^{2/5}$ (we take $c_1 = c_2 = 1$). The graph show the non-monotonic dependency between $\hat{h}^{\text{opt}}$ and $g(t)$.

References

Chen, S., & Goodman, J. (1998). *An empirical study of smoothing techniques for language modelling* (Technical Report). Harvard university.

Forman, G. (2006). Tackling concept drift by temporal inductive transfer. *Proc. of the ACM SIGIR Conference.*

Helmbold, D. P., & Long, P. M. (1994). Tracking drifting concepts by minimizing disagreements. *Machine Learning, 14.*

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. *Proc. of the ACM SIGKDD Conference.*

Kuh, A., Petsche, T., & Rivest, R. L. (1990). Learning time-varying concepts. *Advances in Neural Information Processing Systems, 3.*

Lewis, D., Yang, Y., Rose, T., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research, 5.*

Loader, C. (1999). *Local regression and likelihood.* Springer.

Manning, C. D., & Schutze, H. (1999). *Foundations of statistical natural language processing.* MIT Press.

Mosteller, F., & Wallace, D. (1964). *Inference and disputed authorship: The federalist.* Addison Wesley.

Mulligan, G. (1999). *Removing the spam: Email processing and filtering.* Addison Wesley.

Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationship for sentiment categorization with respect to rating scales. *Proc. of the ACL conference.*

Sharan, U., & Neville, J. (2007). Exploiting time-varying relationships in statistical relational models. *Proc. of the joint WebKDD and SNA-KDD workshop.*

Tibshirani, R., & Hastie, T. (1987). Local likelihood estimation. *J. of the American Statistical Association, 82.*

Wand, M. P., & Jones, M. C. (1995). *Kernel smoothing.* Chapman and Hall/CRC.

A Worst-Case Comparison between Temporal Difference and Residual Gradient with Linear Function Approximation

Lihong Li LIHONG@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, 110 Frelinghuysen Road, Piscataway, NJ, USA 08854

Abstract

Residual gradient (RG) was proposed as an alternative to TD(0) for policy evaluation when function approximation is used, but there exists little formal analysis comparing them except in very limited cases. This paper employs techniques from online learning of linear functions and provides a worst-case (non-probabilistic) analysis to compare these two types of algorithms when linear function approximation is used. No statistical assumptions are made on the sequence of observations, so the analysis applies to non-Markovian and even adversarial domains as well. In particular, our results suggest that RG may result in smaller temporal differences, while TD(0) is more likely to yield smaller prediction errors. These phenomena can be observed even in two simple Markov chain examples that are non-adversarial.

1. Introduction

Reinforcement learning (RL) is a learning paradigm for optimal sequential decision making (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998) and has been successfully applied to a number of challenging problems. In the RL framework, the *agent* interacts with the *environment* in discrete timesteps by repeatedly observing its current state, taking an action, receiving a real-valued reward, and transitioning to a next state. A *policy* is a function that maps states to actions; semantically, it specifies what action to take given the current state. The goal of an agent is to optimize its policy in order to maximize the expected long-term return, namely, the discounted sum of rewards it receives by following the policy.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

An important step in this optimization process is *policy evaluation*—the problem of evaluating expected returns of a *fixed* policy. This problem is often the most challenging step in approximate policy-iteration algorithms (Bertsekas & Tsitsiklis, 1996; Lagoudakis & Parr, 2003). *Temporal difference* (TD) is a family of algorithms for policy evaluation (Sutton, 1988) and has received a lot of attention from the community. Unfortunately, it is observed (*e.g.*, Baird (1995)) that TD methods may diverge when they are combined with *function approximation*. An alternative algorithm known as *residual gradient* (RG) was proposed by Baird (1995) and enjoys guaranteed convergence to a local optimum. Since RG is similar to TD(0), a particular instance of the TD family, we will focus on RG, TD(0), and a variant of TD(0) in this paper.

Despite convergence issues, little is known that compares RG and TD(0). Building on previous work on *online learning of linear functions* (Cesa-Bianchi et al., 1996) and a similar analysis by Schapire and Warmuth (1996), we provide a worst-case (non-probabilistic) analysis of these algorithms and focus on two evaluation metrics: (i) total squared prediction error, and (ii) total squared temporal difference. The former measures *accuracy* of the predictions, while the latter measures *consistency* and is closely related to the *Bellman error* (Sutton & Barto, 1998).

Either metric may be preferred over the other in different situations. For instance, Lagoudakis and Parr (2003) argue that TD solutions tend to preserve the shape of the value function and is more suitable for approximate policy iteration, while there is evidence that minimizing squared Bellman errors is more robust in general (Munos, 2003). Our analysis suggests that TD can make more accurate predictions, while RG can result in smaller temporal differences. All terms will be made precise in the next section. Although our theory focuses on worst-case upper bounds, we also provide numerical evidence and expect the resulting insights to give useful guidance to RL practitioners in deciding which algorithm best suits their purposes.

2. Preliminaries

Fully observable environments in RL are often modelled as *Markov decision processes* (Puterman, 1994), which are equivalent to induced *Markov chains* when controlled by a fixed policy. Here, however, we consider a different model that is suitable for worst-case analysis, as introduced in the next subsection. This model makes no statistical assumption about the observations, and thus our results apply to much more general situations including *partially* observable or adversarial environments that subsume Markov chains.

Some notation is in order. We use bold-face, lower-case letters to denote real-valued column vectors such as $\mathbf{v}$. Their components are denoted by the corresponding letter with subscripts such as v_t. We use $\|\cdot\|$ to denote the Euclidean, or ℓ_2-norm: $\|\mathbf{v}\| = \sqrt{\mathbf{v}^\top \mathbf{v}}$ where $\mathbf{v}^\top$ is the transpose of $\mathbf{v}$. For a square matrix M, the set of eigenvalues of M, known as the *spectrum* of M, is denoted $\sigma(M)$. If M is symmetric, its eigenvalues must be real, and its largest eigenvalue is denoted $\rho(M)$.

2.1. The Sequential Online Learning Model

Our learning model is adopted from Schapire and Warmuth (1996) and is an extension of the online-learning model to sequential prediction problems. Let k be the dimension of input vectors. The agent maintains a weight vector of the same dimension and uses it to make predictions. In RL, input vectors are often feature vectors of states or state–action pairs, and are used to approximate value functions (Sutton & Barto, 1998). Learning proceeds in discrete timesteps and terminates after T steps. The agent starts with an initial input vector $\mathbf{x}_1 \in \mathbb{R}^k$ and an initial weight vector $\mathbf{w}_1 \in \mathbb{R}^k$. At timestep $t \in \{1, 2, 3, \cdots, T\}$:

- The agent makes a prediction $\hat{y}_t = \mathbf{w}_t^\top \mathbf{x}_t \in \mathbb{R}$, where $\mathbf{w}_t$ is the weight vector at time t. Throughout the paper, assume $\|\mathbf{x}_t\| \le X$ for some known constant $X > 0$.
- The agent then observes an *immediate reward* $r_t \in \mathbb{R}$ and the next input vector $\mathbf{x}_{t+1}$. Based on this information, it updates its weight vector whose new value is denoted $\mathbf{w}_{t+1}$. The change in weight is $\Delta\mathbf{w}_t = \mathbf{w}_{t+1} - \mathbf{w}_t$.

By convention, $r_t = 0$ and $\mathbf{x}_t = \mathbf{0}$ for $t > T$. Define the *return* at time t by $y_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$, where $\gamma \in [0, 1)$ is the *discount factor*. Since $\gamma < 1$, it effectively diminishes future rewards exponentially fast. A quick observation is that $y_t = r_t + \gamma y_{t+1}$, which is analogous to the Bellman equation for Markov chains (Sutton & Barto, 1998). The agent attempts to mimic y_t by its prediction $\hat{y}_t$, and the *prediction error* is $e_t = y_t - \hat{y}_t$. Our first evaluation metric is the *total squared prediction error*: $\ell_{\mathcal{P}} = \sum_{t=1}^{T} e_t^2 = \|\mathbf{e}\|^2$.

Another useful metric in RL is the *temporal differences* (also known as *TD errors*), which measures how consistent the predictions are. In particular, the temporal difference at time t is $d_t = r_t + \gamma \mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t$, and the *total squared temporal difference* is $\ell_{\mathcal{TD}} = \sum_{t=1}^{T} d_t^2 = \|\mathbf{d}\|^2$.

2.2. Previous Work

Previous convergence results of TD and RG often rely heavily on certain stochastic assumptions of the environment such as the assumption that the sequence of observations, $[(\mathbf{x}_t, r_t)]_{t\in\mathbb{N}}$, are generated by an irreducible and aperiodic Markov chain. Tsitsiklis and Van Roy (1997) first proved convergence of TD with linear function approximation, while they also pointed out the potential divergence risk when nonlinear approximation is used.

To resolve the instability issue of TD(0), Baird (1995) proposed the RG algorithm, but also noted that RG may converge more slowly than TD(0) in some problems. Such an observation was later proved by Schoknecht and Merke (2003), who used spectral analysis to compare the asymptotic convergence rates of the two algorithms. Although their results are interesting, they only apply to quite limited cases where, for example, a certain matrix associated with TD updates has real eigenvalues only (which does not hold in general). More importantly, they study *synchronous* updates while TD and RG are often applied *asynchronously* in practice. Furthermore, their results assume that the value function is represented by a lookup table, but the initial motivation of studying RG was to develop a provably convergent algorithm when function approximation is used.

Schapire and Warmuth (1996) were also concerned with similar worst-case behavior of TD-like algorithms within the model described in Subsection 2.1. They defined a new class of algorithms called TD$^*(\lambda)$, which is very similar to the TD(λ) algorithms of Sutton (1988). They developed worst-case bounds for the total squared prediction error of TD$^*(\lambda)$, but not the total squared temporal difference.

2.3. Algorithms

The algorithms we consider all update the weight vector incrementally and differ only in the update rules. TD(0) uses the following rule:

$$\Delta\mathbf{w}_t = \eta d_t \mathbf{x}_t, \tag{1}$$

where $\eta \in (0,1)$ is the *step-size* parameter controlling aggressiveness of the update. Although TD(0) is widely used in practice, analysis turns out to be easier with a close relative of it, TD*(0). This algorithm differs from TD(0) in that it adapts the step-size based on the input vectors (Schapire and Warmuth (1996) defined TD*(0) in a different, but equivalent, form):

$$\Delta \mathbf{w}_t = \frac{\eta d_t \mathbf{x}_t}{1 - \gamma\eta \mathbf{x}_t^\top \mathbf{x}_{t+1}}. \tag{2}$$

Due to space limitation, we only provide results for TD*(0), but similar results hold for TD(0). It is expected, and also supported by the numerical evidence in Section 4, that TD(0) and TD*(0) have similar behavior and performance in practice. For this reason, we refer to both algorithms as TD in the rest of the paper if there is no risk of confusion. In contrast, RG uses the following update rule:

$$\Delta \mathbf{w}_t = \eta d_t \left(\mathbf{x}_t - \gamma \mathbf{x}_{t+1}\right). \tag{3}$$

3. Main Results

This section contains the main theoretical results. We will first describe how to evaluate an algorithm in the worst-case scenario. For completeness, we also summarize the squared prediction error bounds for TD*(0) due to Schapire and Warmuth (1996). Then, we analyze total squared temporal difference bounds and RG.

Our analysis makes a few uses of matrix theory (see, *e.g.*, Horn and Johnson (1986)), and several technical lemmas are found in the appendix. Two basic facts about $\rho(M)$ will be used repeatedly: (i) if M is negative-definite, then $\rho(M) < 0$; and (ii) the Rayleigh-Ritz theorem (Horn & Johnson, 1986, Theorem 4.2.2) states that $\rho(M) = \max_{\mathbf{v}\neq\mathbf{0}} \frac{\mathbf{v}^\top M \mathbf{v}}{\mathbf{v}^\top \mathbf{v}}$.

3.1. Evaluation Criterion

Analogous to other online-learning analysis, we treat $\ell_{\mathcal{P}}$ and $\ell_{\mathcal{TD}}$ as *total losses*, and compare the total loss of an algorithm to that of an arbitrary weight vector, $\mathbf{u}$. We wish to prove that this difference is small for *all* $\mathbf{u}$, including the *optimal* (in any well-defined sense) but unknown vector $\mathbf{u}^*$.

The prediction using vector $\mathbf{u}$ at time t is $y_t^{\mathbf{u}} = \mathbf{u}^\top \mathbf{x}_t$. Accordingly, the prediction error and temporal difference at time t are $e_t^{\mathbf{u}} = y_t - y_t^{\mathbf{u}}$ and $d_t^{\mathbf{u}} = r_t + \gamma \mathbf{u}^\top \mathbf{x}_{t+1} - \mathbf{u}^\top \mathbf{x}_t$, respectively. The total squared prediction error and total squared temporal difference of $\mathbf{u}$ are $\ell_{\mathcal{P}}^{\mathbf{u}} = \|\mathbf{e}^{\mathbf{u}}\|^2 = \sum_{t=1}^T \left(y_t - \mathbf{u}^\top \mathbf{x}_t\right)^2$ and $\ell_{\mathcal{TD}}^{\mathbf{u}} = \|\mathbf{d}^{\mathbf{u}}\|^2 = \sum_{t=1}^T \left(r_t + \gamma \mathbf{u}^\top \mathbf{x}_{t+1} - \mathbf{u}^\top \mathbf{x}_t\right)^2$, respectively.

3.2. Squared Prediction Errors of TD*(0)

Using step-size $\eta = \frac{1}{X^2+1}$, Schapire and Warmuth (1996) showed a worst-case upper bound:

$$\ell_{\mathcal{P}} \leq \frac{\left(1+X^2\right)\left(\ell_{\mathcal{P}}^{\mathbf{u}} + \|\mathbf{w}_1 - \mathbf{u}\|_2^2\right)}{1-\gamma^2}.$$

Furthermore, if E and W are known beforehand such that $\ell_{\mathcal{P}}^{\mathbf{u}} \leq E$ and $\|\mathbf{w}_1 - \mathbf{u}\| \leq W$, then the step-size η can be optimized by $\eta = \frac{W}{X\sqrt{E}+X^2 W}$ to yield an asymptotically better bound:

$$\ell_{\mathcal{P}} \leq \frac{\ell_{\mathcal{P}}^{\mathbf{u}} + 2WX\sqrt{E} + X^2W^2}{1-\gamma^2}. \tag{4}$$

3.3. Squared Temporal Differences of TD*(0)

We will extend the analysis of Schapire and Warmuth (1996) to the new loss function $\ell_{\mathcal{TD}}$ by examining how the potential function, $\|\mathbf{w}_t - \mathbf{u}\|^2$, evolves when a single update is made at time t. It can be shown (Schapire & Warmuth, 1996, Eqn 8) that $-\|\mathbf{w}_1 - \mathbf{u}\|^2 \leq \eta^2 X^2 \mathbf{e}^\top D^\top D \mathbf{e} + 2\eta \mathbf{e}^\top D^\top (\mathbf{e}^{\mathbf{u}} - \mathbf{e})$, where

$$D = \begin{pmatrix} 1 & -\gamma & 0 & \cdots & 0 & 0 \\ 0 & 1 & -\gamma & \cdots & 0 & 0 \\ & & \ddots & & & \\ 0 & 0 & \cdots & 1 & -\gamma & 0 \\ 0 & 0 & \cdots & 0 & 1 & -\gamma \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}. \tag{5}$$

Define $\mathbf{f} = D\mathbf{e}$. According to Lemma A.1(1), $\mathbf{d}^{\mathbf{u}} = D\mathbf{e}^{\mathbf{u}}$, and hence the inequality above is rewritten as:

$$-\|\mathbf{w}_1 - \mathbf{u}\|^2 \leq \eta^2 X^2 \mathbf{f}^\top \mathbf{f} - 2\eta \mathbf{f}^\top D^{-1} \mathbf{f} + 2\eta \mathbf{f}^\top D^{-1} \mathbf{d}^{\mathbf{u}}.$$

Using the fact that $2\mathbf{p}^\top \mathbf{q} \leq \|\mathbf{p}\|^2 + \|\mathbf{q}\|^2$ for $\mathbf{p} = \frac{\eta}{\sqrt{b}} D^{-\top}\mathbf{f}$, $\mathbf{q} = \sqrt{b}\mathbf{d}^{\mathbf{u}}$, and arbitrary $b > 0$, the inequality becomes $-\|\mathbf{w}_1 - \mathbf{u}\|^2 \leq \mathbf{f}^\top M_1 \mathbf{f} + b\ell_{\mathcal{TD}}^{\mathbf{u}}$, where

$$M_1 = \eta^2 X^2 I + \frac{\eta^2}{b} D^{-1} D^{-\top} - \eta(D^{-1} + D^{-\top}) \tag{6}$$

is a symmetric matrix. Since $\rho(M_1)$ is the largest eigenvalue of M_1, we have $\mathbf{f}^\top M_1 \mathbf{f} \leq \rho(M_1)\|\mathbf{f}\|^2$, and hence, $-\|\mathbf{w}_1 - \mathbf{u}\|^2 \leq \|\mathbf{f}\|^2 \rho(M_1) + b\ell_{\mathcal{TD}}^{\mathbf{u}}$. Combining this with Lemma A.4, we have that $\|\mathbf{f}\|^2$ is at most

$$(1+\gamma)^2 \left(X^2 + \frac{1}{b(1-\gamma)^2}\right)\left(b\ell_{\mathcal{TD}}^{\mathbf{u}} + \|\mathbf{w}_1 - \mathbf{b}\|^2\right),$$

when the step-size is

$$\eta = \frac{1}{(1+\gamma)\left(X^2 + \frac{1}{b(1-\gamma)^2}\right)}. \tag{7}$$

Due to Lemma A.1 (2), we have

$$\begin{aligned} d_t^2 &= \left(1-\gamma\eta\mathbf{x}_t^\top\mathbf{x}_{t+1}\right)^2 f_t^2 \\ &\le \left(1+\gamma\eta X^2\right)^2 f_t^2 \;\le\; \frac{(1+2\gamma)^2}{(1+\gamma)^2} f_t^2. \end{aligned}$$

Therefore, $\ell_{\mathcal{TD}}$ is at most

$$(1+2\gamma)^2\left(X^2+\frac{1}{b(1-\gamma)^2}\right)\left(b\ell_{\mathcal{TD}}^{\mathbf{u}}+\|\mathbf{w}_1-\mathbf{u}\|^2\right).$$

Using $b=1$, we have thus proved the first main result.

Theorem 3.1. *Let η be given by Eqn 7 using $b=1$, then the following holds for TD*(0):*

$$\ell_{\mathcal{TD}} \le (1+2\gamma)^2\left(X^2+\frac{1}{(1-\gamma)^2}\right)\left(\ell_{\mathcal{TD}}^{\mathbf{u}}+\|\mathbf{w}_1-\mathbf{u}\|^2\right).$$

Theorem 3.2. *If E and W are known beforehand such that $\ell_{\mathcal{TD}}^{\mathbf{u}} \le E$ and $\|\mathbf{w}_1-\mathbf{u}\| \le W$, then η can be optimized in TD*(0) so that*

$$\ell_{\mathcal{TD}} \le (1+2\gamma)^2 \left(\frac{\ell_{\mathcal{TD}}^{\mathbf{u}}}{(1-\gamma)^2}+\frac{2XW\sqrt{E}}{1-\gamma}+X^2\|\mathbf{w}_1-\mathbf{u}\|^2\right). \quad (8)$$

Proof. Previous analysis for Theorem 3.1 yields

$$\begin{aligned} \ell_{\mathcal{TD}} &\le (1+2\gamma)^2\left(\left(bX^2\ell_{\mathcal{TD}}^{\mathbf{u}}+\frac{\|\mathbf{w}_1-\mathbf{u}\|^2}{b(1-\gamma)^2}\right)+\left(\frac{\ell_{\mathcal{TD}}^{\mathbf{u}}}{(1-\gamma)^2}+X^2\|\mathbf{w}_1-\mathbf{u}\|^2\right)\right) \\ &\le (1+2\gamma)^2\left(\left(bX^2E+\frac{W^2}{b(1-\gamma)^2}\right)+\left(\frac{\ell_{\mathcal{TD}}^{\mathbf{u}}}{(1-\gamma)^2}+X^2\|\mathbf{w}_1-\mathbf{u}\|^2\right)\right) \end{aligned}$$

for any $b>0$. We may simply choose $b=\frac{W}{X(1-\gamma)\sqrt{E}}$, and the step-size in Eqn 7 becomes

$$\eta=\frac{1}{(1+\gamma)\left(X^2+\frac{X\sqrt{E}}{W(1-\gamma)}\right)}. \quad \square$$

3.4. Squared Prediction Errors of RG

By the update rule in Eqn 3 and simple algebra,

$$\begin{aligned} \Delta\mathbf{w}_t^\top(\mathbf{w}_t-\mathbf{u}) &= \eta d_t(\mathbf{x}_t-\gamma\mathbf{x}_{t+1})^\top(\mathbf{w}_t-\mathbf{u}) \\ &= \eta d_t\left(\left(\mathbf{w}_t^\top\mathbf{x}_t-\gamma\mathbf{w}_t^\top\mathbf{x}_{t+1}-r_t\right)-\left(\mathbf{u}^\top\mathbf{x}_t-\gamma\mathbf{u}^\top\mathbf{x}_{t+1}-r_t\right)\right) \\ &= \eta d_t(d_t^{\mathbf{u}}-d_t), \\ \|\Delta\mathbf{w}_t\|^2 &= \eta^2 d_t^2\|\mathbf{x}_t-\gamma\mathbf{x}_{t+1}\|_2^2 \\ &\le \eta^2 d_t^2 X^2(1+\gamma)^2. \end{aligned}$$

Similar to the previous section, we use the potential function $\|\mathbf{w}_t-\mathbf{u}\|^2$ to measure progress of learning:

$$\begin{aligned} -\|\mathbf{w}_1-\mathbf{u}\|^2 &\le \sum_{t=1}^T\left(\|\mathbf{w}_{t+1}-\mathbf{u}\|^2-\|\mathbf{w}_t-\mathbf{u}\|^2\right) \\ &= \sum_{t=1}^T\left(2\Delta\mathbf{w}_t^\top(\mathbf{w}_t-\mathbf{u})+\Delta\mathbf{w}_t^\top\Delta\mathbf{w}_t\right) \\ &\le \sum_{t=1}^T\left(2\eta d_t(d_t^{\mathbf{u}}-d_t)+\eta^2 d_t^2X^2(1+\gamma)^2\right) \\ &= 2\eta\mathbf{d}^\top\mathbf{d}^{\mathbf{u}}-2\eta\mathbf{d}^\top\mathbf{d}+\eta^2X^2(1+\gamma)^2\mathbf{d}^\top\mathbf{d}. \end{aligned}$$

According to Lemma A.1 (1) and using the fact that $2\mathbf{p}^\top\mathbf{q}\le\|\mathbf{p}\|^2+\|\mathbf{q}\|^2$ for $\mathbf{p}=\frac{\eta}{\sqrt{b}}D^\top\mathbf{d}$, $\mathbf{q}=\sqrt{b}\mathbf{e}^{\mathbf{u}}$, and arbitrary $b>0$, the inequality above is written as:

$$-\|\mathbf{w}_1-\mathbf{u}\|^2 \le b\|\mathbf{e}^{\mathbf{u}}\|^2+\frac{\eta^2}{b}\mathbf{d}^\top DD^\top\mathbf{d}+\left(\eta^2X^2(1+\gamma)^2-2\eta\right)\|\mathbf{d}\|^2$$

Due to Lemma A.1 (3), $\mathbf{d}=\Sigma D\mathbf{e}$, where

$$\Sigma = \operatorname{diag}\left(\frac{1}{1+\gamma\eta(\mathbf{x}_1-\gamma\mathbf{x}_2)^\top\mathbf{x}_2}, \frac{1}{1+\gamma\eta(\mathbf{x}_2-\gamma\mathbf{x}_3)^\top\mathbf{x}_3},\cdots, \frac{1}{1+\gamma\eta(\mathbf{x}_{T-1}-\gamma\mathbf{x}_T)^\top\mathbf{x}_T},1\right). \quad (9)$$

Then, the inequality above becomes:

$$-\|\mathbf{w}_1-\mathbf{u}\|^2 \le b\|\mathbf{e}^{\mathbf{u}}\|^2+\mathbf{e}^\top M_2\mathbf{e},$$

where

$$M_2=D^\top\Sigma\left(\frac{\eta^2}{b}DD^\top+\left(\eta^2X^2(1+\gamma)^2-2\eta\right)I\right)\Sigma D. \quad (10)$$

Since $\mathbf{e}^\top M_2\mathbf{e}\le\rho(M_2)\|\mathbf{e}\|^2$, Lemma A.5 implies the following theorems when the step-size is

$$\eta=\frac{1}{(1+\gamma)^2\left(X^2+\frac{1}{b}\right)}. \quad (11)$$

Theorem 3.3. *Let η be given by Eqn 11 using $b=1$, then the following holds for RG:*

$$\ell_{\mathcal{P}} \le \frac{(1+2\gamma)^2\left(X^2+1\right)}{(1-\gamma)^2}\left(\ell_{\mathcal{P}}^{\mathbf{u}}+\|\mathbf{w}_1-\mathbf{u}\|^2\right).$$

Theorem 3.4. *If E and W are known beforehand such that $\ell_{\mathcal{P}}^{\mathbf{u}} \le E$ and $\|\mathbf{w}_1-\mathbf{u}\| \le W$, then η can be optimized in RG so that*

$$\ell_{\mathcal{P}} \le \frac{(1+2\gamma)^2}{(1-\gamma)^2}\left(\ell_{\mathcal{P}}^{\mathbf{u}}+2XW\sqrt{E}+X^2\|\mathbf{w}_1-\mathbf{u}\|^2\right). \quad (12)$$

Proof. Previous analysis in this subsection yields

$$\begin{aligned}\ell_{\mathcal{P}} &\leq \frac{(1+2\gamma)^2}{(1-\gamma)^2}\left(\left(\ell^{\mathbf{u}}_{\mathcal{P}} + X^2\left\|\mathbf{w}_1-\mathbf{u}\right\|^2\right) + \left(X^2 b\ell^{\mathbf{u}}_{\mathcal{P}} + \frac{\left\|\mathbf{w}_1-\mathbf{u}\right\|^2}{b}\right)\right)\\ &\leq \frac{(1+2\gamma)^2}{(1-\gamma)^2}\left(\left(\ell^{\mathbf{u}}_{\mathcal{P}} + X^2\left\|\mathbf{w}_1-\mathbf{u}\right\|^2\right) + \left(X^2 bE + \frac{W^2}{b}\right)\right).\end{aligned}$$

We simply choose $b = \frac{W}{X\sqrt{E}}$ and accordingly the step-size in Eqn 11 becomes

$$\eta = \frac{1}{(1+\gamma)^2\left(X^2 + \frac{X\sqrt{E}}{W}\right)}. \quad \square$$

3.5. Squared Temporal Differences of RG

It is most convenient to turn this problem into one of analyzing the total squared prediction error in the original online-learning-of-linear-function framework (Cesa-Bianchi et al., 1996). In particular, define $\mathbf{z}_t = \mathbf{x}_t - \gamma\mathbf{x}_{t+1}$ and thus $\|\mathbf{z}_t\| \leq (1+\gamma)X$. Now, RG can be viewed as a gradient descent algorithm operating over the sequence of data $[(\mathbf{z}_t, r_t)]_{t\in\{1,2,\cdots,T\}}$. Due to Theorem IV.1 of Cesa-Bianchi et al. (1996), we immediately have

$$\ell_{\mathcal{TD}} \leq 2.25\left(\ell^{\mathbf{u}}_{\mathcal{TD}} + X^2(1+\gamma)^2\left\|\mathbf{u}\right\|^2\right),$$

for any $\mathbf{u}$ when the step-size is $\eta = \frac{2}{3X^2(1+\gamma)^2}$. If E and W are known beforehand so that $\ell^{\mathbf{u}}_{\mathcal{TD}} \leq E$ and $\|\mathbf{u}\| \leq W$, then η can be optimized (Theorem IV.3 of Cesa-Bianchi et al. (1996)) by $\eta = \frac{W}{X(1+\gamma)(WX(1+\gamma)+\sqrt{E})}$ to obtain the following improved bound:

$$\ell_{\mathcal{TD}} \leq \ell^{\mathbf{u}}_{\mathcal{TD}} + 2WX(1+\gamma)\sqrt{E} + (1+\gamma)^2W^2X^2. \quad (13)$$

3.6. Discussions

Based on Eqns 4, 8, 12, and 13, Table 1 summarizes the asymptotic upper bounds (when $T \to \infty$) assuming E and W are known beforehand to optimize η.[1] Although our bounds are all upper bounds, results in the table *suggest* that, in worst cases, TD*(0) (and also TD(0)) tend to make smaller prediction errors, while RG tends to make smaller temporal differences. The gaps between corresponding bounds increase as $\gamma \to 1$. On the other extreme where $\gamma = 0$, all these asymptotic bounds coincide, which is not surprising as TD(0), TD*(0), and RG are all identical when $\gamma = 0$.

[1] Strictly speaking, the validity of these asymptotic results relies on the assumptions that (i) $\sqrt{E} = o(\ell^{\mathbf{u}}_{\mathcal{P}})$, and (ii) W and X remain constant as $T \to \infty$. Both assumptions are reasonable in practice.

Table 1. Asymptotic upper bounds for total squared prediction error and total squared temporal difference of TD*(0) and RG.

	$\ell_{\mathcal{P}}/\ell^{\mathbf{u}}_{\mathcal{P}}$	$\ell_{\mathcal{TD}}/\ell^{\mathbf{u}}_{\mathcal{TD}}$
TD*(0)	$\frac{1}{1-\gamma^2} + o(1)$	$\frac{(1+2\gamma)^2}{(1-\gamma)^2} + o(1)$
RG	$\frac{(1+2\gamma)^2}{(1-\gamma)^2} + o(1)$	$1 + o(1)$

Since it is unknown whether the leading constants in Table 1 are optimal, the next section will provide numerical evidence to support our claims about the relative strengths of these algorithms.

It is worth mentioning that in sequential prediction or decision problems, the factor $\frac{1}{1-\gamma}$ often plays a role similar to the *decision horizon* (Puterman, 1994). Therefore, in some sense, our bounds also characterize how prediction errors and temporal differences may scale with decision horizon, in the worst-case sense.

When $\ell_{\mathcal{P}}$ or $\ell_{\mathcal{TD}}$ are relatively small, the asymptotic bounds in Table 1 are less useful as the $\|\mathbf{w}_1-\mathbf{u}\|^2$ in the bounds dominate $\ell_{\mathcal{P}}$ or $\ell_{\mathcal{TD}}$. However, we still get similar qualitative results by comparing the constant factors of the term $\|\mathbf{w}_1-\mathbf{u}\|^2$ in the bounds.

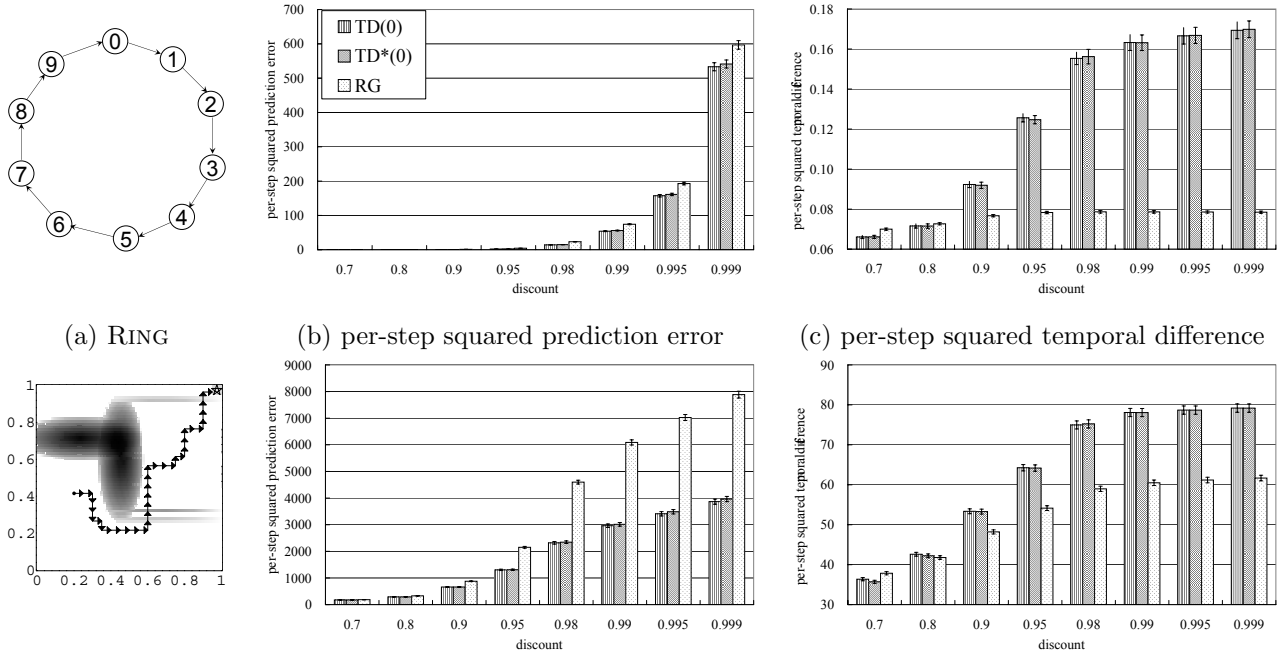
Since our setting is quite different from that of Schoknecht and Merke (2003), our results are not comparable to theirs.

4. Experiments

This section presents empirical evidence in two Markov chains that supports our claims in Section 3.6.

The first is the RING Markov chain (Figure 1 (a)), a variant of the HALL problem introduced by Baird (1995) in which RG was observed to converge to the optimal weights more slowly than TD(0). The state space is a ring consisting of 10 states numbered from 0 through 9. Each state is associated with a randomly selected feature vector of dimension $k = 5$: $\mathbf{x}^{(0)}, \cdots, \mathbf{x}^{(9)} \in \mathbb{R}^k$. Transitions are deterministic and are indicated by arrows. The reward in every state is stochastic and is distributed uniformly in $[-0.1, 0.1]$. As in HALL , the value of every state is exactly 0.

The second problem is a benchmark problem known as PUDDLEWORLD (Boyan & Moore, 1995). The state space is a unit square (Figure 1 (d)), and a start state of an episode is randomly selected in $[0, 0.2] \times [0, 0.2]$.



(a) RING (b) per-step squared prediction error (c) per-step squared temporal difference

(d) PUDDLEWORLD (e) per-step squared prediction error (f) per-step squared temporal difference

Figure 1. Two Markov chains we used: (a) RING and (d) PUDDLEWORLD (Boyan & Moore, 1995). All results are averaged over 500 runs, with 99% confidence intervals plotted. RING and PUDDLEWORLD results are in (b,c) and (e,f), respectively.

The agent adopts a fixed policy that goes north or east with probability 0.5 each. Every episode takes about 40 steps to terminate. The reward is -1 unless the agent steps into the puddles and receives penalty for that; the smallest possible reward is -41. We used 16 RBF features of width 0.3, whose centers were evenly distributed in the state space. We also tried a degree-two polynomial feature: for a state $\mathbf{s} = (s_1, s_2)^\top$, the feature vector had six components: $\mathbf{x_s} = \left(1, s_1, s_2, s_1 s_2, s_1^2, s_2^2\right)^\top$. Since the results are similar to those for RBF features, they are not included.

We ran three algorithms in the experiments: TD(0), TD*(0), and RG. For a fair comparison, all algorithms started with the all-one weight vector and were given the same sequence of $(\mathbf{x}_t, r_t)$ for learning. The procedure was repeated 500 times. For RING , each run used a different realization of feature $\mathbf{x}^{(s)}$ and $T = 500$; for PUDDLEWORLD , each run consisted of 50 episodes (yielding slightly less than 2000 steps in total). A wide range of step-sizes were tried, and the best choices for each discount-factor–algorithm combination were used to evaluate $\ell_{\mathcal{P}}$ and $\ell_{\mathcal{TD}}$, respectively. Figure 1 (b,c,e,f) gives the average per-step squared prediction errors and squared temporal differences for these two problems, with 99% confidence intervals plotted.

These results are consistent with our analysis: TD(0) and TD*(0) tended to make more accurate predictions, while RG did a better job at minimizing temporal differences; the differences between these algorithms were even larger as the discount factor γ approached 1.[2] Finally, as a side effect, it is verified that TD(0) and TD*(0) had essentially identical performance, although their best learning rates might differ.

5. Conclusion

We have carried out a worst-case analysis to compare two policy-evaluation algorithms, TD and RG, when linear function approximation is used. Together with previously known results due to Schapire and Warmuth (1996) and Cesa-Bianchi et al. (1996), our results suggest that, although the TD algorithms may make more accurate predictions, RG may be a better choice when small temporal differences are desired. This claim is supported by empirical evidence in two simple Markov chains. Although the analysis is purely mathematical, we expect the implications to deepen the understanding of these two types of algorithms and can provide useful insights to RL practitioners.

[2]This effect was less obvious when γ got too close to 1. This was because the trajectories in our experiments were not long enough for such γ to have full impacts.

There has been relatively little attention to this sort of online-learning analysis within the RL community. Our analysis shows that this kind of analysis may be helpful and provide useful insights. A few directions are worth pursuing. First, we have focused on worst-case upper bounds, but it remains open whether matching lower bounds can be found. More extensive empirical studies are also necessary to see if such worst-case behavior can be observed in realistic problems. Second, we wish to generalize the analysis of total squared temporal difference from TD(0) and TD*(0) to TD(λ) and TD*(λ), respectively. Finally, we would like to mention that, in their original forms, both TD and RG use *additive* updates. Another class of updates known as *multiplicative* updates (Kivinen & Warmuth, 1997) has been useful when the number of features (*i.e.*, the k in Subsection 2.1) is large but only a few of them are relevant for making predictions. Such learning rules have potential uses in RL (Precup & Sutton, 1997), but it remains open whether these algorithms converge or whether worst-case error bounds similar to the ones given in this paper can be obtained.

A. Lemmas and Proofs

Lemma A.1. *This lemma collects a few basic facts useful in our analysis (D is given in Eqn 5):*

1. *In all three algorithms,* $\mathbf{d^u} = D\mathbf{e^u}$.
2. *In TD*(0),* $d_t = (1 - \gamma\eta\mathbf{x}_t^\top\mathbf{x}_{t+1})(e_t - \gamma e_{t+1})$.
3. *In RG,* $d_t = \frac{e_t - \gamma e_{t+1}}{1+\gamma\eta(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})^\top\mathbf{x}_{t+1}}$.

Proof. 1. Since $y_t = r_t + \gamma y_{t+1}$, we have

$$\begin{aligned} d_t^{\mathbf{u}} &= r_t + \gamma\mathbf{u}^\top\mathbf{x}_{t+1} - \mathbf{u}^\top\mathbf{x}_t \\ &= \left(y_t - \mathbf{u}^\top\mathbf{x}_t\right) - \left(y_t - r_t - \gamma\mathbf{u}^\top\mathbf{x}_{t+1}\right) \\ &= \left(y_t - \mathbf{u}^\top\mathbf{x}_t\right) - \gamma\left(y_{t+1} - \mathbf{u}^\top\mathbf{x}_{t+1}\right) \\ &= e_t^{\mathbf{u}} - \gamma e_{t+1}^{\mathbf{u}}. \end{aligned}$$

In matrix form, this is $\mathbf{d^u} = D\mathbf{e^u}$.

2. Since $\mathbf{w}_t = \mathbf{w}_{t+1} - \Delta\mathbf{w}_t$ and $y_t = r_t + \gamma y_{t+1}$,

$$\begin{aligned} d_t &= r_t + \gamma\mathbf{w}_t^\top\mathbf{x}_{t+1} - \mathbf{w}_t^\top\mathbf{x}_t \\ &= r_t + \gamma(\mathbf{w}_{t+1} - \Delta\mathbf{w}_t)^\top\mathbf{x}_{t+1} - \mathbf{w}_t^\top\mathbf{x}_t + \\ &\quad (y_t - r_t - \gamma y_{t+1}) \\ &= (y_t - \mathbf{w}_t^\top\mathbf{x}_t) - \gamma(y_{t+1} - \mathbf{w}_{t+1}^\top\mathbf{x}_{t+1}) - \\ &\quad \gamma\Delta\mathbf{w}_t^\top\mathbf{x}_{t+1} \\ &= e_t - \gamma e_{t+1} - \frac{\gamma\eta d_t\mathbf{x}_t^\top\mathbf{x}_{t+1}}{1 - \gamma\eta\mathbf{x}_t^\top\mathbf{x}_{t+1}}. \end{aligned}$$

Reorganizing terms will complete the proof.

3. Similar to the proof for part (2) except that $\Delta\mathbf{w}_t$ is computed by Eqn 3. □

Two technical lemmas are useful to prove Lemma A.4. It should be noted that the bounds they give are tight.

Lemma A.2. *For D given in Eqn 5, let A be $D^\top D$ or $DD^\top$, and B be $D^{-1}D^{-\top}$ or $D^{-\top}D^{-1}$. Then, $\sigma(A) \subseteq \left[(1-\gamma)^2, (1+\gamma)^2\right]$ and $\sigma(B) \subseteq \left[(1+\gamma)^{-2}, (1-\gamma)^{-2}\right]$.*

Proof. It can be verified that $D^\top D$ equals

$$\begin{pmatrix} 1 & -\gamma & 0 & \cdots & 0 \\ -\gamma & 1+\gamma^2 & -\gamma & \cdots & 0 \\ & & \ddots & & \\ 0 & 0 & \cdots & 1+\gamma^2 & -\gamma \\ 0 & 0 & \cdots & -\gamma & 1+\gamma^2 \end{pmatrix}.$$

Since $D^\top D$ is symmetric, $\sigma\left(D^\top D\right) \subset \mathbb{R}$. It follows from Geršgorin's theorem (Horn & Johnson, 1986, Theorem 6.1.1) that $\sigma\left(D^\top D\right) \subseteq \left[(1-\gamma)^2, (1+\gamma)^2\right]$. The same holds for $\sigma\left(DD^\top\right)$. The second part follows immediately by observing that $D^{-1}D^{-\top} = \left(D^\top D\right)^{-1}$ and $D^{-\top}D^{-1} = \left(DD^\top\right)^{-1}$. □

Lemma A.3. *Let D be given by Eqn 5, then*

$$\sigma(D^{-1} + D^{-\top}) \subseteq \left[\frac{2}{1+\gamma}, \frac{2}{1-\gamma}\right].$$

Proof. It can be verified that $D^{-1} + D^{-\top}$ equals

$$G = \begin{pmatrix} 2 & \gamma & \gamma^2 & \cdots & \gamma^{T-2} & \gamma^{T-1} \\ \gamma & 2 & \gamma & \cdots & \gamma^{T-3} & \gamma^{T-2} \\ & & \ddots & & & \\ \gamma^{T-3} & \gamma^{T-4} & \cdots & 2 & \gamma & \gamma^2 \\ \gamma^{T-2} & \gamma^{T-3} & \cdots & \gamma & 2 & \gamma \\ \gamma^{T-1} & \gamma^{T-2} & \cdots & \gamma^2 & \gamma & 2 \end{pmatrix},$$

and that $(G-I)^{-1}$ equals

$$\begin{pmatrix} \frac{1}{1-\gamma^2} & \frac{-\gamma}{1-\gamma^2} & 0 & \cdots & 0 & 0 \\ \frac{-\gamma}{1-\gamma^2} & \frac{1+\gamma^2}{1-\gamma^2} & \frac{-\gamma}{1-\gamma^2} & \cdots & 0 & 0 \\ & & \ddots & & & \\ 0 & 0 & \cdots & \frac{1+\gamma^2}{1-\gamma^2} & \frac{-\gamma}{1-\gamma^2} & 0 \\ 0 & 0 & \cdots & \frac{-\gamma}{1-\gamma^2} & \frac{1+\gamma^2}{1-\gamma^2} & \frac{-\gamma}{1-\gamma^2} \\ 0 & 0 & \cdots & 0 & \frac{-\gamma}{1-\gamma^2} & \frac{1}{1-\gamma^2} \end{pmatrix}.$$

Clearly, $(G-I)^{-1}$ is symmetric, and it follows from Geršgorin's theorem that

$$\sigma\left((G-I)^{-1}\right) \subseteq \left[\frac{1-\gamma}{1+\gamma}, \frac{1+\gamma}{1-\gamma}\right].$$

Therefore,

$$\begin{aligned} \sigma(G-I) &\subseteq \left[\left(\frac{1+\gamma}{1-\gamma}\right)^{-1}, \left(\frac{1-\gamma}{1+\gamma}\right)^{-1}\right] \\ &= \left[\frac{1-\gamma}{1+\gamma}, \frac{1+\gamma}{1-\gamma}\right]. \end{aligned}$$

Consequently,

$$\sigma(G) \subseteq \left[1 + \frac{1-\gamma}{1+\gamma}, 1 + \frac{1+\gamma}{1-\gamma}\right] = \left[\frac{2}{1+\gamma}, \frac{2}{1-\gamma}\right]. \quad \square$$

We are now ready to prove the following lemma.

Lemma A.4. *$\rho(M_1) \leq -\frac{2\eta}{1+\gamma} + \eta^2\left(X^2 + \frac{1}{b(1-\gamma)^2}\right)$ where M_1 is given in Eqn 6.*

Proof. By Weyl's theorem (Horn & Johnson, 1986, Theorem 4.3.1),

$$\begin{aligned}\rho(M_1) \quad &\leq \quad \rho\left(\eta^2 X^2 I\right) + \rho\left(\frac{\eta^2}{b} D^{-1} D^{-\top}\right) + \\ &\qquad \rho\left(-\eta\left(D^{-1} + D^{-\top}\right)\right).\end{aligned}$$

The lemma then follows immediately from Lemmas A.2 and A.3. $\square$

Lemma A.5. *Let M_2 be defined by Eqn 10 and suppose the step-size is given by Eqn 11, then*

$$\rho(M_2) \leq -\frac{(1-\gamma)^2}{(1+2\gamma)^2\left(X^2 + \frac{1}{b}\right)}.$$

Proof. Let $\alpha = \frac{\eta^2}{b}$ and $\beta = \eta^2 X^2 (1+\gamma)^2 - 2\eta$, then $M_2 = D^\top \Sigma \left(\alpha D D^\top + \beta I\right) \Sigma D$. It is known that

$$\rho(M_2) = \max_{\mathbf{v}_1 \neq \mathbf{0}} \frac{\mathbf{v}_1^\top M_2 \mathbf{v}_1}{\mathbf{v}_1^\top \mathbf{v}_1}.$$

Define $\mathbf{v}_2 = D\mathbf{v}_1$ and we have:

$$\begin{aligned}\rho(M_2) \quad &= \quad \max_{\mathbf{v}_2 \neq \mathbf{0}} \frac{\mathbf{v}_2 \Sigma \left(\alpha D D^\top + \beta I\right) \Sigma \mathbf{v}_2}{\mathbf{v}_2^\top D^{-\top} D^{-1} \mathbf{v}_2} \\ &\leq \quad \max_{\mathbf{v}_2 \neq \mathbf{0}} \frac{(1-\gamma)^2 \mathbf{v}_2 \Sigma \left(\alpha D D^\top + \beta I\right) \Sigma \mathbf{v}_2}{\mathbf{v}_2^\top \mathbf{v}_2},\end{aligned}$$

where the last step is due to Lemma A.2 and the fact that M_2 is negative-definite for $\eta \ll 1$. Similarly, we define $\mathbf{v}_3 = \Sigma \mathbf{v}_2$ and use the fact that

$$0 \leq \mathbf{v}_2^\top \mathbf{v}_2 = \mathbf{v}_3^\top \Sigma^{-2} \mathbf{v}_3 \leq \left(1 + \gamma(1+\gamma)\eta X^2\right)^2 \|\mathbf{v}_3\|^2$$

to obtain:

$$\begin{aligned}\rho(M_2) \quad &\leq \quad \max_{\mathbf{v}_3 \neq \mathbf{0}} \frac{(1-\gamma)^2 \mathbf{v}_3^\top \left(\alpha D D^\top + \beta I\right) \mathbf{v}_3^\top}{\left(1 + \gamma(1+\gamma)\eta X^2\right)^2 \mathbf{v}_3^\top \mathbf{v}_3} \\ &= \quad \frac{(1-\gamma)^2 \rho\left(\alpha D D^\top + \beta I\right)}{\left(1 + \gamma(1+\gamma)\eta X^2\right)^2} \\ &\leq \quad \frac{(1-\gamma)^2\left(\alpha(1+\gamma)^2 + \beta\right)}{\left(1 + \gamma(1+\gamma)\eta X^2\right)^2}.\end{aligned}$$

If we choose η as in Eqn 11, then the lemma follows immediately from the fact that

$$1 + \frac{\gamma X^2}{(1+\gamma)\left(X^2 + \frac{1}{b}\right)} \leq 1 + \frac{\gamma}{1+\gamma} = \frac{1+2\gamma}{1+\gamma}. \quad \square$$

Acknowledgment

We thank Michael Littman, Hengshuai Yao, and the anonymous reviewers for helpful comments that improved the presentation of the paper. The author is supported by NSF under grant IIS-0325281.

References

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. *Proceedings of the Twelfth International Conference on Machine Learning (ICML-95)* (pp. 30–37).

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming.* Athena Scientific.

Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. *Advances in Neural Information Processing Systems 7 (NIPS-94)* (pp. 369–376).

Cesa-Bianchi, N., Long, P. M., & Warmuth, M. (1996). Worst-case quadratic loss bounds for prediction using linear functions and gradient descent. *IEEE Transactions on Neural Networks*, *7*, 604–619.

Horn, R. A., & Johnson, C. R. (1986). *Matrix analysis.* Cambridge University Press.

Kivinen, J., & Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, *132*, 1–63.

Lagoudakis, M. G., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4*, 1107–1149.

Munos, R. (2003). Error bounds for approximate policy iteration. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)* (pp. 560–567).

Precup, D., & Sutton, R. S. (1997). Exponentiated gradient methods for reinforcement learning. *Proceedings of the Fourteenth International Conference on Machine Learning (ICML-97)* (pp. 272–277).

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming.* New York: Wiley-Interscience.

Schapire, R. E., & Warmuth, M. K. (1996). On the worst-case analysis of temporal-difference learning algorithms. *Machine Learning*, *22*, 95–122.

Schoknecht, R., & Merke, A. (2003). TD(0) converges provably faster than the residual gradient algorithm. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)* (pp. 680–687).

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction.* Cambridge, MA: MIT Press.

Tsitsiklis, J. N., & Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, *42*, 674–690.

Knows What It Knows: A Framework For Self-Aware Learning

Lihong Li LIHONG@CS.RUTGERS.EDU
Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
Thomas J. Walsh THOMASWA@CS.RUTGERS.EDU
Department of Computer Science, Rutgers University, Piscataway, NJ 08854 USA

Abstract

We introduce a learning framework that combines elements of the well-known PAC and mistake-bound models. The KWIK (knows what it knows) framework was designed particularly for its utility in learning settings where active exploration can impact the training examples the learner is exposed to, as is true in reinforcement-learning and active-learning problems. We catalog several KWIK-learnable classes and open problems.

1. Motivation

At the core of recent reinforcement-learning algorithms that have polynomial sample complexity guarantees (Kearns & Singh, 2002; Kearns & Koller, 1999; Kakade et al., 2003; Strehl et al., 2007) lies the idea of distinguishing between instances that have been learned with sufficient accuracy and those whose outputs are still unknown.

The Rmax algorithm (Brafman & Tennenholtz, 2002), for example, estimates transition probabilities for each state–action–next-state triple of a Markov decision process (MDP). The estimates are made separately, as licensed by the Markov property, and the accuracy of the estimate is bounded using Hoeffding bounds. The algorithm explicitly distinguishes between probabilities that have been estimated accurately (known) and those for which more experience will be needed (unknown). By encouraging the agent to gather more experience in the unknown states, Rmax can guarantee a polynomial bound on the number of timesteps in which it has a non-near-optimal policy (Kakade, 2003).

In this paper, we make explicit the properties that are sufficient for a learning algorithm to be used in efficient exploration algorithms like Rmax. Roughly, the learning algorithm needs to make only accurate predictions, although it can opt out of predictions by saying "I don't know" ($\perp$). However, there must be a (polynomial) bound on the number of times the algorithm can respond $\perp$. We call such a learning algorithm KWIK ("know what it knows").

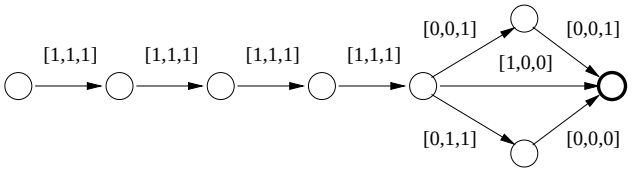


Figure 1. A cost-vector navigation graph.

Section 2 provides a motivating example and sketches possible uses for KWIK algorithms. Section 3 defines the KWIK conditions more precisely and relates them to established models from learning theory. Sections 4 and 5 survey a set of hypothesis classes for which KWIK algorithms can be created.

2. A KWIK Example

Consider the simple navigation task in Figure 1. There is a set of nodes connected by edges, with the node on the left as the source and the dark one on the right as the sink. Each edge in the graph is associated with a binary *cost vector* of dimension $d = 3$, indicated in the figure. The cost of traversing an edge is the dot product of its cost vector with a fixed weight vector $w = [1, 2, 0]$. Assume that w is not known to the agent, but the graph topology and all cost vectors are. In each episode, the agent starts from the source and moves along some path to the sink. Each time it crosses an edge, the agent observes its true cost. Once the sink is reached, the next episode begins. The learning task is to take a non-cheapest path in as few episodes as possible. There are 3 distinct paths in this example. Given the w above, the top has a cost of 12, the middle

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

13, and the bottom 15.

A simple approach for this task is for the agent to assume edge costs are uniform and walk the shortest (middle) path to collect data. It would gather 4 examples of $[1,1,1] \rightarrow 3$ and one of $[1,0,0] \rightarrow 1$. Standard regression algorithms could use this dataset to find a $\hat{w}$ that fits this data. Here, $\hat{w} = [1,1,1]$ is a natural choice. The learned weight vector could then be used to estimate costs for the three paths: 14 for the top, 13 for the middle, 14 for the bottom. Using these estimates, an agent would continue to take the middle path forever, never realizing it is not optimal.

In contrast, consider a learning algorithm that "knows what it knows". Instead of creating an approximate weight vector $\hat{w}$, it reasons about whether the costs for each edge can be obtained from the available data. The middle path, since we've seen all its edge costs, is definitely 13. The last edge of the bottom path has cost vector $[0,0,0]$, so its cost must be zero, but the penultimate edge of this path has cost vector $[0,1,1]$. This vector is a linear combination of the two observed cost vectors, so, regardless of w,

$$w{\cdot}[0,1,1] = w{\cdot}([1,1,1]-[1,0,0]) = w{\cdot}[1,1,1]-w{\cdot}[1,0,0],$$

which is just $3 - 1 = 2$. Thus, we know the bottom path's cost is 14—worse than the middle path.

The vector $[0,0,1]$ on the top path is linearly independent of the cost vectors we've seen, so its cost is unconstrained. We know we don't know. A safe thing to assume provisionally is that it's zero, encouraging the agent to try the top path in the second episode. Now, it observes $[0,0,1] \rightarrow 0$, allowing it to solve for w and accurately predict the cost for any vector (the training data spans $\Re^d$). It now knows that it knows all the costs, and can confidently take the optimal (top) path.

In general, any algorithm that guesses a weight vector may never find the optimal path. An algorithm that uses linear algebra to distinguish known from unknown costs will either take an optimal route *or* discover the cost of a linearly independent cost vector on each episode. Thus, it can never choose suboptimal paths more than d times.

The motivation for studying KWIK learning grew out of its use in multi-state sequential decision making problems like this one. However, other machine-learning problems could benefit from this perspective and from the development of efficient algorithms. For instance, action selection in bandit problems (Fong, 1995) and associative bandit problems (Strehl et al., 2006) (bandit problems with inputs) can both be addressed in the KWIK setting by choosing the better arm when both payoffs are known and an unknown arm otherwise.

KWIK could also be a useful framework for studying active learning (Cohn et al., 1994) and anomaly detection (Lane & Brodley, 2003), both of which are machine-learning problems that require some degree of reasoning about whether a recently presented input is predictable from previous examples. When mistakes are costly, as in utility-based data mining (Weiss & Tian, 2006) or learning robust control (Bagnell et al., 2001), having explicit predictions of certainty can be very useful for decision making.

3. Formal Definition

This section provides a formal definition of KWIK learning and its relationship to existing frameworks.

3.1. KWIK Definition

KWIK is an objective for supervised learning algorithms. In particular, we begin with an *input set* X and *output set* Y. The *hypothesis class* H consists of a set of functions from X to Y: $H \subseteq (X \rightarrow Y)$. The *target function* $h^* \in H$ is the source of training examples and is unknown to the learner. Note that the setting is "realizable", meaning we assume the target function is in the hypothesis class.

The protocol for a "run" is:

- The hypothesis class H and accuracy parameters ϵ and δ are known to both the learner and environment.
- The environment selects a target function $h^* \in H$ adversarially.
- Repeat:
  - The environment selects an input $x \in X$ adversarially and informs the learner.
  - The learner predicts an output $\hat{y} \in Y \cup \{\perp\}$.
  - If $\hat{y} \neq \perp$, it should be accurate: $|\hat{y} - y| \leq \epsilon$, where $y = h^*(x)$. Otherwise, the entire run is considered a failure. The probability of a failed run must be bounded by δ.
  - Over a run, the total number of steps on which $\hat{y} = \perp$ must be bounded by $B(\epsilon, \delta)$, ideally polynomial in $1/\epsilon$, $1/\delta$, and parameters defining H. Note that this bound should hold even if $h^* \notin H$, although, obviously, outputs need not be accurate in this case.
  - If $\hat{y} = \perp$, the learner makes an observation $z \in Z$ of the output, where $z = y$ in the deterministic case, $z = 1$ with probability y and

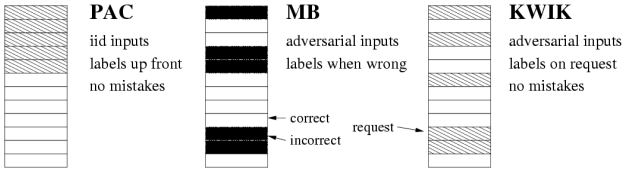


Figure 2. Relationship of KWIK to existing PAC and MB (mistake bound) frameworks in terms of how labels are provided for inputs.

0 with probability $1-y$ in the Bernoulli case, or $z = y + \eta$ for zero-mean random variable η in the additive noise case.

3.2. Connection to PAC and MB

Figure 2 illustrates the relationship of KWIK to the similar PAC (probably approximately correct) (Valiant, 1984) and MB (mistake bound) (Littlestone, 1987) frameworks. In all three cases, a series of inputs (instances) is presented to the learner. Each input is depicted in the figure by a rectangular box.

In the PAC model, the learner is provided with labels (correct outputs) for an initial sequence of inputs, depicted by cross-hatched boxes. After that point, the learner is responsible for producing accurate outputs (empty boxes) for all new inputs. Inputs are drawn from a fixed distribution.

In the MB model, the learner is expected to produce an output for every input. Labels are provided to the learner whenever it makes a mistake (filled boxes). Inputs are selected adversarially, so there is no bound on when the last mistake might be made. However, MB algorithms guarantee that the *total* number of mistakes is small, so the ratio of incorrect to correct outputs must go to zero asymptotically. Any MB algorithm for a hypothesis class can be used to provide a PAC algorithm for the same class, but not necessarily vice versa (Blum, 1994).

The KWIK model has elements of both PAC and MB. Like PAC, a KWIK algorithm is not allowed to make mistakes. Like MB, inputs to a KWIK algorithm are selected adversarially. Instead of bounding mistakes, a KWIK algorithm must have a bound on the number of label requests ($\perp$) it can make. By requiring performance to be independent of the distribution, a KWIK algorithm can be used in cases in which the input distribution is dependent in complex ways on the KWIK algorithm's behavior, as can happen in on-line or active learning settings. And, like PAC and MB, the definition of KWIK algorithms can be naturally extended to enforce low computational complexity.

Note that any KWIK algorithm can be turned into a MB algorithm with the same bound by simply having the algorithm guess an output each time it is not certain. However, some hypothesis classes are exponentially harder to learn in the KWIK setting than in the MB setting. An example is conjunctions of n Boolean variables, in which MB algorithms can guess "false" when uncertain and learn with $n+1$ mistakes, but a KWIK algorithm may need $\Omega(2^{n/2})$ $\perp$s to acquire the negative examples required to capture the target hypothesis.

3.3. Other Online Learning Models

The notion of allowing the learner to opt out of some inputs by returning $\perp$ is not unique to KWIK. Several other authors have considered related models. For instance, sleeping experts (Freund et al., 1997) can respond $\perp$ for some inputs, although they need not learn from these experiences. Learners in the settings of Selective Sampling (SS) (Cesa-Bianchi et al., 2006) and Label Efficient Prediction (Cesa-Bianchi et al., 2005) request labels randomly with a changing probability and achieve bounds on the expected number of mistakes and the expected number of label requests for a finite number of interactions. These algorithms cannot be used unmodified in the KWIK setting because, with high probability, KWIK algorithms must not make mistakes at any time. In the MB-like Apple-Tasting setting (Helmbold et al., 2000), the learner receives feedback asymmetrically only when it predicts a particular label (a positive example, say), which conflates the request for a sample with the prediction of a particular outcome.

Open Problem 1 *Is there a way of modifying SS algorithms to satisfy the KWIK criteria?*

4. Some KWIK Learnable Classes

This section describes some hypothesis classes for which KWIK algorithms are available. It is not meant to be an exhaustive survey, but simply to provide a flavor for the properties of hypothesis classes KWIK algorithms can exploit. The complexity of many learning problems has been characterized by defining the *dimensionality* of hypothesis classes (Angluin, 2004). No such definition has been found for the KWIK model, so we resort to enumerating examples of learnable classes.

Open Problem 2 *Is there a way of characterizing the "dimension" of a hypothesis class in a way that can be used to derive KWIK bounds?*

4.1. Memorization and Enumeration

We begin by describing the simplest and most general KWIK algorithms.

Algorithm 1 *The* memorization algorithm *can learn any hypothesis class with input space X with a KWIK bound of $|X|$. This algorithm can be used when the input space X is finite and observations are noise free.*

To achieve this bound, the algorithm simply keeps a mapping $\hat{h}$ initialized to $\hat{h}(x) = \perp$ for all $x \in X$. When the environment chooses an input x, the algorithm reports $\hat{h}(x)$. If $\hat{h}(x) = \perp$, the environment will provide a label y and the algorithm will assign $\hat{h}(x) := y$. It will only report $\perp$ once for each input, so the KWIK bound is $|X|$.

Algorithm 2 *The* enumeration algorithm *can learn any hypothesis class H with a KWIK bound of $|H|-1$. This algorithm can be used when the hypothesis class H is finite and observations are noise free.*

The algorithm keeps track of $\hat{H}$, the version space, and initially $\hat{H} = H$. Each time the environment provides input $x \in X$, the algorithm computes $\hat{L} = \{h(x) | h \in \hat{H}\}$. That is, it builds the set of all outputs for x for all hypotheses that have not yet been ruled out. If $|\hat{L}| = 0$, the version space has been exhausted and the target hypothesis is not in the hypothesis class ($h^* \notin H$).

If $|\hat{L}| = 1$, it means that all hypotheses left in $\hat{H}$ agree on the output for this input, and therefore the algorithm knows what the proper output must be. It returns $\hat{y} \in \hat{L}$. On the other hand, if $|\hat{L}| > 1$, two hypotheses in the version space disagree. In this case, the algorithm returns $\perp$ and receives the true label y. It then computes an updated version space

$$\hat{H}' = \{h | h \in \hat{H} \wedge h(x) = y\}.$$

Because $|\hat{L}| > 1$, there must be some $h \in \hat{H}$ such that $h(x) \neq y$. Therefore, the new version space must be smaller $|\hat{H}'| \leq |\hat{H}| - 1$. Before the next input is received, the version space is updated $\hat{H} := \hat{H}'$.

If $|\hat{H}| = 1$ at any point, $|\hat{L}| = 1$, and the algorithm will no longer return $\perp$. Therefore, $|H|-1$ is the maximum number of $\perp$s the algorithm can return.

Example 1 *You own a bar that is frequented by a group of n patrons P. There is one patron $\mathbf{f} \in P$ who is an instigator—whenever a group of patrons is in the bar $G \subseteq P$, if $\mathbf{f} \in G$, a fight will break out. However, there is another patron $\mathbf{p} \in P$, who is a peacemaker. If $\mathbf{p}$ is in the group, it will prevent a fight, even if $\mathbf{f}$ is present.*

You want to predict whether a fight will break out among a subset of patrons, initially without knowing the identities of $\mathbf{f}$ and $\mathbf{p}$. The input space is $X = 2^P$ and the output space is $Y = \{$fight, no fight$\}$.

The memorization algorithm achieves a KWIK bound of 2^n for this problem, since it may have to see each possible subset of patrons. However, the enumeration algorithm can KWIK learn this hypothesis class with a bound of $n(n-1)$ since there is one hypothesis for each possible assignment of a patron to $\mathbf{f}$ and $\mathbf{p}$. Each time it reports $\perp$, it is able to rule out at least one possible instigator–peacemaker combination.

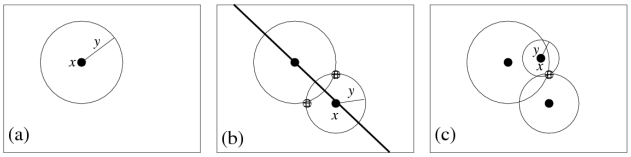


Figure 3. Schematic of behavior of the planar-distance algorithm after the first (a), second (b), and third (c) time it returns $\perp$.

4.2. Real-valued Functions

The previous two examples exploited the finiteness of the hypothesis class and input space. KWIK bounds can also be achieved when these sets are infinite.

Algorithm 3 *Define $X = \Re^2$, $Y = \Re$, and*

$$H = \{f | f : X \to Y, c \in \Re^2, f(x) = \|x - c\|_2\}.$$

This is, there is an unknown point and the target function maps input points to the distance from the unknown point. The planar-distance algorithm *can learn in this hypothesis class with a KWIK bound of 3.*

The algorithm proceeds as follows, illustrated in Figure 3. First, given initial input x, the algorithm says $\perp$ and receives output y. Since y is the distance between x and some unknown point c, we know c must lie on the circle illustrated in Figure 3(a). (If $y = 0$, then $c = x$.) Let's call this input–output pair x_1, y_1. The algorithm will return y_1 for any future input that matches x_1. Otherwise, it will need to return $\perp$ and will obtain a new input–output pair x, y, as shown in Figure 3(b). They become x_2 and y_2.

Now, the algorithm can narrow down the location of c to the two hatch-marked points. In spite of this ambiguity, for any input on the dark diagonal line the algorithm will be able to return the correct distance—all

these points are equidistant from the two possibilities. The two circles must intersect, assuming the target hypothesis is in H[1].

Once an input x is received that is not co-linear with x_1 and x_2, the algorithm returns $\perp$ and obtains another x, y pair, as illustrated in Figure 3(c). Finally, since three circles will intersect at at most one point, the algorithm can identify the location of c and use it to correctly answer any future query. Thus, three $\perp$s suffice for KWIK learning in this setting. The algorithm generalizes to d-dimensional versions of the setting with a KWIK bound of $d + 1$.

Algorithm 3 illustrates a number of important points. First, since learners have no control over their inputs in the KWIK setting, they must be robust to degenerate inputs such as inputs that lie precisely on a line. Second, they can often return valid answers for some inputs even before they have learned the target function over the entire input space.

4.3. Noisy Observations

Up to this point, observations have been noise free. Next, we consider the simplest noisy KWIK learning problem in the Bernoulli case.

Algorithm 4 *The* coin-learning algorithm *can accurately predict the probability that a biased coin will come up heads given Bernoulli observations with a KWIK bound of* $B(\epsilon, \delta) = \frac{1}{2\epsilon^2} \ln \frac{2}{\delta} = O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}\right)$.

We have a biased coin whose unknown probability of heads is p. In the notation of this paper, $|X| = 1$, $Y = [0, 1]$, and $Z = \{0, 1\}$. We want to learn an estimate $\hat{p}$ that is accurate ($|\hat{p} - p| \leq \epsilon$) with high probability $(1 - \delta)$.

If we could observe p, then this problem would be trivial: Say $\perp$ once, observe p, and let $\hat{p} = p$. The KWIK bound is thus 1. Now, however, observations are noisy. Instead of observing p, we see either 1 (with probability p) or 0 (with probability $1 - p$).

Each time the algorithm says $\perp$, it gets an independent trial that it can use to compute $\hat{p} = \frac{1}{T} \sum_{t=1}^{T} z_t$, where $z_t \in Z$ is the tth observation in T trials. The number of trials needed before we are $1-\delta$ certain our estimate is within ϵ can be computed using a Hoeffding bound:

$$T \geq \frac{1}{2\epsilon^2} \ln \frac{2}{\delta} = O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}\right).$$

[1]They can also intersect at one point, if the circles are tangent, in which case the algorithm can identify c unambiguously.

Algorithm 5 *Define* $X = \Re^d$, $Y = \Re$, *and*

$$H = \{f | f : X \to Y, w \in \Re^d, f(x) = w \cdot x\}.$$

That is, H is the linear functions on d variables. Given additive noise, the noisy linear-regression algorithm *can learn in H with a KWIK bound of* $B(\epsilon, \delta) = \tilde{O}(d^3/\epsilon^4)$, *where* $\tilde{O}(\cdot)$ *suppresses log factors.*

The deterministic case was described in Section 2 with a bound of d. Here, the algorithm must be cautious to average over the noisy samples to make predictions accurately. This problem was solved by Strehl and Littman (2008). The algorithm uses the least squares estimate of the weight vector for inputs with high certainty. Certainty is measured by two terms representing (1) the number and proximity of previous samples to the current point and (2) the appropriateness of the previous samples for making a least squares estimate. When certainty is low for either measure, the algorithm reports $\perp$ and observes a noisy sample of the linear function.

Here, solving a noisy version of a problem resulted in an increased KWIK bound (from d to essentially d^3). Note that the deterministic Algorithm 3 also has a bound of d, but no bound has been found for the stochastic case.

Open Problem 3 *Is there a general scheme for taking a KWIK algorithm for a deterministic class and updating it to work in the presence of noise?*

5. Combining KWIK Learners

This section provides examples of how KWIK learners can be combined to provide learning guarantees for more complex hypothesis classes.

Algorithm 6 *Let* $F : X \to Y$ *be the set of functions mapping input set X to output set Y. Let* $H_1, \ldots, H_k$ *be a set of KWIK learnable hypothesis classes with bounds of* $B_1(\epsilon, \delta), \ldots, B_k(\epsilon, \delta)$ *where* $H_i \subseteq F$ *for all* $1 \leq i \leq k$. *That is, all the hypothesis classes share the same input/output sets. The* union algorithm *can learn the joint hypothesis class* $H = \bigcup_i H_i$ *with a KWIK bound of* $B(\epsilon, \delta) = (1 - k) + \sum_i B_i(\epsilon, \delta)$.

The union algorithm is like a higher-level version of the enumeration algorithm (Algorithm 2) and applies in the deterministic setting. It maintains a set of active algorithms $\hat{A}$, one for each hypothesis class: $\hat{A} = \{1, \ldots, k\}$. Given an input x, the union algorithm queries each algorithm $i \in \hat{A}$ to obtain a prediction $\hat{y}_i$ from each active algorithm. Let $\hat{L} = \{\hat{y}_i | i \in \hat{A}\}$.

If $\perp \in \hat{L}$, the union algorithm reports $\perp$ and obtains the correct output y. Any algorithm i for which $\hat{y} = \perp$ is then sent the correct output y to allow it to learn. If $|\hat{L}| > 1$, then there is disagreement among the subalgorithms. The union algorithm reports $\perp$ in this case because at least one of the algorithms has learned the wrong hypothesis and it needs to know which.

Any algorithm that made a prediction other than y or $\perp$ is "killed"—removed from consideration. That is,

$$\hat{A}' = \{i | i \in \hat{A} \wedge (\hat{y}_i = \perp \vee \hat{y}_i = y)\}.$$

On each input for which the union algorithm reports $\perp$, either one of the subalgorithms reported $\perp$ (at most $\sum_i B_i(\epsilon, \delta)$ times) or two algorithms disagreed and at least one was removed from $\hat{A}$ (at most $k - 1$ times). The KWIK bound follows from these facts.

Example 2 *Let $X = Y = \Re$. Now, define $H_1 = \{f | f(x) = |x - c|, c \in \Re\}$. That is, each function in H_1 maps x to its distance from some unknown point c. We can learn H_1 with a KWIK bound of 2 using a 1-dimensional version of Algorithm 3. Next, define $H_2 = \{f | f(x) = yx + b, m \in \Re, b \in \Re\}$. That is, H_2 is the set of lines. We can learn H_2 with a KWIK bound of 2 using the regression algorithm in Section 2. Finally, define $H = H_1 \cup H_2$, the union of these two classes. We can use Algorithm 6 to KWIK learn H.*

Assume the first input is $x_1 = 2$. The union algorithm asks the learners for H_1 and H_2 the output for x_1 and neither has any idea, so it returns $\perp$ and receives the feedback $y_1 = 2$, which it passes to the subalgorithms. The next input is $x_2 = 8$. The learners for H_1 and H_2 still don't have enough information, so it returns $\perp$ and sees $y_2 = 4$, which it passes to the subalgorithms. Next, $x_3 = 1$. Now, the learner for H_1 unambiguously computes $c = 4$, because that's the only interpretation consistent with the first two examples ($|2 - 4| = 2$, $|8 - 4| = 4$), so it returns $|1 - 4| = 3$. On the other hand, the learner for H_2 unambiguously computes $m = 1/3$ and $b = 4/3$, because that's the only interpretation consistent with the first two examples ($2 \times 1/3 + 4/3 = 2$, $8 \times 1/3 + 4/3 = 4$), so it returns $1 \times 1/3 + 4/3 = 5/3$. Since the two subalgorithms disagree, the union algorithm returns $\perp$ one last time and finds out that $y_3 = 3$. It makes all future predictions (accurately) using the algorithm for H_1.

Next, we consider a variant of Algorithm 1 that combines learners across disjoint input spaces.

Algorithm 7 *Let $X_1, \ldots, X_k$ be a set of disjoint input spaces ($X_i \cap X_j = \emptyset$ if $i \neq j$) and Y be an output space. Let $H_1, \ldots, H_k$ be a set of KWIK learnable hypothesis classes with bounds of $B_1(\epsilon, \delta), \ldots, B_k(\epsilon, \delta)$ where $H_i \in (X_i \to Y)$. The* input-partition algorithm *can learn the hypothesis class $H \in (X_1 \cup \cdots \cup X_k \to Y)$ with a KWIK bound of $B(\epsilon, \delta) = \sum_i B_i(\epsilon, \delta/k)$.*

The input-partition algorithm runs the learning algorithm for each subclass H_i. When it receives an input $x \in X_i$, it queries the learning algorithm for class H_i and returns its response, which is ϵ accurate, by request. To achieve $1 - \delta$ certainty, it insists on $1 - \delta/k$ certainty from each of the subalgorithms. By the union bound, the overall failure probability must be less than the sum of the failure probabilities for the subalgorithms.

Example 3 *An MDP consists of n states and m actions. For each combination of state and action and next state, the transition function returns a probability. As the reinforcement-learning agent moves around in the state space, it observes state–action–state transitions and must predict the probabilities for transitions it has not yet observed. In the model-based setting, an algorithm learns a mapping from the size n^2m input space of state–action–state combinations to probabilities via Bernoulli observations. Thus, the problem can be solved via the input-partition algorithm (Algorithm 7) over a set of individual probabilities learned via Algorithm 4. The resulting KWIK bound is $B(\epsilon, \delta) = O\left(\frac{n^2 m}{\epsilon^2} \ln \frac{nm}{\delta}\right)$.*

Note that this approach is precisely what is found in most efficient RL algorithms in the literature (Kearns & Singh, 2002; Brafman & Tennenholtz, 2002).

Algorithm 7 combines hypotheses by partitioning the input space. In contrast, the next example concerns combinations in input and output space.

Algorithm 8 *Let $X_1, \ldots, X_k$ and $Y_1, \ldots, Y_k$ be a set of input and output spaces and $H_1, \ldots, H_k$ be a set of KWIK learnable hypothesis classes with bounds of $B_1(\epsilon, \delta), \ldots, B_k(\epsilon, \delta)$ on these spaces. That is, $H_i \in (X_i \to Y_i)$. The* cross-product algorithm *can learn the hypothesis class $H \in ((X_1 \times \cdots \times X_k) \to (Y_1 \times \cdots \times Y_k))$ with a KWIK bound of $B(\epsilon, \delta) = \sum_i B_i(\epsilon, \delta/k)$.*

Here, each input consists of a vector of inputs from each of the spaces $X_1, \ldots, X_k$ and outputs are vectors of outputs from $Y_1, \ldots, Y_k$. Like Algorithm 7, each component of this vector can be learned independently via the corresponding algorithm. Each is learned to within an accuracy of ϵ and confidence $1 - \delta/k$. Any time any component returns $\perp$, the cross-product algorithm returns $\perp$. Since each $\perp$ returned can be traced to one of the subalgorithms, the total is bounded as

described above. By the union bound, total failure probability is no more than $k \times \delta/k = \delta$.

Example 4 *Transitions in factored-state MDP can be thought of as mappings from vectors to vectors. Given known dependencies, the cross-product algorithm can be used to learn each component of the transition function. Each component is, itself, an instance of Algorithm 7 applied to the coin-learning algorithm. This three-level KWIK algorithm provides an approach to learn the transition function of a factored-state MDP with a polynomial KWIK bound. This insight can be used to derive the factored-state-MDP learning algorithm used by Kearns and Koller (1999).*

The previous two algorithms apply to both deterministic and noisy observations. We next provide a powerful algorithm that generalizes the union algorithm (Algorithm 6) to work with noisy observations as well.

Algorithm 9 *Let $F : X \to Y$ be the set of functions mapping input set X to output set $Y = [0, 1]$. Let $Z = \{0, 1\}$ be a binary observation set. Let $H_1, \ldots, H_k$ be a set of KWIK learnable hypothesis classes with bounds of $B_1(\epsilon, \delta), \ldots, B_k(\epsilon, \delta)$ where $H_i \subseteq F$ for all $1 \leq i \leq k$. That is, all the hypothesis classes share the same input/output sets. The* noisy union algorithm *can learn the joint hypothesis class $H = \bigcup_i H_i$ with a KWIK bound of $B(\epsilon, \delta) = O\left(\frac{k}{\epsilon^2} \ln \frac{k}{\delta}\right) + \sum_{i=1}^{k} B_i(\frac{\epsilon}{4}, \frac{\delta}{k+1})$.*

For simplicity, we sketch the special case of $k = 2$. The general case will be briefly discussed at the end. The noisy union algorithm is similar to the union algorithm (Algorithm 6), except that it has to deal with noisy observations. The algorithm proceeds by running the KWIK algorithms, using parameters (ϵ_0, δ_0), as subalgorithms for each of the H_i hypothesis classes, where $\epsilon_0 = \frac{\epsilon}{4}$ and $\delta_0 = \frac{\delta}{3}$. Given an input x_t in trial t, it queries each algorithm i to obtain a prediction $\hat{y}_{ti}$. Let $\hat{L}_t$ be the set of responses.

If $\perp \in \hat{L}_t$, the noisy union algorithm reports $\perp$, obtains an observation $z_t \in Z$, and sends it to all subalgorithms i with $\hat{y}_{ti} = \perp$ to allow them to learn. In the following, we focus on the other case where $\perp \notin \hat{L}_t$.

If $|\hat{y}_{t1} - \hat{y}_{t2}| \leq 4\epsilon_0$, then these two predictions are sufficiently consistent, and we claim that, with high probability, the prediction $\hat{p}_t = (\hat{y}_{t1} + \hat{y}_{t2})/2$ is ϵ-close to $y_t = \Pr(z_t = 1)$. This claim follows because, by assumption, one of the predictions, say $\hat{y}_{t1}$, deviates from y_t by at most ϵ_0 with probability at least $1 - \delta/3$, and hence $|\hat{p}_t - y_t| = |\hat{p}_t - \hat{y}_{t1} + \hat{y}_{t1} - y_t| \leq |\hat{p}_t - \hat{y}_{t1}| + |\hat{y}_{t1} - \hat{y}_t| = |\hat{y}_{t1} - \hat{y}_{t2}|/2 + |\hat{y}_{t1} - \hat{y}_t| \leq 2\epsilon_0 + \epsilon_0 < \epsilon$.

If $|\hat{y}_{t1} - \hat{y}_{t2}| > 4\epsilon_0$, then the individual predictions are not consistent enough for the noisy union algorithm to make an ϵ-accurate prediction. Thus, the noisy union algorithm reports $\perp$ and needs to know which subalgorithm provided an inaccurate response. But, since the observations are noisy in this problem, it cannot eliminate h_i on the basis of a single observation. Instead, it maintains the total squared prediction error for every subalgorithm i: $\ell_i = \sum_{t \in I} (\hat{y}_{ti} - z_t)^2$, where $I = \{t \,|\, |\hat{y}_{t1} - \hat{y}_{t2}| > 4\epsilon_0\}$ is the set of trials in which the subalgorithms gave inconsistent predictions. We observe that $|I|$ is the number of $\perp$s returned by the noisy union algorithm alone (not counting those returned by the subalgorithms). Our last step is to show ℓ_i provides a robust measure for eliminating invalid predictors when $|I|$ is sufficiently large.

Applying the Hoeffding bound and some algebra, we find $\Pr(\ell_1 > \ell_2) \leq$

$$\exp\left(-\frac{\sum_{t \in I} |\hat{y}_{t1} - \hat{y}_{t2}|^2}{8}\right) \leq \exp\left(-2\epsilon_0^2 |I|\right).$$

Setting the righthand side to be $\delta/3$ and solving for $|I|$, we have $|I| = \frac{1}{2\epsilon_0^2} \ln \frac{3}{\delta} = O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}\right)$.

Since each h_i succeeds with probability $1 - \frac{\delta}{3}$, and the comparison of ℓ_1 and ℓ_2 also succeeds with probability $1 - \frac{\delta}{3}$, a union bound implies that the noisy union algorithm succeeds with probability at least $1 - \delta$. All $\perp$s are either from a subalgorithm (at most $\sum_i B_i(\epsilon_0, \delta_0)$) or from the noisy union algorithm ($O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}\right)$).

The general case where $k > 2$ can be reduced to the $k = 2$ case by pairing the k learners and running the noisy union algorithm described above on each pair. Here, each subalgorithm is run with parameter $\frac{\epsilon}{4}$ and $\frac{\delta}{k+1}$. Although there are $\binom{k}{2} = O(k^2)$ pairs, a slightly improved reduction and analysis can reduce the dependence of $|I|$ on k from quadratic to linearithmic, leading to the bound given in the statement.

Example 5 *Without known dependencies, learning a factored-state MDP is more challenging. Strehl et al. (2007) showed that each possible dependence structure can be viewed as a separate hypothesis and provided an algorithm for learning the dependencies in a factored-state MDP while learning the transition probabilities. The algorithm can be viewed as a four-level KWIK algorithm with a noisy union algorithm at the top (to discover the dependence structure), a cross-product algorithm beneath it (to decompose the transitions for the separate components of the factored-state representation), an input-partition algorithm below that (to handle the different combinations of state component and action), and a coin-learning algorithm*

at the very bottom (to learn the transition probabilities themselves). Note that Algorithm 9 is conceptually simpler, significantly more efficient ($k \log k$ vs. k^2 dependence on k), and more generally applicable than the one due to Strehl et al. (2007).

6. Conclusion and Future Work

We described the KWIK ("knows what it knows") model of supervised learning, which identifies and generalizes a key step common to a class of algorithms for efficient exploration. We provided algorithms for a set of basic hypothesis classes given deterministic and noisy observations as well as methods for composing hypothesis classes to create more complex algorithms. One example algorithm consisted of a four-level decomposition of an existing learning algorithm from the reinforcement-learning literature.

By providing a set of example algorithms and composition rules, we hope to encourage the use of KWIK algorithms as a component in machine-learning applications as well as spur the development of novel algorithms. One concern of particular interest in applying the KWIK framework to real-life data we leave as an open problem.

Open Problem 4 *How can KWIK be adapted to apply in the unrealizable setting in which the target hypothesis can be chosen from outside the hypothesis class H?*

References

Angluin, D. (2004). Queries revisited. *Theoretical Computer Science*, *313*, :175–194.

Bagnell, J., Ng, A. Y., & Schneider, J. (2001). *Solving uncertain Markov decision problems* (Technical Report CMU-RI-TR-01-25). Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Blum, A. (1994). Separating distribution-free and mistake-bound learning models over the Boolean domain. *SIAM Journal on Computing*, *23*, 990–1000.

Brafman, R. I., & Tennenholtz, M. (2002). R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Worst-case analysis of selective sampling for linear classification. *Journal of Machine Learning Research*, *7*, 1205–1230.

Cesa-Bianchi, N., Lugosi, G., & Stoltz, G. (2005). Minimizing regret with label efficient prediction. *IEEE Transactions on Information Theory*, *51*, 2152–2162.

Cohn, D. A., Atlas, L., & Ladner, R. E. (1994). Improving generalization with active learning. *Machine Learning*, *15*, 201–221.

Fong, P. W. L. (1995). A quantitative study of hypothesis selection. *Proceedings of the Twelfth International Conference on Machine Learning (ICML-95)* (pp. 226–234).

Freund, Y., Schapire, R. E., Singer, Y., & Warmuth, M. K. (1997). Using and combining predictors that specialize. *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (pp. 334–343).

Helmbold, D. P., Littlestone, N., & Long, P. M. (2000). Apple tasting. *Information and Computation*, *161*, 85–139.

Kakade, S., Kearns, M., & Langford, J. (2003). Exploration in metric state spaces. *Proceedings of the 20th International Conference on Machine Learning*.

Kakade, S. M. (2003). *On the sample complexity of reinforcement learning*. Doctoral dissertation, Gatsby Computational Neuroscience Unit, University College London.

Kearns, M. J., & Koller, D. (1999). Efficient reinforcement learning in factored MDPs. *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 740–747).

Kearns, M. J., & Singh, S. P. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

Lane, T., & Brodley, C. E. (2003). An empirical study of two approaches to sequence learning for anomaly detection. *Machine Learning*, *51*, 73–107.

Littlestone, N. (1987). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Strehl, A. L., Diuk, C., & Littman, M. L. (2007). Efficient structure learning in factored-state MDPs. *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07)*.

Strehl, A. L., & Littman, M. L. (2008). Online linear regression and its application to model-based reinforcement learning. *Advances in Neural Information Processing Systems 20*.

Strehl, A. L., Mesterharm, C., Littman, M. L., & Hirsh, H. (2006). Experience-efficient learning in associative bandit problems. *Proceedings of the Twenty-third International Conference on Machine Learning (ICML-06)*.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*, 1134–1142.

Weiss, G. M., & Tian, Y. (2006). Maximizing classifier utility when training data is costly. *SIGKDD Explorations*, *8*, 31–38.

Acknowledgments

Support provided by NSF IIS-0325281 and DARPA TL.

Pairwise Constraint Propagation by Semidefinite Programming for Semi-Supervised Classification

Zhenguo Li ZGLI5@IE.CUHK.EDU.HK
Jianzhuang Liu JZLIU@IE.CUHK.EDU.HK
Xiaoou Tang XTANG@IE.CUHK.EDU.HK
Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong

Abstract

We consider the general problem of learning from both pairwise constraints and unlabeled data. The pairwise constraints specify whether two objects belong to the same class or not, known as the must-link constraints and the cannot-link constraints. We propose to learn a mapping that is smooth over the data graph and maps the data onto a unit hypersphere, where two must-link objects are mapped to the same point while two cannot-link objects are mapped to be orthogonal. We show that such a mapping can be achieved by formulating a semidefinite programming problem, which is convex and can be solved globally. Our approach can effectively propagate pairwise constraints to the whole data set. It can be directly applied to multi-class classification and can handle data labels, pairwise constraints, or a mixture of them in a unified framework. Promising experimental results are presented for classification tasks on a variety of synthetic and real data sets.

1. Introduction

Learning from both labeled and unlabeled data, known as semi-supervised learning, has attracted considerable interest in recent years (Chapelle et al., 2006), (Zhu, 2005). The key to the success of semi-supervised learning is the cluster assumption (Zhou et al., 2004), stating that nearby objects and objects on the same manifold structure are likely to be in the same class. Different algorithms actually implement the cluster assumption from different viewpoints (Zhu et al.,), (Zhou et al., 2004), (Belkin et al., 2006), (Chapelle & Zien, 2005), (Zhang & Ando, 2006), (Szummer et al., 2002). When the cluster assumption is appropriate, we can properly classify the whole data set with only one labeled object for each class.

However, the distributions of real-world data are often more complex than expected, where there are circumstances that a class may consist of multiple separate groups or manifolds, and different classes may be close to or even overlap with each other. For example, a common experience is that face images of the same person under different poses and illuminations can be drastically different, while those with similar appearances may originate from two different persons. To handle the classification problems of such practical data, additional assumptions should be made and more supervisory information should be exploited when available.

Class labels of data are the most widely used supervisory information. In addition, pairwise constraints are also often seen, which specify whether two objects belong to the same class or not, known as the must-link constraints and the cannot-link constraints. Such pairwise constraints may arise from domain knowledge automatically or with a little human effort (Wagstaff & Cardie, 2000), (Klein et al., 2002), (Kulis et al., 2005), (Chapelle et al., 2006). They can also be obtained from data labels where objects with the same label are must-link while objects with different labels are cannot-link. Generally, we cannot infer data labels from only pairwise constraints, especially for multi-class data. In this sense, pairwise constraints are inherently weaker and thus more general than labels of data. Pairwise constrains have been widely used in the contexts of clustering with side information (Wagstaff et al., 2001), (Klein et al., 2002), (Xing et al., 2003), (Kulis et al., 2005), (Kamvar et al., 2003), (Globerson & Roweis, 2006), (Basu et al., 2004), (Bilenko

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

et al., 2004), (Bar-Hillel et al., 2003), (Hoi et al., 2007), where it has been shown that the presence of appropriate pairwise constraints can often improve the clustering performance.

In this paper, we consider a more general problem of semi-supervised classification from pairwise constraints and unlabeled data, which includes the traditional semi-supervised classification as a subproblem that considers labeled and unlabeled data. Note that the label propagation techniques, which are often used in the traditional semi-supervised classification (Zhou et al., 2004), (Zhu et al.,), (Belkin et al., 2006), cannot be readily generalized to propagate pairwise constraints. Recently, two methods (Goldberg et al., 2007), (Tong & Jin, 2007) are proposed to incorporate dissimilarity information in semi-supervised classification, which is similar to the cannot-link constraints. It is important to notice that the similarities between objects are not identical to the must-link constraints imposed on them. The former reflects their distances in the input space while the latter is often obtained using domain knowledge or specified by the user.

We propose an approach, called *pairwise constraint propagation* (PCP), that can effectively propagate pairwise constraints to the whole data set. PCP intends to learn a mapping that is smooth over the data graph and maps the data onto a unit hypersphere, where two must-link objects are mapped to the same point while two cannot-link objects are mapped to be orthogonal. Such a mapping can be implicitly achieved using the kernel trick via semidefinite programming, which is convex and can be solved globally. Our approach can be directly applied to multi-class classification and can handle data labels, pairwise constraints, or a mixture of them in a unified framework.

2. Motivation

Given a data set of n objects $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ and two sets of pairwise must-link and cannot-link constraints, denoted respectively by $\mathcal{M} = \{(\mathbf{x}_i, \mathbf{x}_j)\}$ where $\mathbf{x}_i$ and $\mathbf{x}_j$ should be in the same class and $\mathcal{C} = \{(\mathbf{x}_i, \mathbf{x}_j)\}$ where $\mathbf{x}_i$ and $\mathbf{x}_j$ should be in different classes, our goal is to classify $\mathcal{X}$ into k classes such that not only the constraints are satisfied, but also those unlabeled objects similar to two must-link objects respectively are classified into the same class and those similar to two cannot-link objects respectively are classified into different classes.

To better illustrate our purpose, let us consider the classification task on a toy data set shown in Fig. 1(a). Although this data set consists of three separate

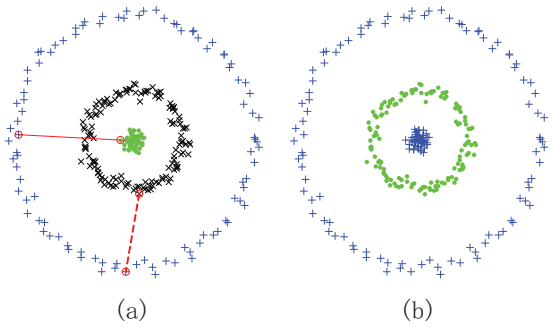


Figure 1. Classification on Three-Circle. (a) A data set with one must-link constraint and one cannot-link constraint, denoted by the solid and the dashed red lines, respectively. (b) Ideal classification (two classes) we hope to obtain where different classes are denoted by different colors and symbols.

groups (denoted by different colors and symbols in Fig. 1(a)), it has only two classes (Fig. 1(b)). We argue that the must-link constraint asks merging the outer circle and the inner circle into one class, instead of just merging the two must-link objects; and the cannot-link constraint asks for keeping the middle circle and the outer circles into different classes, not just keeping the two cannot-link objects into different classes. Consequently, the desired classification result is the one shown in Fig. 1(b). It is such a global implication that we interpret the pairwise constraints.

From this simple example, we can see that the cluster assumption is still valid, i.e., nearby objects tend to be in the same class and objects on the same manifold structure also tend to be in the same class. However, this assumption does not concern those objects that are not close to each other and do not share the same manifold structure. We argue that the classification for such objects should accord with the input pairwise constraints. For example, any two objects on the outer and inner circles in Fig. 1(a) should be in the same class because they respectively share the same manifold structures with the two must-link objects, and any two objects on the outer and middle circles should be in different classes because they respectively share the same manifold structures with the two cannot-link objects. We refer to this assumption as the *pairwise constraint assumption*.

In this paper, we seek to implement both the cluster assumption and the pairwise constraint assumption in a unified framework. A dilemma is that one may specify nearby objects or objects sharing the same manifold structure to be cannot-link. In this case, we choose to respect the pairwise constraint assumption first and then the cluster assumption, considering that the prior pairwise constraints are from reliable knowledge. This

is true in most practical applications.

3. Pairwise Constraint Propagation

3.1. A General Framework

As mentioned in the last section, our goal is to propagate the given pairwise constraints to the whole data set in a global implication for classification. Intuitively, it is hard to implement our idea in the input space. Therefore, we seek a mapping (usually non-linear) to map the objects to a new and possibly higher-dimensional space such that the objects are reshaped in this way: two must-link objects become close while two cannot-link objects become far apart; objects respectively similar to two must-link objects also become close while objects respectively similar to two cannot-link objects become far apart.

Let $\boldsymbol{\phi}$ be a mapping from $\mathcal{X}$ to some space $\mathcal{F}$,

$$\mathbf{x}_i \in \mathcal{X} \mapsto \boldsymbol{\phi}(\mathbf{x}_i) \in \mathcal{F}. \tag{1}$$

The above analysis motivates us to consider the following optimization framework:

$$\min_{\boldsymbol{\phi}} : \ \mathcal{S}(\boldsymbol{\phi}) \tag{2}$$

$$\text{s.t.} : \ \|\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j)\|_{\mathcal{F}} < \varepsilon, \ \forall(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}, \tag{3}$$

$$\|\boldsymbol{\phi}(\mathbf{x}_i) - \boldsymbol{\phi}(\mathbf{x}_j)\|_{\mathcal{F}} > \delta, \ \forall(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}, \tag{4}$$

where $\mathcal{S}(\boldsymbol{\phi})$ is a smoothness measure for $\boldsymbol{\phi}$ such that the smaller is $\mathcal{S}(\boldsymbol{\phi})$, the smoother is $\boldsymbol{\phi}$; ε is a small positive number; δ is a large positive number; $\|\cdot\|_{\mathcal{F}}$ is a distance metric in $\mathcal{F}$; $\mathcal{M}$ is the set of the must-link constraints; $\mathcal{C}$ is the set of cannot-link constraints. The inequality constraints (3) and (4) require $\boldsymbol{\phi}$ to map two must-link objects to be close and two cannot-link objects far apart. By enforcing the smoothness on $\boldsymbol{\phi}$ (minimizing the objective (2)), we actually require $\boldsymbol{\phi}$ to map any two objects respectively similar to two must-link objects to be close and map any two objects respectively similar to two cannot-link objects far apart. Hopefully, after the mapping, each class becomes relatively compact and different classes become far apart. Once such a mapping is derived, the classification task can be done much easier.

This optimization framework is quite general and the details have to be developed. We propose a unit hypersphere model to substantialize it in Section 3.2, and then solve the resulting optimization problem in Section 3.3.

3.2. The Unit Hypersphere Model

Recall that our goal is to find a smooth mapping that maps two must-link objects close and two cannot-link objects far apart. To this end, we consider it better to put the images of all the objects under a uniform scale. The unit hypersphere in $\mathcal{F}$ is a good choice because there is a natural way to impose the pairwise constraints on it. Our key idea is to map all the objects onto the unit hypersphere in $\mathcal{F}$, where two must-link objects are mapped to the same point and two cannot-link objects to be orthogonal. Mathematically, we require $\boldsymbol{\phi}$ to satisfy

$$< \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_i) >_{\mathcal{F}} = 1, \ i = 1, 2, ..., n, \tag{5}$$

$$< \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) >_{\mathcal{F}} = 1, \ \forall(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}, \tag{6}$$

$$< \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) >_{\mathcal{F}} = 0, \ \forall(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}, \tag{7}$$

where $< \cdot, \cdot >_{\mathcal{F}}$ denotes the dot product in $\mathcal{F}$.

Next we impose smoothness on $\boldsymbol{\phi}$ using the spectral graph theory where the graph Laplacian plays an essential role (Chung, 1997). Let $G = (V, W)$ be an undirected, weighted graph with the node set $V = \mathcal{X}$ and the weight matrix $W = [w_{ij}]_{n \times n}$, where w_{ij} is the weight on the edge connecting nodes $\mathbf{x}_i$ and $\mathbf{x}_j$, denoting how similar they are. W is commonly assumed to be symmetric and non-negative. The graph Laplacian L of G is defined as $L = D - W$, where $D = [d_{ij}]_{n \times n}$ is a diagonal matrix with $d_{ii} = \sum_j w_{ij}$. The normalized graph Laplacian $\bar{L}$ of G is defined as

$$\bar{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}, \tag{8}$$

where I is the identity matrix. W is also called the affinity matrix, and $\bar{W} = D^{-1/2} W D^{-1/2}$ the normalized affinity matrix. $\bar{L}$ is symmetric and positive semidefinite, with eigenvalues in the interval $[0, 2]$ (Chung, 1997).

Following the idea of regularization in spectral graph theory (e.g., see (Zhou et al., 2004)), we define the smoothness measure $\mathcal{S}(\cdot)$ by

$$\mathcal{S}(\boldsymbol{\phi}) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \| \frac{\boldsymbol{\phi}(\mathbf{x}_i)}{\sqrt{d_{ii}}} - \frac{\boldsymbol{\phi}(\mathbf{x}_j)}{\sqrt{d_{jj}}} \|_{\mathcal{F}}^2, \tag{9}$$

where $\boldsymbol{\phi}(\mathbf{x}_i) \in \mathcal{F}$, and $\|\cdot\|_{\mathcal{F}}$ is a distance metric in $\mathcal{F}$. Note that $\mathcal{F}$ is possibly an infinite-dimensional space. By this definition, we can see that $\mathcal{S}(\boldsymbol{\phi}) \geq 0$ since W is non-negative, and the value $\mathcal{S}(\boldsymbol{\phi})$ penalizes the large change of the mapping $\boldsymbol{\phi}$ between two nodes linked with a large weight. In other words, minimizing $\mathcal{S}(\cdot)$ encourages the smoothness of a mapping over the data graph. Next we rewrite $\mathcal{S}(\boldsymbol{\phi})$ in matrix form.

Let $k_{ij} =< \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) >_{\mathcal{F}}$. Then the matrix $K = [k_{ij}]_{n \times n}$ is symmetric and positive semidefinite, denoted by $K \succeq 0$, and thus can be thought as a kernel

over $\mathcal{X}$ (Smola & Kondor, 2003). From (9), we have

$$\begin{aligned}\mathcal{S}(\boldsymbol{\phi}) &= \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{1}{d_{ii}}k_{ii} + \frac{1}{d_{jj}}k_{jj} - 2\frac{1}{\sqrt{d_{ii}d_{jj}}}k_{ij}\right) \\ &= \sum_{i=1}^{n} k_{ii} - \sum_{i,j=1}^{n}\frac{w_{ij}}{\sqrt{d_{ii}d_{jj}}}k_{ij} \quad (10) \\ &= I \bullet K - (D^{-1/2}WD^{-1/2}) \bullet K \quad (11) \\ &= (I - D^{-1/2}WD^{-1/2}) \bullet K = \bar{L} \bullet K, \quad (12)\end{aligned}$$

where $\bullet$ denotes the dot product between two matrices, defined as $A \bullet B = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}b_{ij}$, for $A = [a_{ij}]_{n\times m}$ and $B = [b_{ij}]_{n\times m}$.

3.3. Learning a Kernel Matrix

With the above analysis (5)–(7), and (12), we have arrived at the following optimization problem:

$$\begin{aligned}\min_{K}: \ & \bar{L} \bullet K \quad (13) \\ \text{s.t.}: \ & k_{ii} = 1, \ i = 1, 2, ..., n, \quad (14) \\ & k_{ij} = 1, \ \forall(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}, \quad (15) \\ & k_{ij} = 0, \ \forall(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}, \quad (16) \\ & K \succeq 0, \quad (17)\end{aligned}$$

which can be recognized as a semidefinite programming (SDP) problem (Boyd & Vandenberghe, 2004). This problem is convex and thus the global optimal solution is guaranteed. To solve this problem, we can use the highly optimized software packages, such as SeDuMi (Sturm, 1999) and CSDP (Borchers, 1999).

We can also express the above SDP problem in a more familiar matrix form. Let E_{ij} be a $n \times n$ matrix consisting of all 0's except the $(i, j)^{th}$ entry being 1. Then the above SDP problem becomes

$$\begin{aligned}\min_{K}: \ & \bar{L} \bullet K \quad (18) \\ \text{s.t.}: \ & E_{ii} \bullet K = 1, \ i = 1, 2, ..., n, \quad (19) \\ & E_{ij} \bullet K = 1, \ \forall(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}, \quad (20) \\ & E_{ij} \bullet K = 0, \ \forall(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}, \quad (21) \\ & K \succeq 0. \quad (22)\end{aligned}$$

It should be noted that we have transformed the problem of learning a mapping $\boldsymbol{\phi}$ stated in (2)–(4) into the problem of learning a kernel matrix K such that $\boldsymbol{\phi}$ is the feature mapping induced by K. The kernel trick (Schölkopf & Smola, 2002) indicates that we can implicitly derive $\boldsymbol{\phi}$ by explicitly pursuing K. Note that the kernel matrix K captures the distribution of the point set $\{\boldsymbol{\phi}(\mathbf{x}_i)\}_{i=1}^{n}$ in the feature space. The equality constraints (14) constrain $\boldsymbol{\phi}(\mathbf{x}_i)$'s to be on the unit hypersphere, the inequality constraints (15) and (16) force $\boldsymbol{\phi}(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_j)$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are must-link and $\boldsymbol{\phi}(\mathbf{x}_i)$ and $\boldsymbol{\phi}(\mathbf{x}_j)$ to be orthogonal if $\mathbf{x}_i$ and $\mathbf{x}_j$ are cannot-link. By minimizing the objective function (13), which is equivalent to enforcing smoothness on $\boldsymbol{\phi}$, $\boldsymbol{\phi}(\mathbf{x}_i)$ and $\boldsymbol{\phi}(\mathbf{x}_j)$ will move close to each other if $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar (lie on the same group or manifold). This process will continue until a global stable state is achieved (the objective function is minimized and the constraints are satisfied). We call this process the *pairwise constraint propagation*. It is expected that after the propagation, each class becomes compact and different classes become far apart (being nearly orthogonal on the unit hypersphere). This phenomenon is also observed by our experiments (see Section 5.2). The idea of reshaping the data in a high-dimensional space by propagating the spatial information among objects is previously appeared in our recent work (Li et al., 2007) where the problem of clustering highly noisy data is addressed.

3.4. Classification

Let K^* be the kernel matrix obtained by solving the SDP problem stated in (18)–(22). The final step of our approach is to obtain k classes from K^*. We apply the kernel K-means algorithm (Shawe-Taylor & Cristianini, 2004) to K^* to form k classes.

4. The Algorithm

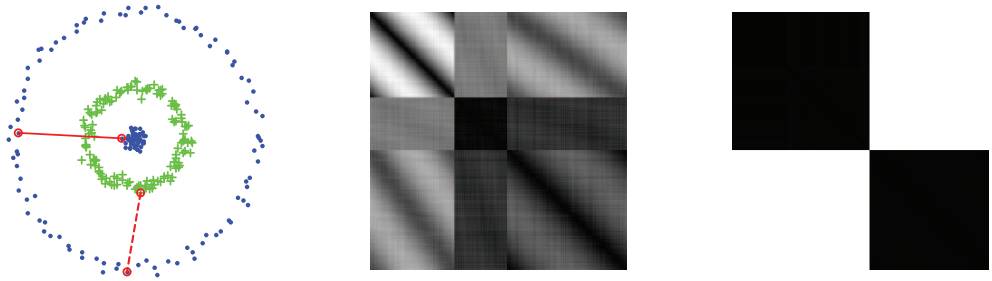
Based on the previous analysis, we develop a semi-supervised classification algorithm listed in Algorithm 1, which we called the *Pairwise Constraint Propagation* (PCP). The scale factor σ in Step 1 needs to be tuned, which is discussed in Section 5.1.

Algorithm 1 Pairwise Constraint Propagation

Input: A data set of n objects $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, the set of must-link constraints $\mathcal{M} = \{(\mathbf{x}_i, \mathbf{x}_j)\}$, the set of cannot-link constraints $\mathcal{C} = \{(\mathbf{x}_i, \mathbf{x}_j)\}$, and the number of classes k.

Output: The class labels of the objects in $\mathcal{X}$.

1. Form the affinity matrix $W = [w_{ij}]_{n\times n}$ with $w_{ij} = \exp(-d^2(\mathbf{x}_i, \mathbf{x}_j)/2\sigma^2)$ if $i \neq j$ and $w_{ii} = 0$.
2. Form the normalized graph Laplacian $\bar{L} = I - D^{-1/2}WD^{-1/2}$, where $D = \text{diag}(d_{ii})$ is the diagonal matrix with $d_{ii} = \sum_{j=1}^{n} w_{ij}$.
3. Obtain the kernel matrix K^* by solving the SDP problem stated in (18)–(22).
4. Form k classes by applying the kernel K-means to K^*.



(a) (b) (c)

Figure 2. (a) Three-Circle with classes denoted by different colors and symbols. The solid red line denotes a must-link constraint and the dashed red line denotes a cannot-link constraint. (b) & (c) Distance matrices for Three-Circle in the input space and in the feature space, respectively, where, for illustration purpose, the data are arranged such that points within a class appear consecutively. The darker is a pixel, the smaller is the distance the pixel represents.

5. Experimental Results

In this section, we evaluate the proposed algorithm PCP on a number of synthetic and real data sets. By comparison, the results of two notable and most related algorithms, Kamvar et al.'s spectral learning algorithm (SL) (Kamvar et al., 2003) and Kulis et al's semi-supervised kernel K-means algorithm (SSKK) (Kulis et al., 2005), are also presented. Note that most semi-supervised classification algorithms cannot be directly applied to the tasks of classification from pairwise constraints we consider here, because they perform classification from labeled and unlabeled data and cannot be readily generalized to address classification from pairwise constraints and unlabeled data.

In order to evaluate these algorithms, we compare the results with available ground-truth data labels, and employ the *Normalized Mutual Information* (NMI) as the performance measure (Strehl & Ghosh, 2003). For two random variables $\mathbf{X}$ and $\mathbf{Y}$, the NMI is defined as:

$$\mathrm{NMI}(\mathbf{X},\mathbf{Y}) = \frac{I(\mathbf{X},\mathbf{Y})}{\sqrt{H(\mathbf{X})H(\mathbf{Y})}}, \tag{23}$$

where $I(\mathbf{X},\mathbf{Y})$ is the mutual information between $\mathbf{X}$ and $\mathbf{Y}$, and $H(\mathbf{X})$ and $H(\mathbf{Y})$ are the entropies of $\mathbf{X}$ and $\mathbf{Y}$, respectively. Note that $0 \le \mathrm{NMI} \le 1$, and $\mathrm{NMI} = 1$ when a result is the same as the ground-truth. The larger is the NMI, the better is a result.

To evaluate the algorithms under different settings of pairwise constraints, we generate a varying number of pairwise constraints randomly for each data set. For a data set of k classes, we randomly generate j must-link constraints for each class, and j cannot-link constraints for every two classes, giving total $j \times (k + k(k-1)/2)$ constraints for each j, where j ranges from 1 to 10. The averaged NMI is reported for each number of pairwise constraints over 20 different realizations. Since all the three algorithms employ the K-means or kernel K-means in the final step, for each experiment we run the K-means or kernel K-means 20 times with random initializations, and report the averaged result.

5.1. Parameter Selection

The three algorithms are all graph-based and thus the inputs are assumed to be graphs. We use the weighted graphs for all the algorithms, where the similarity matrix $W = [w_{ij}]$ is given by

$$w_{ij} = \begin{cases} e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2} & i \neq j \\ 0 & i = j \end{cases}. \tag{24}$$

The most suitable scale factor σ is found over the set $S(0.1r, r, 5) \bigcup S(r, 10r, 5)$, where $S(r_1, r_2, m)$ denotes the set of m linearly equally spaced numbers between r_1 and r_1, and r denotes the averaged distance from each node to its 10-th nearest neighbor.

We use the SDP solver CSDP 6.0.1[1] (Borchers, 1999) to solve the SDP problem in the proposed PCP. For SSKK, we use its normalized cut version since it performs best in the experiments given in (Kulis et al., 2005). The constraint penalty in SSKK is set to $n/(kc)$, as suggested in (Kulis et al., 2005), where n is the number of objects, k is the number of classes, and c is the total number of pairwise constraints. All the algorithms are implemented in MATLAB 7.6, running on a 3.4 GHz, 2GB RAM Pentium IV PC.

5.2. A Toy Example

In this subsection, we illustrate the proposed PCP using a toy example. We mainly study its capability of propagating pairwise constraints to the whole data

[1] https://projects.coin-or.org/Csdp/.

Table 1. Description of the four sensory data sets from UCI.

Data	Iris	Wine	Ionosphere	Soybean
Number of objects	150	178	351	47
Dimension	4	13	34	35
Number of classes	3	3	2	4

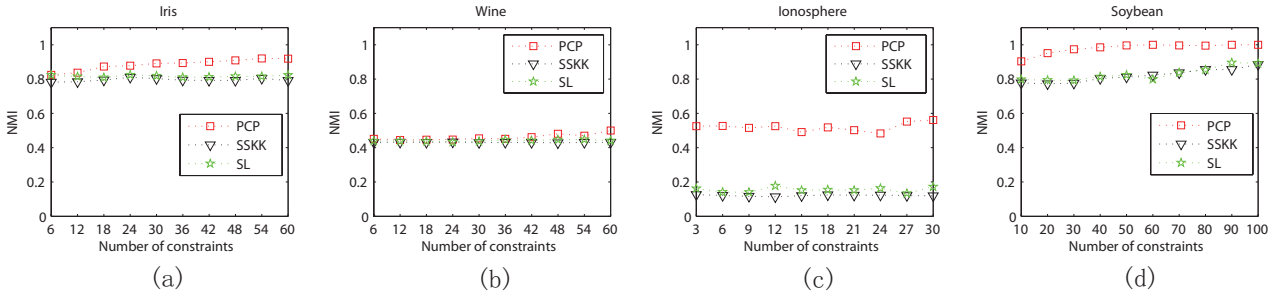


Figure 3. Classification results on the four sensory data sets: NMI vs. Number of constraints. (a) Results on Iris. (b) Results on Wine. (c) Results on Ionosphere. (d) Results on Soybean.

set. Specifically, we want to see how PCP reshapes the data in the feature space according to the original data structure and the given pairwise constraints. The classification task on the Three-Circle data is shown in Fig. 2(a), where the ground-truth classes are denoted by different colors and symbols, and one must-link (solid red line) and one cannot-link (dashed red line) constraints are also provided. At first glance, Three-Circle is composed of a mixture of curve-like and Gaussian-like groups. A more detailed observation is that there is one class composed of separate groups.

The distance matrices for Three-Circle in the input space and in the feature space are shown in Figs. 2(b) and (c), where the data are ordered such that all the objects in the outer circle appear first, all the objects in the inner circle appear second, and all the objects in the middle circle appear finally. Note that this arrangement does not affect the classification results but only for better illustration of the distance matrices. We can see that the distance matrix in the feature space, compared to the one in the input space, exhibits a clear block structure, meaning that each class becomes highly compact (although in the input space one of the two classes consists of two well-separated groups) and the two classes become far apart. Our computations show that the distance between any two points in the feature space in different classes falls in $[\sqrt{2} - 1.9262 \times 10^{-5}, \sqrt{2} + 1.3214 \times 10^{-6}]$, implying that the two classes are nearly $\sqrt{2}$ from each other, which comes from the requirement that two cannot-link objects are mapped to be orthogonal on the unit hypersphere.

5.3. On Sensory Data

Four sensory data sets from the UCI Machine Learning Repository[2] are used for testing in this experiment. The data sets are described in Table 1. These four data sets are widely used for evaluation of the classification and clustering methods in the machine learning community.

The results are shown in Fig. 3, from which two observations can be drawn. First, PCP performs better than SSKK and SL on all the four data sets under different settings of pairwise constraints, especially on the Ionosphere data. Second, as the number of constraints grows, the performances of all the algorithms increase accordingly on Soybean, but vary little on Wine and Ionosphere. On Iris, as the number of constraints grows the performance of PCP improves accordingly but those of SSKK and SL are almost unchanged.

5.4. On Imagery Data

In this subsection, we test the algorithms on three image databases USPS, MNIST, and CMU PIE (Pose, Illumination, and Expression). Both USPS and MNIST consist of images of handwritten digits with significantly different fashion and of sizes 16×16 and 28×28. The CMU PIE contains 41,368 images of 68 people, each person with 13 different poses, 43 different illumination conditions, and 4 different expressions. From these databases, we draw four data sets, which are described in Table 2. The USPS0123 and MNIST0123 are drawn respectively from USPS and MNIST, and PIE-10-20 and PIE-22-23 are drawn from CMU PIE.

[2] http://archive.ics.uci.edu/ml.

Table 2. Description of the four imagery data sets.

Data	USPS0123	MNIST0123	PIE-10-20	PIE-22-23
Number of objects	400	400	340	340
Dimension	256	784	1024	1024
Number of classes	4	4	2	2

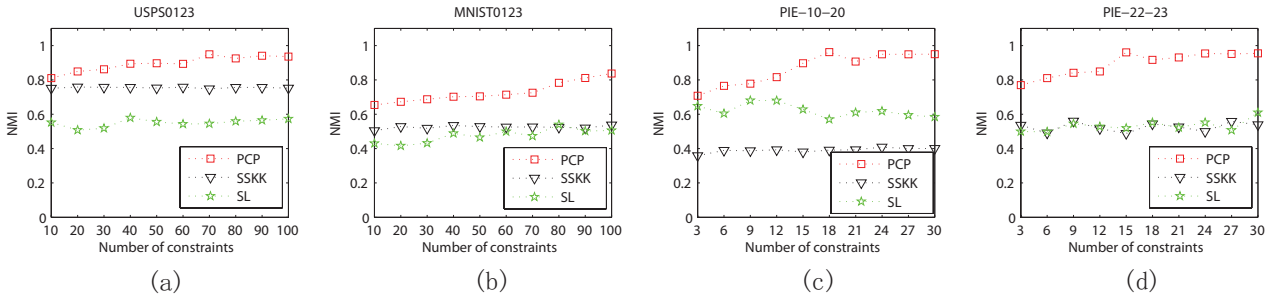


Figure 4. Classification results on the four imagery data sets: NMI vs. Number of constraints. (a) Results on USPS0123. (b) Results on MNIST0123. (c) Results on PIE-10-20. (d) Results on PIE-22-23.

USPS0123 (MNIST0123) consists of digits 0 to 3, with first 100 instances from each class. PIE-10-20 (PIE-22-23) contains five near frontal poses (C05, C07, C09, C27, C29) of two individuals indexed as 10 and 20 (22 and 23) under different illuminations and expressions. Original images in PIE-10-20 and PIE-22-23 are manually aligned (two eyes are aligned at the fixed positions), cropped, and then down-sampled to 32×32. Each image is represented by a vector of size equal to the product of its width and height.

The results are shown in Fig. 4, from which we can see that the proposed PCP consistently and significantly outperforms SSKK and SL on all the four data sets under different settings of pairwise constraints. As the number of constraints grows, the performance of PCP improves more significantly than those of SSKK and SL.

We also look at the computational costs of different algorithms. For example, for each run on USPS0123 (of size 400) with 100 pairwise constraints, PCP takes about 17 seconds while both SSKK and SL take less than 0.5 second. PCP does take more execution time than SSKK and SL since it involves solving for a kernel matrix with SDP, while either SSKK or SL uses predefined kernel matrix. The main computational cost in PCP is in solving the SDP problem.

6. Conclusions

A semi-supervised classification approach, *Pairwise Constraint Propagation* (PCP), for learning from pairwise constraints and unlabeled data is proposed. PCP seeks a smooth mapping to map the data onto a unit hypersphere, where any two must-link objects are mapped to the same point and any two cannot-link objects are mapped to be orthogonal. Consequently, PCP simultaneously implements the cluster assumption and the pairwise constraint assumption stated in Section 2. PCP implicitly derives such a mapping by explicitly finding a kernel matrix via semidefinite programming. In contrast to label propagation in traditional semi-supervised learning, PCP can effectively propagate pairwise constraints to the whole data set. Experimental results on a variety of synthetic and real data sets have demonstrated the superiority of PCP.

Note that PCP falls into semi-supervised learning since it performs learning from both constrained and unconstrained data. Most previous metric learning methods, however, belong to supervised learning. PCP always keeps every two must-link objects close and every two cannot-link objects far apart. Therefore it essentially addresses hard constrained classification.

Although extensive experiments have confirmed the effectiveness of the PCP algorithm, there are several issues worthy to be further investigated in future work. One issue is to accelerate PCP where solving the associated SDP problem is the bottleneck. Another issue is to handle noisy constraints effectively.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong SAR, China (Project No. CUHK 414306).

References

Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall, D. (2003). Learning distance functions using equivalence relations. *ICML* (pp. 11–18).

Basu, S., Bilenko, M., & Mooney, R. (2004). A probabilistic framework for semi-supervised clustering. *SIGKDD* (pp. 59–68).

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, *7*, 2399–2434.

Bilenko, M., Basu, S., & Mooney, R. (2004). Integrating constraints and metric learning in semi-supervised clustering. *ICML*.

Borchers, B. (1999). CSDP, a C library for semidefinite programming. *Optimization Methods & Software*, *11-2*, 613–623.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. MIT Press.

Chapelle, O., & Zien, A. (2005). Semi-supervised classification by low density separation. *AISTATS*.

Chung, F. (1997). *Spectral graph theory*. American Mathematical Society.

Globerson, A., & Roweis, S. (2006). Metric learning by collapsing classes. *Advances in Neural Information Processing Systems* (pp. 451–458).

Goldberg, A., Zhu, X., & Wright, S. (2007). Dissimilarity in graph-based semisupervised classification. *AISTATS*.

Hoi, S., Jin, R., & Lyu, M. (2007). Learning nonparametric kernel matrices from pairwise constraints. *ICML* (pp. 361–368).

Kamvar, S., Klein, D., & Manning, C. (2003). Spectral learning. *IJCAI* (pp. 561–566).

Klein, D., Kamvar, S., & Manning, C. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. *ICML* (pp. 307–314).

Kulis, B., Basu, S., Dhillon, I., & Mooney, R. (2005). Semi-supervised graph clustering: A kernel approach. *ICML* (pp. 457–464).

Li, Z., Liu, J., Chen, S., & Tang, X. (2007). Noise robust spectral clustering. *ICCV*.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge University Press.

Smola, A., & Kondor, R. (2003). Kernels and regularization on graphs. *COLT*.

Strehl, A., & Ghosh, J. (2003). Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, *3*, 583–617.

Sturm, J. F. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods & Software*, *11-2*, 625–653.

Szummer, M., Jaakkola, T., & Cambridge, M. (2002). Partially labeled classification with markov random walks. *Advances in Neural Information Processing Systems*.

Tong, W., & Jin, R. (2007). Semi-supervised learning by mixed label propagation. *AAAI*.

Wagstaff, K., & Cardie, C. (2000). Clustering with instance-level constraints. *ICML* (pp. 1103–1110).

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained k-means clustering with background knowledge. *ICML* (pp. 577–584).

Xing, E., Ng, A., Jordan, M., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems* (pp. 505–512).

Zhang, T., & Ando, R. (2006). Analysis of spectral kernel design based semi-supervised learning. *Advances in Neural Information Processing Systems* (pp. 1601–1608).

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. *Advances in Neural Information Processing Systems*.

Zhu, X. (2005). *Semi-supervised learning literature survey* (Technical Report 1530). Computer Sciences, University of Wisconsin-Madison.

Zhu, X., Ghahramani, Z., & Lafferty, J. Semi-supervised learning using gaussian fields and harmonic functions. *ICML* (pp. 912–919).

An Asymptotic Analysis of Generative, Discriminative, and Pseudolikelihood Estimators

Percy Liang PLIANG@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, CA, USA

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics, University of California, Berkeley, CA, USA

Abstract

Statistical and computational concerns have motivated parameter estimators based on various forms of likelihood, e.g., joint, conditional, and pseudolikelihood. In this paper, we present a unified framework for studying these estimators, which allows us to compare their relative (statistical) efficiencies. Our asymptotic analysis suggests that modeling more of the data tends to reduce variance, but at the cost of being more sensitive to model misspecification. We present experiments validating our analysis.

1. Introduction

Probabilistic models play a prominent role in domains such as natural language processing, bioinformatics, and computer vision, where they provide methods for jointly reasoning about many interdependent variables. For prediction tasks, one generally models a conditional distribution over outputs given an input. There can be reasons, however, for pursuing alternatives to conditional modeling. First, we might be able to leverage additional statistical strength present in the input by using generative methods rather than discriminative ones. Second, the exact inference required for a full conditional likelihood could be intractable; in this case, one might turn to computationally more efficient alternatives such as pseudolikelihood (Besag, 1975).

The generative-discriminative distinction has received much attention in machine learning. The standing intuition is that while discriminative methods achieve lower asymptotic error, generative methods might be better when training data are limited. This intuition is supported by the theoretical comparison of Naive Bayes and logistic regression (Ng & Jordan, 2002) and the recent empirical success of hybrid methods (McCallum et al., 2006; Lasserre et al., 2006).

Computational concerns have also spurred the development of alternatives to the full likelihood; these methods can be seen as optimizing an alternate objective or performing approximate inference during optimization. Examples include pseudolikelihood (Besag, 1975), composite likelihood (Lindsay, 1988), tree-reweighted belief propagation (Wainwright et al., 2003), piecewise training (Sutton & McCallum, 2005), agreement-based learning (Liang et al., 2008), and many others (Varin, 2008).

We can think of all these schemes as simply different estimators operating in a single model family. In this work, we analyze the statistical properties of a class of convex composite likelihood estimators for exponential families, which contains the generative, discriminative, and pseudolikelihood estimators as special cases.

The main focus of our analysis is on prediction error. Standard tools from learning theory based on uniform convergence typically only provide upper bounds on this quantity. Moreover, they generally express estimation error in terms of the overall complexity of the model family.[1] In our case, since all estimators operate in the same model family, these tools are inadequate for comparing different estimators.

Instead, we turn to asymptotic analysis, a mainstay of theoretical statistics. There is much relevant statistical work on the estimators that we treat; note in particular that Lindsay (1988) used asymptotic arguments to show that composite likelihoods are generally

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] There are more advanced techniques such as local Rademacher complexities, which focus on the relevant regions of the model family, but these typically only apply to empirical risk minimization.

less efficient than the joint likelihood. The majority of these results are, however, focused on parameter estimation. In the current paper, our focus is on prediction, and we also consider model misspecification.

We draw two main conclusions from our analysis: First, when the model is well-specified, conditioning on fewer variables increases statistical efficiency; this to some extent accounts for the better generalization enjoyed by generative estimators and the worse performance of pseudolikelihood estimators. Second, model misspecification can severely increase both the approximation and estimation errors of generative estimators. We confirm our theoretical results by comparing our three estimators on a toy example to verify the asymptotics and on a Markov model for part-of-speech tagging.

2. Exponential Family Estimators

In structured prediction tasks, we are interested in learning a mapping from an input space $\mathcal{X}$ to an output space $\mathcal{Y}$. Probabilistic modeling is a common platform for solving such tasks, allowing for the natural handling of missing data and the incorporation of latent variables.

In this paper, we focus on regular exponential families, which define distributions over an outcome space $\mathcal{Z}$ as follows:

$$p_\theta(z) \stackrel{\text{def}}{=} \exp\{\phi(z)^\top \theta - A(\theta)\} \text{ for } z \in \mathcal{Z}, \tag{1}$$

where $\phi(z) \in \mathbb{R}^d$ is a vector of sufficient statistics (features), $\theta \in \mathbb{R}^d$ is a vector of parameters, and $A(\theta) \stackrel{\text{def}}{=} \log \int e^{\phi(z)^\top \theta} \nu(dz)$ is the log-partition function. In our case, the outcomes are input-output pairs: $z = (x, y)$ and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.

Exponential families include a wide range of popular models used in machine learning. For example, for a conditional random field (CRF) (Lafferty et al., 2001) defined on a graph $G = (V, E)$, we have an output variable for each node ($y = \{y_i\}_{i \in V}$), and the features are $\phi(x, y) = \sum_{i \in V} \phi_{\text{node}}(y_i, \mathbf{x}, i) + \sum_{(i,j) \in E} \phi_{\text{edge}}(y_i, y_j)$.

From the density $p_\theta(z)$, we can compute event probabilities as follows:

$$p_\theta(z \in s) = \exp\{A(\theta; s) - A(\theta)\}, \tag{2}$$

where $A(\theta; s) = \log \int e^{\phi(z)^\top \theta} \mathbb{1}[z \in s] \nu(dz)$ is a conditional log-partition function.

2.1. Composite Likelihood Estimators

In this paper, we consider a class of *composite likelihood* estimators (Lindsay, 1988), which is incidentally equivalent to the multi-conditional learning framework of McCallum et al. (2006). A *composite likelihood* consists of a weighted sum of *component* likelihoods, each of which is the probability of one subset of variables conditioned on another. In this work, we only consider the case where the first set is all the variables.

We adopt the following more fundamental way of specifying the components: Each component r is defined by a *partitioning* of the outcome space $\mathcal{Z}$. We represent a partitioning by an associated *equivalence function* that maps each $z \in \mathcal{Z}$ to its partition:

Definition 1 (Equivalence function)**.** *An* equivalence function r *is a measurable map from* $\mathcal{Z}$ *to measurable subsets of* $\mathcal{Z}$ *such that for each* $z \in \mathcal{Z}$ *and* $z' \in r(z)$, $r(z) = r(z')$.

The component likelihood associated with r takes the following form:

$$p_\theta(z \mid z \in r(z)) = \exp\{\phi(z)^\top \theta - A(\theta; r(z))\}. \tag{3}$$

By maximizing this quantity, we are intuitively taking probability mass away from some neighborhood $r(z)$ of z and putting it on z.

Without loss of generality, assume the component weights sum to 1, so we can think of taking an expectation over a random component R drawn from some fixed distribution P_{r}. We then define the *criterion function*:

$$m_\theta(z) \stackrel{\text{def}}{=} \mathbb{E}_{R \sim P_{\text{r}}} \log p_\theta(z \mid z \in R(z)). \tag{4}$$

Given data points $Z^{(1)}, \ldots, Z^{(n)}$ drawn i.i.d. from some true distribution p^* (not necessarily in the exponential family), the maximum composite likelihood estimator is defined by averaging the criterion function over these data points:

$$\hat{\theta} = \operatorname*{argmax}_\theta \hat{\mathbb{E}} m_\theta(Z), \tag{5}$$

where $\hat{\mathbb{E}} m_\theta(Z) = \frac{1}{n} \sum_{i=1}^n m_\theta(Z^{(i)})$.

We can now place the three estimators of interest in our framework:

Generative: We have one component $r_{\text{g}}(x, y) = \mathcal{X} \times \mathcal{Y}$, which has one partition—the whole outcome space.

Fully discriminative: We have one component $r_{\text{d}}(x, y) = x \times \mathcal{Y}$. The outcomes in each partition have the same value of x, but different y.

Pseudolikelihood discriminative: Assume $y = \{y_i\}_{i \in V}$. For each $i \in V$, we have a component $r_i(x, y) = \{(x', y') : x' = x, y' \in \mathcal{Y}, y'_j = y_j \text{ for } j \neq i\}$. P_{r} is uniform over these components.

$v^{\otimes} = vv^{\top}$	
Parameter estimates	
p^*	true distribution of the data
m_θ	criterion function (defines the estimator)
$\mathcal{R}$	risk (expected log-loss)
$\hat{\theta}$	$= \operatorname{argmax}_\theta \hat{\mathbb{E}} m_\theta(Z)$ [empirical parameter estimate]
θ°	$= \operatorname{argmax}_\theta \mathbb{E} m_\theta(Z)$ [limiting parameter vector]
Random variables for asymptotic analysis	
R	$\sim P_{\mathrm{r}}$ [choose composite likelihood component]
$r_{\mathrm{d}}(x, y) = x \times \mathcal{Y}$	[fully discriminative component]
ϕ	$= \phi(Z)$, $Z \sim p^*$ [sample from true distribution]
ϕ^{t}	$= \phi(Z^{\mathrm{t}})$, $Z^{\mathrm{t}} \sim p^*(\cdot \mid \cdot \in R(Z))$ [from true distrib.]
ϕ^{m}	$= \phi(Z^{\mathrm{m}})$, $Z^{\mathrm{m}} \sim p_{\theta^\circ}(\cdot \mid \cdot \in R(Z))$ [for estimation]
ϕ^{e}	$= \phi(Z^{\mathrm{e}})$, $Z^{\mathrm{e}} \sim p_{\theta^\circ}(\cdot \mid \cdot \in r_{\mathrm{d}}(Z))$ [for prediction]

Table 1. Notation used in the paper.

2.2. Prediction and Evaluation

Given a parameter estimate $\hat{\theta}$, we make predictions based on $p_{\hat{\theta}}(y \mid x)$. In this paper, we evaluate our model according to log-loss; the risk is the expected log-loss:

$$\mathcal{R}(\theta) = \mathbb{E}_{(X,Y)\sim p^*}[-\log p_\theta(Y \mid X)]. \tag{6}$$

The quality of an estimator is determined by the gap between the risk of the estimate $\mathcal{R}(\hat{\theta})$ and the Bayes risk $\mathcal{R}^* = H(Y \mid X)$. It will be useful to relate these two via the risk of $\theta^\circ = \operatorname{argmax}_\theta \mathbb{E}_{Z\sim p^*} m_\theta(Z)$, which leads to the following standard decomposition:

$$\underbrace{\mathcal{R}(\hat{\theta}) - \mathcal{R}^*}_{\text{total error}} = \underbrace{(\mathcal{R}(\hat{\theta}) - \mathcal{R}(\theta^\circ))}_{\text{estimation error}} + \underbrace{(\mathcal{R}(\theta^\circ) - \mathcal{R}^*)}_{\text{approx. error}}. \tag{7}$$

The estimation error is due to having only finite data; the approximation error is due to the intrinsic suboptimality of the estimator.[2]

3. Asymptotic Analysis

We first compute the asymptotic estimation errors of composite likelihood estimators in general (Sections 3.1 and 3.2). Then we use these results to compare the estimators of interest (Sections 3.3 and 3.4).

In this paper, we assume that our exponential family is identifiable.[3] Also assume that our estimators converge ($\hat{\theta} \xrightarrow{P} \theta^\circ$) and are consistent when the model is well-specified (if $p^* = p_{\theta^*}$, then $\theta^\circ = \theta^*$). Note, however, that in general we do not assume that our model is well-specified.

Our asymptotic analysis is driven by Taylor expansions, so we need to compute a few derivatives. The derivatives of the log-partition function are moments of the sufficient statistics (a standard result, see Wainwright and Jordan (2003)):

$$\dot{A}(\theta; s) = \mathbb{E}_{Z\sim p_\theta(\cdot\mid\cdot\in s)}(\phi(Z)) \tag{8}$$

$$\ddot{A}(\theta; s) = \operatorname{var}_{Z\sim p_\theta(\cdot\mid\cdot\in s)}(\phi(Z)). \tag{9}$$

From these moments, we can obtain the derivatives of m_{θ° and $\mathcal{R}$ (to simplify notation, we express these in terms of random variables whose distributions are defined in Table 1):

$$\dot{m}_{\theta^\circ} = \phi - \mathbb{E}(\phi^{\mathrm{m}} \mid Z) \tag{10}$$

$$\ddot{m}_{\theta^\circ} = -\mathbb{E}[\operatorname{var}(\phi^{\mathrm{m}} \mid R(Z)) \mid Z] \tag{11}$$

$$\dot{\mathcal{R}}(\theta^\circ) = \mathbb{E}(\phi^{\mathrm{e}} - \phi) \tag{12}$$

$$\ddot{\mathcal{R}}(\theta^\circ) = \mathbb{E}\operatorname{var}(\phi^{\mathrm{e}} \mid Z). \tag{13}$$

3.1. Asymptotics of the Parameters

We first analyze how fast $\hat{\theta}$ converges to θ° by computing the asymptotic distribution of $\hat{\theta} - \theta^\circ$. In Section 3.2 we use this result to get the asymptotic distribution of the estimation error $\mathcal{R}(\hat{\theta}) - \mathcal{R}(\theta^\circ)$.

The following standard lemma will prove to be very useful in our analysis:

Lemma 1. *For random vectors X, Y, Z, we have* $\operatorname{var}(X \mid Z) = \mathbb{E}[\operatorname{var}(X \mid Y, Z) \mid Z] + \operatorname{var}[\mathbb{E}(X \mid Y, Z) \mid Z]$.

The important implication of this lemma is that conditioning on another variable Y reduces the variance of X. This lemma already hints at how conditioning on more variables can lead to poorer estimators: conditioning reduces the variance of the data, which can make it harder to learn about the parameters.

The following theorem gives us the asymptotic variance of a general composite likelihood estimator:

Theorem 1 (Asymptotic distribution of the parameters)**.** *Assume $\hat{\theta} \xrightarrow{P} \theta^\circ$. Then*

$$\sqrt{n}(\hat{\theta} - \theta^\circ) \to \mathcal{N}(0, \Sigma). \tag{14}$$

The asymptotic variance is

$$\Sigma = \Gamma^{-1} + \Gamma^{-1}(C_{\mathrm{c}} + C_{\mathrm{m}})\Gamma^{-1}, \tag{15}$$

where $\Gamma = \mathbb{E}\operatorname{var}(\phi^{\mathrm{m}} \mid R(Z))$ is the sensitivity, $C_{\mathrm{c}} = \mathbb{E}\operatorname{var}[\mathbb{E}(\phi^{\mathrm{m}} \mid R(Z)) \mid Z]$ is the component correction, and $C_{\mathrm{m}} = \mathbb{E}[\operatorname{var}(\phi^{\mathrm{t}} \mid Z) - \operatorname{var}(\phi^{\mathrm{m}} \mid Z)] + \mathbb{E}[\mathbb{E}(\phi^{\mathrm{t}} \mid Z) - \mathbb{E}(\phi^{\mathrm{m}} \mid Z)]^{\otimes}$ is the misspecification correction.

[2] Note that θ° is not necessarily the minimum risk parameter vector in the model family.

[3] In the non-identifiable case, the analysis becomes more cluttered, but the results are essentially the same, since predictions depend on only the distributions induced by the parameters. See the longer version of this paper for an in-depth discussion.

Proof. The standard asymptotic normality result for M-estimators (Theorem 5.21 of van der Vaart (1998)), which includes composite likelihood estimators, gives us the asymptotic variance:

$$\Sigma = (\mathbb{E}\ddot{m}_{\theta^\circ})^{-1}(\mathbb{E}\dot{m}_{\theta^\circ}^{\otimes})(\mathbb{E}\ddot{m}_{\theta^\circ})^{-1}. \tag{16}$$

The remainder of the proof simply re-expresses Σ in terms of more interpretable quantities. Algebraic manipulation of (10) yields:

$$\mathbb{E}\dot{m}_{\theta^\circ}^{\otimes} = \mathbb{E}[(\phi - \mathbb{E}(\phi^{\mathrm{t}} \mid Z)) + (\mathbb{E}(\phi^{\mathrm{t}} \mid Z) - \mathbb{E}(\phi^{\mathrm{m}} \mid Z))]^{\otimes}.$$

Note that cross terms cancel conditioned on Z and that $\mathbb{E}[\phi - \mathbb{E}(\phi^{\mathrm{t}} \mid Z)]^{\otimes} = \mathbb{E}[\phi^{\mathrm{t}} - \mathbb{E}(\phi^{\mathrm{t}} \mid Z)]^{\otimes}$, so

$$\mathbb{E}\dot{m}_{\theta^\circ}^{\otimes} = C_{\mathrm{m}} + \mathbb{E}\,\mathrm{var}(\phi^{\mathrm{m}} \mid Z). \tag{17}$$

We then apply Lemma 1 to decompose the second term of the right-hand side:

$$\mathbb{E}\,\mathrm{var}(\phi^{\mathrm{m}} \mid Z) = \mathbb{E}\,\mathrm{var}(\phi^{\mathrm{m}} \mid R(Z)) + \mathbb{E}\,\mathrm{var}[\mathbb{E}(\phi^{\mathrm{m}} \mid R(Z)) \mid Z]. \tag{18}$$

Substitute (18) into (17) to get an expression for $\mathbb{E}\dot{m}_{\theta^\circ}^{\otimes}$; (11) already provides one for $\mathbb{E}\ddot{m}_{\theta^\circ}$. Substitute these two expressions into (16) and simplify to get (15). □

The decomposition in (15) allows us to make several qualitative judgments. First, the *sensitivity* $\Gamma = \mathbb{E}\,\mathrm{var}(\phi^{\mathrm{m}} \mid R(Z))$ is the expected amount of variation in the features given Z and R (equivalently, given $R(Z)$). The larger the sensitivity, the more the data can tell us about the parameters, and thus the lower the asymptotic variance will be.

The *component correction* C_{c} intuitively measures how different the feature expectations $\mathbb{E}(\phi^{\mathrm{m}} \mid R(Z))$ under the various components are. C_{c} is zero for the generative and fully discriminative estimators, but the pseudolikelihood discriminative estimator pays a penalty for having more than one component.

The *misspecification correction* C_{m} is zero when the model is well-specified (in this case, $\phi^{\mathrm{m}} \mid Z \stackrel{d}{=} \phi^{\mathrm{t}} \mid Z$), but is in general nonzero under model misspecification. In this latter case, one incurs a nonzero approximation error (defined in (7)) as expected, but we see that there is also a nonzero effect on estimation error.

3.2. Asymptotics of the Risk

The following theorem turns Theorem 1 from a statement about the asymptotic distribution of the parameters into one about the risk:

Theorem 2 (Asymptotic distribution of the risk)**.** *Let Σ be the asymptotic variance as defined in (15). Denote $\dot{\mathcal{R}} \stackrel{\mathrm{def}}{=} \dot{\mathcal{R}}(\theta^\circ)$ and $\ddot{\mathcal{R}} \stackrel{\mathrm{def}}{=} \ddot{\mathcal{R}}(\theta^\circ)$. Then*

$$\sqrt{n}(\mathcal{R}(\hat{\theta}) - \mathcal{R}(\theta^\circ)) \xrightarrow{d} \mathcal{N}\left(0, \dot{\mathcal{R}}^\top \Sigma \dot{\mathcal{R}}\right). \tag{19}$$

Furthermore, if $\dot{\mathcal{R}} = 0$, then

$$n(\mathcal{R}(\hat{\theta}) - \mathcal{R}(\theta^\circ)) \xrightarrow{d} \frac{1}{2}\mathrm{tr}\,\mathcal{W}\left(\ddot{\mathcal{R}}^{\frac{1}{2}} \Sigma \ddot{\mathcal{R}}^{\frac{1}{2}}, 1\right), \tag{20}$$

where $\mathcal{W}(V, n)$ is the Wishart distribution with n degrees of freedom.

Proof. Perform a Taylor expansion of the risk function around θ°:

$$\mathcal{R}(\hat{\theta}) = \mathcal{R}(\theta^\circ) + \dot{\mathcal{R}}^\top(\hat{\theta} - \theta^\circ) + \frac{1}{2}(\hat{\theta} - \theta^\circ)^\top \ddot{\mathcal{R}}(\hat{\theta} - \theta^\circ) + o(||\hat{\theta} - \theta^\circ||^2). \tag{21}$$

We use a standard argument known as the delta method (van der Vaart, 1998). Multiplying (21) on both sides by $\sqrt{n}$, rearranging terms, and applying Slutsky's theorem, we get (19). However, when $\dot{\mathcal{R}} = 0$, the first-order term of the expansion (21) is zero, so we must consider the second-order term to get a non-degenerate distribution. Note that $\ddot{\mathcal{R}}$ is positive semidefinite. Multiplying (21) by n and rearranging yields the following:

$$n(\mathcal{R}(\hat{\theta}) - \mathcal{R}(\theta^\circ)) = \frac{1}{2}\mathrm{tr}\left([\ddot{\mathcal{R}}^{\frac{1}{2}}\sqrt{n}(\hat{\theta} - \theta^\circ)]^{\otimes}\right) + \cdots$$

Since $\ddot{\mathcal{R}}^{\frac{1}{2}}\sqrt{n}(\hat{\theta} - \theta^\circ) \xrightarrow{d} \mathcal{N}(0, \ddot{\mathcal{R}}^{\frac{1}{2}}\Sigma\ddot{\mathcal{R}}^{\frac{1}{2}})$, applying the continuous mapping theorem with the outer product function yields a Wishart as the limiting distribution. Thus, $n(\mathcal{R}(\hat{\theta}) - \mathcal{R}(\theta^\circ))$ is asymptotically equal in distribution to $\frac{1}{2}$ times the trace of a sample from that Wishart distribution. □

We can also understand (20) in the following way. Let $V = \ddot{\mathcal{R}}^{\frac{1}{2}}\Sigma\ddot{\mathcal{R}}^{\frac{1}{2}}$. Note that $\frac{1}{2}\mathrm{tr}\,\mathcal{W}(V, 1) \stackrel{d}{=} \frac{1}{2}\mathrm{tr}\,(V\mathcal{W}(I, 1))$, which is the distribution of a weighted sum of independent χ_1^2 variables, where the weights are determined by the diagonal elements of V. The mean of this distribution is $\frac{1}{2}\mathrm{tr}(V)$ and the variance is $\mathrm{tr}(V \bullet V)$, where $\bullet$ denotes elementwise product.

An important question is when we obtain the ordinary $O(n^{-\frac{1}{2}})$ convergence (19) versus the much better $O(n^{-1})$ convergence (20). A sufficient condition for $O(n^{-1})$ convergence is $\dot{\mathcal{R}}(\theta^\circ) = 0$. When the model is well-specified, this is true for any consistent estimator.

Even if the model is misspecified, the fully discriminative estimator still achieves the $O(n^{-1})$ rate. The

reason is that whenever a training criterion m_θ is the same (up to constants) as the test criterion $\mathcal{R}(\cdot)$, $\dot{\mathcal{R}}$ vanishes and we obtain the $O(n^{-1})$ rate. This is in concordance with a related observation made by Wainwright (2006) that it is better to use the same inference procedure at both training and test time.

When the model is well-specified, there is another appealing property that holds if the training and test criterion are the same up to constants: the asymptotic distribution of the risk depends on only the dimensionality of the exponential family, not the actual structure of the model. In particular, for composite likelihood estimators with one component, $\Sigma = \Gamma^{-1} = (-\mathbb{E}\ddot{m}_{\theta^\circ})^{-1} = \ddot{\mathcal{R}}^{-1}$. Therefore, $\ddot{\mathcal{R}}^{\frac{1}{2}}\Sigma\ddot{\mathcal{R}}^{\frac{1}{2}} = I_d$ and so $n(\mathcal{R}(\hat{\theta}) - \mathcal{R}(\theta^\circ)) \xrightarrow{d} \frac{1}{2}\mathrm{tr}\mathcal{W}(I_d, 1) \stackrel{d}{=} \frac{1}{2}\chi^2_d$, where d is the number of parameters. This result is essentially another way of looking at the fact that the likelihood ratio test statistic is asymptotically distributed as χ^2.

3.3. Comparing Estimation Errors

In the previous section, we analyzed the asymptotics of a single estimator. Now, given two estimators, we would like to be able to tell which one is better. In order to compare two estimators, it would be convenient if they converged to the same limit. In this section, we ensure this by assuming that the model is well-specified and that our estimators are consistent.

Since all parameter estimates are used in the same way for prediction, it suffices to analyze the relative efficiencies of the parameter estimates. The following theorem says that coarser partitionings of $\mathcal{Z}$ generally lead to more efficient estimators:

Theorem 3 (Asymptotic relative efficiency)**.** *Let $\hat{\theta}_1$ and $\hat{\theta}_2$ be two consistent estimators with asymptotic variances Σ_1 and Σ_2 as defined in (15). Assume that R_1 is constant ($\hat{\theta}_1$ has exactly one component) and $R_1(z) \supset R_2(z)$ for all $z \in \mathcal{Z}$. If the model is well-specified, then $\Sigma_1 \preceq \Sigma_2$ ($\hat{\theta}_1$ is no worse than $\hat{\theta}_2$).*

Proof. We first show that $\Gamma_1^{-1} \preceq \Gamma_2^{-1}$, where Γ_1 and Γ_2 are the sensitivities of the two estimators. Because the model is well-specified, $\Gamma_k = \mathbb{E}\,\mathrm{var}(\phi^{\mathrm{t}} \mid R_k(Z))$ for $k = 1, 2$. The assumption $R_1(Z) \supset R_2(Z)$ means that $R_2(Z)$ provides more information about Z than $R_1(Z)$; formally, the σ-fields satisfy $\sigma(R_1(Z)) \subset \sigma(R_2(Z))$. Thus, we can use Lemma 1 to decompose the variance: $\Gamma_1 = \mathbb{E}\,\mathrm{var}(\phi^{\mathrm{t}} \mid R_2(Z)) + \mathbb{E}\,\mathrm{var}[\mathbb{E}(\phi^{\mathrm{t}} \mid R_2(Z)) \mid R_1(Z)]$. The first term of the right-hand side is exactly Γ_2 and the second term is positive semidefinite, so $\Gamma_1 \succeq \Gamma_2$, which implies $\Gamma_1^{-1} \preceq \Gamma_2^{-1}$.

Let $C_{\mathrm{c}1}$ and $C_{\mathrm{c}2}$ be the component corrections of the two estimators. Note that $C_{\mathrm{c}1} = 0$ because the R_1 is constant, so $C_{\mathrm{c}1} \preceq C_{\mathrm{c}2}$. The misspecification corrections are both zero. Putting these results together yields the theorem. □

One might wonder if we really need R_1 to be constant. Is it not enough to just assume that $R_1(z) \supset R_2(z)$ (for some coupling of R_1 and R_2)? The answer is no, as the following counterexample shows:

Counterexample Let $\mathcal{Z} = \{1, 2, 3\}$. The general shape of the distribution is given by the single feature $\phi(1) = 1, \phi(2) = 3, \phi(3) = 2$ and a scalar parameter θ controls the peakiness of the distribution. Let the true parameter be $\theta^* = 1$. Consider two estimators: $\hat{\theta}_1$ has two components, $r_{1a} = \{\{1, 2\}, \{3\}\}$ and $r_{1b} = \{\{1\}, \{2, 3\}\}$; $\hat{\theta}_2$ also has two components, $r_{2a} = \{\{1, 2\}, \{3\}\}$ and $r_{2b} = \{\{1\}, \{2\}, \{3\}\}$.

Coupling r_{1a} with r_{2a} and r_{1b} with r_{2b}, we have $R_1(z) \supset R_2(z)$. However, we computed and found that $\Gamma_1 \approxeq 4.19$ and $\Gamma_2 \approxeq 3.15$, so $\hat{\theta}_2$ actually has lower asymptotic variance although it has finer partitionings.

To explain this, note that the contribution of r_{2b} to the criterion function is zero, so the second estimator is equivalent to just using the single component r_{2a} ($= r_{1a}$), so the first estimator actually suffers by using the additional component r_{1b}. In general, while we would still expect coarser partitionings to be better even for estimators with many components, this counterexample shows that we must exercise caution.

3.4. Comparing Estimators

Finally, we use Theorem 3 to compare the estimation and approximation errors of the generative ($\hat{\theta}_{\mathrm{g}}$), fully discriminative ($\hat{\theta}_{\mathrm{d}}$), and pseudolikelihood discriminative ($\hat{\theta}_{\mathrm{p}}$) estimators. The subscripts $\mathrm{g}, \mathrm{d}, \mathrm{p}$ will be attached to other variables to refer to the quantities associated with the corresponding estimators. In the following corollaries, we use the word "lower" loosely to mean "no more than," although in general we expect the inequality to be strict.

Corollary 1 (Generative versus fully discriminative)**.** *(1) If the model is well-specified, $\hat{\theta}_{\mathrm{g}}$ has lower asymptotic estimation error than $\hat{\theta}_{\mathrm{d}}$; both have zero approximation error. (2) If the model is misspecified, $\hat{\theta}_{\mathrm{d}}$ has lower approximation and asymptotic estimation errors than $\hat{\theta}_{\mathrm{g}}$.*

Proof. For (1), since $R_{\mathrm{d}}(z) \subset R_{\mathrm{g}}(z)$, we have $\Sigma_{\mathrm{g}} \preceq \Sigma_{\mathrm{d}}$ by Theorem 3. Zero approximation error follows from consistency. For (2), since the discriminative estimator

achieves the minimum risk in the model family, it has the lowest approximation error. Also, by Theorem 2 and the ensuing discussion, it always converges at a $O(n^{-1})$ rate, whereas the generative estimator will in general converge at a $O(n^{-\frac{1}{2}})$ rate. □

Note that there is a qualitative change of asymptotics in going from the well-specified to the misspecified scenario. This discontinuity demonstrates one weakness of asymptotic analyses: we would expect that for a very minor model misspecification, the generative estimator would still dominate the discriminative estimator for moderate sample sizes, but even a small misspecification is magnified in the asymptotic limit.

In the following toy example where the model is well-specified, we see concretely that the generative estimator has smaller asymptotic estimation error:

Example Consider a model where x and y are binary variables: $\phi(x,y)^\top\theta = \theta_0\mathbb{1}[x=0,y=1] + \theta_1\mathbb{1}[x=1,y=1]$, where the true parameters are $\theta^* = (0,0)$. We can compute $\Gamma_{\mathrm{g}} = \mathrm{var}(\phi) = \frac{1}{16}\begin{pmatrix} 3 & -1 \\ -1 & 3 \end{pmatrix}$ and $\ddot{\mathcal{R}}(\theta^*) = \Gamma_{\mathrm{d}} = \mathbb{E}\,\mathrm{var}(\phi \mid X) = \frac{1}{16}\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$. The mean asymptotic estimation error (scaled by n) of the generative estimator is $\frac{1}{2}\mathrm{tr}(\Gamma_{\mathrm{d}}\Gamma_{\mathrm{g}}^{-1}) = \frac{3}{4}$ while that of the discriminative estimator is $\frac{1}{2}\mathrm{tr}(\Gamma_{\mathrm{d}}\Gamma_{\mathrm{d}}^{-1}) = 1$.

We now show that fully discriminative estimators are statistically superior to pseudolikelihood discriminative estimators in all regimes, but of course pseudolikelihood is computationally more efficient.

Corollary 2 (Fully discriminative versus pseudolikelihood discriminative)**.** *(1) If the model is well-specified, $\hat{\theta}_{\mathrm{d}}$ has lower asymptotic estimation error than $\hat{\theta}_{\mathrm{p}}$; both have zero approximation error. (2) If the model is misspecified, $\hat{\theta}_{\mathrm{d}}$ has lower approximation and asymptotic estimation errors than $\hat{\theta}_{\mathrm{p}}$.*

Proof. For (1), since $R_{\mathrm{p}}(z) \subset R_{\mathrm{d}}(z)$, $\Sigma_{\mathrm{d}} \preceq \Sigma_{\mathrm{p}}$ by Theorem 3. Zero approximation error follows from consistency. For (2), the same arguments as the corresponding part of the proof of Corollary 1 apply. □

4. Experiments

In this section, we validate our theoretical analysis empirically. First, we evaluate the three estimators on a simple graphical model which allows us to plot the real asymptotics of the estimation error (Section 4.1). Then we show that in the non-asymptotic regime, the qualitative predictions of the asymptotic analyses are also valid (Section 4.2).

4.1. A Simple Graphical Model

Consider a four-node binary-valued graphical model where $z = (x_1, x_2, y_1, y_2)$. The true model family p^* is an Markov random field parametrized by $\theta^* = (\alpha^*, \beta^*, \gamma^*)$ as follows:

$$\phi(z)^\top\theta = \alpha\mathbb{1}[y_1 = y_2] + \beta(\mathbb{1}[x_1 = y_1] + \mathbb{1}[x_2 = y_2]) + \gamma(\mathbb{1}[x_1 = y_2] + \mathbb{1}[x_2 = y_1]).$$

To emulate misspecification, we set γ^* to be nonzero and force $\gamma = 0$ during parameter estimation.

In the first experiment, we estimated the variance (by running 10K trials) of the estimation error as we increased the number of data points. We set $\alpha^* = \beta^* = 1$ for the true model. When $\gamma^* = 0$ (the model is well-specified), Figures 1(a)–(c) show that scaling the variance by n yields a constant; this implies that all three estimators achieve $O(n^{-1})$ convergence.

When the model is misspecified with $\gamma^* = 0.5$ (Figures 1(d)–(f)), there is a sharp difference between the rates of the generative and discriminative estimators. The fully discriminative estimator still enjoys the $O(n^{-1})$ convergence; scaling by n reveals that the generative and pseudolikelihood discriminative estimators are only attaining a $O(n^{-\frac{1}{2}})$ rate as predicted by Theorem 2 (Figure 1(f)). Note that the generative estimator is affected most severely.

Figures 1(g)–(h) demonstrate the non-asymptotic impact of varying the parameters of the graphical model in terms of the total error. In (g), as we increase the amount of misspecification γ, the error increases for all estimators, but most sharply for the generative estimator. In (h), as we increase the strength of the edge potential α, the pseudolikelihood discriminative estimator suffers, the fully discriminative estimator is unaffected, and the generative estimator actually improves.

4.2. Part-of-speech Tagging

In this section, we present experiments on part-of-speech (POS) tagging. In POS tagging, the input is a sequence of words $x = (x_1, \ldots, x_\ell)$ and the output is a sequence of POS tags $y = (y_1, \ldots, y_\ell)$, e.g., noun, verb, etc. (There are 45 tags total.) We consider the following model, specified by the following features (roughly 2 million total):

$$\phi(x,y) = \sum_{i=1}^{\ell}\phi_{\mathrm{node}}(y_i, x_i) + \sum_{i=1}^{\ell-1}\phi_{\mathrm{edge}}(y_i, y_{i+1}), \quad (22)$$

where the node features $\phi_{\mathrm{node}}(y_i, x_i)$ are a vector of indicator functions of the form $\mathbb{1}[y_i = a, x_i = b]$, and

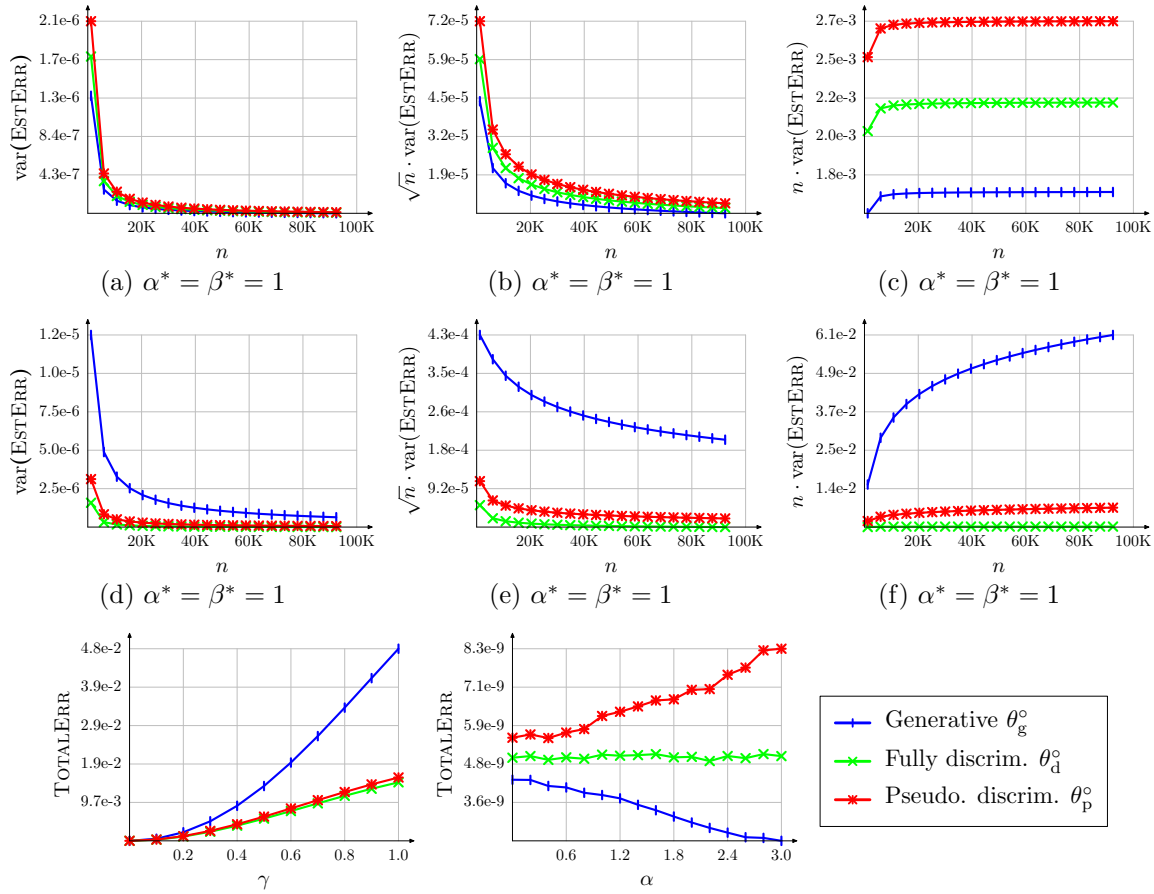


Figure 1. Asymptotics of the simple four-node graphical model. In (a)–(c), $\alpha^* = \beta^* = 1$ and $\gamma^* = 0$; we plot the asymptotic variance of the estimation error, scaled by 1, $\sqrt{n}$, and n. In (d)–(f), we repeat with $\gamma^* = 0.5$. In (g), we take $n = 20000$ examples, $\alpha^* = \beta^* = 1$ and vary γ. In (h), we take $n = 20000$, $\beta^* = 1, \gamma^* = 0$ and vary α.

the edge features $\phi_{\text{edge}}(y_i, y_{i+1})$ are a vector of indicator functions of the form $\mathbb{1}[y_i = a, y_{i+1} = b]$. Trained generatively, this model is essentially an HMM, but slightly more expressive. Trained (fully) discriminatively, this model is a CRF.

We used the Wall Street Journal (WSJ) portion of the Penn Treebank, with sections 0–21 for training (38K sentences) and 22–24 for testing (5.5K sentences). Table 2(a) shows that the discriminative estimators perform better than the generative one. This is not surprising given that the model is misspecified (language does not come from an HMM).

To verify that the generative estimator is superior when the model is well-specified, we used the learned generative model in the previous experiment to sample 1000 synthetic training and 1000 synthetic test examples. We then applied the estimators as before on this artificial data. Table 2(b) shows that the generative es-

	Accuracy		Log-loss	
	Train	Test	Train	Test
Gen.	0.940	0.935	4.628	4.945
Fully dis.	**0.977**	**0.956**	**1.480**	**3.120**
Pseudo dis.	0.975	0.955	1.562	3.170

(a) Real data (misspecified)

	Accuracy		Log-loss	
	Train	Test	Train	Test
Gen.	0.989	**0.898**	0.570	**7.297**
Full dis.	**0.992**	0.879	**0.407**	12.431
Pseudo dis.	0.990	0.891	0.469	10.840

(b) Synthetic data (well-specified)

Table 2. Part-of-speech tagging results. Discriminative estimators outperform the generative estimator (on both test accuracy and log-loss) when the model is misspecified, but the reverse is true when the model is well-specified.

timator has an advantage over the fully discriminative estimator, and both are better than the pseudolikelihood estimator.

5. Discussion and Extensions

We believe our analysis captures the essence of the generative-discriminative distinction: by modeling the input, we reduce the variance of the parameter estimates. In related work, Ng and Jordan (2002) showed that Naive Bayes requires exponentially fewer examples than logistic regression to obtain the same estimation error. The key property needed in their proof was that the Naive Bayes estimator decouples into d independent closed form optimization problems, which does not seem to be the defining property of generative estimation. In particular, this property does not apply to general globally-normalized generative models, but one would still expect those models to have the advantages of being generative.

Given that the generative and discriminative estimators are complementary, one natural question is how to interpolate between the two to get the benefits of both. Our framework naturally suggests two ways to go about this. First, we could vary the coarseness of the partitioning. Generative and discriminative estimators differ only in this coarseness and there is a range of intermediate choices corresponding to conditioning on more or fewer of the input variables. Second, we could take a weighted combination of estimators (e.g., Bouchard and Triggs (2004); McCallum et al. (2006)). For one-parameter models, Lindsay (1988) derived the optimal weighting of the component likelihoods, but unfortunately these results cannot be applied directly in practice.

It would also be interesting to perform a similar asymptotic analysis on other estimators used in practice, for example marginal likelihoods with latent variables, tree-reweighted belief propagation (Wainwright et al., 2003; Wainwright, 2006), piecewise training (Sutton & McCallum, 2005), etc. Another important extension is to curved exponential families, which account for many of the popular generative models based on directed graphical models.

6. Conclusion

We have analyzed the asymptotic distributions of composite likelihood estimators in the exponential family. The idea of considering different partitionings of the outcome space allows a clean and intuitive characterization of the asymptotic variances, which enables us to compare the commonly used generative, discriminative, and pseudolikelihood estimators as special cases. Our work provides new theoretical support for existing intuitions and a basis for developing new estimators which balance the tradeoff between computational and statistical efficiency.

Acknowledgments We thank Peter Bartlett for useful discussions and Simon Lacoste-Julien for comments. We also wish to acknowledge NSF grant 0509559 and a grant from Microsoft Research.

References

Besag, J. (1975). The analysis of non-lattice data. *The Statistician*, *24*, 179–195.

Bouchard, G., & Triggs, B. (2004). The trade-off between generative and discriminative classifiers. *International Conference on Computational Statistics* (pp. 721–728).

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling data. *International Conference on Machine Learning (ICML)*.

Lasserre, J. A., Bishop, C. M., & Minka, T. P. (2006). Principled hybrids of generative and discriminative models. *Computer Vision and Pattern Recognition (CVPR)* (pp. 87–94).

Liang, P., Klein, D., & Jordan, M. I. (2008). Agreement-based learning. *Advances in Neural Information Processing Systems (NIPS)*.

Lindsay, B. (1988). Composite likelihood methods. *Contemporary Mathematics*, *80*, 221–239.

McCallum, A., Pal, C., Druck, G., & Wang, X. (2006). Multi-conditional learning: Generative/discriminative training for clustering and classification. *Association for the Advancement of Artificial Intelligence (AAAI)*.

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems (NIPS)*.

Sutton, C., & McCallum, A. (2005). Piecewise training of undirected models. *Uncertainty in Artificial Intelligence (UAI)*.

van der Vaart, A. W. (1998). *Asymptotic Statistics*. Cambridge University Press.

Varin, C. (2008). On composite marginal likelihoods. *Advances in Statistical Analysis*, *92*, 1–28.

Wainwright, M. (2006). Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research*, *7*, 1829–1859.

Wainwright, M., Jaakkola, T., & Willsky, A. (2003). Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. *Artificial Intelligence and Statistics (AISTATS)*.

Wainwright, M., & Jordan, M. I. (2003). *Graphical models, exponential families, and variational inference* (Technical Report). Department of Statistics, University of California at Berkeley.

Structure Compilation: Trading Structure for Features

Percy Liang PLIANG@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, CA, USA

Hal Daumé III ME@HAL3.NAME
School of Computing, University of Utah, Salt Lake City, UT, USA

Dan Klein KLEIN@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, CA, USA

Abstract

Structured models often achieve excellent performance but can be slow at test time. We investigate *structure compilation*, where we replace structure with features, which are often computationally simpler but unfortunately statistically more complex. We analyze this tradeoff theoretically and empirically on three natural language processing tasks. We also introduce a simple method to transfer predictive power from structure to features via unlabeled data, while incurring a minimal statistical penalty.

1. Introduction

Structured models have proven to be quite effective for tasks which require the prediction of complex outputs with many interdependencies, e.g., sequences, segmentations, trees, etc. For example, conditional random fields (CRFs) can be used to predict tag sequences where there are strong dependencies between adjacent tags (Lafferty et al., 2001). In part-of-speech tagging, for instance, a CRF can easily model the fact that adjectives tend to precede nouns in English. However, the excellent performance of structured models comes at a computational cost: inference in loopy graphs requires approximate inference, and even for sequences, there is a quadratic dependence on the number of tags.

In this paper, we ask a bold question: do we really need structure? Consider replacing the edges in a CRF with additional contextual features, i.e., having an independent logistic regression (ILR) at each position. A fundamental question is whether there is a gap between the expressive power of the ILR and that of the CRF. Punyakanok et al. (2005) investigated this question for margin-based models[1] and concluded that structure was not needed when the independent problems were "easy." They characterized difficulty in terms of classifier separability, which is a very rigid notion. In Section 3.1, we provide an information-theoretic analysis, decomposing the gap between the ILR and CRF into three terms, each one representing a shortcoming of the ILR. The impact of each is investigated empirically.

Even if the ILR were made as expressive as the CRF by adding additional features, an important remaining question is whether the ILR could generalize as well as the CRF given limited labeled data. Indeed, the ILR overfits more easily, and we provide generalization bounds in Section 3.2 to quantify this effect.

At this point, it seems as though we are forced to make a tradeoff between the computational simplicity of the ILR and the statistical simplicity of the CRF. However, we propose *structure compilation* as a way to have the best of both worlds. Our strategy is to label a plethora of unlabeled examples using the CRF and then train the ILR on these automatically labeled examples. If we label enough examples, the ILR will be less likely to overfit. Although training now takes longer, it is only a one-time cost, whereas prediction at test time should be made as fast as possible.

Many authors have used unlabeled data to transfer the predictive power of one model to another, for example, from high accuracy neural networks to more interpretable decision trees (Craven, 1996), or from high accuracy ensembles to faster and more compact neural networks (Bucilă et al., 2006). Our fo-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] In their case, the independent models are not endowed with extra features, but coherence of the predictions is enforced at test time.

cus is on structured classification tasks, specifically on studying the tradeoff between structure and features. We ran experiments on three tasks: part-of-speech tagging (POS), named-entity recognition (NER), and constituency parsing (Figure 1).

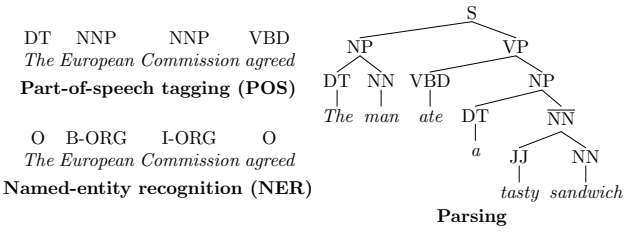


Figure 1. Examples of inputs and their outputs for the three tasks we experimented on. The input (what's seen at test time) is italicized.

2. From Structure to Features

In this section, we will walk through the process of replacing structure with features, using empirical results on POS[2] and NER[3] as running examples. Table 1 summarizes the notation we will use.

2.1. Conditional Random Fields (CRFs)

In structured classification, our goal is to learn to predict an output $\mathbf{y} \in \mathcal{Y}$ (e.g., a tag sequence, a segmentation, or a parse tree) given an input $\mathbf{x} \in \mathcal{X}$ (e.g., a sentence). In this paper, we consider *conditional exponential family* models, which have the following form:

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \exp\{\phi(\mathbf{x}, \mathbf{y})^\top \theta - A(\theta; \mathbf{x})\}, \quad (1)$$

where $\phi(\mathbf{x}, \mathbf{y})$ are the sufficient statistics (features), $\theta \in \mathbb{R}^d$ are the parameters, and $A(\theta; \mathbf{x}) = \log \sum_{\mathbf{y}} \exp\{\phi(\mathbf{x}, \mathbf{y})^\top \theta\}$ is the log-partition function.

One important type of conditional exponential family is a conditional random field (CRF) defined on a graph $G = (V, E)$. In this case, the output $\mathbf{y} = \{y_i\}_{i \in V}$ is a collection of labels, one for each node $i \in V$, with $y_i \in \{1, \ldots, K\}$. The features include functions over both nodes and edges:

$$\phi(\mathbf{x}, \mathbf{y}) = \sum_{i \in V} f(y_i, \mathbf{x}, i) + \sum_{(i,j) \in E} g(y_i, y_j).$$

[2]We used the Wall Street Journal (WSJ) portion of the Penn Treebank, with sections 0–21 as the training set (38K sentences) and sections 22–24 as the test set (5.5K sentences).

[3]We used the English data from the 2003 CoNLL Shared Task, consisting of 14.8K training sentences and 3.5K test sentences (set A).

In this work, we use a generic set of features for both POS and NER. The components of the node features $f(y_i, \mathbf{x}, i)$ are all indicator functions of the form $\mathbb{I}[y_i = a, s(x_{i+o}) = b]$, where a ranges over tags, $s(\cdot)$ ranges over functions on words,[4] b are values in the range of $s(\cdot)$, and $-L \le o \le L$ is an offset within a radius L window of the current position i (we used $L = 0$ for POS, $L = 1$ for NER). The components of the edge features $g(y_i, y_j)$ are of the form $\mathbb{I}[y_i = a, y_j = b]$. Let f_1 denote this *base feature set*.

Training Suppose we are given n labeled examples $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$, which for the purposes of our theoretical analysis are assumed to be drawn i.i.d. from some unknown true distribution p^*. We train the CRF using standard maximum likelihood:[5] $\max_\theta \mathbb{E}_{p^{\text{L}}(\mathbf{x},\mathbf{y})} \log p(\mathbf{y} \mid \mathbf{x}; \theta)$, where $p^{\text{L}}(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^n \mathbb{I}[\mathbf{x} = \mathbf{x}^{(i)}, \mathbf{y} = \mathbf{y}^{(i)}]$ denotes the empirical distribution of the labeled data. Later, we will consider other training regimes, so we need to establish some new notation. Let $p_{\text{C}} = \text{TR}(\text{CRF}, f_1, p^{\text{L}})$ denote the CRF trained with the base feature set f_1 on the labeled data.

CRFs achieve state-of-the-art performance on POS and NER. Using just our generic feature set, we obtain 96.9% tagging accuracy on POS (training with 30K examples) and 85.3% F_1 on NER (training with 10K examples). However, the performance does come at a computational cost, since inference scales quadratically with the number of tags K. This cost only increases with more complex models.

2.2. Independent Logistic Regression (ILR)

Let us try something drastic: remove the edges from the CRF to get an independent logistic regression (ILR), where now $\phi(\mathbf{x}, \mathbf{y}) = \sum_{i \in V} f(y_i, \mathbf{x}, i)$. For an ILR trained on the labeled data with our base feature set (denoted formally as $\text{TR}(\text{ILR}, f_1, p^{\text{L}})$), inference can be done independently for each node. For POS, the ILR takes only 0.4ms to process one sentence whereas the CRF takes 2.7ms, which is a speedup of 5.8x, not including the time for precomputing features.[6] Unfortunately, the accuracy of POS drops from 96.9% to 93.7%. For NER, F_1 drops from 85.3% to 81.4%.

[4]We used 10 standard NLP functions which return the word, prefixes and suffixes (up to length 3) of the word, word signatures (e.g., *McPherson* maps to *AaAaaaaaa* and *AaAa*), and whether the word is capitalized.

[5]In our experiments, we ran stochastic gradient for 50 iterations with a $1/(\text{iteration} + 3)$ step-size.

[6]If we include the time for computing features, the speedup drops to 2.3x.

2.3. Adding New Features

Without edges, the ILR has less expressiveness compared to the CRF. We can compensate for this loss by expanding our base feature set. We will use f_2 to denote the *expanded feature set*.

We use a simple recipe to automatically construct f_2 from f_1, but in general, we could engineer the features more carefully for better performance (see the parsing experiments in Section 4, for example). Essentially, our recipe is to allow the ILR at node i to use the base features f_1 applied to the nodes in a local window around i. For the chain CRF, this amounts to simply increasing the window size from L to $L + r$ (Section 2.1), where we call r the *expansion radius*.

For POS, we used an expansion radius of $r = 1$; for NER, $r = 2$. By training with these new features ($\text{TR}(\text{ILR}, f_2, p^{\text{L}})$), we get 96.8% on POS (compared to the 96.9% of the CRF), taking 0.8ms per example (compared to 2.7ms for the CRF). In this case, we have successfully traded structure for features with a negligible loss in performance and a 3.4x speedup. For NER, the story is quite different: adding features actually makes F_1 drop from 81.1% to 78.8%.

We believe there are two reasons for the poor performance on NER. First, since NER is a segmentation problem, the structure plays a more integral role and thus cannot be easily replaced with features. In other words, the *approximation error* of the ILR with respect to the CRF is higher for NER than POS. Section 3.1 provides a more formal treatment of this matter. Second, adding more features increases the risk of overfitting. In other words, the *estimation error* is larger when there are more features. Section 3.2 analyzes this error theoretically.

2.4. Using Unlabeled Data

There seems to be a tradeoff between approximation error and estimation error: More features can provide a better substitute for structure (decreasing the approximation error), but at the risk of overfitting the data (increasing the estimation error).

The algorithmic contribution of this paper is using unlabeled data to reduce the estimation error of the ILR via structure compilation. Suppose we have m unlabeled examples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}$ (which we assume are also generated from p^*); let $p^{\text{U}}(\mathbf{x})$ denote the corresponding empirical distribution. We can use the CRF $p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ (which has been trained on p^{L}) to label this unlabeled data. Let $p^{\text{U}}_{\text{C}}(\mathbf{x}, \mathbf{y}) = p^{\text{U}}(\mathbf{x}) p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ denote this new plentiful source of labeled data. Instead of training the ILR on the limited amount of originally labeled data p^{L}, we instead train it on our automatically labeled data p^{U}_{C}.[7]

f_1	base feature set
f_2	expanded feature set
$p^*(\mathbf{x}, \mathbf{y})$	true data distribution
$p^{\text{L}}(\mathbf{x}, \mathbf{y})$	original labeled examples (few)
$p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$	$= \text{TR}(\text{CRF}, f_1, p^{\text{L}})$ [trained CRF]
$p_{\text{C}^*}(\mathbf{y} \mid \mathbf{x})$	$= \text{TR}(\text{CRF}, f_1, p^*)$ [limiting CRF]
$p^{\text{U}}(\mathbf{x})$	unlabeled examples (many)
$p^{\text{U}}_{\text{C}}(\mathbf{x}, \mathbf{y})$	$= p^{\text{U}}(\mathbf{x}) p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ [labeled with CRF]
$p^*_{\text{C}}(\mathbf{x}, \mathbf{y})$	$= p^*(\mathbf{x}) p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ [labeled with CRF]
$p_{\text{I}}(\mathbf{y} \mid \mathbf{x})$	$= \text{TR}(\text{ILR}, f_2, p^{\text{U}}_{\text{C}})$ [trained ILR]
$p_{\text{I}^*}(\mathbf{y} \mid \mathbf{x})$	$= \text{TR}(\text{ILR}, f_2, p^*_{\text{C}})$ [limiting ILR]

Table 1. Notation used in this paper. In general, superscripts denote marginal distributions over $\mathbf{x}$ and subscripts denote conditional distributions over $\mathbf{y}$ given $\mathbf{x}$.

Structure compilation is most useful when there are few original labeled examples. Figure 2 shows the performance of a ILR obtained using structure compilation when the CRF is trained on only 2K labeled examples. We see that using more unlabeled data reduces the performance gap between the CRF and ILR as the estimation error is reduced. For POS, the gap is closed entirely, whereas for NER, there is a remaining approximation error.

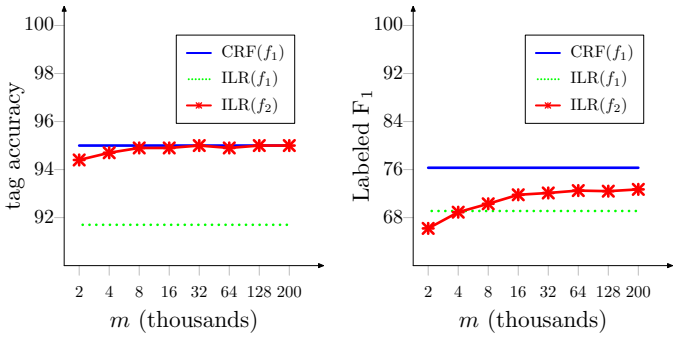


Figure 2. $\text{CRF}(f_1)$ is the CRF trained using the base feature set on 2K examples. $\text{ILR}(f_1)$ is the ILR trained the same way; performance suffers. However, by using the expanded feature set and training on m examples (New York Times articles from Gigawords) which are labeled with the CRF, $\text{ILR}(f_2)$ can recover all of the performance for POS and more than half of the performance for NER.

[7] Strictly speaking, training on p^{U}_{C} would involve creating many examples weighted by $p_{\text{C}}(y_i \mid \mathbf{x})$ for each original $\mathbf{x}$. But since the posteriors are sharply peaked, using $\text{argmax}_{\mathbf{y}} p_{\text{C}}(\mathbf{y} \mid \mathbf{x})$ was faster and an adequate approximation for our applications.

3. Analysis

In this section, we try to get a better understanding of when structure compilation would be effective. For the theoretical analysis, we will measure performance in terms of log-loss (negative cross-entropy):

$$\epsilon(p_1, p_2) \stackrel{\text{def}}{=} \mathbb{E}_{p_1(\mathbf{x},\mathbf{y})}[-\log p_2(\mathbf{y} \mid \mathbf{x})], \tag{2}$$

where $p_1(\mathbf{x}, \mathbf{y})$ is the data generating distribution and $p_2(\mathbf{y} \mid \mathbf{x})$ is the model under evaluation. We will also use conditional KL-divergence to quantify approximation error:

$$\kappa(p_1, p_2) \stackrel{\text{def}}{=} \epsilon(p_1, p_2) - \epsilon(p_1, p_1). \tag{3}$$

We are interested in $\epsilon(p^*, p_{\text{I}})$, the loss of the final compiled ILR. As we will later show in (7), this loss can be expressed in terms of two parts: (1) $\epsilon(p^*, p_{\text{C}})$, the loss of the CRF; and (2) $\kappa(p_{\text{C}}, p_{\text{I}})$, the penalty due to structure compilation.

The CRF loss can be decomposed into approximation and estimation errors as follows, using telescoping sums (refer to Table 1 for notation):

$$\begin{aligned}\epsilon(p^*, p_{\text{C}}) \;=\; & \underbrace{\epsilon(p^*, p_{\text{C}}) - \epsilon(p^{\text{L}}, p_{\text{C}})}_{\text{CRF-EST-ERR}} + \\ & \underbrace{\epsilon(p^{\text{L}}, p_{\text{C}}) - \epsilon(p^{\text{L}}, p_{\text{C}^*})}_{\leq 0} + \\ & \underbrace{\epsilon(p^{\text{L}}, p_{\text{C}^*}) - \epsilon(p^*, p_{\text{C}^*})}_{\text{CRF-EST-ERR}} + \underbrace{\epsilon(p^*, p_{\text{C}^*})}_{\text{CRF-APX-ERR}}.\end{aligned} \tag{4}$$

The second term on the RHS is non-positive because because p_{C} is chosen to minimize log-loss on p^{L}. The first and third terms are the estimation errors resulting from using p^{L} instead of p^*; these will be uniformly bounded in Section 3.2. Finally, the last term is an approximation error reflecting the modeling limitations of the CRF.

The structure compilation penalty can be decomposed analogously:

$$\begin{aligned}\kappa(p^*_{\text{C}}, p_{\text{I}}) \;=\; & \underbrace{\kappa(p^*_{\text{C}}, p_{\text{I}}) - \kappa(p^{\text{U}}_{\text{C}}, p_{\text{I}})}_{\text{ILR-EST-ERR}} + \\ & \underbrace{\kappa(p^{\text{U}}_{\text{C}}, p_{\text{I}}) - \kappa(p^{\text{U}}_{\text{C}}, p_{\text{I}^*})}_{\leq 0} + \\ & \underbrace{\kappa(p^{\text{U}}_{\text{C}}, p_{\text{I}^*}) - \kappa(p^*_{\text{C}}, p_{\text{I}^*})}_{\text{ILR-EST-ERR}} + \underbrace{\kappa(p^*_{\text{C}}, p_{\text{I}^*})}_{\text{ILR-APX-ERR}}.\end{aligned} \tag{5}$$

We would now like to combine $\epsilon(p^*, p_{\text{C}})$ and $\kappa(p_{\text{C}}, p_{\text{I}})$ to get a handle on $\epsilon(p^*, p_{\text{I}})$, but unfortunately, KL-divergence does not satisfy a triangle inequality. We can, however, derive the following approximate triangle inequality, where we pay an extra multiplicative factor (see Appendix A.1 for the proof):

Theorem 1. *Consider a conditional exponential family $\mathcal{P}$ with features ϕ. Let Θ be a compact subset of parameters, and define $\mathcal{P}_\Theta = \{p_\theta : \theta \in \Theta\}$. For any $p_1, p_2 \in \mathcal{P}_\Theta$ and any distribution p_0 such that $p'_0 = \operatorname{argmax}_{p \in \mathcal{P}} \mathbb{E}_{p^*(\mathbf{x})p_0(\mathbf{y}|\mathbf{x})} \log p(\mathbf{y} \mid \mathbf{x}) \in \mathcal{P}_\Theta$,*

$$\begin{aligned}\mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_0(\mathbf{y} \mid \mathbf{x}) \,||\, p_2(\mathbf{y} \mid \mathbf{x})\right) \leq \\ \alpha\left[\mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_0(\mathbf{y} \mid \mathbf{x}) \,||\, p_1(\mathbf{y} \mid \mathbf{x})\right) + \right. \\ \left. \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_1(\mathbf{y} \mid \mathbf{x}) \,||\, p_2(\mathbf{y} \mid \mathbf{x})\right)\right],\end{aligned} \tag{6}$$

where $\alpha = 2\frac{\sup_{\theta\in\Theta} \lambda_{\max}(\mathbb{E}\,\mathrm{var}_\theta(\phi|\mathbf{x}))}{\inf_{\theta\in\Theta} \lambda^+_{\min}(\mathbb{E}\,\mathrm{var}_\theta(\phi|\mathbf{x}))}$. Here, $\lambda_{\max}(\Sigma)$ and $\lambda^+_{\min}(\Sigma)$ are the largest and smallest nonzero eigenvalues of Σ, respectively.

Theorem 1 generalizes Lemma 3 of Crammar et al. (2007) to conditional distributions and the case where p_0 is not necessarily in an exponential family.

Let us apply Theorem 1 with p^*, p_{C}, p_{I}. We then add the conditional entropy $\mathbb{E}\,H(p^*(\mathbf{y} \mid \mathbf{x}))$ to the LHS of the resulting inequality and $\alpha\mathbb{E}\,H(p^*(\mathbf{y} \mid \mathbf{x}))$ to the RHS (note that $\alpha \geq 1$), thus obtaining a bound for the total loss of the final compiled ILR:

$$\begin{aligned}\epsilon(p^*, p_{\text{I}}) \;&\leq\; \alpha(\epsilon(p^*, p_{\text{C}}) + \kappa(p^*_{\text{C}}, p_{\text{I}})) \\ &\leq\; \alpha(\textsc{crf-apx-err} + \textsc{ilr-apx-err}) + \\ &\quad\; 2\alpha(\textsc{crf-est-err} + \textsc{ilr-est-err}).\end{aligned} \tag{7}$$

In the remaining sections, we analyze the various pieces of this bound.

3.1. Approximation Error

We start by analyzing $\textsc{ilr-apx-err} = \kappa(p^*_{\text{C}}, p_{\text{I}^*})$, which measures how well the ILR can approximate the CRF. Specifically, we show that $\kappa(p^*_{\text{C}}, p_{\text{I}^*})$ decomposes into three terms, each reflecting a limitation of the ILR: (1) I_{C}, the inability to produce a coherent output; (2) I_{N}, the inability to express nonlinearities; and (3) I_{G}, the inability to use information about the input outside a local window. The following theorem formalizes these concepts (see Appendix A.2 for the proof):

Theorem 2 (Decomposition of approximation error)**.** $\kappa(p^*_{\text{C}}, p_{\text{I}^*}) = I_{\text{C}} + I_{\text{N}} + I_{\text{G}}$*, where*

$$\begin{aligned}I_{\text{C}} \;&=\; \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_{\text{C}}(\mathbf{y} \mid \mathbf{x}) \,||\, \prod_{i\in V} p_{\text{C}}(y_i \mid \mathbf{x})\right), \\ I_{\text{N}} \;&=\; \mathbb{E}_{p^*(\mathbf{x})} \sum_{i\in V} \mathrm{KL}\left(p_{\text{C}}(y_i \mid \mathbf{x}) \,||\, p_{\text{A}^*}(y_i \mid \mathbf{x})\right), \\ I_{\text{G}} \;&=\; \mathbb{E}_{p^*(\mathbf{x})} \sum_{i\in V} \mathrm{KL}\left(p_{\text{A}^*}(y_i \mid \mathbf{x}) \,||\, p_{\text{I}^*}(y_i \mid \mathbf{x})\right),\end{aligned}$$

with $p_{\text{A}^*} = \text{TR}(\text{ILR}, f_\infty, p_{\text{C}}^*)$*, where the node features* f_∞ *are constructed from* f_1 *with an expansion radius of* ∞ *(so the entire input sequence* $\mathbf{x}$ *is used).*

Coherence One advantage of structured models is their ability to predict the output jointly. This could be especially important for NER, where the output tag sequence actually codes for a segmentation of the input. I_{C} measures the information lost by using the independent marginals of the CRF rather than the joint.[8]

For a chain CRF defined on $\mathbf{y} = (y_1, \ldots, y_\ell)$, one can check that I_{C} is the sum of mutual information terms along the edges: $I_{\text{C}} = \mathbb{E} \sum_{i=1}^{\ell-1} I(y_i, y_{i+1} \mid \mathbf{x})$. We computed I_{C} empirically: for POS, $I_{\text{C}} = 0.003$ and for NER, $I_{\text{C}} = 0.009$ (normalized by sequence length). Also, when we predict using the CRF marginals, the performance drops from 76.3% to 76.0% for NER but stays at 95.0% for POS. From these results, we conclude that coherence is not a big concern for our applications, although it is slightly more serious for NER, as we would expect.

Nonlinearities Although we think of CRFs as linear models, their marginal predictions actually behave nonlinearly. I_{N} captures the importance of this nonlinearity by comparing $p_{\text{C}}(y_i \mid \mathbf{x})$ and $p_{\text{A}^*}(y_i \mid \mathbf{x})$. Both depend on $\mathbf{x}$ through the same sufficient statistics $f_1(\cdot, \mathbf{x}, \cdot)$, but p_{A^*} acts on these sufficient statistics in a linear way whereas p_{C} allows the information about $\mathbf{x}$ to propagate in a nonlinear way through the other hidden labels $\mathbf{y}_{-i} = \{y_j : j \neq i\}$ in a manner roughly similar to that of a neural network. However, one difference is that the parameters of $p_{\text{C}}(y_i \mid \mathbf{x})$ are learned with $\mathbf{y}_{-i}$ fixed at training time; they are not arbitrary hidden units in service of y_i. A neural network therefore offers more expressive power but could be more difficult to learn.

We would like to measure the effect of nonlinearity empirically, but p_{A^*} has too many parameters to learn effectively. Thus, instead of comparing p_{A^*} and p_{C}, we compare p_{I^*} and a *truncated CRF* p_{TC}, which we train as follows:[9] For each labeled example $(\mathbf{x}, \mathbf{y})$ (which are sequences of length ℓ), we create ℓ new examples: $(\mathbf{x}_{i-L-r..i+L+r}, \mathbf{y}_{i-r..i+r})$ for $i = 1, \ldots, \ell$, where r is the expansion radius (Section 2.3). Then we train a CRF with features f_1 on these new examples to get p_{TC}. To label node i, we use $p_{\text{TC}}(y_i \mid \mathbf{x}_{i-L-r..i+L+r})$, marginalizing out $\mathbf{y}_{i-r..i-1,i+1..i+r}$. By this setup, both $p_{\text{TC}}(y_i \mid \mathbf{x})$ and $p_{\text{I}^*}(y_i \mid \mathbf{x})$ depend on $\mathbf{x}$ through the same features. Table 2 compares the NER performance of p_{TC} and p_{I}. As we can see, the truncated CRF significantly outperforms the ILR, demonstrating the power of nonlinearities.

Expansion radius r	1	2	3	∞
compiled ILR	0.725	0.727	0.721	—
truncated CRF	0.748	0.760	0.762	0.760

Table 2. NER F_1 (2K originally labeled examples, 200K automatically labeled examples for structure compilation) showing the importance of nonlinearity. Both the ILR and truncated CRF depend on the input $\mathbf{x}$ in the same way, but only the latter permits nonlinearities.

Global information I_{G} compares p_{A^*} and p_{I^*}, both of which are independent linear classifiers. The difference is that p_{A^*} uses all features of $\mathbf{x}$, while p_{I^*} uses only features of $\mathbf{x}$ in a local window.

Instead of comparing p_{A^*} and p_{I^*}, we compare their nonlinear counterparts p_{C} and p_{TC}. From Table 2, we can see that the truncated CRF with just an expansion radius of 2 has the same performance as the original CRF (expansion radius ∞). Therefore, we suspect that the features on $\mathbf{x}$ outside a local window have little impact on performance, and that most of the approximation error is due to lacking nonlinearities.

3.2. Estimation Error

In this section, we quantify the estimation errors for the CRF and ILR. First, we establish a general result about the estimation error of log-loss for exponential families. Our strategy is to uniformly bound the difference between empirical and expected log-losses across all parameter values of the exponential family. Our proof uses covering numbers and is based on Collins (2001), which derived an analogous result for a 0-1 margin loss.

Assume our parameters and features are bounded: $\Theta = \{\theta : ||\theta||_2 \leq B\}$ and $R = \sup_{\mathbf{x},\mathbf{y}} ||\phi(\mathbf{x}, \mathbf{y})||_2$. The following theorem relates the difference between empirical and expected log-loss to the number of examples n (see Appendix for proof):

Theorem 3. *For* $\delta > 0$*, with probability* $\geq 1 - \delta$*, for* $|\epsilon(p^*, p_\theta) - \epsilon(p^{\text{L}}, p_\theta)| \leq \eta$ *to hold for all* $\theta \in \Theta$*, it suffices to use* $n = \Omega(B^4 R^4 \log^2 |\mathcal{Y}| \eta^{-4} \log(1/\delta))$ *examples, where we have suppressed logarithmic factors.*

The above result gives uniform convergence of $\epsilon(\cdot, \cdot)$, which allows us to bound CRF-EST-ERR. In order to bound ILR-EST-ERR, we need uniform convergence of

[8] On the other hand, if we evaluate predictions using Hamming distance, it could actually be better to use the marginals. In that case, coherence is irrelevant.

[9] Truncated CRFs are closely related to piecewise-trained CRFs (Sutton & McCallum, 2005).

$\kappa(\cdot,\cdot)$ (5). This requires the convergence of one additional term: $|\epsilon(p_{\mathrm{C}}^{\mathrm{U}}, p_{\mathrm{C}}) - \epsilon(p_{\mathrm{C}}^{*}, p_{\mathrm{C}})| \xrightarrow{P} 0$ (p_{C} is non-random in this context). The asymptotics for n in Theorem 3 therefore remain unchanged.

We now apply Theorem 3 to the CRF and ILR with the features described in Section 2.1. For both models, $\log|\mathcal{Y}| = K|V|$. Where they differ is on the norms of the parameters and features (B and R). Let d_1 be the total number of features in the base feature set; d_2, the expanded feature set. Let c_k be the number of nonzero entries in $f_k(y_i, \mathbf{x}, i)$. Natural language processing is typified by sparse binary feature vectors, so $d_k \gg c_k$. For the CRF, $||\phi(\mathbf{x}, \mathbf{y})||_2$ is bounded by $R \leq \sqrt{c_1|V|^2 + |E|^2} \leq \sqrt{c_1}|V| + |E|$. The ILR has no edge potentials so $R \leq \sqrt{c_2}|V|$. In general, R is small for NLP applications.

On the other hand, $B \sim \sqrt{d_1}$ for the CRF and $B \sim \sqrt{d_2}$ for the ILR if the magnitude of the individual parameters are comparable. Since d_2 is significantly larger than d_1, the generalization bound for the ILR is much worse than for the CRF. While comparing upper bounds is inconclusive, showing that one upper bound is larger than another via the same methodology is weak evidence that the actual quantities obey a similar inequality.

4. Parsing Experiments

We now apply structure compilation to parsing. In this case, our structured model is a log-linear parser (Petrov & Klein, 2008), which we would like to replace with independent logistic regressions. For simplicity, we consider unlabeled binary trees. We could always use another independent classifier to predict node labels.

Standard parsing algorithms require $O(|G|\ell^3)$ time to parse a sentence with ℓ words, where $|G|$ is the size of the grammar. The ILR-based parser we will describe only requires $O(\ell^3)$ time. An extension to the labeled case would require $O(\ell^3 + K\ell)$ time, where K is the number of labels. This is a significant gain, since the grammars used in real-world parsers are quite large ($|G| \gg \ell, K$).

4.1. Independent Model

An example parse tree is shown in Figure 1 (recall that we do not predict the labels). For each span (i, j) ($1 \leq i < j \leq \ell$), the independent model predicts whether a node in the parse tree dominates the span. For example, of the 14 non-trivial spans in the sentence in Figure 1, the positively labeled spans are $(1, 2)$, $(5, 6)$, $(4, 6)$, and $(3, 6)$. To parse a sentence at test time, we first use the independent model to assign each span a probability and then use dynamic programming to find the most likely tree.

The independent model uses the following features evaluated (a) within the span, (b) at the boundary of the span, and (c) within a window of 3 words on either side of the span: identity, parts-of-speech, prefixes/suffixes (length 1-3), and case patterns. Additional features include 3- and 4-grams of the words and POS tags that occur within the span; the entire POS sequence; the entire word sequence; the 3-character suffix sequence; the case sequence within the span; the length of the span; the position of the span relative to the entire sentence; the number of verbs, conjunctions and punctuation marks within the span; and whether the span centers on a conjunction and has symmetric POS tags to the left and right.

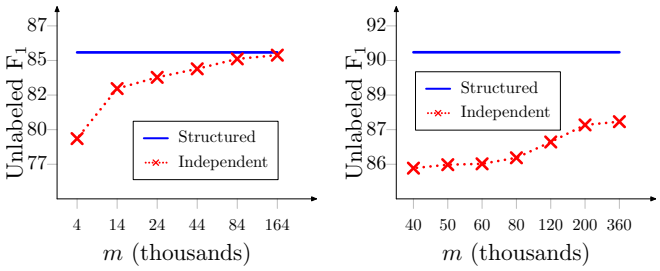
4.2. Results

In all of our experiments, we evaluate according to the F_1 score on unlabeled, binarized trees (using right-binarization). This scoring metric is slightly non-standard, but equally difficult: a parser that achieves a labeled F_1 (the usual metric) of 89.96% on the Treebank test data (section 23) achieves 90.29% under our metric; on 10% of the data, the two metrics are 82.84% and 85.16%, respectively.

To test our independent parser, we trained a structured parser on 10% of the WSJ portion of the Penn Treebank (4K sentences). We then used the structured parser to parse 160K unlabeled sentences,[10] which were then used, along with the original 4K sentences, to train the independent model. Figure 3(a) shows the F_1 scores of the various models. When only 4K is used, the independent parser achieves a score of 79.18%, whereas the structured parser gets 85.16%. With more automatically labeled examples, the performance of the independent parser approaches that of the structured parser (84.97% with 164K sentences).

On the other hand, if we trained the structured parser on 40K sentences, then the independent parser has a much harder time catching up, improving from 85.42% (40K sentences) to just 87.57% (360K sentences) compared to the structured parser's 90.78% (Figure 3(b)). Since parsing is a much more complex task compared to POS or NER, we believe that richer features would be needed to reduce this gap.

[10] These sentences are from the North American National Corpus, selected by test-set relativization.



(a) 4K sentences (b) 40K sentences

Figure 3. A comparison of the structured and independent parsers when the structured parser is trained on 4K (a) and 40K (b) sentences. m is the number of examples (original + automatically labeled) used to train the independent parser.

5. Conclusion

The importance of deploying fast classifiers at test time motivated our investigation into the feasibility of replacing structure with features. We presented a method to compile structure into features and conducted theoretical and empirical analyses of the estimation and approximation errors involved in structure compilation. We hope that a better understanding of the role structure plays will lead to more computationally efficient methods that can still reap the benefits of structure.

References

Bucilă, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. *International Conference on Knowledge Discovery and Data Mining (KDD)*.

Collins, M. (2001). Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. *International Workshop on Parsing Technologies*.

Crammar, K., Kearns, M., & Wortman, J. (2007). Learning from multiple sources. *Advances in Neural Information Processing Systems (NIPS)*.

Craven, M. W. (1996). *Extracting comprehensible models from trained neural networks*. Doctoral dissertation, University of Wisconsin at Madison.

Csiszár, I., & Shields, P. (2004). Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory, 1*, 417–528.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling data. *International Conference on Machine Learning (ICML)*.

Petrov, S., & Klein, D. (2008). Discriminative log-linear grammars with latent variables. *Advances in Neural Information Processing Systems (NIPS)*.

Pollard, D. (1984). *Convergence of stochastic processes*. Springer-Verlag.

Punyakanok, V., Roth, D., Yih, W., & Zimak, D. (2005). Learning and inference over constrained output. *International Joint Conference on Artificial Intelligence (IJCAI)*.

Sutton, C., & McCallum, A. (2005). Piecewise training of undirected models. *Uncertainty in Artificial Intelligence (UAI)*.

Zhang, T. (2002). Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research, 2*, 527–550.

A. Proofs

Lemma 1 gives conditions under which a conditional KL-divergence can be decomposed exactly. Lemma 2 specializes to the exponential family. These lemmas are variants of standard results from information geometry (see Csiszár and Shields (2004)). We will use them in the proofs of Theorems 1 and 2.

Lemma 1 (Conditional Pythagorean identity)**.** *Let* $d(p, p') = \mathbb{E}_{p^*(\mathbf{x})p(\mathbf{y}|\mathbf{x})} \log p'(\mathbf{y} \mid \mathbf{x})$ *be the negative conditional cross-entropy. For any three conditional distributions* p_0, p_1, p_2*, if* $d(p_0, p_1) = d(p_1, p_1)$ *and* $d(p_0, p_2) = d(p_1, p_2)$*, then*

$$\begin{aligned}\mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_0(\mathbf{y} \mid \mathbf{x}) \,||\, p_2(\mathbf{y} \mid \mathbf{x})\right) = \\ \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_0(\mathbf{y} \mid \mathbf{x}) \,||\, p_1(\mathbf{y} \mid \mathbf{x})\right) + \\ \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_1(\mathbf{y} \mid \mathbf{x}) \,||\, p_2(\mathbf{y} \mid \mathbf{x})\right).\end{aligned} \tag{8}$$

Proof. Use the fact that $\mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p \,||\, p'\right) = d(p, p) - d(p, p')$ and perform algebra. □

Lemma 2 (Conditional Pythagorean identity for exponential families)**.** *Let* $\mathcal{P}$ *be a conditional exponential family. If* $p_1 = \operatorname{argmax}_{p \in \mathcal{P}} \mathbb{E}_{p^*(\mathbf{x})p_0(\mathbf{y}|\mathbf{x})} \log p(\mathbf{y} \mid \mathbf{x})$ *and* $p_2 \in \mathcal{P}$*, (8) holds for* p_0, p_1, p_2*.*

Proof. Since p_1 is the maximum likelihood solution, $\mu \stackrel{\text{def}}{=} \mathbb{E}_{p^*(\mathbf{x})p_0(\mathbf{y}|\mathbf{x})}\phi(\mathbf{x}, \mathbf{y}) = \mathbb{E}_{p^*(\mathbf{x})p_1(\mathbf{y}|\mathbf{x})}\phi(\mathbf{x}, \mathbf{y})$, where ϕ are the features of $\mathcal{P}$. Then for $p \in \mathcal{P}$ with parameters θ, $d(p_0, p) = \mu^\top \theta - \mathbb{E}\, A(\theta; \mathbf{x}) = d(p_1, p)$ (follows from (1)). Plug in $p = p_1, p_2$ and apply Lemma 1. □

A.1. Proof of Theorem 1

Proof. The first part of the proof is similar to that of Lemma 3 in Crammar et al. (2007). Denote $k(p, p') = \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p(\mathbf{y} \mid \mathbf{x}) \,||\, p'(\mathbf{y} \mid \mathbf{x})\right)$ and $B(\theta) = \mathbb{E}_{p^*(\mathbf{x})} A(\theta; \mathbf{x})$.

The key is to note that for $p, p' \in \mathcal{P}$, the conditional KL-divergence is the residual term in the first-order approximation of B: $k(p, p') = \nabla B(\theta)^\top (\theta - \theta') - (B(\theta) - B(\theta'))$, where θ, θ' are the parameters of p, p'. Thus, we can use Taylor's theorem to get that

$k(p,p') = \frac{1}{2}||\theta - \theta'||^2_{V_{p,p'}}$, where $V_{p,p'} = \nabla^2 B(\tilde{\theta}) = \mathbb{E}\,\mathrm{var}_{\tilde{\theta}}(\phi \mid \mathbf{x})$ for some $\tilde{\theta} \in \Theta$.

One can check that $||\theta'_0 - \theta_2||^2_{V_{0,2}} \le 2[||\theta'_0 - \theta_1||^2_{V_{0,2}} + ||\theta_1 - \theta_2||^2_{V_{0,2}}]$, where $V_{0,2} = V_{p'_0,p_2}$. By definition of α, $V_{0,2} \preceq \frac{\alpha}{2} V_{0,1}$ and $V_{0,2} \preceq \frac{\alpha}{2} V_{1,2}$,[11] so $||\theta'_0 - \theta_2||^2_{V_{0,2}} \le \alpha[||\theta'_0 - \theta_1||^2_{V_{0,1}} + ||\theta_1 - \theta_2||^2_{V_{1,2}}]$. Rewritten another way: $k(p'_0, p_2) \le \alpha[k(p'_0, p_1) + k(p_1, p_2)]$.

Applying Lemma 2 twice to p_0, p'_0, p_1 and p_0, p'_0, p_2 yields $k(p_0,p_2) - k(p_0,p'_0) \le \alpha[k(p_0,p_1) - k(p_0,p'_0) + k(p_1,p_2)]$. Subtracting $k(p_0,p'_0)$ from both sides and noting $\alpha \ge 1$ yields the theorem. □

A.2. Proof of Theorem 2

Proof. Define $p_{\text{MC}}(\mathbf{y} \mid \mathbf{x}) = \prod_{i \in V} p_{\text{C}}(y_i \mid \mathbf{x})$. Check that $d(p_{\text{C}}(\mathbf{y} \mid \mathbf{x}), \prod_{i \in V} p(y_i \mid \mathbf{x})) = d(\prod_{i \in V} p_{\text{C}}(y_i \mid \mathbf{x}), \prod_{i \in V} p(y_i \mid \mathbf{x}))$ for any p, in particular, p_{MC} and p_{I^*}. Thus we can apply Lemma 1 with $p_{\text{C}}, p_{\text{MC}}, p_{\text{I}^*}$ to get $\kappa(p^*_{\text{C}}, p_{\text{I}^*}) = I_{\text{C}} + \mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_{\text{MC}}(\mathbf{y} \mid \mathbf{x}) \,||\, p_{\text{I}^*}(\mathbf{y} \mid \mathbf{x})\right)$.

Since f_∞ is a superset of f_2, both p_{I^*} and p_{A^*} are members of the f_∞-exponential family, with p_{A^*} being the maximum likelihood solution. Apply Lemma 2 with $p_{\text{MC}}, p_{\text{A}^*}, p_{\text{I}^*}$ to get $\mathbb{E}_{p^*(\mathbf{x})}\mathrm{KL}\left(p_{\text{MC}}(\mathbf{y} \mid \mathbf{x}) \,||\, p_{\text{I}^*}(\mathbf{y} \mid \mathbf{x})\right) = I_{\text{N}} + I_{\text{G}}$. □

A.3. Proof of Theorem 3

We use covering numbers to bound the complexity of the class of log-losses. Our proof is inspired by Collins (2001), who works with a 0-1 margin-based loss. Define the loss class:

$$\mathcal{M} \stackrel{\text{def}}{=} \{(\mathbf{x},\mathbf{y}) \mapsto -\log p_\theta(\mathbf{y} \mid \mathbf{x}) : \theta \in \Theta\}. \tag{9}$$

We first show that the elements of $\mathcal{M}$ are bounded:

Lemma 3. *For each $f \in \mathcal{M}$ and $(\mathbf{x},\mathbf{y})$, $0 \le f(\mathbf{x},\mathbf{y}) \le L$, where $L \stackrel{\text{def}}{=} BR(1 + \log|\mathcal{Y}|)$.*

Proof. The lower bound holds since probabilities are bounded above by 1. For the upper bound, consider the absolute value of the two terms in (1) separately. For the linear term, $|\phi(\mathbf{x},\mathbf{y})^\top \theta| \le BR$ by the Cauchy-Schwartz inequality. The log-partition function can be bounded by $BR\log|\mathcal{Y}|$ by applying the linear term result to the exponent. Add the two bounds. □

Theorem 1 of Zhang (2002) (originally due to Pollard (1984)) applied to $\mathcal{M}$: With probability $\ge 1 - \delta$,

$$P\left(\sup_{\theta \in \Theta} |\epsilon(p^*, p_\theta) - \epsilon(p^{\text{L}}, p_\theta)| > \eta\right) \le 8\mathcal{N}_1(\mathcal{M}, \eta/8, n) \exp\left\{\frac{-n\eta^2}{128L^2}\right\}, \tag{10}$$

where $\mathcal{N}_p(\mathcal{F}, \epsilon, n)$, the covering number of function class $\mathcal{F}$, is the supremum over all points $\mathbf{z}_1, \ldots, \mathbf{z}_n$, of the size of the smallest cover $\{g_1, \ldots, g_k\}$ such that for all $f \in \mathcal{F}$, there exists a g_j in the cover with $(\frac{1}{n}\sum_{i=1}^n |f(\mathbf{z}_i) - g_j(\mathbf{z}_i)|^p)^{1/p} \le \epsilon$.

We now upper bound $\mathcal{N}_\infty(\mathcal{M}, \epsilon/8, n)$, adapting the method used in Collins (2001). First define the set of linear functions:

$$\mathcal{L} \stackrel{\text{def}}{=} \left\{\mathbf{v} \mapsto \theta^\top \mathbf{v} : \theta \in \Theta\right\}. \tag{11}$$

Theorem 4 of Zhang (2002) (with $p = q = 2$) allows us to bound the complexity of this class:

$$\log_2 \mathcal{N}_\infty(\mathcal{L}, \epsilon, n) \le 36(BR/\epsilon)^2 \log_2(2\lceil 4BR/\epsilon + 2\rceil n + 1). \tag{12}$$

We now relate the covering numbers of $\mathcal{L}$ and $\mathcal{M}$:

Lemma 4. $\mathcal{N}_\infty(\mathcal{M}, \epsilon, n) \le \mathcal{N}_\infty(\mathcal{L}, \epsilon/2, n|\mathcal{Y}|)$.

Proof. Let $S = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}$. We will construct a covering of $\mathcal{M}$ (with respect to S) by reducing to the problem of finding a covering of $\mathcal{L}$. Let $\mathbf{v}_{i\mathbf{y}} = \phi(\mathbf{x}^{(i)}, \mathbf{y})$ and $V = \{\mathbf{v}_{i\mathbf{y}} : i = 1, \ldots, n, \mathbf{y} \in \mathcal{Y}\}$. By (12), we can cover $\mathcal{L}$ with respect to V with a set $C_{\mathcal{L}}$. Consider the corresponding set $C_{\mathcal{M}} \subset \mathcal{M}$ (note the natural 3-way bijections between Θ, $\mathcal{L}$, and $\mathcal{M}$).

To prove the lemma, it suffices to show that $C_{\mathcal{M}}$ is a covering of $\mathcal{M}$. Fix some $g \in \mathcal{M}$, which is associated with some $f \in \mathcal{L}$ and $\theta \in \Theta$. There exists a $\tilde{f} \in C_{\mathcal{L}}$ (corresponding to a $\tilde{\theta} \in \Theta$ and a $\tilde{g} \in C_{\mathcal{M}}$) such that $|f(\mathbf{x},\mathbf{y}) - \tilde{f}(\mathbf{x},\mathbf{y})| = |\theta^\top \phi(\mathbf{x},\mathbf{y}) - \tilde{\theta}^\top \phi(\mathbf{x},\mathbf{y})| \le \epsilon/2$ for all $(\mathbf{x},\mathbf{y}) \in S$ and $\mathbf{y} \in \mathcal{Y}$. We now argue that $\tilde{g}$ is close to g. For each $(\mathbf{x},\mathbf{y}) \in S$,

$$\begin{aligned} g(\mathbf{x},\mathbf{y}) &= -\log p_\theta(\mathbf{y} \mid \mathbf{x}) \\ &= -\theta^\top \phi(\mathbf{x},\mathbf{y}) + \log \sum_{\mathbf{y}' \in \mathcal{Y}} e^{\theta^\top \phi(\mathbf{x},\mathbf{y}')} \\ &\le -(\tilde{\theta}^\top \phi(\mathbf{x},\mathbf{y}) - \epsilon/2) + \log \sum_{\mathbf{y}' \in \mathcal{Y}} e^{\tilde{\theta}^\top \phi(\mathbf{x},\mathbf{y}') + \epsilon/2} \\ &= \tilde{g}(\mathbf{x},\mathbf{y}) + \epsilon. \end{aligned}$$

Similarly, $g(\mathbf{x},\mathbf{y}) \ge \tilde{g}(\mathbf{x},\mathbf{y}) - \epsilon$. Therefore, $C_{\mathcal{M}}$ is a cover of $\mathcal{M}$. □

Substitute (12) into Lemma 4 to get an expression for $\mathcal{N}_\infty(\mathcal{M}, \epsilon, n)$; substitute that into (10), using the fact that $\mathcal{N}_1 \le \mathcal{N}_\infty$. Solving for n yields the theorem.

[11] It suffices to consider nonzero eigenvalues because zero eigenvalues correspond to non-identifiable directions, which are the same for all parameters θ.

ManifoldBoost: Stagewise Function Approximation for Fully-, Semi- and Un-supervised Learning

Nicolas Loeff LOEFF@UIUC.EDU
Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801

David Forsyth DAF@UIUC.EDU
Deepak Ramachandran DRAMACHA@UIUC.EDU
Department of Computer Science, University of Illinois, Urbana, IL 61801

Abstract

We describe a manifold learning framework that naturally accommodates supervised learning, partially supervised learning and unsupervised clustering as particular cases. Our method chooses a function by minimizing loss subject to a manifold regularization penalty. This augmented cost is minimized using a greedy, stagewise, functional minimization procedure, as in Gradientboost. Each stage of boosting is fast and efficient. We demonstrate our approach using both radial basis function approximations and trees. The performance of our method is at the state of the art on many standard semi-supervised learning benchmarks, and we produce results for large scale datasets.

1. Introduction

Manifold Learning algorithms exploit geometric (or correlation) properties of datasets in high-dimensional spaces. The literature is too large to review in detail here (163 references in a recent review (Zhu, 2006)). Many different approaches have been pursued that utilize manifold structure such as constructing an explicit parametrization (e.g. (Tenenbaum et al., 2000; Roweis & Saul, 2000; Donoho & Grimes, 2003)), introducing a penalty term that imposes smoothness conditions on functions restricted to the manifold (e.g. (Sindhwani et al., 2006)), adjusting kernel smoothing bandwidths to account for manifold properties (e.g. (Bickel & Li, 2007)), and infering labels for unlabeled data using a harmonic smoother (e.g. (Zhu et al., 2003)).

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

There is a rough distinction in semi-supervised learning between manifold based algorithms that expect data to lie embedded in a space of lower intrisic dimensionality, and cluster-based algorithms that expect data to lie in clumps (the distinction seems to explain some differences in performance on different datasets (Chapelle et al., 2006)). There is some disagreement about the benefits of using unlabeled data, which may not always improve the asymptotic error rate of a regression estimator (Lafferty & Wasserman, 2007). On the other hand, (Niyogi, 2008) argues that manifold learning is useful insofar as the marginal of the data P_x can be linked with the conditional $P_{y|x}$ via the manifold.

Computational Complexity is a common problem for most semi-supervised approaches. Write l for the number of labeled data items and u for the number of unlabeled data items. Many algorithms scale as badly as $O((l+u)^3)$ (Zhu, 2006). Transductive support vector machines must solve a quadratic programming problem in $(l+u)$ variables (Joachims, 1999). Manifold smoothing of an SVM solves a quadratic programming problem in l variables, followed by a linear problem in $l+u$ variables; the situation is better for a linear SVM if feature vectors are sufficiently sparse (Sindhwani et al., 2006). Harmonic smoothing solves a relatively sparse linear system in l variables. This problem is relatively tractable, because the linear system involves the Laplacian of the smoothing kernel and so should be diagonally dominant (see (Dyn et al., 1986) for relevant observations in the context of radial basis functions). Each method must pay the cost of forming the Laplacian. For functional approximation schemes other than kernel smoothing, the complexity of current manifold learning methods in the number of training examples appears to be high. This is a problem – it is natural to want to use a manifold regularization term

with such methods as tree-structured classifiers, and with very large datasets.

Gradient Boosting poses function approximation as a variational problem, then uses a form of coordinate ascent on that problem ((Friedman, 1999); section 2). In this paper, we describe a variation on gradient boosting that can exploit a manifold regularization term, is fast and efficient for many forms of functional approximation (section 2.1), provides out of sample extensions (section 4), offers performance at the state of the art on standard datasets, and is capable of handling very large datasets (section 6). In the extreme case, when there is no supervision, the generalized method gracefully degrades into a clustering method (section 4). Finally, we show that our framework also easily extends to multi-class problems by choosing suitable loss functions (section 5).

2. Semi-Supervised Boosting

We follow convention by minimizing the sum of an expected loss and a regularization term. We must predict labels $y \in \mathcal{Y}$ for patterns $x \in \mathcal{X}$. We assume a probability distribution $P_{x,y}$ over $\mathcal{X} \times \mathcal{Y}$.

We will further assume the support of the marginal P_x lies on a domain $\mathcal{M} \subset \mathcal{X}$. Typically, this domain is of lower intrinsic dimension than $\mathcal{X}$; the term manifold is widely used to refer to such domains, though we require no manifold properties.

Write the predictor as $F(x)$, and the cost function as $\psi(y, F(x))$. We would like to find the function minimizer $F^* = \arg\min_{F \in \mathcal{H}} V[F]$, of the cost functional

$$\begin{aligned} V[F] &= \underbrace{\int \psi(y, F(x)) dP_{x,y}}_{\text{Expected Loss}} \\ &+ \underbrace{\gamma_{\mathcal{M}} \int_{\mathcal{M}} ||\nabla_{\mathcal{M}} F(x)||^2 dP_x}_{\text{Manifold Regularization}} \end{aligned} \quad (1)$$

restricted to some function family $\mathcal{H}$. Our regularization term is of the same form as that of (Sindhwani et al., 2006), and encourages smoothness of the solution in regions of high probability density. We control the complexity of the solution by choosing $\mathcal{H}$ and using the *shrinkage* approach of (Friedman, 1999).

This expression is very general. There are many possible choices for $\psi[y, F]$. Expressions such as $|y - F|$ and $(y - F)^2$ are typically used for regression. Expressions such as $\exp(-yF)$ and the binomial log likelihood $\log(1 + \exp(-2yF))$ penalize the *margin* yF, and are typically used for classification.

2.1. ManifoldBoost Framework

2.1.1. Stagewise Functional Minimization (P_x Known)

Following the work of Friedman (Friedman, 1999), we will find a *additive* solution of the form

$$F_M(x) = \sum_{m'=0}^{M} f_{m'}(x) \quad (2)$$

We will proceed in a greedy fashion. Assume we have a solution for $M = m$; we will then minimize $V[F_m + f_{m+1}]$ with respect to f_{m+1}. After (Friedman, 1999), we obtain a descent direction from the first variation of V

$$V[F_m + \epsilon f] = V[F_m] + \epsilon \delta V[F_m, f] + O(\epsilon^2) \quad (3)$$

where

$$\delta V[F_m, f] = \frac{d}{d\epsilon} V[F_m + \epsilon f]|_{\epsilon=0}$$

Write $\langle u, v \rangle$ for the usual inner product in L^2. Now $\delta V[F_m, f]$ is a linear functional of f, so there is some $G_V(F_m)$ — which we regard as the "gradient" of the cost — such that $\delta V[F_m, f] = \langle G_V(F_m), f \rangle$. Now we have that $\langle G_V(F_m), f \rangle$ is equal to

$$\int \left\{ \begin{array}{c} f(x) \left[\int_y \frac{\partial}{\partial u} \psi(y, u)\big|_{u=F_m(x)} dP_{y|x} \right] \\ +2\gamma_{\mathcal{M}} \nabla f(x)^t \nabla F_m(x) \end{array} \right\} dP_x \quad (4)$$

Assuming sufficient regularity, recalling that $P_x = 0$ on the boundary of the support of P_x, and using the first Green identity, we have that $\langle G_V(F_m), f \rangle$ is equal to

$$\int \left\{ \begin{array}{c} f(x) \left[\int_y \frac{\partial}{\partial u} \psi(y, u)\big|_{u=F_m(x)} dP_{y|x} \right] \\ +2\gamma_{\mathcal{M}} f(x) \nabla^2_{\mathcal{M}} F_m(x) \end{array} \right\} dP_x \quad (5)$$

where $\nabla^2_{\mathcal{M}} = -\nabla \cdot \nabla_{\mathcal{M}}$ is the *Laplace-Beltrami* operator. The optimal descent direction is a function f that maximizes $-\langle G_V(F_m), f \rangle$ (subject, if necessary to a norm constraint on f). The term $f_{m+1} = \alpha f$ is obtained using line search, minimizing the true cost $V[F_m + \alpha f]$ with respect to α.

2.1.2. Finite Data

Generally, neither P_x nor $P_{x,y}$ are known. Instead, we have sample of labelled data $\{x_i, y_i\}_{i=1}^{l}$, and of unlabelled data $\{x_i\}_{i=l+1}^{u}$. Now integrals become sums over data points. Generally, $\{f_m(x)\}$ will belong to a parametric family of functions (e. g. Radial Basis functions, decision trees, etc).

The Laplacian operator in equation 5 must be discretized. In high dimensions, we cannot triangulate

the data set. A smoothed Laplacian is equivalent to the difference between a short-scale average of the data and a long-scale average (e.g. the use of unsharp masking in photography, or the difference of Gaussians in computer vision). The graph Laplacian is a linear operator that takes a function on the graph to the weighted difference between the function value and the average of the K nearest neighbours. This means it is usual to approximate the Laplacian operator with the graph Laplacian $\mathcal{L}$ (e.g. see (Sindhwani et al., 2006)).

Write the graph Laplacian as $\mathcal{L}^{\mathcal{M}}$. The cost function becomes

$$\begin{aligned} V[F] &= \frac{1}{l}\sum_{i=1}^{l}\psi(y_i, F(x_i)) \\ &+ \frac{\gamma_{\mathcal{M}}}{(l+u)K}\sum_{i,j}F(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F(x_j) \end{aligned} \quad (6)$$

Again, assume we know F_m, and seek f_{m+1}. We will find a function f that maximizes $-\langle G_V(F_m), f\rangle$ then we will weight this function using line search. The inner product is $\langle a, b\rangle = \frac{1}{N}\sum_{i=1}^{N} a(x_i)\cdot b(x_j)$ and we have that

$$\begin{aligned} \langle G_V(F_m), f\rangle &= \frac{1}{l}\sum_i \frac{\partial}{\partial u}\psi(y,u)\Big|_{u=F_m(x_i)} f(x_i) \\ &+ \frac{2\gamma_{\mathcal{M}}}{(l+u)^2}\sum_{i,j} f(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F_{m-1}(x_j) \end{aligned} \quad (7)$$

Now $-\langle G_V, f\rangle$ is linear in f, and so we should maximize subject to a norm constraint on f. If the norm is fixed, then maximizing this expression is equivalent to minimizing $||G_V - f||^2 = ||G_V||^2 - 2\langle G_V, f\rangle + ||f||^2$. This means any squared loss regression algorithm can be used to find the optimal parameters. Our variational formulation explains why Friedman's *choosing* to make f parallel to the gradient G_V and posing the problem as squared error minimization is natural. Once the descent direction f is found, the final $f_{m+1} = \alpha f$ is obtained using line search, minimizing the true cost $V[F_m + \alpha f]$.

3. Two Examples: Tree and RBF ManifoldBoost Algorithms

We offer two example algorithms with calculations to illustrate our extremely general formalism. For each example, we consider the binary case ($y \in \{-1, 1\}$, $y = 0$ for unlabeled data), and use the negative binomial log likelihood as the loss function (Friedman, 1999): $\psi(y, F) = \log(1 + \exp(-2yF))$ For this case, whatever classifier we use represents $F(x) = \frac{1}{2}[\log(p(y=1|x)) - \log(p(y=-1|x))]$ and so at round m, the inner product with the "gradient" becomes,

$$\begin{aligned} \langle G_V(F_m), f\rangle &= \frac{1}{l}\sum_i \frac{2y_i}{1+\exp(2y_iF_m(x_i))} f(x_i) \\ &+ \frac{2\gamma_{\mathcal{M}}}{(l+u)K}\sum_{i,j} f(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F_m(x_j) \end{aligned} \quad (8)$$

The cases now differ by the procedures used to choose the optimal f

Tree-ManifoldBoost: As in L_2 TreeBoost (Friedman, 1999), we use regression trees as base learners. A tree has the form $f_{m+1}(x) = \sum_{s=1}^{S}\eta_{m+1,s}I[x \in R_s]$, where $I[\cdot] = 1$ if the expression inside is true, and $I[\cdot] = 0$ otherwise.

To minimize $||G_V - f||^2$, we must search for the parameters R_s (which determine the geometry of the tree) and η_s (which determine weights within region). Once a tree has been found, we fix R_s and minimize $V(F_m(x) + \sum_{s=1}^{S}\eta_{m,s}[x \in R_s])$ with respect to $\{\eta_s\}$, using a standard continuous optimization method (BFGS; see (Bertsekas, 1996)). In each round, we use a small number of descent steps to prevent overfitting.

Algorithm 1 Tree ManifoldBoost Algorithm

1: $F_0(x) = 1/2[\log(1+\overline{y}) - \log(1-\overline{y})]$
2: **for** $m = 1$ to M **do**
3: Compute G_V as in (8)
4: Obtain regression tree $\{R_{s,m}\}$ by minimizing $\sum_i (G_V(x_i) - \sum_s \eta_{m,s}I[x_i \in R_{m,s}])^2$
5: Find $\{\eta_{m,s}\}$ using BFGS and $\frac{\partial V}{\partial \eta_s}$, and fixing $\{R_{m,s}\}$
6: $F_m(x) = F_{m-1}(x) + \sum_s \eta_{m,s}[x \in R_{m,s}]$
7: **end for**

The algorithm converges when M rounds have been run, or the relative change in the cost function in a round is below a threshold. Probability estimates for each x can then be estimated by inverting the loss function: $p(y=1|x) = 1/(1+\exp(-2F_M(x)))$. This in turn can be used for classification:

$$\tilde{y_i} = \begin{cases} 1 & p(y=1|x)k_{-1,1} > p(y=-1|x)k_{1,-1} \\ -1 & \text{otherwise} \end{cases} \quad (9)$$

where cost $k_{a,b}$ is the penalty for choosing label a when b is the correct label.

Figure 1 shows a toy example for semisupervised classification taken from (Sindhwani et al., 2006) (two moons dataset). The unlabeled datapoints are depicted in green and the diamonds represent the labeled examples (one for each class). The algorithm also can

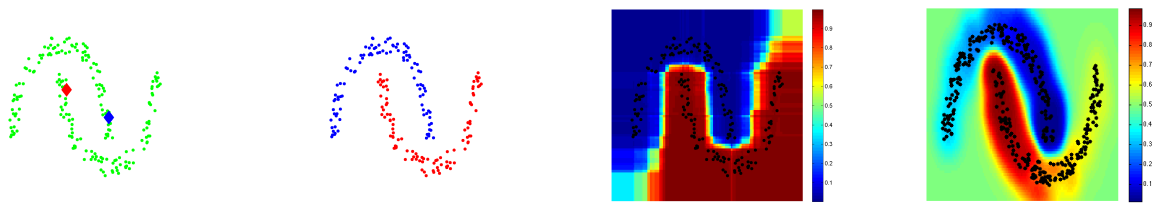


Figure 1. **Semi-supervised learning** using Tree- and RBF-ManifoldBoost. **first** figure from left to right shows a toy example introduced in (Sindhwani et al., 2006) (two moons dataset): the unlabeled datapoints are depicted in green, the diamonds represent the labeled examples (one for each class). The output classification of both algorithm is the same and is depicted on the **second** image (for datapoints). The **third** and **fourth** figures depict the likelihood predicted by the classifiers for the whole space for the tree- and rbf-based classifier respectively.

provide likelihood estimates, as seen in the right figures.

RBF-ManifoldBoost: Tree functions are not the only possible approximation to the "gradient". Step 4 in algorithm 1 can be modified so that R radial basis function of width σ, each with a weight w_r and centered in a datapoint are chosen as approximation. Again, a BFGS step can be performed to improve the loss by fitting the weights w_r. Algorithm 2 describes this.

Algorithm 2 RBF ManifoldBoost Algorithm

1: $F_0(x) = 1/2[\log(1+\overline{y}) - \log(1-\overline{y})]$
2: **for** $m = 1$ to M **do**
3: Compute G_V as in (8)
4: Choose R RBFs greedily to minimize $\sum_i (G_V(x_i) - \sum_r w_r RBF_{r,\sigma}(x_i)])^2$
5: Find $\{w_r\}$ using BFGS and $\frac{\partial V}{\partial w_r}$
6: $F_m(x) = F_{m-1}(x) + \nu \sum_r w_r RBF_{r,\sigma}(x)$
7: **end for**

Complexity: The procedure itself is linear in $n = l+u$, in the Laplacian neighborhood K, the dimensionality of x and the number of rounds. The complexity of the algorithm depends then on the base regressor, and the computation of the Laplacian matrix. Influence trimming can also be used to get tenfold speedups (Friedman, 1999), although the algorithm is still linear in the number of datapoints.

4. Unsupervised Boosting

The essential step in semi-supervised learning is the observation that similar data items should tend to have similar labels, which means that semi-supervised learning method should be capable of clustering. Our framework can naturally be extended to unsupervised learning, where one wishes to cluster data and the choice of label for a cluster is arbitrary. As there are no labeled data, the first term in equation 6 becomes zero and the problem is,

$$F^* = \arg\min_{F\in\mathcal{H}} \sum_{i,j} F(x_i)\mathcal{L}^{\mathcal{M}}_{i,j}F(x_j) \qquad (10)$$

under the constraints $\sum_i F(x_i) = 0$, $\sum_i F(x_i)^2 = N$ (this is a form of spectral clustering problem, see (Sindhwani et al., 2006); without the constraints, the problem is ill-posed). Our formalism yields a greedy method for this problem, rather than the usual generalized eigenvalue problem. To manage constraints, we use the Augmented Lagrangian method (Bertsekas, 1996), which adds a penalty in each round for constraint violations in the unconstrained problem. We choose F^* to be

$$\arg\min_{F\in\mathcal{H}} \sum_{i,j} F(x_i)\mathcal{L}_{i,j}F(x_j) + \lambda_1^m \sum_i F(x_i) + \dots$$
$$\lambda_2^m \left(\sum_i F(x_i)^2 - N\right) + \frac{c_1^m}{2}\left(\sum_i F(x_i)\right)^2 + \dots$$
$$\frac{c_2^m}{2}\left(\sum_i F(x_i)^2 - N\right)^2 \qquad (11)$$

for non-decreasing sequences $\{c_1^m, c_2^m\}_{m=1}^M$. After each round, the values of the Lagrange multipliers are increased by the constraint violation (Bertsekas, 1996) $\lambda_1^{m+1} \leftarrow \lambda_1^m + c_1^m(\sum_i F(x_i))$ and $\lambda^{m+1} \leftarrow \lambda_1^m + c_1^m(\sum_i F(x_i)^2 - N)$. As before, BFGS is applied in each round. Algorithm 3 describes the tree-based version.

The algorithm converges to a local minimum of the constrained problem. This formulation, unlike ISOMAP, naturally takes care of out-of-sample evaluation. Compared to (Sindhwani et al., 2006), the computational complexity is greatly reduced. On the other hand the solution is greedy, and there is no straightforward term for controlling the complexity of the function in the ambient space; this is achieved through the depth of the trees used in the algorithm.

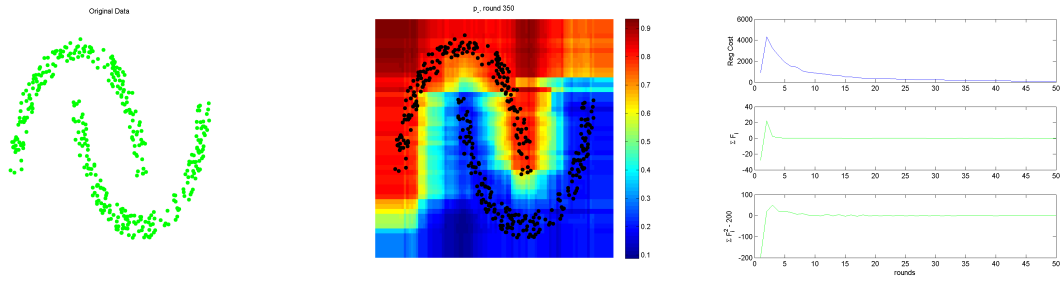


Figure 2. **Unsupervised learning**. The framework can be extended to unsupervised learning. The same problem as in figure 1 is presented without labels (**left** image). The result of the Tree-based algorithm is shown in the **center** image. The plots on the **right** show the constraint violations go to zero as learning progresses. This figure is best viewed in color.

Algorithm 3 Unsupervised Tree ManifoldBoost Algorithm

1: Initialize $F_0(x)$ randomly, with zero mean and low variance.
2: **for** $m = 1$ to M **do**
3: Compute G_V of the penalized, uncontrained problem.
4: Obtain regression tree $\{R_{m,s}\}$ by minimizing $\sum_i (G_V(x_i) - \sum_s \eta_{m,s}[x_i \in R_{m,s}])^2$
5: Find $\{\eta_{m,s}\}$ using BFGS and fixing $\{R_{m,s}\}$
6: $F_m(x) = F_{m-1}(x) + \sum_s \eta_{m,s}[x \in R_{m,s}]$
7: Update Lagrange multipliers using the constrain violations.
8: **end for**

5. Multiclass Case

Algorithm 1 can be extended to K-class problems by introducing a multinomial cost in equation 1,

$$\psi(\{y^{(c)}, F^{(c)}(x)\}_{c=1}^{C}) = -\sum_{c'=1}^{C} y^{(c')} \log p^{(c')}(x) \quad (12)$$

where $p^{(c)}(x)$ represents the belief example x belongs to class c, and $y^{(c)}$ is a binary variable which is one if example x belongs to class c. As in (Friedman, 1999) we use the symmetric multiple logistic transform

$$p^{(c)}(x) = \exp F^{(c)}(x) \cdot \left(\sum_{c'=1}^{C} \exp F^{(c')}(x) \right)^{-1} \quad (13)$$

Smoothness of $F^{(c)}$ is enforced by defining the cost $V(\{F^{(c)}\})$ to be

$$\frac{1}{l} \sum_i \psi(\{y_i^{(c)}, F^{(c)}(x_i)\}_{c=1}^{C}) + \dots$$
$$\frac{1}{C \cdot (l+u) \cdot K} \sum_{c'} \sum_{i,j} \gamma_{\mathcal{M}}^{(c')} F^{(c')}(x_i) \mathcal{L}_{i,j}^{\mathcal{M}} F^{(c')}(x_j)$$

The inner product of $f^{(c)}$ with gradient of V becomes, for class c,

$$\begin{aligned} \langle G_V^{(c)}(F_m^{(c)}), f \rangle &= \frac{1}{l} \sum_i (-y_i^{(c)} + p_m^{(c)}(x_i)) f(x_i) \quad (14) \\ &+ \frac{2\gamma_c^{\mathcal{M}}}{C \cdot (l+u)^2} \sum_{i,j} f(x_i) \mathcal{L}_{i,j}^{\mathcal{M}} F_m^{(c)}(x_j) \end{aligned}$$

Now one regression tree is fitted per class at each round to approximate each descent direction. As in the two class problem, the S regions $\{R_{m,s}^{(c)}\}_{s=1}^{S}$ defined by the terminal nodes are fixed, and the parameters $\eta_{m,s}^{(c)}$ for regions in each tree are learned in order to minimize the total cost V. We use a couple of BFGS iterations per round to find these parameters. In order to do this, the derivatives of the cost with respect to $\eta_{m,s}^{(c)}$ have to be computed.

Once the final $\{F_M^{(c)}(x)\}$ are computed, the probability for a given example of each label can be estimated and thus the label can be classified as $\hat{c}(x_i) = \arg\min_c \sum_{c'=1}^{C} k_{c,c'} p_M^{(c')}(x)$ for costs $k_{c,c'}$ when label c is assigned when label c' is correct. The complexity of this algorithm is also linear in the number of classes, but it scales highly sub-linearly with the number of rounds M when inluence trimming is used (Friedman, 1999).

6. Experiments and Discussion

6.1. Comparison to Other Regularized Boosting Algorithms

Kegl et. al. (Kégl & Wang, 2005) introduce RegBoost, an extension to AdaBoost which incorporates a weight decay that depends on a Laplacian regularizer. Our approach is different several senses: first, ours is based on the GradientBoost framework while theirs is based on AdaBoost, second, in the sense that ManifoldBoost does not require manifold-regularized base learners. This makes their approach limited in

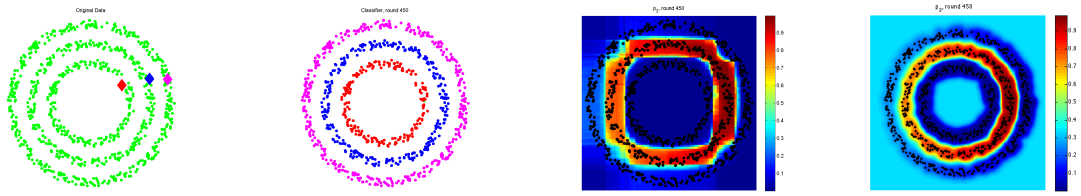


Figure 3. The framework also handles **multiclass learning**. **Left** figure shows another toy example (three rings): the unlabeled datapoints are depicted in green, the diamonds represent the labeled examples (one for each class). The output of the algorithm is depicted on the **center** figure. The blue classification function $F^2(x)$ is shown on the **right**. This figure is best viewed in color.

Algorithm 4 K-Tree ManifoldBoost Algorithm

Let $p_0^{(c)}$ be the frequencies of each class c.
$F_0^{(c)}(x) = \log p_0^{(c)} - \frac{1}{C}\sum_{c'=1}^{C} \log p_0^{(c)}$
for $m = 1$ to C **do**
  Compute $p_m^{(c)}(x)$ as in eq. 13 for all c.
  for $c = 1$ to C **do**
    Compute $G_V^{(c)}$ as in (14)
    Obtain regression tree $\{R_{m,s}^{(c)}\}$ by minimizing $\sum_i (G_V^{(c)}(x_i) - \sum_s \eta_{m,s}^{(c)} I[x_i \in R_{m,s}^{(c)}])^2$
  end for
  Find $\{\eta_{m,s}^{(c)}\}$ using BFGS and $\frac{\partial V}{\partial \eta_{m,s}}$, and fixing $\{R_{m,s}^{(c)}\}$ for all c.
  $F_m^{(c)}(x) = F_{m-1}^{(c)}(x) + \sum_s \eta^{(}(c))_{m,s} I[x \in R_{m,s}^{(c)}]$ for all c.
end for

the types of learners to be used (they use stumps only). Also, the ensemble classifier should be smooth on the manifold, but regularizing each of the base learners may result in over-smoothing of the overall solution.

We compare our results with (Kégl & Wang, 2005) on standard UCI benchmark datasets. Whenever possible we tried to use the same configuration as (Kégl & Wang, 2005)[1]. We set number of nearest neighbors $K = 8$ and used binary weights to compute the graph Laplacian. We used regression trees of fixed depth 3 as learners. The datasets were normalized to zero mean and unit variance. The learning rate was set to $\nu = 0.1$ after (Friedman, 1999). Only γ was explored for different values. We used 5-fold cross validation for determining parameters and 10-fold cross validation for error estimation. Table 1 compares our performance with that of AdaBoost, RegBoost, and (M. Belkin & Niyogi, 2004) as reported in (Kégl & Wang, 2005).

In the fully supervised problems, there is a difference in performance for the Sonar dataset, an impovement in the Ionosphere dataset, and a very slight decrease in performance in the Pima Indians dataset with respect to RegBoost, well within a standard deviation. It should be noted that the variance in the performance of the algorithm is consistently smaller for our algorithm. (Kégl & Wang, 2005) also tests the algorithm under semi-supervision, using 100 labeled and 251 unlabeled examples. We ran our algorithm under the same conditions, using the stumps to prevent overfitting. In this case our algorithm outperforms (Kégl & Wang, 2005) and (M. Belkin & Niyogi, 2004), as our mean performance over 10 runs is more than a standard deviation above theirs. No variance of results is reported in (Kégl & Wang, 2005).

(Chen & Wang, 2008) proposes an interesting alternative approach to regularized boosting based on the more traditional framework of boosting "weak" learners outlined in (Mason et al., 2000). As a consequence, they need to assign pseudo-class labels to unlabeled data (labels assigned with the current $F_t(x)$) while learning the ensemble. In contrast, ManifoldBoost uses base regressors to measure the confidence of the prediction and does not commit to $\{-1, +1\}$ classification at each step. Smoothing this seems more natural in a formulation that penalizes second derivatives (Laplacian cost).

6.2. Comparison to Other Semi-Supervised Learning Algorithms

We measured the performance of our two-class RBF-ManifoldBoost algorithm on the SSL data sets, a standard benchmark for semi-supervised learning problems introduced in (Chapelle et al., 2006), and compared with the 14 other state-of-the-art semi-supervised learning algorithms discussed there. In table 2 we present results for five data sets, 2 of which are cluster-like and 3 manifold-like. On the manifold-like data sets, we are at the state of the art and no single algorithm does uniformly better than us. On the cluster-

[1]The breast cancer dataset was not used because (Kégl & Wang, 2005) does not explain what metric they use for categorical data in the Lalacian, making any comparison meaningless.

Table 1. Performance results of TREE-MANIFOLDBOOST and different boosting approaches on standard UCI datasets, as reported in (Kégl & Wang, 2005) (variance in parenthesis)

Algorithm	Ionosphere Train / Test	Pima Indians Train / Test	Sonar Train / Test	Ionosphere (semisup.)
AdaBoost	0% / 9.2% (7.1)	10.9% / 25.3% (5.3)	0% / 32.5% (19.8)	-
RegBoost	0% / 7.7% (6.0)	16.0% / 23.3% (6.8)	0% / 29.8% (18.8)	12%
Tree-ManifoldBoost	0% (0) / 6.5% (4.8)	6.5% (0.5) / 24.0% (5.2)	0% (0) / 18.7% (6.1)	10.4% (0.8)
Belkin et al., 04 [2]	- / -	- / -	- / -	18%

like data sets, our performance is good compared to most other regularization-based and manifold learners but is not as good as the specialized clustering algorithms Cluster-Kernel and SGT (Spectral Graph Transducer).

Parameter search was performed following section 21.2.5 of (Chapelle et al., 2006) when possible, using the same ranges for γ, the RBF width σ, distance metric, K, etc. For the base regressors, we used $R \in \{15, 30\}$ as the numbers of RBFs, and $M = 500$ rounds. The learning rate was again chosen as $\nu = 0.1$. These parameters were obtained in small-scale experiments and then fixed. Results reported are the means over the different splits.

The running times for a MATLAB implementation on a 2 GHz machine was in the order of minutes. Unfortunately, running times for the other algorithms were not reported in (Chapelle et al., 2006).

6.3. `SecStr` Data Set

We also ran experiments on the `SecStr` data set (Chapelle et al., 2006), which is a problem of predicting the secondary structure of protein sequences from their amino acid chains. This is a large-scale and challenging data set with 83,000 labeled and 1.2 million unlabeled examples. Semi-supervised algorithms have made little improvement to this benchmark so far (Table 3), and the best result is the manifold-regularized learning algorithm (Sindhwani et al., 2006), which yields a 29% error rate on a subset of the data with 10,000 labeled and 73,000 unlabeled examples.

Tree-ManifoldBoost with $\gamma \in \{0, 10^{-5}, 10^{-3}, 0.1, 1\}$ achieved similar performance on the same subset in approximately 45 minutes of training time (after computing the Laplacian matrix). We used stumps, $K = 6$ and $\nu = 0.05$. No model selection was performed. We used as similarity measure the Hamming distance between the best alignment of sequences. The results reported are the mean over the 10 splits.

When we used the whole dataset (1.3 million sequences) with $\gamma \in \{0, 10^{-3}, 1\}$, there is virtually no performance improvement. This may be due to the smaller parameter search space, or to peculiarities of the dataset. When we analysed the structure of the manifold on the labeled subset, we observed that almost 20% of sequences at distance 1 (that is, shifted by one position to the left or right) had a different label. Therefore the manifold assumption is not strong on this set.

As far as we know, this is the first time results are reported on the complete `SecStr` dataset. Our algorithm is efficient and therefore can handle datasets of this size. Learning time is in the order of three hours for 1.3 million samples (leaving aside the computation of the graph Laplacian, which took significantly longer)

Table 3. Error rates on `SecStr` dataset. l is the number of labeled examples.

l	100	1000	10000
SVM	44.59	33.71	
Cluster Kernel	42.95	34.03	
QC randsub (CMN)	42.32	40.84	
QC smartonly (CMN)	42.14	40.71	
QC smartsub (CMN)	42.26	40.84	
Boosting (assemble)			32.21
LapRLS	42.59	34.17	28.55
LapSVM	43.42	33.96	28.53
Tree-ManifoldBoost (83K)	42.70	33.43	28.96
Tree-ManifoldBoost (1.3M)	43.28	33.42	29.07

7. Conclusion

We have presented a new boosting framework for regularized learning in a greedy, stage-wise procedure. It is flexible enough to handle the whole range of supervision, from fully supervised classification to unsupervised clustering. The framework is general, accepts many different function approximation techniques, is

Table 2. Error rates for data sets from (Chapelle et al., 2006). l is the number of labeled examples.

	Manifold-like						Cluster-like			
	$l = 10$			$l = 100$			$l = 10$		$l = 100$	
	BCI	Digit1	USPS	BCI	Digit1	USPS	g241c	g241d	g241c	g241d
1-NN	49.00	13.65	16.66	48.67	3.89	5.81	47.88	46.72	43.93	42.45
SVM	49.85	30.60	20.03	34.31	5.53	9.75	47.32	46.66	23.11	24.64
21.2.8 MVU + 1-NN	47.95	14.42	23.34	47.89	2.83	6.50	47.15	45.56	43.01	38.20
21.2.8 LEM + 1-NN	48.74	23.47	19.82	44.83	6.12	7.64	44.05	43.22	40.28	37.49
21.2.4 QC + CMN	50.36	9.80	13.61	46.22	3.15	6.36	39.96	46.55	22.05	28.20
21.2.6 Discrete Reg.	49.51	12.64	16.07	47.67	2.77	4.68	49.59	49.05	43.65	41.65
21.2.1 TSVM	49.15	17.77	25.20	33.25	6.15	9.77	24.71	50.08	18.46	22.42
21.2.1 SGT	49.59	8.92	25.36	45.03	2.61	6.80	22.76	18.64	17.41	9.11
21.2.10 Cluster-Kernel	48.31	18.73	19.41	35.17	3.79	9.68	48.28	42.05	13.49	4.95
21.2.3 Data-Dep. Reg.	50.21	12.49	17.96	47.47	2.44	5.10	41.25	45.89	20.31	32.82
21.2.11 LDS	49.27	15.63	17.57	43.97	3.46	4.96	28.85	50.63	18.04	23.74
21.2.5 Laplacian RLS	48.97	5.44	18.99	31.36	2.92	4.68	43.95	45.68	24.36	26.46
21.2.7 CHM (normed)	46.90	14.86	20.53	36.03	3.79	7.65	39.03	43.01	24.82	25.67
RBF-ManifoldBoost	47.12	19.42	19.97	32.17	4.29	6.65	42.17	42.80	22.87	25.00

efficient and fast at each round of boosting, handles multi-class and wholly unsupervised problems, and produces results at the state of the art. We are working on understanding important aspects of the algorithm, in particular, generalization, error bounds, convergence and local minima.

References

Bertsekas, D. P. (1996). *Constrained optimization and lagrange multiplier methods*. Academic Press.

Bickel, P., & Li, B. (2007). Local polynomial regression on unknown manifolds. *IMS Lecture Notes Monograph Series: Complex Datasets and Inverse Problems: Tomography, Networks and Beyond* (pp. 177–186).

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised Learning*. MIT Press.

Chen, K., & Wang, S. (2008). Regularized boost for semi-supervised learning. *NIPS 20* (pp. 281–288).

Donoho, D. L., & Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci. USA*, *100*, 5591–5596.

Dyn, N., Levin, D., & Rippa, S. (1986). Numerical procedures for surface fitting of scattered data by radial functions. *SIAM Journal on Scientific and Statistical Computing*, *7*, 639–659.

Friedman, J. (1999). *Greedy function approximation: a gradient boosting machine* (Technical Report). Dept. of Statistics, Stanford University.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *ICML* (pp. 200–209).

Kégl, B., & Wang, L. (2005). Boosting on manifolds: Adaptive regularization of base classifiers. *NIPS 17* (pp. 665–672).

Lafferty, J., & Wasserman, L. (2007). Statistical analysis of semi-supervised regression. *NIPS 20* (pp. 801–808).

M. Belkin, I. M., & Niyogi, P. (2004). Regression and regularization on large graphs. *COLT* (pp. 824–831).

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Boosting algorithms as gradient descent. *NIPS 12* (pp. 512–518).

Niyogi, P. (2008). *Manifold regularization and semi-supervised learning: Some theoretical analyses* (Technical Report). University of Chicago. Technical Report TR-2008-01, Computer Science Dept.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Sindhwani, V., Belkin, M., & Niyogi, P. (2006). The geometric basis of semi-supervised learning. In Chapelle, Schoelkopf and Zien (Eds.), *Semi-supervised learning*. MIT Press.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Zhu, X. (2006). *Semi-supervised learning literature review* (Technical Report). University of Wisconsin.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *ICML* (pp. 912–919).

Random Classification Noise Defeats All Convex Potential Boosters

Philip M. Long PLONG@GOOGLE.COM
Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043

Rocco A. Servedio ROCCO@CS.COLUMBIA.EDU
Computer Science Department, Columbia University, New York, NY 10027

Abstract

A broad class of boosting algorithms can be interpreted as performing coordinate-wise gradient descent to minimize some potential function of the margins of a data set. This class includes AdaBoost, LogitBoost, and other widely used and well-studied boosters. In this paper we show that for a broad class of convex potential functions, any such boosting algorithm is highly susceptible to random classification noise. We do this by showing that for any such booster and any nonzero random classification noise rate η, there is a simple data set of examples which is efficiently learnable by such a booster if there is no noise, but which cannot be learned to accuracy better than 1/2 if there is random classification noise at rate η. This negative result is in contrast with known branching program based boosters which do not fall into the convex potential function framework and which can provably learn to high accuracy in the presence of random classification noise.

1. Introduction

1.1. Background

Much work has been done on viewing boosting algorithms as greedy iterative algorithms that perform a coordinate-wise gradient descent to minimize a potential function of the margin of the examples, see e.g. [3, 12, 19, 7, 18, 2]. In this framework every potential function ϕ defines an algorithm that may possibly be a boosting algorithm; we denote the algorithm corresponding to ϕ by $\mathcal{B}_\phi$. For example, AdaBoost [11] and its confidence-rated generalization [20] may be viewed as the algorithm $\mathcal{B}_\phi$ corresponding to the potential function $\phi(z) = e^{-z}$. The MadaBoost algorithm of Domingo and Watanabe [5] may be viewed as the algorithm $\mathcal{B}_\phi$ corresponding to

$$\phi(z) = \begin{cases} 1 - z & \text{if } z \leq 0 \\ e^{-z} & \text{if } z > 0. \end{cases} \quad (1)$$

(We give a more detailed description of exactly what the algorithm $\mathcal{B}_\phi$ is for a given potential function ϕ in Section 2.2.)

1.2. Motivation: noise-tolerant boosters?

It has been widely observed that AdaBoost can suffer poor performance when run on noisy data, see e.g. [10, 17, 4]. The most commonly given explanation for this is that the exponential reweighting of examples which it performs (a consequence of the exponential potential function) can cause the algorithm to invest too much "effort" on correctly classifying noisy examples. Boosting algorithms such as MadaBoost [5] and LogitBoost [12] based on a range of other potential functions have subsequently been provided, sometimes with an explicitly stated motivation of rectifying AdaBoost's poor noise tolerance. However, we are not aware of rigorous results establishing provable noise tolerance for any boosting algorithms that fit into the potential functions framework, even for mild forms of noise such as random classification noise (henceforth abbreviated RCN) at low noise rates. This motivates the following question: are Adaboost's difficulties in dealing with noise due solely to its exponential weighting scheme, or are these difficulties inherent in the potential function approach to boosting?

1.3. Our results: convex potential boosters cannot withstand random classification noise

This paper shows that the potential function boosting approach provably cannot yield learning algorithms that tolerate even low levels of random classification noise when convex potential functions are used. More

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

precisely, we exhibit a fixed natural set of base classifiers $h_1, \ldots, h_n$ and show that for every convex function ϕ satisfying some very mild conditions and every noise rate $\eta > 0$, there is a multiset S of labeled examples such that the following holds:

- There is a linear separator $\mathrm{sgn}(\alpha_1 h_1 + \cdots + \alpha_n h_n)$ over the base classifiers $h_1, \ldots, h_n$ that correctly labels every example in S with margin $\gamma > 0$ (and hence it is easy for a boosting algorithm trained on S to efficiently construct a final hypothesis that correctly classifies all examples in S). However,
- When the algorithm $\mathcal{B}_\phi$ is run on the distribution $\mathcal{D}_{\eta,S}$, it constructs a classifier that has error rate 1/2 on the examples in S. Here $\mathcal{D}_{\eta,S}$ is the uniform distribution over S but where examples are corrupted with random classification noise at rate η, i.e. labels are independently flipped with probability η.

This result shows that random classification noise can cause convex potential function boosters to fail in a rather strong sense. We note that as discussed in Section 7, there do exist known boosting algorithms [13, 16] that can tolerate random classification noise, and in particular can efficiently achieve perfect accuracy on S, after at most $\mathrm{poly}(1/\gamma)$ stages of boosting, when run on $\mathcal{D}_{\eta,S}$ in the scenario described above.

Recently Bartlett and Traskin proved that the AdaBoost algorithm is consistent if it is stopped after a suitable number of iterations, given certain conditions on a random source generating the data [1]. Our analysis does not contradict theirs because the source in our construction does not satisfy Condition 1 of their paper. To see why this is the case it is useful, as has become customary, to think of the contribution that a given example makes to the potential as a "loss" paid by the learning algorithm. Informally, Condition 1 from [1] requires linear combinations of base classifier predictions to have total loss arbitrarily close to the best possible loss for any measurable function. Our analysis takes advantage of the fact that, for linear combinations of base classifiers with a convex loss function, large-margin errors are especially egregious: we present the learner with a choice between a lot of cheap errors and relatively few expensive errors. If optimization were to be performed over all measurable functions, roughly speaking, it would be possible to make all errors cheap.

Though the analysis required to establish our main result is somewhat delicate, the actual construction is quite simple and admits an intuitive explanation (see Section 4.2). For every convex potential function ϕ we use the same set of only $n = 2$ base classifiers (these are confidence-rated base classifiers which output real values in the range $[-1, 1]$), and the multiset S contains only three distinct labeled examples; one of these occurs twice in S, for a total multiset size of four. We expect that many other constructions which similarly show the brittleness of convex potential boosters to random classification noise can be given. We describe experiments with one such construction that uses Boolean-valued weak classifiers rather than confidence-rated ones in Section 6.

2. Background and Notation

Throughout the paper X will denote the instance space. $\mathcal{H} = \{h_1, \ldots, h_n\}$ will denote a fixed finite collection of *base classifiers* over X, where each base classifier is a function $h_i : X \to [-1, 1]$; i.e. we shall work with confidence-rated base classifiers. $S = (x^1, y^1), \ldots, (x^m, y^m) \in (X \times \{-1, 1\})^m$ will denote a multiset of m examples with binary labels.

2.1. Convex potential functions

We adopt the following natural definition which, as we discuss in Section 5, captures a broad range of different potential functions that have been studied.

Definition 1 *We say that $\phi : \mathbf{R} \to \mathbf{R}$ is a* convex potential function *if ϕ satisfies the following properties:*

1. *ϕ is convex and nonincreasing and $\phi \in C^1$ (i.e. ϕ is differentiable and ϕ' is continuous);*
2. *$\phi'(0) < 0$ and $\lim_{x \to +\infty} \phi(x) = 0$.*

2.2. Convex potential boosters

Let ϕ be a convex potential function, $\mathcal{H} = \{h_1, \ldots, h_n\}$ a fixed set of base classifiers, and $S = (x^1, y^1), \ldots, (x^m, y^m)$ a multiset of labeled examples.

Similarly to Duffy and Helmbold [6, 7], we consider an iterative algorithm which we denote $\mathcal{B}_\phi$. The algorithm performs a coordinatewise gradient descent through the space of all possible coefficient vectors for the weak hypotheses, in an attempt to minimize the convex potential function of the margins of the examples. We now give a more precise description of how $\mathcal{B}_\phi$ works when run with $\mathcal{H}$ on S.

Algorithm $\mathcal{B}_\phi$ maintains a vector $(\alpha_1, ..., \alpha_n)$ of voting weights for the base classifiers $h_1, ..., h_n$. The weights are initialized to 0. In a given round T, the algorithm chooses an index i_T of a base classifier, and modifies

the value of α_{i_T}. If α_{i_T} had previously been zero, this can be thought of as adding base classifier number i_T to a pool of voters, and choosing a voting weight.

Let $F(x;\alpha_1,...,\alpha_n) = \sum_{i=1}^n \alpha_i h_i(x)$ be the master hypothesis that the algorithm has constructed prior to stage T (so at stage $T = 1$ the hypothesis F is identically zero.) We write $P_{\phi,S}$ to denote the "global" potential function over S

$$P_{\phi,S}(\alpha_1,...,\alpha_n) = \sum_{i=1}^m \phi(y^i F(x^i;\alpha_1,...,\alpha_n)) \qquad (2)$$

which represents the overall potential of a hypothesis vector $(\alpha_1,\ldots,\alpha_n)$ on the sample S. It is easy to check that this is a convex function from $\mathbf{R}^n$ (the space of all possible $(\alpha_1,\ldots,\alpha_n)$ coefficient vectors for F) to $\mathbf{R}$.

In stage T the algorithm $\mathcal{B}_\phi$ first chooses a base classifier by choosing i_T to be the index $i \in [n]$ which maximizes

$$-\frac{\partial}{\partial \alpha_i} P_{\phi,S}(\alpha_1,...,\alpha_n),$$

and then choosing a new value of α_{i_T} in order to minimize $P_{\phi,S}(\alpha_1,...,\alpha_n)$ for the resulting $\alpha_1,...,\alpha_n$. Thus, in the terminology of [6] we consider "un-normalized" algorithms which preserve the original weighting factors α_1, α_2, etc. The AdaBoost algorithm is an example of an algorithm that falls into this framework, as are the other algorithms we discuss in Section 5. Note that the fact that $\mathcal{B}_\phi$ can determine the exactly optimal weak classifier to add in each round errs on the side of pessimism in our analysis.

In our analysis, we will consider the case in which $\mathcal{B}_\phi$ as being run on a distribution $\mathcal{D}_{\eta,S}$ obtained by starting with a finite multiset of examples, and adding independent misclassification noise. One can naturally extend the definition of $\mathcal{B}_\phi$ to apply to probability distributions over $X \times \{-1,1\}$ by extending the definition of potential in (2) as follows

$$P_{\phi,\mathcal{D}}(\alpha_1,...,\alpha_n) = \mathbf{E}_{(x,y)\sim\mathcal{D}}(\phi(yF(x;\alpha_1,...,\alpha_n))). \qquad (3)$$

For rational values of η, running $\mathcal{B}_\phi$ on (3) for $\mathcal{D} = \mathcal{D}_{\eta,S}$ is equivalent to running $\mathcal{B}_\phi$ over a finite multiset in which each element of S occurs a number of times proportional to its weight under $\mathcal{D}$.

2.3. Boosting

Fix a classifier $c : X \to \{-1,1\}$ and a multiset $S = (x^1,y^1),\ldots,(x^m,y^m)$ of labeled examples. We say that a set of base classifiers $\mathcal{H} = \{h_1,\ldots,h_n\}$ is *boostable with respect to c and S* if there is a vector $\alpha \in \mathbf{R}^n$ such that for all $i = 1,\ldots,m$, we have

$$\mathrm{sgn}[\alpha_1 h_1(x^i) + \cdots + \alpha_n h_n(x^i)] = y^i.$$

If $\gamma > 0$ is such that

$$\frac{y^i \cdot \big(\alpha_1 h_1(x^i) + \cdots + \alpha_n h_n(x^i)\big)}{\sqrt{\alpha_1^2 + \cdots + \alpha_n^2}} \geq \gamma$$

for all i, we say that $\mathcal{H}$ is *boostable w.r.t. c and S with margin γ*.

It is well known that if $\mathcal{H}$ is boostable w.r.t. c and S with margin γ, then a range of different boosting algorithms (such as AdaBoost) can be run on the noise-free data set S to efficiently construct a final classifier that correctly labels every example in S. As one concrete example, after $O(\frac{\log m}{\gamma^2})$ stages of boosting AdaBoost will construct a linear combination $F(x) = \sum_{i=1}^n \gamma_i h_i(x)$ of the base classifiers such that $\mathrm{sgn}(F(x^i)) = y^i$ for all $i = 1,\ldots,m$; see [11, 20] for details.

2.4. Random classification noise and noise-tolerant boosting

Random classification noise is a simple, natural, and well-studied model of how benign (nonadversarial) noise can affect data. Given a multiset S of labeled examples and a value $0 < \eta < \frac{1}{2}$, we write $\mathcal{D}_{\eta,S}$ to denote the distribution corresponding to S corrupted with random classification noise at rate η. A draw from $\mathcal{D}_{\eta,S}$ is obtained by drawing (x,y) uniformly at random from S and independently flipping the binary label y with probability η.

We say that an algorithm $\mathcal{B}$ is a *boosting algorithm which tolerates RCN at rate η* if $\mathcal{B}$ has the following property. Let c be a target classifier, S be a multiset of m examples, and $\mathcal{H}$ be a set of base classifiers such that $\mathcal{H}$ is boostable w.r.t. c and S. Then for any $\epsilon > 0$, if $\mathcal{B}$ is run with $\mathcal{H}$ as the set of base classifiers on $\mathcal{D}_{\eta,S}$, at some stage of boosting $\mathcal{B}$ constructs a classifier g which has accuracy

$$\frac{|\{(x^i,y^i) \in S : g(x^i) = y^i\}|}{m} \geq 1 - \eta - \epsilon.$$

The accuracy rate above is in some sense optimal, since known results [13] show that no "black-box" boosting algorithm can be guaranteed to construct a classifier g whose accuracy exceeds $1-\eta$ in the presence of RCN at rate η. As we discuss in Section 7, there are known boosting algorithms [13, 16] which can tolerate RCN at rate η for any $0 < \eta < 1/2$. These algorithms, which do not follow the convex potential function approach but instead build a branching program over the base classifiers, use $\mathrm{poly}(1/\gamma, \log(1/\epsilon))$ stages to achieve accuracy $1-\eta-\epsilon$ in the presence of RCN at rate η if $\mathcal{H}$ is boostable w.r.t. c and S with margin γ.

3. Main Result

As was just noted, there do exist boosting algorithms (based on branching programs) that can tolerate RCN. Our main result is that no convex potential function booster can have this property:

Theorem 2 *Fix any convex potential function ϕ. For any noise rate $0 < \eta < 1/2$, the algorithm $\mathcal{B}_\phi$ does not tolerate RCN at rate η.*

We obtain Theorem 2 as a direct consequence of the following stronger result, which shows that there is a simple RCN learning problem for which $\mathcal{B}_\phi$ will in fact misclassify half the examples in S.

Theorem 3 *Fix the instance space $X = [-1,1]^2 \subset \mathbf{R}^2$ and the set $\mathcal{H} = \{h_1(x) = x_1, h_2(x) = x_2\}$ of confidence-rated base classifiers over X.*

For any noise rate $0 < \eta < 1/2$ and any convex potential function ϕ, there is a target classifier c, a value $\gamma > 0$, and a multiset S of four labeled examples (three of which are distinct) such that (a) $\mathcal{H}$ is boostable w.r.t. c and S with margin γ, but (b) when $\mathcal{B}_\phi$ is run on the distribution $\mathcal{D}_{\eta,S}$, it constructs a classifier which misclassifies two of the four examples in S.

4. Proof of Theorem 3

We are given an RCN noise rate $0 < \eta < 1/2$ and a convex potential function ϕ.

4.1. The basic idea

Before specifying the sample S we explain the high-level structure of our argument. Recall from (3) that $P_{\phi,\mathcal{D}}$ is defined as

$$P_{\phi,\mathcal{D}}(\alpha_1,\alpha_2) = \sum_{(x,y)} \mathcal{D}_{\eta,S}(x,y)\phi(y(\alpha_1 x_1 + \alpha_2 x_2)). \quad (4)$$

As noted in Section 2.2 the function $P_{\phi,\mathcal{D}}(\alpha_1,\alpha_2)$ is convex. It follows immediately from the definition of a convex potential function that $P_{\phi,\mathcal{D}}(\alpha_1,\alpha_2) \geq 0$ for all $(\alpha_1,\alpha_2) \in \mathbf{R}^2$.

The high-level idea of our proof is as follows. We shall construct a multiset S of four labeled examples in $[-1,1]^2$ (actually in the unit disc $\{x : \|x\| \leq 1\} \subset \mathbf{R}^2$) such that there is a global minimum (α_1^*,α_2^*) of the corresponding $P_{\phi,\mathcal{D}}(\alpha_1,\alpha_2)$ which has the following two properties:

1. **("high error")** The corresponding classifier $g(x) = \mathrm{sgn}(\alpha_1^* x_1 + \alpha_2^* x_2)$ misclassifies two of the points in S (and thus has error rate 1/2); and
2. **("steep slope")** At the point $(0,0)$, the directional derivative of $P_{\phi,\mathcal{D}}(\alpha_1,\alpha_2)$ in any direction orthogonal to (α_1^*,α_2^*) is not as steep as the directional derivative toward (α_1^*,α_2^*).

We now show that it suffices to establish these two properties to prove part (b) of Theorem 3.[1] Suppose we have such an S. Since $P_{\phi,\mathcal{D}}(\alpha_1,\alpha_2)$ depends only on the inner product between (α_1,α_2) and the (normalized) example vectors (yx_1, yx_2), it follows that rotating the set S around the origin by any fixed angle induces a corresponding rotation of the function $P_{\phi,\mathcal{D}}$, and in particular of its minima. (Note that we have used here the fact that every example point in S lies within the unit disc; this ensures that for any rotation of S each weak hypothesis x_i will always give outputs in $[-1,1]$ as required.) Consequently a suitable rotation of S to S' will result in the corresponding rotated function $P_{\phi,\mathcal{D}}$ having a global minimum at a vector which lies on one of the two coordinate axes (say a vector of the form $(0,\tau)$). If this is the case, then the "steep slope" property (2) ensures that the directional derivative at $(0,0)$ in this direction will be steepest, so the convex potential booster $\mathcal{B}_\phi$ will pick a base classifier corresponding to this direction (in this case h_2). Since a globally optimal weight vector is available in this direction (the vector of length $\sqrt{(\alpha_1^*)^2 + (\alpha_2^*)^2}$ is such a vector), $\mathcal{B}_\phi$ will select such a vector. Once it has achieved such a global optimum it will not change its hypothesis in any subsequent stage, and thus $\mathcal{B}_\phi$'s hypothesis will have error rate 1/2 on the points in the rotated set S' by the "high error" property (1).

4.2. The sample S

Now let us define the multiset S of examples. S consists of three distinct examples, one of which is repeated twice. (We shall specify the value of γ later and show that $0 < \gamma < \frac{1}{6}$.)

- S contains one copy of the example $x = (1,0)$ with label $y = +1$. (We call this the "large margin" example.)
- S contains two copies of the example $x = (\gamma, -\gamma)$ with label $y = +1$. (We call these examples the "penalizers" since they are the points that $\mathcal{B}_\phi$ will misclassify.)
- S contains one copy of the example $x = (\gamma, 5\gamma)$ with label $y = +1$. (We call this example the "puller" for reasons described below.)

[1] To prove part (a) we need to show that $\mathcal{H}$ is boostable w.r.t. some classifier c and S with margin γ, but as we shall see this is easy to achieve.

Thus all examples in S are positive. It is immediately clear that the classifier $c(x) = \mathrm{sgn}(x_1)$ correctly classifies all examples in S with margin $\gamma > 0$, so the set $\mathcal{H} = \{h_1(x) = x_1, h_2(x) = x_2\}$ of base classifiers is boostable w.r.t. c and S with margin γ. We further note that since $\gamma < \frac{1}{6}$, each example in S does indeed lie in the unit disc $\{x : \|x\| \leq 1\}$.

Let us give some intuition for why this set S has the "high error" property. The halfspace whose normal vector is $(1, 0)$ classifies all examples correctly, but the noisy (negative labeled) version of the "large margin" example causes a convex potential function to incur a very large cost for this hypothesis vector. Consequently a lower cost hypothesis can be obtained with a vector that points rather far away from $(1, 0)$. The "puller" example (whose y-coordinate is 5γ) outweights the two "penalizer" examples (whose y-coordinates are $-\gamma$), so it "pulls" the minimum cost hypothesis vector to point up into the first quadrant – in fact, so far up that the two "penalizer" examples are misclassified by the optimal hypothesis vector for the potential function ϕ.

In Section 4.3 below we make this intuition precise and show that there is a global minimum (α_1^*, α_2^*) of $P_{\phi,\mathcal{D}}$ for which $\alpha_1^* < \alpha_2^*$. This immediately implies that the corresponding classifier $g(x) = \mathrm{sgn}(\alpha_1^* x_1 + \alpha_2^* x_2)$ misclassifies the two copies of $(\gamma, -\gamma)$ in S and gives us the "high error" property (1). In Section 4.4 we show that this (α_1^*, α_2^*) moreover has the "steep slope" property (2).

4.3. The "high error" property: analyzing a global minimum of $P_{\phi,\mathcal{D}}$

Let $1 < N < \infty$ be such that $\eta = \frac{1}{N+1}$, so $1-\eta = \frac{N}{N+1}$.

We have that

$$\begin{aligned} P_{\phi,\mathcal{D}}(\alpha_1, \alpha_2) &= \sum_{(x,y)} \mathcal{D}_{\eta,S}(x, y)\phi(y(\alpha_1 x_1 + \alpha_2 x_2)) \\ &= \frac{1}{4} \sum_{(x,y)\in S} [(1-\eta)\phi(\alpha_1 x_1 + \alpha_2 x_2) \\ &\qquad + \eta\phi(-\alpha_1 x_1 - \alpha_2 x_2)]\,. \end{aligned}$$

It is clear that minimizing $4(N+1)P_{\phi,\mathcal{D}}$ is the same as minimizing $P_{\phi,\mathcal{D}}$ so we shall henceforth work with $4(N+1)P_{\phi,\mathcal{D}}$ since it gives rise to cleaner expressions. We have that $4(N+1)P_{\phi,\mathcal{D}}(\alpha_1, \alpha_2)$ equals

$$\begin{aligned} &\sum_{(x,y)\in S} [N\phi(\alpha_1 x_1 + \alpha_2 x_2) + \phi(-\alpha_1 x_1 - \alpha_2 x_2)] \\ &= N\phi(\alpha_1) + \phi(-\alpha_1) \\ &\quad +2N\phi(\alpha_1\gamma - \alpha_2\gamma) + 2\phi(-\alpha_1\gamma + \alpha_2\gamma) \\ &\quad +N\phi(\alpha_1\gamma + 5\alpha_2\gamma) + \phi(-\alpha_1\gamma - 5\alpha_2\gamma). \quad (5) \end{aligned}$$

Let $L_1(\alpha_1, \alpha_2)$ and $L_2(\alpha_1, \alpha_2)$ be defined as follows:

$$\begin{aligned} L_1(\alpha_1, \alpha_2) &\stackrel{\text{def}}{=} \frac{\partial}{\partial\alpha_1} 4(N+1)P_{\phi,\mathcal{D}}(\alpha_1, \alpha_2) \text{ and} \\ L_2(\alpha_1, \alpha_2) &\stackrel{\text{def}}{=} \frac{\partial}{\partial\alpha_2} 4(N+1)P_{\phi,\mathcal{D}}(\alpha_1, \alpha_2). \end{aligned}$$

For $B > 1$ to be fixed later, let us write $L_1(\alpha)$ to denote $L_1(\alpha, B\alpha)$ and similarly write $L_2(\alpha)$ to denote $L_2(\alpha, B\alpha)$. It is easy to verify that we have

$$\begin{aligned} L_1(\alpha) \;=\; & N\phi'(\alpha) - \phi'(-\alpha) + 2\gamma N\phi'(-(B-1)\alpha\gamma) \\ & -2\gamma\phi'((B-1)\alpha\gamma) + N\gamma\phi'((5B+1)\alpha\gamma) \\ & -\gamma\phi'(-(5B+1)\alpha\gamma) \end{aligned}$$

and

$$\begin{aligned} L_2(\alpha) \;=\; & -2\gamma N\phi'(-(B-1)\alpha\gamma) + 2\gamma\phi'((B-1)\alpha\gamma) \\ & +5\gamma N\phi'((5B+1)\alpha\gamma) - 5\gamma\phi'(-(5B+1)\alpha\gamma). \end{aligned}$$

We introduce the following function to help in the analysis of $L_1(\alpha)$ and $L_2(\alpha)$:

$$\text{for } \alpha \in \mathbf{R}, \qquad Z(\alpha) \stackrel{\text{def}}{=} N\phi'(\alpha) - \phi'(-\alpha).$$

Let us establish some basic properties of this function. Since ϕ is differentiable and convex, we have that ϕ' is a non-decreasing function. This is easily seen to imply that $Z(\cdot)$ is a non-decreasing function. We moreover have $Z(0) = \phi'(0)(N-1) < 0$. The definition of a convex potential function implies that as $\alpha \to +\infty$ we have $\phi'(\alpha) \to 0^-$, and consequently we have

$$\lim_{\alpha\to+\infty} Z(\alpha) = 0 + \lim_{\alpha\to+\infty} -\phi'(-\alpha) > 0,$$

where the inequality holds since $\phi'(\alpha)$ is a nonincreasing function and $\phi'(0) < 0$. Since ϕ' and hence Z is continuous, we have that over the interval $[0, +\infty)$ the function $Z(\alpha)$ assumes every value in the range $[\phi'(0)(N-1), -\phi'(0))$.

Next observe that we may rewrite $L_1(\alpha)$ and $L_2(\alpha)$ as

$$L_1(\alpha) = Z(\alpha) + 2\gamma Z(-(B-1)\alpha\gamma) + \gamma Z((5B+1)\gamma\alpha) \quad (6)$$

and

$$L_2(\alpha) = -2\gamma Z(-(B-1)\alpha\gamma) + 5\gamma Z((5B+1)\gamma\alpha). \quad (7)$$

In the rest of this section we shall show that there are values $\alpha > 0$, $0 < \gamma < 1/6$, $B > 1$ such that $L_1(\alpha) = L_2(\alpha) = 0$. Since $P_{\phi,\mathcal{D}}$ is convex, this will imply that $(\alpha_1^*, \alpha_2^*) \stackrel{\text{def}}{=} (\alpha, B\alpha)$ is a global minimum for the dataset constructed using this γ, as required.

Let us begin with the following claim which will be useful in establishing $L_2(\alpha) = 0$.

Claim 4 *For any $B \geq 1$ there is a finite value $\epsilon(B) > 0$ such that*

$$2Z(-(B-1)\epsilon(B)) = 5Z((5B+1)\epsilon(B)) < 0 \quad (8)$$

Proof: Fix any value $B \geq 1$. Recalling that $Z(0) = \phi'(0)(N-1) < 0$, at $\epsilon = 0$ the quantity $2Z(-(B-1)\epsilon)$ equals $2\phi'(0)(N-1) < 0$, and as ϵ increases this quantity does not increase. On the other hand, at $\epsilon = 0$ the quantity $5Z((5B+1)\epsilon)$ equals $5\phi'(0)(N-1) < 2\phi'(0)(N-1)$, and as ϵ increases this quantity increases to a limit, as $\epsilon \to +\infty$, which is at least $5(-\phi'(0))$. Since Z is continuous, there must be some $\epsilon > 0$ at which the two quantities are equal and are each at most $2\phi'(0)(N-1) < 0$. □

Observation 5 *The function $\epsilon(B)$ is a continuous and nonincreasing function of B for $B \in [0, \infty)$.*

Proof: The larger $B \geq 1$ is, the faster $-(B-1)\epsilon)$ decreases as a function of ϵ and the faster $(5B+1)\epsilon$ increases as a function of ϵ. Continuity of $\epsilon(\cdot)$ follows from continuity of $Z(\cdot)$. □

We now fix the value of B to be $B \stackrel{\text{def}}{=} 1+\gamma$, where the parameter γ will be fixed later. We shall only consider settings of $\alpha, \gamma > 0$ such that $\alpha\gamma = \epsilon(B) = \epsilon(1+\gamma)$; i.e. given a setting of γ, we shall take $\alpha = \frac{\epsilon(1+\gamma)}{\gamma}$. For any such α, γ we have

$$\begin{aligned} L_2(\alpha) &= (7) = \gamma[-2Z(-(B-1)\epsilon(1+\gamma)) \\ &\qquad +5Z((5B+1)\epsilon(1+\gamma))] = 0 \end{aligned}$$

where the last equality is by Claim 4. Now let us consider (6); our goal is to show that for some $\gamma > 0$ it is also 0. For any (α, γ) pair with $\alpha\gamma = \epsilon(1+\gamma)$, we have by Claim 4 that

$$\begin{aligned} &2\gamma Z(-(B-1)\gamma\alpha) + \gamma Z((5B+1)\gamma\alpha) \\ &= 2\gamma Z(-(B-1)\epsilon(1+\gamma)) + \gamma Z((5B+1)\epsilon(1+\gamma)) \\ &= 6\gamma Z((5B+1)\epsilon(1+\gamma)) \end{aligned}$$

where the second equality is by Claim 4. Plugging this into (6), we have that for $\alpha = \frac{\epsilon(1+\gamma)}{\gamma}$, the quantity $L_1(\alpha)$ equals 0 if and only if

$$\begin{aligned} Z\left(\frac{\epsilon(1+\gamma)}{\gamma}\right) &= -6\gamma Z((5B+1)\epsilon(1+\gamma)) \\ &= 6\gamma \cdot (-Z((6+5\gamma)\cdot\epsilon(1+\gamma))). \quad (9) \end{aligned}$$

Let us analyze (9). We first note that Observation 5 implies that $\epsilon(1+\gamma)$ is a nonincreasing function of γ for $\gamma \in [0, \infty)$. Consequently $\frac{\epsilon(1+\gamma)}{\gamma}$ is a decreasing function of γ, and since Z is a nonincreasing function, the LHS is a nonincreasing function of γ. Recall that at $\gamma = 0$ we have $\epsilon(1+\gamma) = \epsilon(1)$ which is some fixed finite positive value by Claim 4. So we have $\lim_{\gamma\to 0^+}$ LHS $= \lim_{x\to+\infty} Z(x) \geq -\phi'(0)$. On the other extreme, since $\epsilon(\cdot)$ is nonincreasing, we have

$$\lim_{\gamma\to+\infty} \text{LHS} \leq \lim_{\gamma\to+\infty} Z\left(\frac{\epsilon(1)}{\gamma}\right) = Z(0) = \phi'(0)(N-1) < 0.$$

So as γ varies through $(0, \infty)$, the LHS decreases through all values between $-\phi'(0)$ and 0.

On the other hand, at $\gamma = 0$ the RHS of (9) is clearly 0. Moreover the RHS is always positive for $\gamma > 0$ by Claim 4. Since the RHS is continuous (by continuity of $Z(\cdot)$ and $\epsilon(\cdot)$), this together with the previous paragraph implies that there must be some $\gamma > 0$ for which the LHS and RHS of (9) are the same positive value. So we have shown that there are values $\alpha > 0$, $\gamma > 0$, $B = 1+\gamma$ such that $L_1(\alpha) = L_2(\alpha) = 0$. This concludes the proof of the "high error" property (1).

We close this section by showing that the value of $\gamma > 0$ obtained above is indeed at most $1/6$ (and hence every example in S lies in the unit disc as required). To see this, note that we have shown that for this γ, we have $Z((6+5\gamma)\epsilon(1+\gamma)) < 0$ and $Z\left(\frac{\epsilon(1+\gamma)}{\gamma}\right) > 0$. Since Z is a nondecreasing function this implies $6+5\gamma < \frac{1}{\gamma}$ which clearly implies $\gamma < 1/6$ as desired.

4.4. The "steep slope" property: analyzing directional derivatives

Now we turn to proving that the directional derivative in the orthogonal direction is less steep than in the direction of the global minimum (α_1^*, α_2^*). We have just established that $(\alpha, B\alpha) = (\alpha, (1+\gamma)\alpha)$ is a global minimum for the data set as constructed above. The directional derivative at $(0,0)$ in the direction of this optimum is $\frac{L_1(0)+BL_2(0)}{\sqrt{1+B^2}}$.

Since $\phi'(0) < 0$, by (6) and (7) we have

$$\begin{aligned} L_1(0) &= (1+3\gamma)\phi'(0)(N-1) < 0 \\ L_2(0) &= 3\gamma\phi'(0)(N-1) < 0. \end{aligned}$$

This implies that $L_1(0) < L_2(0) < 0$, which, since $B > 1$, implies $BL_1(0) - L_2(0) < 0$. This means that $(B, -1)$ rather than $(-B, 1)$ is the direction orthogonal to the optimal $(1, B)$ which has negative slope.

Recalling that $B = 1+\gamma$, we have the following in-

equalities:

$$\begin{aligned} B &< 1 + 6\gamma = \frac{(1+3\gamma)+3\gamma}{(1+3\gamma)-3\gamma} \\ B &< \frac{-L_1(0) - L_2(0)}{-L_1(0) + L_2(0)} \quad (10) \\ B(-L_1(0) + L_2(0)) &< -L_1(0) - L_2(0) \quad (11) \\ L_1(0) + BL_2(0) &< BL_1(0) - L_2(0) < 0, \quad (12) \end{aligned}$$

where (11) follows from (10) using $L_1(0) < L_2(0) < 0$. So the directional derivative in the optimal direction $(1, B)$ is steeper than in $(B, -1)$, and the proof of the "steep slope" property, and with it Theorem 3, is complete. □

5. Consequences for Known Boosting Algorithms

A wide range of well-studied boosting algorithms are based on potential functions ϕ that satisfy our Definition 1. Theorem 2 thus implies that each of the corresponding convex potential function boosters as defined in Section 2.2 cannot tolerate random classification noise at any noise rate $0 < \eta < \frac{1}{2}$. (In some cases the original versions of the algorithms discussed below are not exactly the same as the $\mathcal{B}_\phi$ algorithm as described in Section 2.2 because of small differences such as the way the step size is chosen at each update. Thus we do not claim that Theorem 2 applies directly to each of the original boosting algorithms; however we feel that our analysis strongly suggests that the original boosters may, like the corresponding $\mathcal{B}_\phi$ algorithms, be highly susceptible to random classification noise.)

AdaBoost and MadaBoost. As discussed in the Introduction and in [6, 18] the Adaboost algorithm [11] is the algorithm $\mathcal{B}_\phi$ obtained by taking the convex potential function to be $\phi(x) = \exp(-x)$. Similarly the MadaBoost algorithm [5] is based on the potential function $\phi(x)$ defined in Equation (1). Each of these functions clearly satisfies Definition 1.

LogitBoost and FilterBoost. As described in [6, 18, 2], the LogitBoost algorithm of [12] is based on the logistic potential function $\ln(1 + \exp(-x))$, which is easily seen to fit our Definition 1. Roughly, FilterBoost [2] combines a variation on the rejection sampling of MadaBoost with the reweighting scheme, and therefore the potential function, of LogitBoost.

6. Experiments with Binary-valued Weak Learners

The analysis of this paper leaves open the possibility that a convex potential booster could still tolerate noise if the base classifiers were restricted to be binary-valued. In this section we describe empirical evidence that this is not the case. We generated 100 datasets, applied three convex potential boosters to each, and calculated the training error.

Data. Each dataset consisted of 4000 examples, divided into three groups, 1000 large margin examples, 1000 pullers, and 2000 penalizers. The large margin examples corresponded to the example $(1, 0)$ in Section 4.2, the pullers play the role of $(\gamma, 5\gamma)$, and the penalizers collectively play the role of $(\gamma, -\gamma)$.

Each labeled example (x, y) in our dataset is generated as follows. First the label y is chosen randomly from $\{-1, 1\}$. There are 21 features $x_1, \ldots, x_{21}$ that take values in $\{-1, 1\}$. Each large margin example sets $x_1 = \cdots = x_{21} = y$. Each puller assigns $x_1 = \cdots = x_{11} = y$ and $x_{12} = \cdots = x_{21} = -y$. Each penalizer is chosen at random in three stages: (1) the values of a random subset of five of the first eleven features $x_1, \ldots, x_{11}$ are set equal to y, (2) the values of a random subset of six of the last ten features $x_{12}, \ldots, x_{21}$ are set equal to y, and (3) the remaining ten features are set to $-y$.

At this stage, if we associate a base classifier with each feature x_i, then each of the 4000 examples is classified correctly by a majority vote over these 21 base classifiers. Intuitively, when an algorithm responds to the pressure exerted by the noisy large margin examples and the pullers to move toward a hypothesis that is a majority vote over the first 11 features only, then it tends to incorrectly classify the penalizers, because in the penalizers only 5 of those first 11 features agree with the class.

Finally, each class designation y is corrupted with classification noise with probability 0.1.

Boosters. We experimented with three boosters: AdaBoost, MadaBoost (which is arguably, loosely speaking, the least convex of the convex potential boosters), and LogitBoost. Each booster was run for 100 rounds.

Results. The average training error of AdaBoost over the 100 datasets was 33%. The average for LogitBoost was 30%, and for MadaBoost, 27%.

7. Discussion

We have shown that any boosting algorithm based on coordinate-wise gradient descent to optimize a convex potential function satisfying mild conditions cannot tolerate random classification noise. While our results imply strong limits on the noise-tolerance of algorithms that fit this framework, they do not apply to other boosting algorithms such as Freund's Boost-By-Majority algorithm [8] and BrownBoost [9] for which the corresponding potential function is non-convex. An interesting direction for future work is to extend our negative results to a broader class of potential functions, or to other types of boosters such as "regularized" boosters [19, 14].

We close by observing that there do exist efficient boosting algorithms (which do not follow the potential function approach) that can provably tolerate random classification noise [13, 16]. These noise-tolerant boosters work by constructing a branching program over the weak classifiers; the original algorithms of [13, 16] were presented only for binary-valued weak classifiers, but recent work [15] extends the algorithm from [16] to work with confidence-rated base classifiers. A standard analysis (omitted because of space constraints) shows that this boosting algorithm for confidence-rated base classifiers can tolerate random classification noise at any rate $0 < \eta < 1/2$ according to our definition from Section 2.4. In particular, for any noise rate η bounded below $1/4$, if this booster is run on the data sets considered in this paper, it can construct a final classifier with accuracy $1-\eta-\epsilon > 3/4$ after $O(\frac{\log 1/\epsilon}{\gamma^2})$ stages of boosting. Since our set of examples S is of size four, though, this means that the booster's final hypothesis will in fact have *perfect* accuracy on these data sets which thwart convex potential boosters.

References

[1] P. L. Bartlett and M. Traskin. Adaboost is consistent. *JMLR*, 8:2347–2368, 2007.

[2] J. Bradley and R. Schapire. Filterboost: Regression and classification on large datasets. In *NIPS*, 2007.

[3] L. Breiman. Arcing the edge. Technical report 486, Department of Statistics, University of California, Berkeley, 1997.

[4] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

[5] C. Domingo and O. Watanabe. Madaboost: a modified version of adaboost. In *COLT*, pages 180–189, 2000.

[6] N. Duffy and D. Helmbold. Potential boosters? In *NIPS*, pages 258–264, 1999.

[7] N. Duffy and D. Helmbold. A geometric approach to leveraging weak learners. *TCS*, 284:67–108, 2002.

[8] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[9] Y. Freund. An adaptive version of the boost-by-majority algorithm. *Machine Learning*, 43(3):293–318, 2001.

[10] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *ICML*, pages 148–156, 1996.

[11] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS*, 55(1):119–139, 1997.

[12] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 1998.

[13] A. Kalai and R. Servedio. Boosting in the presence of noise. *JCSS*, 71(3):266–290, 2005. Preliminary version in *Proc. STOC'03*.

[14] G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. *NIPS*, pages 447–454, 2002.

[15] P. Long and R. Servedio. Adaptive martingale boosting. Unpublished manuscript.

[16] P. Long and R. Servedio. Martingale boosting. In *COLT*, pages 79–94, 2005.

[17] Richard Maclin and David Opitz. An empirical evaluation of bagging and boosting. In *AAAI/IAAI*, pages 546–551, 1997.

[18] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *NIPS*, pages 512–518, 1999.

[19] G. R atsch, T. Onoda, and K.-R. M uller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, 2001.

[20] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

Uncorrelated Multilinear Principal Component Analysis through Successive Variance Maximization

Haiping Lu HPLU@IEEE.ORG
Konstantinos N. Plataniotis KOSTAS@COMM.TORONTO.EDU
Department of Electrical and Computer Engineering, University of Toronto, ON, M5S 3G4, Canada

Anastasios N. Venetsanopoulos TASVENET@RYERSON.CA
Ryerson University, Toronto, ON, M5B 2K3, Canada

Abstract

Tensorial data are frequently encountered in various machine learning tasks today and dimensionality reduction is one of their most important applications. This paper extends the classical principal component analysis (PCA) to its multilinear version by proposing a novel unsupervised dimensionality reduction algorithm for tensorial data, named as uncorrelated multilinear PCA (UMPCA). UMPCA seeks a tensor-to-vector projection that captures most of the variation in the original tensorial input while producing uncorrelated features through successive variance maximization. We evaluate the UMPCA on a second-order tensorial problem, face recognition, and the experimental results show its superiority, especially in low-dimensional spaces, through the comparison with three other PCA-based algorithms.

1. Introduction

Various machine learning problems take multi-dimensional data as input, which are formally called tensors. The elements of a tensor are to be addressed by several indices and the number of indices used in the description defines the order of the tensor object, with each index defining one "mode" (Lathauwer et al., 2000). Many real-world data are naturally tensor objects. For example, matrix data such as gray-level images are second-order tensors, gray-scale video sequences and 3-D objects are third-order tensors. In addition, streaming data and mining data are frequently organized as third-order tensors. For instance, data in environmental sensor monitoring are often organized in three modes of time, location and type, and data in web graph mining are commonly organized in three modes of source, destination and text. Other applications involving tensorial data include data center monitoring, social network analysis, network forensics and face recognition (Faloutsos et al., 2007). In these practical applications, tensor objects are often specified in a high-dimensional tensor space, leading to the so-called curse of dimensionality. Nonetheless, the class of tensor objects in most applications are highly constrained to a subspace, a manifold of intrinsically low dimension (Shakhnarovich & Moghaddam, 2004), and feature extraction or dimensionality reduction is frequently employed to transform a high-dimensional data set into a low-dimensional space of equivalent representation while retaining most of the underlying structure (Law & Jain, 2006).

The PCA is a classical linear method for unsupervised dimensionality reduction that transforms a data set consisting of a large number of interrelated variables to a new set of uncorrelated variables, while retaining as much as possible the variations present in the original data set (Jolliffe, 2002). PCA on tensor objects requires their reshaping (vectorization) into vectors in a very high-dimensional space, which not only results in high computational and memory demands but also breaks the natural structure and correlation in the original data (Ye, 2005; Ye et al., 2004; Lu et al., 2008a). It is believed by many researchers that potentially more compact or useful representations can be obtained from the original form and PCA extensions operating directly on the tensor objects rather than their vectorized versions are emerging recently (Ye et al., 2004; Lu et al., 2008a; Xu et al., 2005).

In (Shashua & Levin, 2001), the tensor rank-one de-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

composition (TROD) is used to represent a class of images based on variance maximization and (greedy) successive residue calculation. A two-dimensional PCA (2DPCA) is proposed in (Yang et al., 2004) that constructs an image covariance matrix using image matrices as inputs. However, linear transformation is applied only to the right side of image matrices so the image data is projected in one mode only, resulting in poor dimensionality reduction. A more general algorithm named generalized low rank approximation of matrices (GLRAM) was introduced in (Ye, 2005), which applies two linear transforms to both the left and right sides of input image matrices and results in a better dimensionality reduction than 2DPCA. GLRAM is developed from the perspective of approximation while the generalized PCA (GPCA) is proposed in (Ye et al., 2004) from the view of variation maximization, as an extension of PCA. Later, the concurrent subspaces analysis (CSA) is formulated in (Xu et al., 2005) for optimal reconstruction of general tensor objects, which can be considered as a generalization of GLRAM, and the multilinear PCA (MPCA) introduced in (Lu et al., 2008a) targets at variation maximization for general tensor objects in the extension of PCA to the multilinear case, which can be considered as a further generalization of GPCA.

However, none of the existing multilinear extensions of PCA mentioned above takes an important property of PCA into account, i.e., PCA derives uncorrelated features, which contain minimum redundancy and ensure independence among features. Instead, most of them produce orthogonal bases in each mode. Although uncorrelated features imply orthogonal projection bases in PCA, this is not necessarily the case for its multilinear extension. With this motivation, this paper investigates multilinear extension of PCA that can produce uncorrelated features. We propose a novel uncorrelated multilinear PCA (UMPCA) for unsupervised tensor object dimensionality reduction (feature extraction). UMPCA is based on the tensor-to-vector projection (TVP) (Lu et al., 2008b) and it follows the classical PCA derivation of successive variance maximization (Jolliffe, 2002). Thus, a number of elementary multilinear projections (EMPs) are solved to maximize the captured variance with the zero-correlation constraint. The solution is iterative in nature, as many other multilinear algorithms (Xu et al., 2005; Ye et al., 2004; Shashua & Levin, 2001).

The rest of this paper is organized as follows. Section 2 reviews basic multilinear notations and operations, as well as the concept of tensor-to-vector projection. In Sec. 3, the problem of UMPCA is formulated and the solution is derived as a sequential iterative process.

Table 1. Notations

Notations	Descriptions
$\mathcal{X}_m$, $m = 1, ..., M$	the m^{th} input tensor sample
$\mathbf{u}^{(n)}$, $n = 1, ..., N$	the n-mode projection vector
$\{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}$	the p^{th} EMP, where p is the index of the EMP
$\mathbf{y}_m$	the projection of $\mathcal{X}_m$ on the TVP $\{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}_{p=1}^{P}$
$\mathbf{y}_m(p) = y_{m_p} = \mathbf{g}_p(m)$	the projection of $\mathcal{X}_m$ on the p^{th} EMP $\{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}$
$\mathbf{g}_p$	the p^{th} coordinate vector

Next, Sec. 4 evaluates the effectiveness of UMPCA in the popular face recognition task through comparison with PCA, MPCA and TROD. Finally, the conclusions are drawn in Sec. 5.

2. Multilinear Fundamentals

This section introduces the multilinear notations, operations and projections needed in the presentation of UMPCA, and for further pursuing of multilinear algebra, (Lathauwer et al., 2000) is a good reference. The important notations used in this paper are listed in Table 1 for handy reference.

2.1. Notations and basic multilinear operations

Due to the multilinear nature of tensor objects, new notations have been introduced in the literature for mathematical analysis. Following the notations in (Lathauwer et al., 2000), we denote vectors by lowercase boldface letters, e.g., $\mathbf{x}$; matrices by uppercase boldface letters, e.g., $\mathbf{U}$; and tensors by calligraphic letters, e.g., $\mathcal{A}$. Their elements are denoted with indices in parentheses. Indices are denoted by lowercase letters and span the range from 1 to the uppercase letter of the index, e.g., $n = 1, 2, ..., N$.

An N^{th}-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ is addressed by N indices i_n, $n = 1, ..., N$, and each i_n addresses the n-mode of $\mathcal{A}$. The n-mode product of a tensor $\mathcal{A}$ by a matrix $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathcal{A} \times_n \mathbf{U}$, is a tensor with entries:

$$(\mathcal{A} \times_n \mathbf{U})(i_1, ..., i_{n-1}, j_n, i_{n+1}, ..., i_N) = \sum_{i_n} \mathcal{A}(i_1, i_2, ..., i_N) \cdot \mathbf{U}(j_n, i_n). \quad (1)$$

The scalar product of two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ is defined as:

$$< \mathcal{A}, \mathcal{B} >= \sum_{i_1} ... \sum_{i_N} \mathcal{A}(i_1, ..., i_N) \cdot \mathcal{B}(i_1, ..., i_N). \quad (2)$$

A rank-one tensor $\mathcal{A}$ equals to the outer product of N

vectors: $\mathcal{A} = \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ ... \circ \mathbf{u}^{(N)}$, which means that $\mathcal{A}(i_1, i_2, ..., i_N) = \mathbf{u}^{(1)}(i_1) \cdot \mathbf{u}^{(2)}(i_2) \cdot ... \cdot \mathbf{u}^{(N)}(i_N)$ for all values of indices.

2.2. Tensor-to-vector projection

In order to extract uncorrelated features from tensorial data directly, we employ the TVP introduced in (Lu et al., 2008b), which is a more general form of the projection in (Shashua & Levin, 2001) and consists of multiple EMPs. An EMP is a multilinear projection $\{\mathbf{u}^{(1)^T}, \mathbf{u}^{(2)^T}, ..., \mathbf{u}^{(N)^T}\}$ consisting of one unit projection vector in each mode, i.e., $\| \mathbf{u}^{(n)} \| = 1$ for $n = 1, ..., N$, where $\| \cdot \|$ is the Euclidean norm for vectors. It projects a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ to a scalar y through the N unit projection vectors as

$$y = \mathcal{X} \times_1 \mathbf{u}^{(1)^T} \times_2 \mathbf{u}^{(2)^T} ... \times_N \mathbf{u}^{(N)^T} = < \mathcal{X}, \mathcal{U} >,$$

where $\mathcal{U} = \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ ... \circ \mathbf{u}^{(N)}$. An EMP can be viewed as a constrained linear projection since $< \mathcal{X}, \mathcal{U} > = < vec(\mathcal{X}), vec(\mathcal{U}) > = [vec(\mathcal{U})]^T vec(\mathcal{X})$, where $vec(\cdot)$ denotes the vectorized representation.

The TVP of a tensor object $\mathcal{X}$ to a vector $\mathbf{y} \in \mathbb{R}^P$ consists of P EMPs $\{\mathbf{u}_p^{(1)^T}, \mathbf{u}_p^{(2)^T}, ..., \mathbf{u}_p^{(N)^T}\}, p = 1, ..., P$, which can be written concisely as $\{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}_{p=1}^P$:

$$\mathbf{y} = \mathcal{X} \times_{n=1}^N \{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}_{p=1}^P, \qquad (3)$$

where the p^{th} component of $\mathbf{y}$ is obtained from the p^{th} EMP as: $\mathbf{y}(p) = \mathcal{X} \times_1 \mathbf{u}_p^{(1)^T} \times_2 \mathbf{u}_p^{(2)^T} ... \times_N \mathbf{u}_p^{(N)^T}$. The TROD (Shashua & Levin, 2001) in fact seeks a TVP to maximize the captured variance, however, it takes a heuristic greedy approach. In the next section, we propose a systematic, more principled formulation by taking consideration of the correlation among features.

In addition, the TVP for dimensionality reduction here is related mathematically to the parallel factor analysis (PARAFAC) originated from psychometrics (Harshman, 1970), also known as the canonical decomposition (CANDECOMP) (Carroll & Chang, 1970), which is popular in factor analysis of multi-way data, i.e., tensors. However, they are developed from different perspectives. The PARAFAC in the factorization literature aims to decompose a higher-order tensor, often formed by arranging lower-order tensors, into a number of rank-one tensorial factors explaining the formation of the data. In contrast, the objective of the TVP for dimensionality reduction here is to learn a low-dimensional (subspace) representation of a class of tensor objects from a number of samples so that the underlying (class) structure is well captured.

3. Uncorrelated Multilinear PCA

This section proposes the UMPCA for unsupervised dimensionality reduction of tensor objects by first formulating the UMPCA objective function and then adopting the successive variance maximization approach and alternating projection method to solve the problem. In the presentation, for the convenience of discussion, the training samples are assumed to be zero-mean [1] so that the constraint of uncorrelated features is the same as orthogonal features (Koren & Carmel, 2004).

3.1. Problem formulation

Following the standard derivation of PCA given in (Jolliffe, 2002), we consider the variance of the principal components (PCs) one by one. In the TVP setting, the p^{th} PCs are $\{y_{m_p}, m = 1, ..., M\}$, where M is the number of training samples and y_{m_p} is the projection of the m^{th} sample $\mathcal{X}_m$ by the p^{th} EMP $\{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}$: $y_{m_p} = \mathcal{X}_m \times_{n=1}^N \{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}$. Accordingly, the variance is measure by their total scatter $S_{T_p}^{\mathbf{y}}$, which is defined as

$$S_{T_p}^{\mathbf{y}} = \sum_{m=1}^{M} (y_{m_p} - \bar{y}_p)^2, \qquad (4)$$

where $\bar{y}_p = \frac{1}{M} \sum_m y_{m_p}$. In addition, let $\mathbf{g}_p$ denote the p^{th} coordinate vector, with its m^{th} component $\mathbf{g}_p(m) = y_{m_p}$. A formal definition of the unsupervised multilinear feature extraction problem to be solved in UMPCA is then given in the following:

A set of M tensor object samples $\{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_M\}$ are available for training. Each tensor object $\mathcal{X}_m \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ assumes values in the tensor space $\mathbb{R}^{I_1} \bigotimes \mathbb{R}^{I_2} ... \bigotimes \mathbb{R}^{I_N}$, where I_n is the n-mode dimension of the tensor and $\bigotimes$ denotes the Kronecker product. The objective of the UMPCA is to find a TVP, which consists of P EMPs $\{\mathbf{u}_p^{(n)} \in \mathbb{R}^{I_n \times 1}, n = 1, ..., N\}_{p=1}^P$, mapping from the original tensor space $\mathbb{R}^{I_1} \bigotimes \mathbb{R}^{I_2} ... \bigotimes \mathbb{R}^{I_N}$ into a vector subspace $\mathbb{R}^P$ (with $P < \prod_{n=1}^N I_n$):

$$\mathbf{y}_m = \mathcal{X}_m \times_{n=1}^N \{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}_{p=1}^P, m = 1, ..., M, \qquad (5)$$

such that the variance of the projected samples, measured by $S_{T_p}^{\mathbf{y}}$, is maximized in each EMP direction, subject to the constraint that the P coordinate vectors $\{\mathbf{g}_p \in \mathbb{R}^M, p = 1, ..., P\}$ are uncorrelated.

[1] When the training sample mean is not zero, it can be subtracted to make the training samples to be zero-mean.

In other words, the UMPCA objective is to determine a set of P EMPs $\{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}_{p=1}^{P}$ that maximize the variance while producing features with zero-correlation. Thus, the objective function for the p^{th} EMP is

$$\begin{aligned} \{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\} &= \arg\max \sum_{m=1}^{M} (y_{m_p} - \overline{y}_p)^2, \\ \texttt{subject to} \quad & \mathbf{u}_p^{(n)^T} \mathbf{u}_p^{(n)} = 1 \text{ and} \\ \frac{\mathbf{g}_p^T \mathbf{g}_q}{\| \mathbf{g}_p \| \| \mathbf{g}_q \|} &= \delta_{pq},\ p, q = 1, ..., P, \end{aligned} \tag{6}$$

where δ_{pq} is the Kronecker delta (defined as 1 for $p = q$ and as 0 otherwise).

3.2. The UMPCA algorithm

To solve the UMPCA problem (6), we follow the successive variance maximization approach in the derivation of PCA in (Jolliffe, 2002). The P EMPs $\{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}_{p=1}^{P}$ are determined one by one in P steps, with the p^{th} step obtaining the p^{th} EMP:

Step 1: Determine the first EMP $\{\mathbf{u}_1^{(n)^T}, n = 1, ..., N\}$ by maximizing $S_{T_1}^{\mathbf{y}}$ without any constraint.

Step 2: Determine the second EMP $\{\mathbf{u}_2^{(n)^T}, n = 1, ..., N\}$ by maximizing $S_{T_2}^{\mathbf{y}}$ subject to the constraint that $\mathbf{g}_2^T \mathbf{g}_1 = 0$.

Step $p (p = 3, ..., P)$**:** Determine the p^{th} EMP $\{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}$ by maximizing $S_{T_p}^{\mathbf{y}}$ subject to the constraint that $\mathbf{g}_p^T \mathbf{g}_q = 0$ for $q = 1, ..., p-1$.

In order to solve for the p^{th} EMP $\{\mathbf{u}_p^{(n)^T}, n = 1, ..., N\}$, we need to determine N sets of parameters corresponding to N projection vectors, $\mathbf{u}_p^{(1)}, \mathbf{u}_p^{(2)}, ...\mathbf{u}_p^{(N)}$, one in each mode. Unfortunately, simultaneous determination of these N sets of parameters in all modes is a complicated non-linear problem without an existing optimal solution, except when $N = 1$, which is the classical PCA where only one projection vector is to be solved. Therefore, we follow the approach in the alternating least square (ALS) algorithm (Harshman, 1970) to solve this multilinear problem. For each EMP to be determined, the parameters of the projection vector $\mathbf{u}_p^{(n^*)}$ for each mode n^* are estimated one mode by one mode separately, conditioned on $\{\mathbf{u}_p^{(n)}, n \neq n^*\}$, the parameter values of the projection vectors in the other modes.

To solve for $\mathbf{u}_p^{(n^*)}$ in the n^*-mode, assuming that $\{\mathbf{u}_p^{(n)}, n \neq n^*\}$ is given, the tensor samples are projected in these $(N-1)$ modes $\{n \neq n^*\}$ first to obtain the vectors

$$\begin{aligned} \tilde{\mathbf{y}}_{m_p}^{(n^*)} = \quad & \mathcal{X}_m \times_1 \mathbf{u}_p^{(1)^T} ... \times_{n^*-1} \mathbf{u}_p^{(n^*-1)^T} \\ & \times_{n^*+1} \mathbf{u}_p^{(n^*+1)^T} ... \times_N \mathbf{u}_p^{(N)^T}, \end{aligned} \tag{7}$$

where $\tilde{\mathbf{y}}_{m_p}^{(n^*)} \in \mathbb{R}^{I_{n^*}}$. This conditional subproblem then becomes to determine $\mathbf{u}_p^{(n^*)}$ that projects the vector samples $\{\tilde{\mathbf{y}}_{m_p}^{(n^*)}, m = 1, ..., M\}$ onto a line so that the variance is maximized, subject to the zero-correlation constraint, which is a PCA problem with the input samples $\{\tilde{\mathbf{y}}_{m_p}^{(n^*)}, m = 1, ..., M\}$. The corresponding total scatter matrix $\tilde{\mathbf{S}}_{T_p}^{(n^*)}$ is then defined as

$$\tilde{\mathbf{S}}_{T_p}^{(n^*)} = \sum_{m=1}^{M} (\tilde{\mathbf{y}}_{m_p}^{(n^*)} - \bar{\tilde{\mathbf{y}}}_p^{(n^*)})(\tilde{\mathbf{y}}_{m_p}^{(n^*)} - \bar{\tilde{\mathbf{y}}}_p^{(n^*)})^T, \tag{8}$$

where $\bar{\tilde{\mathbf{y}}}_p^{(n^*)} = \frac{1}{M} \sum_m \tilde{\mathbf{y}}_{m_p}^{(n^*)}$. With (8), we are ready to solve for the P EMPs. For $p = 1$, the $\mathbf{u}_1^{(n^*)}$ that maximizes the total scatter $\mathbf{u}_1^{(n^*)^T} \tilde{\mathbf{S}}_{T_1}^{(n^*)} \mathbf{u}_1^{(n^*)}$ in the projected space is obtained as the unit eigenvector of $\tilde{\mathbf{S}}_{T_1}^{(n^*)}$ associated with the largest eigenvalue. Next, we show how to determine the p^{th} $(p > 1)$ EMP given the first $(p-1)$ EMPs. Given the first $(p-1)$ EMPs, the p^{th} EMP aims to maximize the total scatter $S_{T_p}^{\mathbf{y}}$, subject to the constraint that features projected by the p^{th} EMP are uncorrelated with those projected by the first $(p-1)$ EMPs. Let $\tilde{\mathbf{Y}}_p^{(n^*)} \in \mathbb{R}^{I_{n^*} \times M}$ be a matrix with $\tilde{\mathbf{y}}_{m_p}^{(n^*)}$ as its m^{th} column, i.e., $\tilde{\mathbf{Y}}_p^{(n^*)} = \left[\tilde{\mathbf{y}}_{1_p}^{(n^*)}, \tilde{\mathbf{y}}_{2_p}^{(n^*)}, ..., \tilde{\mathbf{y}}_{M_p}^{(n^*)}\right]$, then the p^{th} coordinate vector is $\mathbf{g}_p = \tilde{\mathbf{Y}}_p^{(n^*)^T} \mathbf{u}_p^{(n^*)}$. The constraint that $\mathbf{g}_p$ is uncorrelated with $\{\mathbf{g}_q, q = 1, ..., p-1\}$ can be written as

$$\mathbf{g}_p^T \mathbf{g}_q = \mathbf{u}_p^{(n^*)^T} \tilde{\mathbf{Y}}_p^{(n^*)} \mathbf{g}_q = 0, q = 1, ..., p-1. \tag{9}$$

Thus, $\mathbf{u}_p^{(n^*)}$ $(p > 1)$ can be determined by solving the following constrained optimization problem:

$$\begin{aligned} & \mathbf{u}_p^{(n^*)} = \arg\max \mathbf{u}_p^{(n^*)^T} \tilde{\mathbf{S}}_{T_p}^{(n^*)} \mathbf{u}_p^{(n^*)}, \\ & \texttt{subject to } \mathbf{u}_p^{(n^*)^T} \mathbf{u}_p^{(n^*)} = 1 \text{ and} \\ & \mathbf{u}_p^{(n^*)^T} \tilde{\mathbf{Y}}_p^{(n^*)} \mathbf{g}_q = 0, q = 1, ..., p-1, \end{aligned} \tag{10}$$

The solution is given by the following theorem:

Theorem 1. *The solution to the problem (10) is the (unit-length) eigenvector corresponding to the largest eigenvalue of the following eigenvalue problem:*

$$\mathbf{\Psi}_p^{(n^*)} \tilde{\mathbf{S}}_{T_p}^{(n^*)} \mathbf{u} = \lambda \mathbf{u}, \tag{11}$$

where

$$\mathbf{\Psi}_p^{(n^*)} = \mathbf{I}_{I_{n^*}} - \tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{G}_{p-1}\mathbf{\Phi}_p^{-1}\mathbf{G}_{p-1}^T\tilde{\mathbf{Y}}_p^{(n^*)^T}, \quad (12)$$

$$\mathbf{\Phi}_p = \mathbf{G}_{p-1}^T\tilde{\mathbf{Y}}_p^{(n^*)^T}\tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{G}_{p-1}, \quad (13)$$

$$\mathbf{G}_{p-1} = [\mathbf{g}_1 \quad \mathbf{g}_2 \quad ...\mathbf{g}_{p-1}] \in \mathbb{R}^{M\times(p-1)}, \quad (14)$$

and $\mathbf{I}_{I_{n^*}}$ *is an identity matrix of size* $I_{n^*} \times I_{n^*}$.

Proof. First, Lagrange multipliers can be used to transform the problem (10) to the following to include all the constraints:

$$\begin{aligned} F(\mathbf{u}_p^{(n^*)}) &= \mathbf{u}_p^{(n^*)^T}\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)} - \nu\left(\mathbf{u}_p^{(n^*)^T}\mathbf{u}_p^{(n^*)} - 1\right) \\ &- \sum_{q=1}^{p-1}\mu_q\mathbf{u}_p^{(n^*)^T}\tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{g}_q, \end{aligned} \quad (15)$$

where ν and $\{\mu_q, q = 1, ..., p-1\}$ are Lagrange multipliers.

The optimization is performed by setting the partial derivative of $F(\mathbf{u}_p^{(n^*)})$ with respect to $\mathbf{u}_p^{(n^*)}$ to zero:

$$\begin{aligned} \frac{\partial F(\mathbf{u}_p^{(n^*)})}{\partial \mathbf{u}_p^{(n^*)}} &= 2\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)} - 2\nu\mathbf{u}_p^{(n^*)} \\ &- \sum_{q=1}^{p-1}\mu_q\tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{g}_q = 0. \end{aligned} \quad (16)$$

Multiplying (16) by $\mathbf{u}_p^{(n^*)^T}$ results in

$$\begin{aligned} &2\mathbf{u}_p^{(n^*)^T}\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)} - 2\nu\mathbf{u}_p^{(n^*)^T}\mathbf{u}_p^{(n^*)} = 0 \\ &\Rightarrow \nu = \frac{\mathbf{u}_p^{(n^*)^T}\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)}}{\mathbf{u}_p^{(n^*)^T}\mathbf{u}_p^{(n^*)}}, \end{aligned} \quad (17)$$

which indicates that ν is exactly the criterion to be maximized, with the constraint on the norm of the projection vector incorporated.

Next, a set of $(p-1)$ equations are obtained by multiplying (16) by $\mathbf{g}_q^T\tilde{\mathbf{Y}}_p^{(n^*)^T}$, $q = 1, ..., p-1$, respectively:

$$\begin{aligned} &2\mathbf{g}_q^T\tilde{\mathbf{Y}}_p^{(n^*)^T}\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)} - \sum_{q=1}^{p-1}\mu_q\mathbf{g}_q^T\tilde{\mathbf{Y}}_p^{(n^*)^T}\cdot\tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{g}_q \\ &= 0. \end{aligned} \quad (18)$$

Let

$$\boldsymbol{\mu}_{p-1} = [\mu_1 \ \mu_2 \ ... \ \mu_{p-1}]^T \quad (19)$$

and use (13) and (14), then the $(p-1)$ equations of (18) can be represented in a single matrix equation as following:

$$2\mathbf{G}_{p-1}^T\tilde{\mathbf{Y}}_p^{(n^*)^T}\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)} - \mathbf{\Phi}_p\boldsymbol{\mu}_{p-1} = 0. \quad (20)$$

Thus,

$$\boldsymbol{\mu}_{p-1} = 2\mathbf{\Phi}_p^{-1}\cdot\mathbf{G}_{p-1}^T\tilde{\mathbf{Y}}_p^{(n^*)^T}\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)}. \quad (21)$$

Since from (14) and (19),

$$\sum_{q=1}^{p-1}\mu_q\tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{g}_q = \tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{G}_{p-1}\boldsymbol{\mu}_{p-1}, \quad (22)$$

the equation (16) can be written as

$$\begin{aligned} &2\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)} - 2\nu\mathbf{u}_p^{(n^*)} - \tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{G}_{p-1}\boldsymbol{\mu}_{p-1} = 0 \\ &\Rightarrow\nu\mathbf{u}_p^{(n^*)} = \tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)} - \tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{G}_{p-1}\frac{\boldsymbol{\mu}_{p-1}}{2} \\ &= \left[\mathbf{I}_{I_{n^*}} - \tilde{\mathbf{Y}}_p^{(n^*)}\mathbf{G}_{p-1}\mathbf{\Phi}_p^{-1}\mathbf{G}_{p-1}^T\tilde{\mathbf{Y}}_p^{(n^*)^T}\right]\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u}_p^{(n^*)}. \end{aligned}$$

Using the definition in (12), an eigenvalue problem is obtained as $\mathbf{\Psi}_p^{(n^*)}\tilde{\mathbf{S}}_{T_p}^{(n^*)}\mathbf{u} = \nu\mathbf{u}$. Since ν is the criterion to be maximized, the maximization is achieved by setting $\mathbf{u}_p^{(n)^*}$ to be the (unit) eigenvector corresponding to the largest eigenvalue of (11). □

By setting $\mathbf{\Psi}_1^{(n^*)} = \mathbf{I}_{I_{n^*}}$ and from Theorem 1, we have a unified solution for UMPCA: for $p = 1, ..., P$, $\mathbf{u}_p^{(n^*)}$ is obtained as the unit eigenvector of $\mathbf{\Psi}_p^{(n^*)}\tilde{\mathbf{S}}_{T_p}^{(n^*)}$ associated with the largest eigenvalue. Algorithm 1 summarizes the UMPCA developed here.

Algorithm 1 Uncorrelated Multilinear Principal Component Analysis (UMPCA)

Input: A set of tensor samples $\{\mathcal{X}_m \in \mathbb{R}^{I_1\times...\times I_N}, m = 1, ..., M\}$, the subspace dimensionality P, and the maximum number of iterations K.
for $p = 1$ **to** P **do**
  for $n = 1$ **to** N **do**
    Initialize $\mathbf{u}_{p_{(0)}}^{(n)} = \mathbf{1}/\parallel\mathbf{1}\parallel$.
  end for
  for $k = 1$ **to** K **do**
    for $n = 1$ **to** N **do**
      Calculate $\tilde{\mathbf{y}}_{m_p}^{(n)} = \mathcal{X}_m \times_1 \mathbf{u}_{p_{(k)}}^{(1)^T} ... \times_{n-1} \mathbf{u}_{p_{(k)}}^{(n-1)^T} \times_{n+1} \mathbf{u}_{p_{(k-1)}}^{(n+1)^T} ... \times_N \mathbf{u}_{p_{(k-1)}}^{(N)^T}$, for $m = 1, ..., M$.
      Calculate $\mathbf{\Psi}_p^{(n)}$ and $\tilde{\mathbf{S}}_{T_p}^{(n)}$. Set $\mathbf{u}_{p_{(k)}}^{(n)}$ to be the (unit) eigenvector of $\mathbf{\Psi}_p^{(n)}\tilde{\mathbf{S}}_{T_p}^{(n)}$ associated with the largest eigenvalue.
    end for
  end for
  Set $\mathbf{u}_p^{(n)} = \mathbf{u}_{p_k}^{(n)}$ for all n.
  Calculate the coordinate vector $\mathbf{g}_p$.
end for

3.3. Initialization, projection order and termination

As an iterative algorithm, the UMPCA may be affected by the initialization method, the projection order and the termination conditions. Due to the space constraint, these issues, as well as the convergence and computational issues, are not studied here. Instead, we adopt simple implementation strategies for them. First, we use the uniform initialization for UMPCA, where all n-mode projection vectors are initialized to have unit length and the same value along the I_n dimensions in n-mode, which is equivalent to the all ones vector $\mathbf{1}$ with proper normalization. Second, as shown in Algorithm 1, the projection order, which is the mode ordering in computing the projection vectors, is from 1-mode to N-mode, as in other multilinear algorithms (Ye, 2005; Xu et al., 2005; Lu et al., 2008a). Third, the iteration is terminated by setting K, the maximum number of iterations.

4. Experimental Evaluation

The proposed UMPCA can potentially benefit various applications involving tensorial data, as mentioned in Sec. 1. Since face recognition has practical importance in security-related applications such as biometric authentication and surveillance, it has been used widely for evaluation of unsupervised learning algorithms (Shashua & Levin, 2001; Yang et al., 2004; Xu et al., 2005; Ye, 2005). Therefore, in this section, we focus on evaluating the effectiveness of UMPCA on this popular classification task through performance comparison with existing unsupervised dimensionality reduction algorithms.

4.1. The FERET database

The Facial Recognition Technology (FERET) database (Phillips et al., 2000) is widely used for testing face recognition performance, with 14,126 images from 1,199 subjects covering a wide range of variations in viewpoint, illumination, facial expression, races and ages. A subset of this database is selected in our experimental evaluation and it consists of those subjects with each subject having at least eight images with at most 15 degrees of pose variation, resulting in 721 face images from 70 subjects. Since our focus here is on the recognition of faces rather than their detection, all face images are manually cropped, aligned (with manually annotated coordinate information of eyes) and normalized to 80×80 pixels, with 256 gray levels per pixel. Figure 1 shows some sample face images from two subjects in this FERET subset.

Figure 1. Examples of face images from two subjects in the FERET subset used in our experimental evaluation.

4.2. Face recognition performance comparison

In the evaluation, we compare the performance of the UMPCA against three PCA-based unsupervised learning algorithms: the PCA (eigenface) algorithm (Turk & Pentland, 1991), the MPCA algorithm (Lu et al., 2008a)[2] and the TROD algorithm (Shashua & Levin, 2001). The number of iterations in TROD and UMPCA is set to ten, with the same (uniform) initialization used. For MPCA, we obtain the full projection and select the most descriptive P features for recognition. The features obtained by these four algorithms are arranged in descending variation captured (measured by respective total scatter). For classification of extracted features, we use the nearest neighbor classifier (NNC) with Euclidean distance measure.

Gray-level face images are naturally second-order tensors (matrices), i.e., $N = 2$. Therefore, they are input directly as 80×80 tensors to the multilinear algorithms (MPCA, TROD, UMPCA), while for PCA, they are vectorized to 6400×1 vectors as input. For each subject in a face recognition experiment, $L(= 1, 2, 3, 4, 5, 6, 7)$ samples are randomly selected for unsupervised training and the rest are used for testing. We report the results averaged over ten such random splits (repetitions).

Figures 2 and 3 show the detailed results[3] for $L = 1$ and $L = 7$, respectively. $L = 1$ is an extreme small sample size scenario where only one sample per class is available for training, the so-called one training sample (OTS) case important in practice (Wang et al., 2006), and $L = 7$ is the maximum number of training samples we can use in our experiments. Figures 2(a) and 3(a) plot the correct recognition rates against P, the dimensionality of the subspace for $P = 1, ..., 10$, and Figs 2(b) and 3(b) plot those for $P = 15, ..., 80$. From the figures, UMPCA outperforms the other three methods in both cases and across all dimensionality, indicating that the uncorrelated features extracted directly from the tensorial face data are more effective in classifi-

[2]Note that MPCA with $N = 2$ is equivalent to GPCA.

[3]Note that for PCA and UMPCA, there are at most 69 features when $L = 1$ (only 70 faces for training).

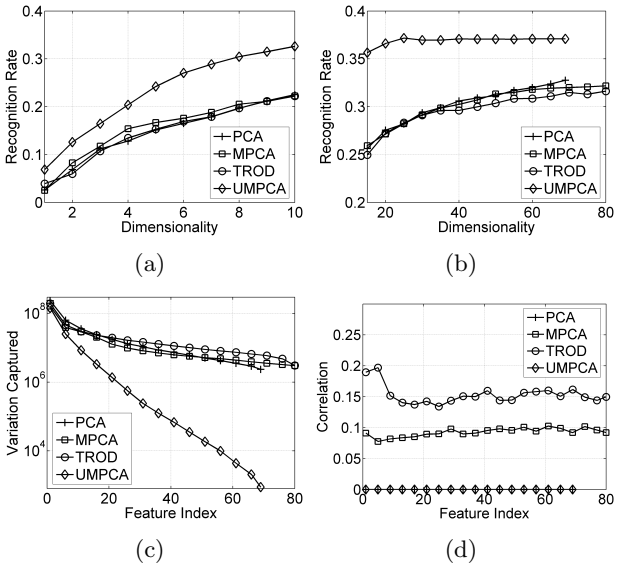


Figure 2. Detailed face recognition results on the FERET database for $L = 1$: (a) performance curves for the low-dimensional case, (b) performance curves for the high-dimensional case, (c) the variation captured by individual features and (d) the correlation among features.

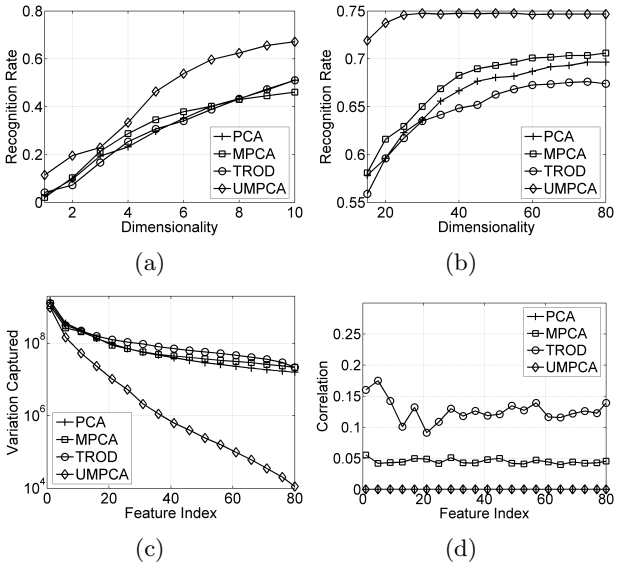


Figure 3. Detailed face recognition results on the FERET database for $L = 7$: (a) performance curves for the low-dimensional case, (b) performance curves for the high-dimensional case, (c) the variation captured by individual features and (d) the correlation among features.

cation. The figures also show that for UMPCA, the recognition rate saturates around $P = 30$, which can be explained by observing the variance captured by individual features as shown in Figs. 2(c) and 3(c) (in log scale). These figures show that the variance captured by UMPCA is considerably lower than those captured by the other methods, which is due to its constraints of zero-correlation and TVP. Despite capturing lower variance, UMPCA is superior in the recognition task performed. Nonetheless, when the variance captured is too low, those corresponding features are no longer descriptive enough to contribute in classification, leading to the saturation.

In addition, we also plot the average correlation of individual features with all the other features in Figs. 2(d) and 3(d). As supported by theoretical derivation, features extracted by PCA and UMPCA are uncorrelated. In contrast, features extracted by MPCA and TROD are correlated, with TROD features have higher correlation on average.

Table 2. Face recognition results on the FERET database: the recognition rates (in percentage) for various Ls and Ps.

L	P	1	5	10	20	50	80
2	PCA	2.8	20.2	32.0	39.1	43.6	45.1
	MPCA	2.6	21.4	28.1	38.9	44.6	**46.0**
	TROD	3.6	19.3	30.6	38.4	43.0	44.3
	UMPCA	**8.1**	**27.6**	**40.6**	**45.0**	**45.8**	45.7
3	PCA	2.7	23.9	37.1	45.9	51.3	52.6
	MPCA	2.3	25.9	34.8	45.5	52.0	53.3
	TROD	4.0	23.5	36.1	44.5	50.1	51.7
	UMPCA	**7.5**	**35.5**	**49.8**	**56.0**	**56.6**	**56.6**
4	PCA	2.7	25.5	41.7	49.4	56.8	57.9
	MPCA	2.3	28.7	39.4	50.2	57.5	58.9
	TROD	4.2	25.3	41.1	49.0	55.1	56.6
	UMPCA	**8.5**	**39.5**	**56.2**	**63.5**	**64.1**	**64.2**
5	PCA	3.0	28.9	47.1	55.6	63.9	64.6
	MPCA	2.6	33.0	43.2	56.8	64.3	65.8
	TROD	4.5	28.4	47.2	55.6	62.0	63.9
	UMPCA	**8.1**	**43.6**	**61.7**	**68.2**	**69.1**	**69.1**
6	PCA	2.8	30.3	49.0	58.5	66.7	68.1
	MPCA	2.2	33.5	45.7	59.7	67.9	69.7
	TROD	4.3	27.3	49.3	58.6	64.7	66.9
	UMPCA	**9.1**	**45.6**	**62.9**	**70.7**	**71.8**	**71.8**

The recognition results for $P = 1, 5, 10, 20, 50, 80$ are listed in Table 2 for $L = 2, 3, 4, 5, 6$, where the best recognition results among the four methods are shown in bold. More detailed results are omitted here to save space. From the table, UMPCA achieves superior recognition results in all cases except for $P = 80$ and $L = 2$, where the difference with the best results by MPCA is small (0.3%). In particular, for smaller P $(1, 5, 10, 20)$, UMPCA outperforms the other algorithms significantly, demonstrating its superior capability in classifying faces in low-dimensional spaces.

5. Conclusions

This paper proposes a novel uncorrelated multilinear PCA algorithm, where uncorrelated features are extracted directly from tensorial representation through a tensor-to-vector projection. The algorithm successively maximizes variance captured by each elementary projection while enforcing the zero-correlation constraint. The solution employs the alternating projection method and is iterative. Experiments on face recognition demonstrate that compared with other unsupervised learning algorithms including the PCA, MPCA and TROD, the UMPCA achieves the best results and it is particularly effective in low-dimensional spaces. Thus, face recognition through unsupervised learning benefits from the proposed UMPCA and in future research, it is worthwhile to investigate whether UMPCA can contribute in other unsupervised learning tasks, such as clustering.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments. This work is partially supported by the Ontario Centres of Excellence through the Communications and Information Technology Ontario Partnership Program and the Bell University Labs - at the University of Toronto.

References

Carroll, J. D., & Chang, J. J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, *35*, 283–319.

Faloutsos, C., Kolda, T. G., & Sun, J. (2007). Mining large time-evolving data using matrix and tensor tools. *Int. Conf. on Data Mining 2007 Tutorial*.

Harshman, R. A. (1970). Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, *16*, 1–84.

Jolliffe, I. T. (2002). *Principal component analysis, second edition*. Springer Serires in Statistics.

Koren, Y., & Carmel, L. (2004). Robust linear dimensionality reduction. *IEEE Trans. Vis. Comput. Graphics*, *10*, 459–470.

Lathauwer, L. D., Moor, B. D., & Vandewalle, J. (2000). On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of higher-order tensors. *SIAM Journal of Matrix Analysis and Applications*, *21*, 1324–1342.

Law, M. H. C., & Jain, A. K. (2006). Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, *28*, 377–391.

Lu, H., Plataniotis, K. N., & Venetsanopoulos, A. N. (2008a). MPCA: Multilinear principal component analysis of tensor objects. *IEEE Trans. Neural Netw.*, *19*, 18–39.

Lu, H., Plataniotis, K. N., & Venetsanopoulos, A. N. (2008b). Uncorrelated multilinear discriminant analysis with regularization and aggregation for tensor object recognition. *IEEE Trans. Neural Netw.* accepted pending minor revision.

Phillips, P. J., Moon, H., Rizvi, S. A., & Rauss, P. (2000). The FERET evaluation method for face recognition algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, *22*, 1090–1104.

Shakhnarovich, G., & Moghaddam, B. (2004). Face recognition in subspaces. *Handbook of Face Recognition* (pp. 141–168). Springer-Verlag.

Shashua, A., & Levin, A. (2001). Linear image coding for regression and classification using the tensor-rank principle. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (pp. 42–49).

Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neurosicence*, *3*, 71–86.

Wang, J., Plataniotis, K. N., Lu, J., & Venetsanopoulos, A. N. (2006). On solving the face recognition problem with one training sample per subject. *Pattern Recognition*, *39*, 1746–1762.

Xu, D., Yan, S., Zhang, L., Zhang, H.-J., Liu, Z., & Shum;, H.-Y. (2005). Concurrent subspaces analysis. *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition* (pp. 203–208).

Yang, J., Zhang, D., Frangi, A. F., & Yang, J. (2004). Two-dimensional PCA: a new approach to appearance-based face representation and recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, *26*, 131–137.

Ye, J. (2005). Generalized low rank approximations of matrices. *Machine Learning*, *61*, 167–191.

Ye, J., Janardan, R., & Li, Q. (2004). GPCA: An efficient dimension reduction scheme for image compression and retrieval. *The* 10^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (pp. 354–363).

A Reproducing Kernel Hilbert Space Framework for Pairwise Time Series Distances

Zhengdong Lu ZHENGDON@CSEE.OGI.EDU
Todd K. Leen TLEEN@CSEE.OGI.EDU
Yonghong Huang HUANG@CSEE.OGI.EDU
Deniz Erdogmus DERDOGMUS@IEEE.ORG
OGI School, Oregon Health & Science University, 20000 NW Walker Rd., Beaverton, OR 97006 USA

Abstract

A good distance measure for time series needs to properly incorporate the temporal structure, and should be applicable to sequences with unequal lengths. In this paper, we propose a distance measure as a principled solution to the two requirements. Unlike the conventional feature vector representation, our approach represents each time series with a summarizing smooth curve in a reproducing kernel Hilbert space (RKHS), and therefore translate the distance between time series into distances between curves. Moreover we propose to learn the kernel of this RKHS from a population of time series with discrete observations using Gaussian process-based non-parametric mixed-effect models. Experiments on two vastly different real-world problems show that the proposed distance measure leads to improved classification accuracy over the conventional distance measures.

1. Introduction

Time series classification is a supervised learning problem aimed at labeling temporally structured sequences of variable length. The most common approach reduces time series classification to a static problem by suitably transforming the input sequences into vectors in Euclidean space. One can either summarize each time series with attributes pertinent to classification (called *feature extraction*)(Keogh & Pazzani, 1998), or use a properly sampled and aligned subsequence (called *sampling*)(Parra et al., 2003). Unfortunately, the feature extraction method is still more art than science, and the performance depends heavily on the designer's domain knowledge and the particular heuristic implemented. The sampling method, although preserving most of the information, is accused of ignoring the important temporal structure of the series. Indeed, the sampled sequences, if treated as vectors in Euclidean space, lead to the same classifiers after any permutation of the vector entries. Moreover, the sampling strategy does not apply to situations where we have only sparse observations that are made at irregular times.

In this paper, we propose a principled non-parametric distance measure for time series by representing each time series with a smooth curve in a reproducing kernel Hilbert space (RKHS) with a kernel learned from data. This new distance measure circumvents the limitations of the two above mentioned strategies.

Paper Roadmap In Section 2, we give the background of the Bregman divergence, and then generalize it to function space for a proper distance measure of smooth curves. In Section 3 we propose a family of new distance measures for time series with only discrete observations. Section 4 is devoted to the non-parametric mixed-effect model, which helps to further specify the proposed distance measure. In Section 5, we apply the proposed distance measure to two real-world time series classification problems. Finally we discuss the related work in Section 6.

2. Gaussian Processes and Functional Bregman Divergence

The Bregman divergence is a natural generalization of squared Euclidean distance and KL-divergence. A Bregman divergence corresponding to a strictly convex function $\phi(x)$ (called seed function) is defined as

$$d_\phi(x_1||x_2) = \phi(x_1) - \phi(x_2) - \langle \nabla\phi(x_2), x_1 - x_2 \rangle . \quad (1)$$

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Bregman divergence is closely connected to the exponential family (Banerjee et al., 2005). For any distribution in the exponential family

$$p(x;\theta) = \exp(\langle x,\theta\rangle - \Phi(\theta))p_0(x),$$

we know that the log likelihood can be re-written as

$$\log p(x;\theta) = -d_\phi(x||\mu(\theta)) + \phi(x) + \log p_0(x), \quad (2)$$

where ϕ is the conjugate function of Φ

$$\phi(x) = \sup_\theta\{\langle x,\theta\rangle - \Phi(\theta)\} \quad (3)$$

and $\mu(\theta) = \nabla\Phi(\theta)$ is the expectation parameter corresponding to θ. We go one step further to argue that $d_\phi(x_1||x_2)$ should be a proper model-weighted divergence measure between any x_1 and x_2. It is straightforward to show that for multi-variate Gaussian distribution $\mathcal{N}(a,\Sigma)$, the corresponding Bregman divergence is given by

$$d_\phi(x_1||x_2) = \frac{1}{2}(x_1-x_2)^T\Sigma^{-1}(x_1-x_2), \quad (4)$$

which is also suggested in (Tipping, 1999) as a model-weighted distance for Gaussian distribution.

2.1. Extension to Function Space

We generalize our discussion on the Bregman divergence and the exponential family to function spaces. To facilitate our discussion, we adopt the language of functional integral, which, although allegedly not rigorously defined, provides a powerful technique for describing the probability on functions (Simon, 1979).

Gaussian processes (GPs) (Rasmussen & Williams, 2006) generalize the multivariate Gaussian distribution to function space, which model any function f with the following probability [1]

$$p[f] \propto \exp(-\frac{1}{2}||f-f_0||^2_{\mathcal{H}}), \quad (5)$$

with f_0 being the *mean function* and $||\cdot||_{\mathcal{H}}$ the norm for the reproducing kernel Hilbert space (RKHS) $\mathcal{H}$. We use K to denote the reproducing kernel, which will also be noted as the covariance function for the Gaussian process expressed in Eq.(5) (Seeger, 2004). In regularization theory, the norm $||\cdot||_{\mathcal{H}}$ is often related to a particular type of smoothness of function, with large (even infinite) $||f||_{\mathcal{H}}$ for non-smooth function f.

After generalizing Eq.(1) to the functional case (Frigyik et al., 2006), we get the Bregman divergence between function f_1 and f_2, with a seed functional $g[\cdot]$

$$d_g(f_1||f_2) = g[f_1] - g[f_2] - \int Dg[f_2](f_1(t)-f_2(t))dt.$$

where $Dg[f]$ is the Fréchet derivative. The Gaussian process expressed in Eq.(5) can be viewed as a member of the exponential family extended to distributions on functions (Altun et al., 2004). Then a direct generalization of Eq.(3) leads to $g[f] = \frac{1}{2}||f||^2_{\mathcal{H}}$, which gives a GP-related divergence for smooth functions

$$d_{\mathcal{H}}(f_1||f_2) = \frac{1}{2}||f_1-f_2||^2_{\mathcal{H}}. \quad (6)$$

3. Distance for Time Series

We consider k time series, using $\mathbf{y}_i$ to denote the N_i observations from the i^{th} time series made at times $\mathbf{t}_i$

$$\mathbf{y}_i \doteq [y_{i1},\cdots,y_{iN_i}]^T, \quad \mathbf{t}_i \doteq [t_{i1},\cdots,t_{iN_i}]^T.$$

The subscript i on $\mathbf{t}_i$ and N_i indicates that the observation times and even the number of observations are generally different for each individual. The time series are called *synchronized* if all the $\mathbf{t}_i$ are the same.

We can define a distance measure for such time series by associating the observations $\{\mathbf{t}_i,\mathbf{y}_i\}$ with a (smooth) curve. We assume the observations for each individual i is generated from a independent Gaussian process f_i with the same covariance function K (and therefore $\mathcal{H}$) and mean f_0. The observation is modeled as

$$y_{in} = f_i(t_{in}) + \epsilon_{in}, \quad n = 1,2,\cdots,N_i, \quad (7)$$

where ϵ_{in} is a white observation noise with standard deviation σ for all i and n.

We choose to summarize each individual time series i with the expectation of $f_i(t)$ given the discrete noisy observation $\{\mathbf{t}_i,\mathbf{y}_i\}$.

$$\hat{f}_i(t) = E[f_i(t)|\mathbf{y}_i,f_0;\mathbf{t}_i,K] \quad (8)$$
$$= f_0 + K(t,\mathbf{t}_i)(K(\mathbf{t}_i,\mathbf{t}_i)+\sigma^2\mathbb{I})^{-1}(\mathbf{y}_i-\mathbf{f}_{0,i}) \quad (9)$$

where $\mathbf{f}_{0,i} \doteq [f_0(t_{i1}),f_0(t_{i2}),\cdots,f_0(t_{iN_i})]^T$ is the values of f_0 at times $\mathbf{t}_i$, and $K(\mathbf{t}_i,\mathbf{t}_i)$ is the $N_i\times N_i$ matrix with the (n,m) entry being $K(t_{in},t_{im})$. With a smooth f_0, we have $||\hat{f}_i||_{\mathcal{H}} < +\infty$, which can be loosely interpreted as that $\hat{f}_i$ is smooth according to K. In Fig.1, we give an example of using such a curve $\hat{f}$ to represent the noisy observations (black crosses).

We then use the distance between $\hat{f}_i$ and $\hat{f}_j$ as the distance between time series $\{\mathbf{t}_i,\mathbf{y}_i\}$ and $\{\mathbf{t}_j,\mathbf{y}_j\}$ [2],

[1] In the remainder of the paper, we use the square brackets [ ] to distinguish functionals from common functions.

[2] Although $E[||f_i-f_j||^2_{\mathcal{H}}|\mathbf{y}_i,\mathbf{y}_j;\mathbf{t}_i,\mathbf{t}_j]$ seems to be a reasonable measure of distance, it goes to infinity since with probability one a sample f from the a Gaussian process with covariance function K has $||f||_{\mathcal{H}} = \infty$ (Seeger, 2004).

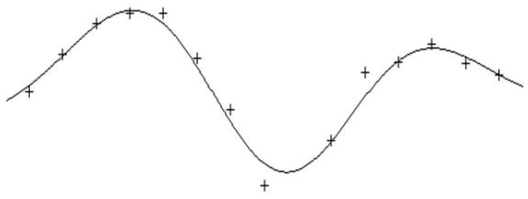


Figure 1. Using smooth curve to represent noisy discrete observations (black crosses). The smooth curve is obtained using Eq.(9) with K being a Gaussian kernel and $f_0 = 0$.

which is given by Eq.(6) as

$$d_{ij} = \frac{1}{2}||\hat{f}_i - \hat{f}_j||^2_{\mathcal{H}}. \tag{10}$$

Since $\mathcal{H}$ is the RKHS induced by the kernel K, this distance measure is well-defined

$$\begin{aligned} d_{ij} &= \frac{1}{2}||\hat{f}_i - \hat{f}_j||^2_{\mathcal{H}} = \frac{1}{2}\left\langle \hat{f}_i - \hat{f}_j, \hat{f}_i - \hat{f}_j \right\rangle_{\mathcal{H}} \\ &= \frac{1}{2}\langle K(t,\mathbf{t}_i)\mathbf{v}_i - K(t,\mathbf{t}_j)\mathbf{v}_j, K(t,\mathbf{t}_i)\mathbf{v}_i - K(t,\mathbf{t}_j)\mathbf{v}_j\rangle_{\mathcal{H}}, \end{aligned}$$

where $\mathbf{v}_i = (K(\mathbf{t}_i,\mathbf{t}_i) + \sigma^2\mathbb{I})^{-1}(\mathbf{y}_i - \mathbf{f}_{0,i})$. Using the reproducing kernel property

$$\forall\, t_n, t_m \quad \langle K(t_n,t), K(t_m,t)\rangle_{\mathcal{H}} = K(t_n,t_m),$$

the distance measurement can be simplified as

$$d_{ij} = \frac{1}{2}\mathbf{v}_i^T K(\mathbf{t}_i,\mathbf{t}_i)\mathbf{v}_i + \frac{1}{2}\mathbf{v}_j^T K(\mathbf{t}_i,\mathbf{t}_i)\mathbf{v}_j - \mathbf{v}_i^T K(\mathbf{t}_i,\mathbf{t}_j)\mathbf{v}_j. \tag{11}$$

It is important to note that this distance does not require all the time series to be synchronized, an advantage when sequences are of different lengths, or the observations are made at different times, as shown in our first experiment in Section 5. When the observations for all individuals are synchronized, we have $\mathbf{t}_i = \mathbf{t} = [t_1, t_2, \cdots, t_N]^T$ with N as the total number of observations for each individual. Letting $\mathbf{K} = K(\mathbf{t},\mathbf{t})$, we can re-write d_{ij} as

$$\begin{aligned} d_{ij} &= \mathbf{v}_i^T\mathbf{K}\mathbf{v}_i + \mathbf{v}_j^T\mathbf{K}\mathbf{v}_j - 2\mathbf{v}_i^T\mathbf{K}\mathbf{v}_j & (12)\\ &= (\mathbf{v}_i - \mathbf{v}_j)^T\mathbf{K}(\mathbf{v}_i - \mathbf{v}_j) & (13)\\ &= (\mathbf{y}_i - \mathbf{y}_j)^T(\mathbf{K} + \sigma^2\mathbb{I})^{-1}\mathbf{K}(\mathbf{K} + \sigma^2\mathbb{I})^{-1}(\mathbf{y}_i - \mathbf{y}_j). & (14) \end{aligned}$$

Temporal Structure In Eq.(11)-(14), the temporal regularity is incorporated in the distance via the kernel K. It is most clear when we notice that K models the correlation of f value at different time

$$K(\mathbf{t}_i,\mathbf{t}_j) = E[(f(\mathbf{t}_i) - f_0(\mathbf{t}_i))^T(f(\mathbf{t}_j) - f_0(\mathbf{t}_j))].$$

The norm $||f_i - f_j||_{\mathcal{H}}$ measures the irregularity defined by K, in contrast to the Euclidean distance $\int(f_i(t) - f_j(t))^2 dt$ which only concerns about the point wise difference between f_i and f_j. It is also important to notice the particular temporal structure incorporated varies greatly with the choice of K. For example, the widely used Matérn (including Gaussian) kernel or rational quadratic kernel promote different types and level of smoothness. On the other hand, the temporal structure is often problem specific and hard to determine beforehand. In the next section, we will discuss learning this temporal structure from the data.

4. Non-parametric Mixed-effect Model

In Section 3, we assume a Gaussian process with known mean and covariance function. However in practice it is often not the case. Instead we may want to learn the characteristic of Gaussian process from examples. One situation of interest to us is when a population of similar time series are available. This prior learning scheme is known in statistics as the empirical Bayesian or the hierarchial Beyesian (Gelman, 2004). Particularly, the model is called mixed-effect model when the hyper-prior is a Gaussian, on which the maximum likelihood (ML) solution can be found with Expectation-Maximization (EM) algorithm.

Traditional mixed-effect models are parametric, which assume a θ-parameterized regression model for each individual. Since the model parameters vary across individuals, it is natural to consider them generated by the sum of a fixed and a random piece $\theta = \alpha + \beta_i$, where α is called the *fixed effect*, and β_i, called *random effect*, is assumed distributed $\mathcal{N}(0,\mathbf{D})$ with unknown covariance $\mathbf{D}$. The fitting of mixed-effect model is to find α, $\mathbf{D}$, and the variance of observation noise.

In non-parametric mixed-effect models, the individual regression models do not take a parametric form. Instead, we assume the observations are generated by k smooth curves $\{f_1, f_2, \cdots, f_k\}$ fluctuating around a mean (fixed-effect) function f_0. We use $\widetilde{f}_i = f_i - f_0$ to denote the deviation of f_i from f_0 (random effect). The prior of both f_0 and $\widetilde{f}_i$ can be summarized with the following equations:

$$\begin{aligned} p_0[f_0] &\propto \exp(-\frac{1}{2}||f_0||^2_{\mathcal{H}_0}) & (15)\\ p_f[\widetilde{f}_i] &\propto \exp(-\frac{1}{2}||\widetilde{f}_i||^2_{\mathcal{H}}) \;\; i = 1,2,\cdots,k, & (16) \end{aligned}$$

where $\mathcal{H}$ and $\mathcal{H}_0$ are generally different Hilbert spaces, with the corresponding reproducing kernel denoted as K and K_0. Also we assume the observation noise to be

white Gaussian with variance σ^2, from which follows

$$p(\mathbf{y}_i|\widetilde{f}_i, f_0; \mathbf{t}_i) \propto \prod_{j=1}^{n_i} \exp(-\frac{(y_{in} - \widetilde{f}_i(t_{in}) - f_0(t_{in}))^2}{2\sigma^2}).$$

We assume $\mathcal{H}_0$ (and thus the form of $p_0[\cdot]$) is predetermined, while the fixed effect f_0 is to be decided. Also unknown are the noise variance σ^2 and the Hilbert space $\mathcal{H}$ for random effects (or equivalently K). Our learning task is therefore to jointly optimize over $\{f_0, K, \sigma\}$ by maximizing the following probability of $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_k\}$.

$$p(\mathbf{Y}|f_0; K, \sigma)p_0[f_0] = p_0[f_0] \prod_{i=1}^{k} \int Df_i \{p(\mathbf{y}_i|\widetilde{f}_i, f_0; \sigma)p_f[\widetilde{f}_i]\}, \quad (17)$$

where the integral $\int D\omega\, g[\omega]$ is a functional integral over ω (Simon, 1979). Using the Gaussian property, Eq.(17) can be further reduced to a standard integral

$$p(\mathbf{Y}|f_0; K, \sigma)p_0[f_0] = p_0[f_0] \prod_{i=1}^{k} \int d\mathbf{f}_i \{p(\mathbf{y}_i|\mathbf{f}_i, f_0; \sigma)p(\mathbf{f}_i; K)\}. \quad (18)$$

where $\mathbf{f}_i = [\widetilde{f}_i(t_{i1}), \widetilde{f}_i(t_{i2}), \cdots, \widetilde{f}_i(t_{iN_i})]^T$ collects the values of f_i on times $\mathbf{t}_i$ and $p(\mathbf{f}_i; K)$ is a standard multivariate Gaussian

$$p(\mathbf{f}_i; K) = \frac{1}{\sqrt{(2\pi)^{N_i}|K(\mathbf{t}_i, \mathbf{t}_i)|}} \exp(-\frac{1}{2}\mathbf{f}_i^T K(\mathbf{t}_i, \mathbf{t}_i)^{-1}\mathbf{f}_i). \quad (19)$$

In general, there is no unique solution of K that maximizes $p(\mathbf{Y}|f_0; K, \sigma)p_0[f_0]$. Indeed, it is easy to verify that if $K(t_{in}, t_{im}) = K'(t_{in}, t_{im})$ for any individual i and time index (n, m), we will have

$$p(\mathbf{Y}|f_0; K, \sigma)p_0[f_0] = p(\mathbf{Y}|f_0; K', \sigma)p_0[f_0].$$

This situation can be circumvented in two ways. First we can restrain K in a particular parametric family, such as the widely used Gaussian kernel. Second, we can instead optimize *only* over the entry $K(t_{in}, t_{im})$ for all individual i, and time index (n, m). Both strategies will be addressed in this paper.

4.1. Optimization with the EM Algorithm

The task is to find the set $\mathcal{M} = \{f_0, K, \sigma\}$ that maximizes the probability $p(\mathbf{Y}|f_0; K, \sigma)p_0[f_0]$. As shown in Eq.(18), we can rewrite the data likelihood $p(\mathbf{y}_i|f_0; K, \sigma)$ using the $\{\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_k\}$ as the latent variables

$$p(\mathbf{y}_i|f_0; K, \sigma) = \int d\mathbf{f}_i p(\mathbf{y}_i|\mathbf{f}_i, f_0, \sigma)p(\mathbf{f}_i; K), \quad (20)$$

which enables us to employ the EM algorithm in finding $\mathcal{M}$. In the following, we will give the results of the expectation step (E-step) and the maximization step (M-step).

E-step: In each EM iteration:

$$Q(\mathcal{M}, \mathcal{M}^g) = E_{\{\mathbf{f}_i|\mathbf{Y};\mathcal{M}^g\}}[\log\{p(\mathbf{Y}, \{\mathbf{f}_i\}; \mathcal{M})p_0[f_0]\}] = \sum_{i=1}^{k} \int d\mathbf{f}_i \log p(\mathbf{y}_i, \mathbf{f}_i; \mathcal{M})p(\mathbf{f}_i|\mathbf{y}_i; \mathcal{M}^g) + \log p[\mathbf{f}_0],$$

where $\mathcal{M}^g$ stands for the parameters from the last iteration. After some algebra, we can re-arrange $Q(\mathcal{M}, \mathcal{M}^g)$ into the following form

$$Q(\mathcal{M}, \mathcal{M}^g) = -\frac{1}{2}||f_0||^2_{\mathcal{H}_0} - n\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{k}\sum_{j=1}^{n_i} E_{\{\mathbf{f}_i|\mathbf{Y};\mathcal{M}^g\}}[(y_{ij} - \widetilde{f}_i(t_{ij}) - f_0(t_{ij}))^2] + \sum_{i=1}^{k} \int d\mathbf{f}_i \log p(\mathbf{f}_i; \mathcal{M})p(\mathbf{f}_i|\mathbf{y}_i; \mathcal{M}^g). \quad (21)$$

M-step: In M-step, we find the

$$\mathcal{M}^* = \arg\max_{\mathcal{M}} Q(\mathcal{M}, \mathcal{M}^g), \quad (22)$$

and use $\mathcal{M}^*$ to update the model parameters. The optimization in Eq.(22) can be divided into two separate parts. The first three terms on the left hand side of Eq.(21) is a function of only (f_0, σ); The last (fourth) term is a function of only K. To find the solution of f_0 and σ, we need to solve the following optimization problem:

$$(\sigma^*, f_0^*) = \arg\min_{\sigma, f_0} \{\frac{1}{2}||f_0||^2_{\mathcal{H}_0} + N\log\sigma + \frac{1}{2\sigma^2}\sum_{i=1}^{k}\sum_{j=1}^{n_i} E_{\{\mathbf{f}_i|\mathbf{Y};\mathcal{M}^g\}}[(y_{ij} - \widetilde{f}_i(t_{ij}) - f_0(t_{ij}))^2]. \quad (23)$$

Particularly, with any fixed σ, maximizing $Q(\mathcal{M}, \mathcal{M}^g)$ over f_0 becomes a regularized regression problem

$$f_0^* = \arg\min_{f_0} \frac{1}{2}||f_0||^2_{\mathcal{H}_0} + \frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{n_i}\{(y_{ij} - E_{\{\mathbf{f}_i|\mathbf{y}_i,\mathcal{M}^g\}}[\widetilde{f}_i(t_{ij})] - f_0(t_{ij}))^2\}.$$

The optimization over K is

$$K = \arg\max_{K\in\mathcal{K}} \sum_{i=1}^{k} \int d\mathbf{f}_i \log p(\mathbf{f}_i; K)p(\mathbf{f}_i|\mathbf{y}_i; K^g) \quad (24)$$

$$= \arg\max_{K\in\mathcal{K}} -\sum_{i=1}^{k}\{\frac{1}{2}\log|K(\mathbf{t}_i,\mathbf{t}_i)| + \frac{1}{2}tr(K(\mathbf{t}_i,\mathbf{t}_i)^{-1}(\mathbf{C}_i^g+\mu_i^g(\mu_i^g)^T))\}, \quad (25)$$

where $\mathcal{K}$ is the set of feasible K, and μ_i is the posterior mean $E[\mathbf{f}_i|\mathbf{y}_i;\mathcal{M}]$ that can be calculated as

$$\mu_i = K(\mathbf{t}_i,\mathbf{t}_i)(K(\mathbf{t}_i,\mathbf{t}_i)+\sigma^2\mathbb{I})^{-1}(\mathbf{y}_i-\mathbf{f}_{0,i})$$

and $\mathbf{C}_i$ is the posterior covariance of $\mathbf{f}_i$

$$\mathbf{C}_i = K(\mathbf{t}_i,\mathbf{t}_i) - K(\mathbf{t}_i,\mathbf{t}_i)(K(\mathbf{t}_i,\mathbf{t}_i)+\sigma^2\mathbb{I})^{-1}K(\mathbf{t}_i,\mathbf{t}_i).$$

4.2. Parametric Covariance Estimation

We assume the covariance function K is of the parametric form $K(x,y;\theta)$. For example, the Gaussian kernel with scale a and kernel width s

$$K(x,y;\{a,s\}) = a\exp(-\frac{||x-y||^2}{2s^2}),$$

or as suggested in (Lanckriet et al., 2004) a convex combination of a set of kernels $\{K_1,K_2,\cdots,K_M\}$

$$K(x,y;\lambda) = \lambda_1K_1(x,y)+\lambda_2K_2(x,y)+\cdots+\lambda_MK_M(x,y).$$

In this case, the optimization of K in the M-step can be reduced to the following parameter estimation

$$\theta^* = \arg\max_{\theta} -\sum_{i=1}^{k}\{\frac{1}{2}\log|K(\mathbf{t}_i,\mathbf{t}_i;\theta)| + \frac{1}{2}tr(K(\mathbf{t}_i,\mathbf{t}_i;\theta)^{-1}(\mathbf{C}_i^g+\mu_i^g(\mu_i^g)^T))\}, \quad (26)$$

where $p(\mathbf{f}_i;\theta) = p(\mathbf{f}_i;K(\mathbf{t}_i,\mathbf{t}_i;\theta))$. This parametric form of K is appealing in either one of the following two situations:

- when the observation are sparse, since the parametric K is generally less prone to overfitting compared to the non-parametric estimation, as will be discussed in Section 4.3.
- when the time series are not synchronized (as in Section 5.1) since the parametric K allows the out-of-sample extension.

4.3. Non-parametric Covariance Estimation

When all the time series all synchronized, we have $\mathbf{t}_i = \mathbf{t}, i = 1,2,\cdots,k$. We can replace $K(\mathbf{t}_i,\mathbf{t}_i)$ in Eq.(25) with $\mathbf{K}\equiv K(\mathbf{t},\mathbf{t})$, and rewrite the optimization into the matrix form

$$\mathbf{K} = \arg\max_{\mathbf{K}\in\mathcal{P}} -\sum_{i=1}^{N}\{\frac{1}{2}\log|\mathbf{K}| + \frac{1}{2}tr(\mathbf{K}^{-1}(\mathbf{C}_i^g+\mu_i^g(\mu_i^g)^T))\}. \quad (27)$$

If we let $\mathcal{P}$ be the set of positive definite matrix, the solution of Eq.(27) is simple

$$\mathbf{K} = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{C}_i^g+\mu_i^g(\mu_i^g)^T). \quad (28)$$

The non-parametric fitting of kernel matrix $\mathbf{K}$ is appealing since it does not assume a particular form for the covariance matrix and thus can fully exploit the information in the samples. However it can only be used when the time series are synchronized. One example of this modeling choice is given in Section 5.2.

5. Experiments

We tested the proposed distance measure on two real-world applications. The first one is an algorithm for cognitive decline detection based on longitudinal clinical observations of motor ability. The second one is an target identifier system based on electroencephalograph (EEG) signal.

In each experiment, we employ support vector machine (SVM) (Burges, 1998) with Gaussian kernel defined as follows

$$\mathbf{G}_{ij} = \exp(-\frac{d_{ij}}{2r^2}) \quad (29)$$

where d_{ij} is the squared distance between the time series i and j and the kernel width r is usually obtained using cross-validation. It is easy to see the $\mathbf{G}$ is a Mercer kernel.

5.1. Cognitive Decline Detection Based on Longitudinal Data

Research by our group and others show that motor changes, such as in walking and finger tapping rates, can effectively predict cognitive decline several years before impairment is manifest (Camicioli et al., 1998). It would be useful to build a system to detect cognitive decline (at least partially) from motor behavior, since they can be obtained by unintrusive in-home assessment (Hayes et al., 2004). Our research focuses on using clinical motor behavior and data from the Oregon Brain Aging Study (OBAS) (Green et al., 2000). All 143 subjects in the cohort were healthy at entry, and when the data were drawn 46 of them had developed into mild cognitive impairment, while 97 remained cognitively healthy. We divide all the subjects into the impaired group and the normal group according to their state when the data were drawn from the database. [3] We intend to predict whether a subject

[3] This grouping is potentially inaccurate due to the possibility that those cognitively healthy subjects can later develop into dementia, which is known as right censoring in survival analysis.

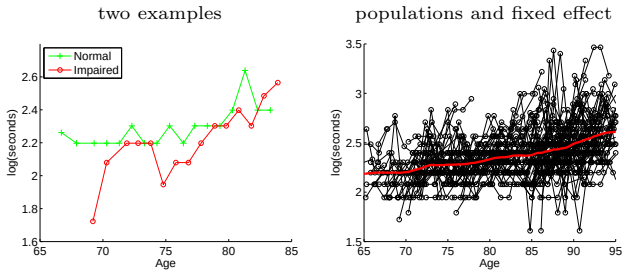


Figure 2. Left panel: sample spaghetti plots of seconds from two groups. Right panel: the population of seconds data and the fit fixed effect model (red line).

would develop into cognitive impairment based on his or her motor behavior before a clinical diagnosis (if any). In this experiment, this task reduces to predicting the group membership for each subject. This classification is difficult due to the fact that motor observations are sparse and noisy, as shown in Fig.2(left panel). We examined four motor behaviors summarized in Table 1. Usually as the subjects age or become impaired, the seconds and steps increase, while tappingD and tappingN decrease.

seconds	# of seconds the subject takes to walk 9 m
steps	# of steps the subject takes to walk 9 m
tappingD	# of the tappings the subject does in 10 seconds with the dominant hand
tappingN	# of the tappings the subject does in 10 seconds with the non-dominant hand

Table 1. Description of data.

We fit the non-parametric mixed-effect model to each motor behavior with the parameterized kernel

$$K_0(t_1,t_2) = \exp(\frac{||t_1-t_2||^2}{2s_0^2}),$$
$$K(t_1,t_2;\{a,s\}) = a\exp(\frac{||t_1-t_2||^2}{2s^2}),$$

where s_0 is predetermined and $\{a,s\}$ are to be learnt. The right panel of Fig.2 shows the seconds time series from the 143 subjects (black $-\circ-$) and the fit fixed effect (red line). Once the model is fit, the distance between any two subjects i and j is calculated as in Eq.(11).

For comparison, we also examined a parametric feature based on the least-square (LSQ) fit coefficients for linear regression: $\mathbf{x}_i = \arg\min_{\mathbf{x}} \sum_{j=1}^{N_i}(x_0 + x_1 t_{ij} - y_{ij})^2$ with $\mathbf{x} = [x_0, x_1]^T$. This feature extraction is justified by the observation that the intercept and the slope of the motor behavior trajectory are predictor of future cognitive decline and dementia (Marquis et al., 2002). Based on the LSQ feature we get another distance measure $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||^2$. We employ a SVM as the classifier with kernels calculated with Eq.(29). Fig.3 compares the ROC curves using the proposed distance measure and the Euclidean distance between the LSQ features. It is clear that SVM with proposed distance measure outperforms the SVM with the LSQ features in terms of the area under curve (AUC). There are two reasons for the superiority of the proposed distance over the LSQ feature:

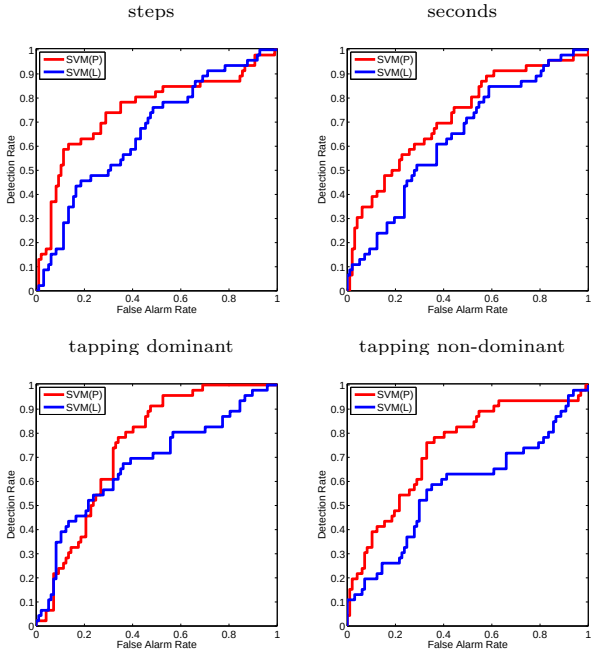


Figure 3. The ROC curve of SVM with two distance measures. SVM(P): SVM with proposed distance. SVM(L): SVM with least-square features.

- The simple heuristic features such as the intercept and the slope cannot capture enough information for the classification.
- The feature extraction is not robust enough for the sparse and noisy observations.

5.2. EEG-based Image Target Detection

The system reported here exploits the perceptual capabilities of expert humans for searching objects of interest (e.g., a golf course in a satellite image) within large image sets. The technique uses event related potentials (ERPs), neural signals linked to critical events, such as interesting/novel visual stimuli. The basic idea of the ERP-based image triage system is to collect electroencephalograph (EEG) signals from a subject's scalp when he or she performs visual target detection, and then detect the ERPs associated with the target

stimuli. We focus on single-trial ERP detection using 32 EEG sensors, which is challenging due to the low signal-to-noise ratio.

This detection task is then boiled down to classifying the EEG segments into target-associated EPRs and distractors. After proper alignment and sampling, the EEG segments are transformed into synchronized sequence of length 4128, which are denoted $\mathbf{y}_i$ for each individual trial i. In this experiment, we collected the EEG data from three human experts, each of them performed 1 training session and 7 test sessions. In each training session, the human expert was fed with ∼600 images with ∼50 targets among them. In each test session, there are 1-4 targets within ∼3000 distractors. Fig.4 (left panel) shows single-trial EEG signals associated with a target and a distractor stimulus.

Due to the high dimensionality, the EM algorithm will be fairly slow due to the extensive use of inverse of $\mathbf{K}$ (4128×4128). To keep the computation at a reasonable level, we simplify the model by assigning a flat prior to the fixed effect f_0, or equivalently letting $||f||_{\mathcal{H}_0} = 0$ for any f. This simplifying assumption instantly leads to the following results.

- The optimal solution of $\mathbf{f}_0$ is simply the data mean $\mathbf{f}_0 = \frac{1}{k}\sum_{i=1}^{k} \mathbf{y}_i$, as shown in Fig.4 (right panel).
- The data likelihood is independent of σ^2 as long as it is less than the smallest eigenvalue of $\hat{\mathbf{K}} = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{y}_i - \mathbf{f}_0)(\mathbf{y}_i - \mathbf{f}_0)^T$.

Based on the above two results, we can pick a σ and then calculate the optimal covariance $\mathbf{K}$ with Eq.(28) in one iteration.

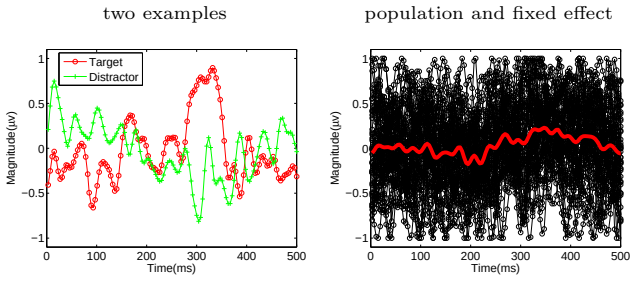


Figure 4. EEG data and the fit mixed-effect model. Left panel: Example of target-associated and distractor-associated EEG signals. Right panel: The population of EEG signals (black –○–) and the fit $\mathbf{f}_0$ (red curve).

Once the optimal $\mathbf{f}_0$ and $\mathbf{K}$ are obtained, the distance between any time series i and j can be calculated using Eq.(14). In addition to directly using the distance, we isometrically embed the time series $\{\mathbf{y}_i\}$ into Euclidean space while preserving the distance expressed in Eq.(14). The embedded vectors, called *ISO feature*, will then be used directly in linear classifiers. One obvious choice is the non-degenerated linear transformation

$$\mathbf{x}_i = \mathbf{K}^{1/2}(\mathbf{K} + \sigma^2\mathbb{I})^{-1}\mathbf{y}_i \tag{30}$$

where $\mathbf{K}^{\frac{1}{2}}$ could be any matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ with $\mathbf{A}\mathbf{A}^T = \mathbf{K}$. We tested both the proposed distance and the (squared) Euclidean distance [4] $||\mathbf{y}_i - \mathbf{y}_j||^2$ as the distance term d_{ij} in the Gaussian kernel $\mathbf{G}$ and compared the performance of the SVM with the two distance measures. In addition, we also tried a linear logistic classifier (LLC) with both the raw feature $\mathbf{y}_i$ and ISO feature $\mathbf{x}_i$ as the input. In our experiment, the SVM parameters and kernel width were selected using 10-fold cross validation.

Due to the extremely low probability of targets and the high cost of misdetection, we aim for zero-miss and minimum false alarm rate (MFAR), which is defined as the percentage of false alarms among all classifications while all targets are correctly detected. We test both SVM and LLC on the 21 ($=3 \times 7$) test sessions. Table 1 summarizes the detection results when different distance or features are used. The criteria of comparison include the average MFAR across the 21 sessions, the number of sessions with low MFAR ($\leq 10\%$)and very low MFAR ($\leq 2\%$). Clearly, the LLC with ISO features outperforms the LLC with raw feature by giving low average MFAR, more low MFAR sessions, and more very low MFAR sessions. The story is similar when using SVM as the classifier: the proposed distance outperforms the the Euclidean distance on all three criteria.

Clearly the temporal structure is important in describing the EEG signal, and thus plays a crucial role in deciding the distances between EEG time series. The proposed distance measure successfully incorporates the temporal structure information learnt with a rather simple algorithm, and yields significantly better classification than the Euclidean distance that simply adds the index-by-index differences.

6. Related Work

The connection between Bregman divergence and exponential family is first proposed by (Forster & Warmuth, 2000), and later used by several authors in deriving a proper distance measure for either clustering (Banerjee et al., 2005) or dimension reduction (Collins et al., 2001). Our work also depends heavily on the functional Bregman divergence, an idea first fully explored in (Frigyik et al., 2006). The non-parametric

[4] It would be fair to learn the covariance for a Mahalanobis distance. However, we have not performed this comparison for the sake of simplicity.

	Aver. MFAR	$\# \leq 2\%$	$\# \leq 10\%$
LLC(I)	8.99%	12	16
LLC(R)	18.18%	2	12
SVM(P)	4.91%	13	19
SVM(E)	6.31%	7	16

Table 2. The detection results with different classifier settings. *Columns*: AverMFAR: the average MFAR across 21 sessions; #≤ 2%: the number of sessions with MFAR≤ 2%; #≤ 10%:the number of sessions with MFAR≤ 10%. *Rows:* LLC(I): LLC with the ISO feature; LLC(R): LLC with raw feature; SVM(P): SVM with the proposed distance; SVM(E): SVM with Euclidean distance.

mixed-effect model is a natural generalization to the hierarchical Bayesian Gaussian process proposed by (Schwaighofer et al., 2005) to functional form where synchronized and non-synchronized time series can be treated in a unified framework.

This work can be viewed as a particular example of the functional data analysis (Ramsay & Silverman, 1997). Particularly, in an early effort towards the functional PCA (Ramsay & Dalzell, 1991), the authors suggested to map the discrete observations $(\mathbf{t}_i, \mathbf{y}_i)$ to a smooth function through the following regularized regression

$$\hat{f}_i(t) = \arg\min_f \frac{1}{2}\sum_{n=1}^{N_i}(y_{in} - f(t_{in}))^2 + \frac{1}{2}\lambda||\mathcal{D}f||^2, \quad (31)$$

where $\mathcal{D}$ is a linear operator. The solution to Eq.(31) is the expectation in Eq.(9) if we let $\lambda = \sigma^2$ and K be the Green's function of the operator $\mathcal{D}^*\mathcal{D}$. The difference, however, are that (1) our model also assumes a non-zero mean (fixed effect) f_0 and (2) the kernel K is learned from a population of time series.

Acknowledgments

ZL wants to thank Inderjit Dhillon, Arindam Banerjee, Jeff Kaye, and Misha Pavel for helpful discussions, and Robin Guariglia and Milar Moore for help with the OBAS data. TL, ZL, and DE were partially funded by the OHSU Behavioral and Intervention Commons project funded by Intel Corporation. YH is supported by NSF: ECS-0524835, ECS-0622239, and IIS-0713690.

References

Altun, Y., Hofmann, T., & Smola, A. (2004). Exponential families for conditional random fields. *Uncertainty in Artificial Intelligence(UAI)*.

Banerjee, A., Merugu, S., Dhillon, I. S., & Ghosh, J. (2005). Clustering with bregman divergences. *JMLR*, *6*, 1705–1749.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167.

Camicioli, R., Howieson, D., Oken, B., Sexton, G., & Kaye, J. (1998). Motor slowing precedes cognitive impairment in the oldest old. *Neurology*, *50*.

Collins, M., Dasgupta, S., & Schapire, R. (2001). A generalization of principal component analysis to the exponential family. *NIPS 13*.

Forster, J., & Warmuth, M. K. (2000). Relative expected instantaneous loss bounds. *COLT 13*.

Frigyik, B., Srivastava, S., & Gupta, M. (2006). Functional bregman divergence and bayesian estimation of distributions. arXiv:cs/0611123.

Gelman, A. (2004). *Bayesian data analysis, second edition*. Chapman & Hall.

Green, M., Kaye, J., & Ball, M. (2000). The oregon brain aging study: Neuropathology accompanying healthy aging in the oldest old. *International Journal of Neural System*, *54*, 105–113.

Hayes, T., Pavel, M., & Kaye, J. (2004). An unobtrusive in-home monitoring system for detection of key motor changes preceding cognitive decline. *26th Annual International Conference of the IEEE EMBS*.

Keogh, E., & Pazzani, M. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. *KDD 98* (pp. 239–241). ACM Press.

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L., & Jordan, M. (2004). Learning the kernel with semidefinite Programming. *Journal of Machine Learning Research*, *5*, 27–72.

Marquis, S., Moore, M., Howieson, D. B., Sexton, G., Payami, H., Kaye, J. A., & Camicioli, R. (2002). Indpendent predictors of cognitive decline in healthy elderly persons. *Arch. Neurol.*, *59*, 601–606.

Parra, L., Alvino, C., Tang, A., Pearlmutter, B., Yeung, N., Osman, A., & Sajda, P. (2003). Single trial detection in eeg and meg: Keeping it linear. *Neurocomputing*, *52-54*, 177–183.

Ramsay, J., & Silverman, B. (1997). *Functional data analysis*. Springer-Verlag.

Ramsay, J. O., & Dalzell, C. J. (1991). Some tools for functional data analysis. *Journal of the Royal Statistical Society, Series B (Methodological)*, *53*, 539–572.

Rasmussen, C., & Williams, C. (2006). *Gaussian processes for machine learning*. Cambridge, MA: MIT Press.

Schwaighofer, A., Tresp, V., & Yu, K. (2005). Learning gaussian process kernels via hierarchical bayes. *NIPS17*.

Seeger, M. (2004). Gaussian process for machine learning. *Internation Journal of Neural System*, *14*, 69–106.

Simon, B. (1979). *Functional integration and quantum physics*. Academic Press.

Tipping, M. (1999). Deriving cluster analytic distance functions from gaussian mixture models.

On-line Discovery of Temporal-Difference Networks

Takaki Makino MAK@SCINT.DPC.U-TOKYO.AC.JP
Division of Project Coordinate, Tokyo University, 5-1-5 Kashiwa-no-ha, Kashiwa-shi, Chiba 277-8568 Japan

Toshihisa Takagi TT@K.U-TOKYO.AC.JP
Database Center for Life Science, Research Organization of Information and Systems, Tokyo 113-0022 Japan

Abstract

We present an algorithm for on-line, incremental discovery of temporal-difference (TD) networks. The key contribution is the establishment of three criteria to expand a node in TD network: a node is expanded when the node is well-known, independent, and has a prediction error that requires further explanation. Since none of these criteria requires centralized calculation operations, they are easily computed in a parallel and distributed manner, and scalable for bigger problems compared to other discovery methods of predictive state representations. Through computer experiments, we demonstrate the empirical effectiveness of our algorithm.

1. Introduction

Predictive representations (Littman et al., 2002; Jaeger, 2000) are a relatively new group of approaches for expressing and learning grounded knowledge about dynamical systems. These approaches represent the state of a dynamical system as a vector of predictions, based on the hypothesis that important knowledge about the world can be represented strictly in terms of relationships between predictions of observable quantities. In the *predictive state representations* (PSRs) introduced by Littman et al. (2002), each prediction is an estimate of the probability of *tests*, defined as some sequence of observations given a sequence of actions. Sutton and Tanner (2005) proposed another approach for predictive representations, namely Temporal-Difference (TD) networks. TD networks are developed as a generalization of PSRs: in TD networks, each prediction is an estimate of the probability or expected value of some function of future predictions, actions and observations. The predictions can be considered as "answers" to a set of "questions" represented in the TD network.

One important problem in the research of predictive representations is the *discovery problem*, that is, the problem of determining the set of questions (or *core tests*) so that the state of a dynamical system is correctly represented by the vector of predictions for these questions. Many of the existing studies on this problem (Rosencrantz et al., 2004; James & Singh, 2004; Wolfe et al., 2005) utilize off-line discovery and learning on PSRs, and therefore they are hardly applicable to both implementing live-interaction agents and finding corresponding activity in the human brain. There is an on-line discovery algorithm (McCracken & Bowling, 2006) for PSR core tests, but it requires complex operations on a large matrix, such as calculation of the maximum linear independent set, and is therefore not suitable for parallel and distributed computing. As far as we are aware, no algorithm has been proposed for discovery of questions in TD networks. Study of a TD-network discovery algorithm that is suitable for parallel distributed computing would contribute not only to research on predictive representations but also to research on cognitive science by providing a hypothesis for the algorithm actually used in the human brain.

In this study, we propose an algorithm for discovering the correct set of tests by incremental node expansion in a TD network. Our key contribution is in the criteria that we have developed for node expansion: a node is expanded when the node is well-known, independent, and has a prediction error that requires further explanation. To check these criteria, our algorithm maintains the average squared error and average variance for each node, and it introduces a dependency detection network. Since none of these criteria requires centralized operations such as calculation of linear independence in the whole representation matrix, they

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

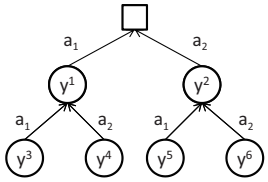


Figure 1. Example TD network we focus in this paper.

are easily computed in a parallel and distributed manner, which is an important property for seeking the algorithm used in the human brain. Although the algorithm has no theoretical guarantee to find the question network (indeed, it is known to be failed in some cases), our simulation experiments demonstrate the empirical effectiveness of our algorithm.

Section 2 reviews PSR and TD network that is necessary to understand our algorithm. Section 3 describes our incremental discovery algorithm. Experiments and results are shown in Section 4. After that, we discuss related work and future directions in Sections 5 and 6.

2. TD Networks

In this section, we make a brief review on TD network, based on the work by Tanner and Sutton (2005a).

The purpose of TD networks is to learn prediction of future observation obtained from the environment. Consider a partially observable environment, which changes its state according to an agent's action $a_t \in \mathcal{A}$ at every time step t, but the agent can only have a partial (and possibly noisy) observation $o_{t+1} \in \mathcal{O}$ of the state. Generally o_t can be a vector consisting of l bits, but in this paper, we consider the case that the observation is a single bit ($l = 1$).

A TD network consists of a set of nodes and links, and each node represents a single scalar prediction. A node has one or more links directed to other nodes or the observation from the environment, which denotes the targets for prediction of the node. A link may have a condition, which indicates that the node is a conditional prediction of the target. This set of nodes and links are called the *question network* since each node is some question about the environment.

As in the previous studies (Tanner & Sutton, 2005b; Tanner & Sutton, 2005a), we focus on a subset of TD networks, in which every node has a single target (hereafter, the parent node) and every link is conditioned with an action. Figure 1 is an example of such a TD network. The node y^1 predicts the observation at the next step if action a_1 is taken. The node y^4 predicts the value of the node y^1 at the next step if action a_2 is taken, and so on.

To provide an answer for the questions asked by the question network, each node in a TD network works also as a function approximator. The inputs to the function approximator of a node are defined by *answer network*, taking values from other nodes, available observations, and actions to be taken. These function approximators are trained so that the output of the nodes becomes the answers to the question asked by the question network. However, to provide an accurate answer, the set of nodes have to be a sufficient representation for the environmental state; in other words, a correct set of questions have to be posed by the question network. The focus of this paper is the discovery of the question network, i.e., to find the structure of the question network that is sufficient for prediction.

Formally, we denote the prediction for node i at time step t as $y_t^i \in [0, 1]$, $i = 1, \ldots, n$. The prediction vector $\mathbf{y}_t = (y_t^1, \ldots, y_t^n)^T$ is given by the answer network:

$$\mathbf{y}_t = \sigma(\mathbf{W}_t \mathbf{x}_t) \quad , \tag{1}$$

where $\mathbf{x}_t \in \Re^m$ is a feature vector, $\mathbf{W}_t$ is a $n \times m$ matrix of modifiable weights, and σ is the S-shaped logistic function $\sigma(s) = (1 + e^{-x})^{-1}$.

The feature vector is a function of the preceding action, observation, and node values:

$$\mathbf{x}_t = \mathbf{x}(a_{t-1}, o_t, \mathbf{y}_{t-1}) \in \mathbb{R}^m \quad . \tag{2}$$

We used the similar form of feature vector as appeared in the work of Tanner and Sutton (2005a); in our experiments, where two actions (L and R) are possible,

$$\mathbf{x}(a, o, \mathbf{y}) = \begin{cases} (o, 1-o, y^1, \ldots, y^n, 0, \ldots, 0)^T & a = \mathrm{L} \\ (0, \ldots, 0, o, 1-o, y^1, \ldots, y^n)^T & a = \mathrm{R} \end{cases} \tag{3}$$

This is equivalent to separate W for each action.

The question network, which gives the target of predictions in terms of the node values at the next time steps, is represented by a $n \times (n + l)$ matrix Z^a and a vector c. Without eligibility traces, the vector of the target values is

$$\mathbf{z}_{t-1} = \mathbf{c}_t \odot \mathbf{Z} \begin{pmatrix} \mathbf{y}_t \\ o_t \end{pmatrix} + \bar{\mathbf{c}}_t \odot \mathbf{y}_{t-1} \quad , \tag{4}$$

where $\odot$ is element-by-element multiplication, and each element of $\mathbf{Z}$, $\mathbf{c}$ and $\bar{\mathbf{c}}$ is:

$$z^{ij} = \begin{cases} 1 & y^i \text{ is the parent node of } y^j \\ 0 & \text{otherwise} \end{cases} , \tag{5}$$

$$c_t^i = \begin{cases} 1 & a_t \text{ satisfies the node } y^i\text{'s condition} \\ 0 & \text{otherwise} \end{cases} , \tag{6}$$

$$\bar{c}_t^i = 1 - c_t^i \quad . \tag{7}$$

The elements of Z are assigned so that $z_{t-1}^i = y_t^{p(i)}$ for any i with $c_t^i = 1$, where $p(i)$ is the parent node of i.

On the other hand, if $c_t^i = 0$, $z_{t-1}^i = y_{t-1}^i$, and the TD error of the node becomes zero so that only the weights for the nodes that satisfy the condition are updated.

In this study we employ eligibility traces (Tanner & Sutton, 2005a), which is a technique to accelerate learning in TD-error learning by incorporating further prediction into the learning target. In a forward view,

$$\mathbf{z}_{t-1} = \mathbf{c}_t \odot \mathbf{Z}(\lambda \mathbf{z}_t + (1-\lambda)\mathbf{y}_t) + \bar{\mathbf{c}}_t \odot \mathbf{y}_{t-1} \quad , \quad (8)$$

where $\lambda \in [0,1]$ is a parameter that controls the balance of temporally distant results in the learning target. When $\lambda = 0$, eq. (8) is equivalent to eq. (4), and no eligibility traces are used.

However, this formula recursively contains future values of z, and it is not easy to be calculated on-line. Tanner and Sutton (2005a) proposes an algorithm that performs on-line update of the weight vector to make the equivalent update as (8). Then each component w_t^{ij} of $\mathbf{W}_t$ is updated by the learning rule:

$$\begin{aligned} w_{t+1}^{ij} &= w_t^{ij} + \alpha(z_t^i - y_t^i)\frac{\partial y_t^i}{\partial w_t^{ij}} \\ &= w_t^{ij} + \alpha(z_t^i - y_t^i)y_t^i(1-y_t^i)x_t^j \quad , \end{aligned} \quad (9)$$

in which the second line is derived from eq. (1).

Roughly, the operation of a TD network proceeds by repeating the following steps: (1) Choose an action a_{t-1} and receive an observation o_t from the environment. (2) Operate the answer network, i.e., calculate feature vector $\mathbf{x}_t = \mathbf{x}(a_{t-1}, o_t, \mathbf{y}_{t-1})$ and obtain the new predictions $\mathbf{y}_t = \sigma(\mathbf{W}_t\mathbf{x}_t)$. (3) Use the question network to obtain the target value for the previous predictions $\mathbf{z}_{t-1} = \mathbf{z}(\mathbf{y}_t, o_t)$, and update the weights W according to the TD error $\mathbf{z}_{t-1} - \mathbf{y}_{t-1}$. For details, readers should consult the original paper of the TD network (Sutton & Tanner, 2005) to see subtle points, such as the precise order of calculation.

3. On-line Discovery Algorithm

We propose an algorithm that performs on-line discovery of the question network. Our algorithm starts with the minimal network, which consists of the observation node and a set of prediction nodes for the observation nodes, one for each action. During learning, the algorithm grows the network by *expanding* leaf nodes by adding a set of prediction nodes for the node. Intuitively, a node is expanded when the following three criteria holds:

1. **The node is well-known**: The agent has sufficient experience to learn the node. This criterion prevents relatively unexplored nodes to be expanded.

2. **The node is independent**: The prediction of the node cannot be calculated in terms of other, formerly known node values. In terms of PSRs, the node represents a core test. This criterion avoids redundant expansion of the nodes.

3. **The node's error requires further explanation**: The node's prediction error is not smaller than expected from the prediction error of the node's parent node. This criterion chooses the node that has the best descriptive power for the error in the parent node, and stops expansion when unpredictability is solved.

In the following, we first describe the variables that our algorithm maintains to check these criteria, and we present more detailed conditions for the criteria.

3.1. Variables

3.1.1. Dependency Detection Network

Our algorithm uses a dependency detection network, which tries to represent a prediction of the node y^i in terms of observation o and values of the nodes with younger index y^j $(j < i)$. If the network succeeds to represent y^i with small error, we can see that y^i is dependent to the predictions with younger index (namely, not a core test of PSRs), and exclude it from the candidate of node expansion. Otherwise, we can assume that y^i is an independent node. Note that it corresponds to the core test in non-linear PSRs (Rudary & Singh, 2004) because the nodes in TD networks correspond to e-tests in non-linear PSRs (Sutton & Tanner, 2005) and we use sigmoidal function in the answer network.

Formally, the dependency detection network is represented by $\mathbf{D}_t$, a $n \times (n+l)$ matrix of modifiable weights. In the matrix d^{ij} is restricted to zero if $i \geq j - l$. The output of the network is $\mathbf{d}_t = \sigma(\mathbf{D}_t \binom{o_t}{\mathbf{y}_t})$. The network is trained so that $\mathbf{d}_t$ is close to $\mathbf{y}_t$; in other words, the network tries to represent a prediction of the node y^i in terms of observation o and predictions with nodes with smaller index y^j $(j < i)$. Since the indices of the nodes are numbered in order, newly expanded nodes are always given higher indices, and are not used by the dependency detection network for describing older nodes with lower indices.

Each component d_t^{ij} of $\mathbf{D}_t$ is updated by the learning rule similar to eq. 9:

$$d_{t+1}^{ij} = d_t^{ij} + \alpha_D(y_t^i - d_t^i)\frac{\partial d_t^i}{\partial d_t^{ij}} \quad (10)$$

$$= d_t^{ij} + \alpha_D(y_t^i - d_t^i)d_t^i(1-d_t^i)y_t^j \quad (11)$$

But the update is limited in the area $i < j - l$. We assign the learning rate of the dependency detection network α_D to be larger than that of the answer network α to allow the dependency detection network to track changes in the calculated node values during the learning of the answer network (as long as the values are dependent).

3.1.2. Average Errors

Our algorithm gathers statistical information about the prediction and error of the TD network and the dependency detection network. To allow on-line learning, the statistical variables are calculated in forms of exponential moving averages:

$$\mathbf{y}_{t+1}^{\mathrm{LERR}} = \rho_2 \mathbf{y}_t^{\mathrm{LERR}} + (1-\rho_2)((\mathbf{z}_t - \mathbf{y}_t) \odot (\mathbf{z}_t - \mathbf{y}_t)) \quad (12)$$

$$\mathbf{d}_{t+1}^{\mathrm{SERR}} = \rho_1 \mathbf{d}_t^{\mathrm{SERR}} + (1-\rho_1)((\mathbf{y}_t - \mathbf{d}_t) \odot (\mathbf{y}_t - \mathbf{d}_t)) \quad (13)$$

$$\mathbf{d}_{t+1}^{\mathrm{LERR}} = \rho_2 \mathbf{d}_t^{\mathrm{LERR}} + (1-\rho_2)((\mathbf{y}_t - \mathbf{d}_t) \odot (\mathbf{y}_t - \mathbf{d}_t)) \quad (14)$$

where $\mathbf{d}_t^{\mathrm{SERR}}$ is a short-term average of squared prediction errors, $\mathbf{d}_t^{\mathrm{LERR}}$ is a long-term average of squared prediction errors, and $0 < \rho_1 < \rho_2 < 1$ is a temporal factor. When a node y^i is added to the network, statistical variables are initialized as $y^{\mathrm{LERR}\ i} = d_t^{\mathrm{SERR}\ i} = d_t^{\mathrm{LERR}\ i} = 1.0$so that the variables show larger errors during the initial period of the node.

Without eligibility traces, the target variable $\mathbf{z}_t$ is available at time $t+1$, so these parameters are easily calculated on-line. However, since eq. (12) contains quadratic term for $\mathbf{z}_t$, on-line calculation technique with eligibility traces such as used in Tanner and Sutton's work (2005a) cannot be used directly. Our implementation keeps the record of last k steps of node values, where k is the maximum depth of the current question network, and calculates errors of $\mathbf{y}_t$ and $\mathbf{d}_t$ at time $t+k$.

3.2. Expansion Criteria

Using these variables, we check the criteria described in the beginning of Section 3 as follows. The criteria are checked for every time step. Since all criteria are described on exponential moving averages, the precise timing of criteria check and node expansion is not important; it should be inserted somewhere in the TD(λ) network learning algorithm (Tanner & Sutton, 2005a). The expansion criterion are designed to avoid redundant expansion as much as possible because no shrinking criterion is given.

3.2.1. The node is well-known

To avoid expanding relatively unexplored nodes, we use the following criteria to determine whether the node y^i is well-known.

- Learning error is not in a decreasing trend (we assume the error is always decreasing during initial learning phase).
- Learning error of the node gets smaller compared with that of its parent node.

In formal representation,

$$d_t^{\mathrm{SERR}\ i} \geq d_t^{\mathrm{LERR}\ i} \quad and \quad (15)$$

$$d_t^{\mathrm{LERR}\ i} \leq d_t^{\mathrm{LERR}\ p(i)} . \quad (16)$$

If $p(i)$ is the observation bit, then $d_t^{\mathrm{LERR}\ p(i)}$ is considered as 1 (the largest possible value).

Eq. 15 works because these variables, representing moving averages of errors, are initialized with the highest possible value (see Section 3.1.2). Thus it is expected that the short-term average error variables stay lower than the long-term ones until the end of the initial learning period, in which the learning error is constantly decreasing.

Eq. 16 is usually satisfied with a plenty amount of experience because the dependency network has no connection from a child node to a parent node. The parent node always has fewer inputs in the dependency network than the child node; if the child node has an unexplained dependency (large $d_t^{\mathrm{LERR}\ i}$), it is likely that the parent node also has an unexplained dependency.

3.2.2. The node is independent

To prevent dependent (non-core test) nodes to be expanded, we require that the learning error in the dependency detection network is not small.

$$d_t^{\mathrm{LERR}\ i} \geq \theta_1 \quad (17)$$

θ_1 is a threshold parameter that controls the requirement for independence.

When all the nodes that match this criterion are expanded, then all the leaf nodes in the question network becomes dependent nodes, and no further expansion occurs. Thus, this criterion is equivalent to the assumption that independent (core test) nodes does not exist as a child of dependent (non-core test) nodes.

3.2.3. The node's error requires further explanation

If the prediction error of a node is larger than expected from its parent node, it is reasonable to require further prediction on the node to reduce the error. Otherwise, we can infer that the error in the parent node has other causes (e.g. another prediction node with different action conditions has large prediction error), and further

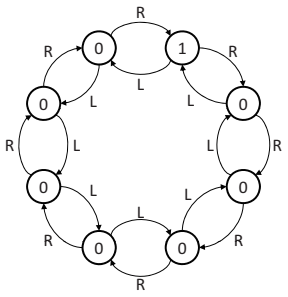


Figure 2. 8-state ring world. The digit in a node represents observation from the state, and edge labels denote actions.

Table 1. Tests for various θ_1 and θ_2 values in the 8-state ring world. Values are [final number of nodes] / [steps ($\times 10^4$) until learned (MSE reduces less than 10^{-4})].

$\theta_1 \backslash \theta_2$	0.005	0.0075	0.01	0.0125	0.015
0.001	18/47	20/46	16/49	6/–	6/–
0.002	18/49	18/48	18/51	6/–	6/–
0.003	18/53	18/55	16/55	6/–	6/–
0.004	18/54	18/57	16/58	6/–	6/–
0.005	18/59	18/59	16/60	6/–	6/–

prediction for the node is less important to reduce the error of the final prediction for the observation.

In case that the parent node $p(i)$ is purely probabilistic, error distribution of the $p(i)$'s child nodes is proportional to the probability that the conditions of the nodes are matched; the following condition checks that the error of the node is greater than that:

$$y_t^{\mathrm{LERR}\ i} \geq \gamma \frac{\#y^i}{\#y^{p(i)}} y_t^{\mathrm{LERR}\ p(i)} + \theta_2 \quad , \qquad (18)$$

where $\#y^i$ is the frequency that the conditions on the chain from the observation bit to node y^i is matched (thus, $\frac{\#y^i}{\#y^{p(i)}}$ is the relative probability that the condition of the node is matched). If $p(i)$ is the observation bit, then $y_t^{\mathrm{LERR}\ p(i)}$ is assumed to be zero. θ_2 is a threshold parameter that controls tolerance for noise.

4. Experiments

To test the efficiency of our algorithm, we conducted a series of computer experiments on n-state ring world ($n = 5, 8$) (Tanner & Sutton, 2005b). Figure 2 illustrates 8-state ring world. There are two observations, 0 or 1, and two actions, L and R.

Since the ring worlds contains only deterministic state transitions and observations, we also tested our algorithm on some standard probabilistic POMDP environments taken from repository (Cassandra, 1999). Among them, environments with one-bit observation are used, and adapted to non-reward situation (we followed a previous work that describe details; McCracken, 2005).

We generated a sequence of experience by a uniform random policy and applied our algorithm online. Through all experiments we used $\rho_1 = 0.99$, $\rho_2 = 0.999$, $\alpha_D = 0.2$, $\lambda = 0.9$, $\theta_1 = 0.0045$, and $\theta_2 = 0.0005$. α is initialized with 0.1, and after 800,000 steps, α is halved with every 100,000 step. We measured the error of the prediction for the selected action, compared to the oracle (observation probability calculated from the structure of the environment), and the mean squared errors for every 10,000 steps are plotted.

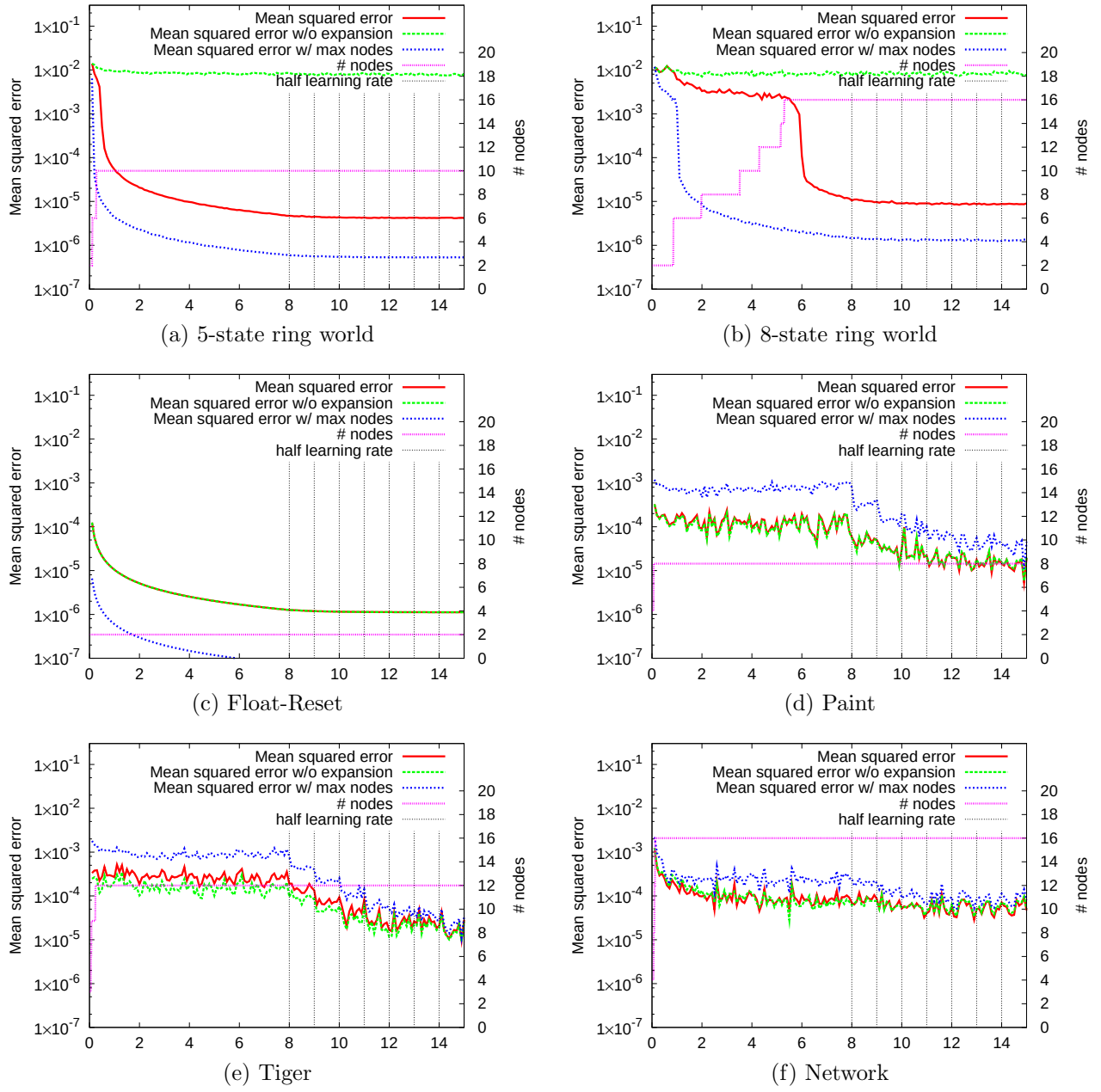
Figures 3(a) and 3(b) are the results in 5- and 8-state ring worlds. We can see that the number of nodes in the TD network increases as a result of node expansion until the prediction error decreases. This indicates that nodes in the TD-networks are expanded only when it is required.

We made additional tests with various parameters θ_1 and θ_2 on the 8-state ring world (Table 1). We found that θ_2 affects the final number of nodes. With larger θ_2, algorithm failed to make a required node expansion; with smaller θ_2, the algorithm made some spurious node expansions (though the learning was successful). On the other hand, θ_1 mainly affects the learning time, but less related to the number of nodes.

Figures 3(c) to 3(f) show the results of our algorithm in other well-known POMDP environments. Our algorithm has successfully learned predictions in all cases. However, we found that the initial form of the TD network without node expansion can learn equally well (compared to the case started with a large TD network, which is mechanically generated by expanding all nodes), due to the high generalization capacity of TD-network with sigmoid function. Thus these experiments are not helpful for evaluating our discovery algorithm. However, we see that some node expansions are occurred in these experiments. This indicates that our algorithm sometimes makes spurious node expansions, especially in these probabilistic environments.

We also evaluated the criteria we selected in terms of the discovered question network for the 8-state ring world. Figure 4 compares the mean square errors and the number of nodes in the discovered network with various settings of criteria (in these experiments α is kept constant to 0.1). Although the prediction error approaches to zero on all conditions, increase in the number of nodes is observed if any one of the criteria is removed.

Figure 5 further examines the difference of the discovered TD network. In the TD network discovered using all of the criteria (Figure 5(a)), all the expanded nodes



(a) 5-state ring world

(b) 8-state ring world

(c) Float-Reset

(d) Paint

(e) Tiger

(f) Network

Figure 3. Results of experiments. X-axis shows time steps ($\times 10^5$)

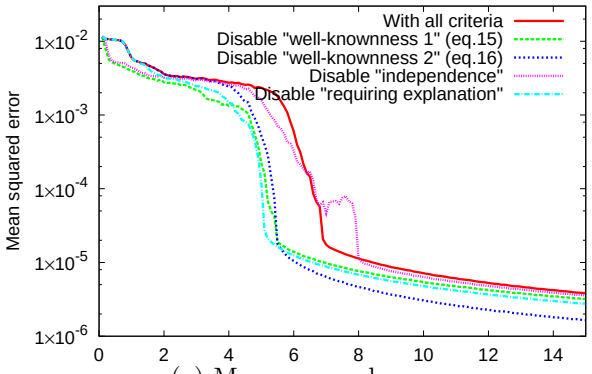
are independent of each other, thus the discovered network is the best network for the 8-state ring world that can be discovered by our algorithm. This shows clear contrast to TD network discovered solely by dependency detection network, shown in Figure 5(b). Although this network has correctly learned to predict the 8-state ring world, some spurious node expansions are observed. This indicates that the dependency detection network is not powerful enough to choose the right node to be expanded, and shows importance of other criteria in our algorithm.

5. Discussion

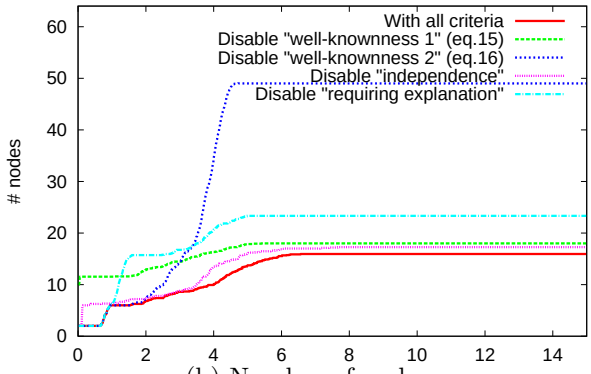
The algorithm we have presented has some limitations. It seems unavoidable because the algorithm has to make inference using an incomplete question network. In this section, we discuss some of the limitations that arose in our approach.

5.1. Deep Dependency

In our algorithm, Criterion 3 (requires more explanation) are devoted for distinguishing apparently unpredictable events, caused by an incomplete question net-

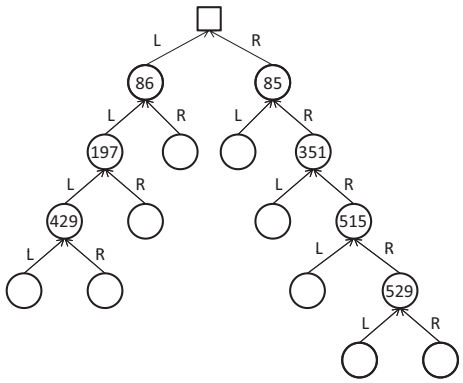


(a) Mean squared error

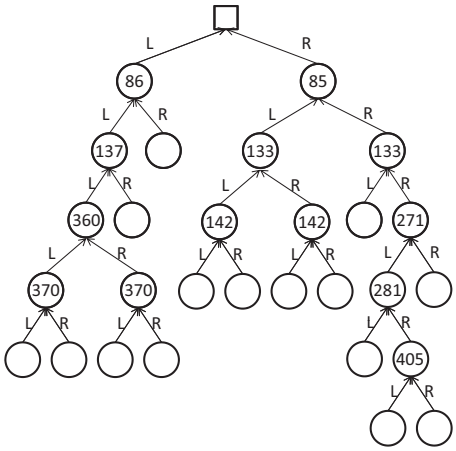


(b) Number of nodes

Figure 4. Results of experiments (average of 30 trials) on 8-state ring world with disabled criteria.



(a) With all criteria



(b) Without criterion "requires further explanation"

Figure 5. An example of discovered TD network for 8-state ring world. The number in a node denotes the time the node is expanded ($\times 10^3$).

work, from inherently random events. However, we can imagine cases in which this criterion does not work well, especially when the observation depends only on the actions several steps before.

As an example, suppose an n-step delay world: $\mathcal{A} = \mathcal{O} = \{0, 1\}$, and the agent observes its own action with n-step delay. If the agent chooses actions randomly, then the observation is also random string. Moreover, unless the depth of the question network reaches n, the agent can predict nothing. For $n \geq 2$, the nodes in a question network are likely to fail satisfying Criterion 3, and as a result, the algorithm fails to achieve the correct test set for the environment.

A partial solution may be to introduce active learning (planned exploration). In the example above, if the agent could adopt some biased choice of actions for a while, one would observe informative result that might satisfy Criterion 3.

5.2. Selection of Expanding Node

The algorithm always expands the node that requires more explanation, but it is possible that the node is not the best one to be expanded. The algorithm depends on an assumption that, if there is a better node to be expanded, the node also satisfies the expansion criteria sooner or later. We need to work for some theoretical support for the assumption. However, if the assumption is correct, the algorithm may perform some spurious expansion due to its distributed-processing nature. A complete solution would require either centralized processing or node shrinking.

5.3. Parameter Selection

The algorithm depends on a number of parameters. Our selection of parameters seems working for the tested problems, but not guaranteed for others. In particular, the algorithm is sensitive to θ_2, a threshold value used in Criterion 3 for distinguishing apparently unpredictable events from inherently random events. Due to the limitation of our approach, it seems impossible to provide a universal parameter set for producing the minimum network for any environment; a better solution would be to use lower thresholds to overgen-

erate the question network and shrink it afterwards.

5.4. Handling Wider Observation

In this paper, we considered only the simple case with one observation bit ($l = 1$). When $l \geq 2$, it would be necessary to consider sets of nodes that share a common action conditions and associated to different observations, and to expand all nodes in a set simultaneously. In addition, it is necessary to extend our criteria have to handle the node sets.

6. Related Work

McCracken and Bowling (2006) proposes the on-line discovery and learning algorithm for predictive state representation. However, their algorithm has to keep substantially long (in their example, 1000 steps) history in memory and calculate linear independency, and requires quite complex efforts to normalize the probability in their representation. On the other hand, our algorithm requires only a small amount of memory (up to the size required for eligibility traces) and, thanks to the sigmoidal function used in the TD network, no effort is required to normalize the result. We argue that these differences cause larger difference in calculation costs in a complex environment.

7. Summary

We presented an algorithm for on-line, incremental discovery of temporal-difference (TD) networks. The key contribution is the establishment of criteria to expand a leaf node in TD network: the algorithm expands a node when the node is well-known, independent, and has a prediction error that requires further explanation. Since none of these criteria requires centralized calculation operations, they can be computed in a parallel and distributed manner. Through computer experiments on n-state ring worlds, we demonstrated the empirical effectiveness of our algorithm.

Among the future work the most important is to evaluate our algorithm on various environments for comparison with other discovery algorithms. Agent planning with TD network should be also studied to combine with our algorithm for developing an agent that explores environments (Bowling et al., 2006).

Acknowledgments

I'd like to thank Prof. Steven Kraines for his kind help. This research is partially supported by the MEXT Grant-in-Aid for Young Scientists (B) (20700126).

References

Bowling, M., McCracken, P., James, M., Neufeld, J., & Wilkinson, D. (2006). Learning predictive state representations using non-blind policies. *In Proc. of ICML '06* (pp. 129–136). ACM Press.

Cassandra, A. (1999). Tony's POMDP file repository page. URL http://www.cs.brown.edu/research/ai/pomdp/examples/index.html.

Jaeger, H. (2000). Observable operator models for discrete stochastic time series. *Neural Computation*, *12*, 1371–1398.

James, M. R., & Singh, S. (2004). Learning and discovery of predictive state representations in dynamical systems with reset. *Proc. of ICML'04* (p. 53). ACM Press.

Littman, M. L., Sutton, R. S., & Singh, S. (2002). Predictive representations of state. In *Advances in neural information processing systems 14*, 1555–1561. MIT Press.

McCracken, P. (2005). An online algorithm for discovery and learning of predictive state representations. Master's thesis, University of Alberta.

McCracken, P., & Bowling, M. (2006). Online discovery and learning of predictive state representations. In *Advances in neural information processing systems 18*, 875–882. MIT Press.

Rosencrantz, M., Gordon, G., & Thrun, S. (2004). Learning low dimensional predictive representations. *Proc. of ICML '04* (p. 88). ACM Press.

Rudary, M. R., & Singh, S. (2004). A nonlinear predictive state representation. In *Advances in neural information processing systems 16*. MIT Press.

Sutton, R. S., & Tanner, B. (2005). Temporal-difference networks. In *Advances in neural information processing systems 17*, 1377–1384. MIT Press.

Tanner, B., & Sutton, R. S. (2005a). TD(λ) networks: temporal-difference networks with eligibility traces. *Proc. of ICML'05* (pp. 888–895). ACM Press.

Tanner, B., & Sutton, R. S. (2005b). Temporal-difference networks with history. In *Proc. of IJCAI'05*, 865–870.

Wolfe, B., James, M. R., & Singh, S. (2005). Learning predictive state representations in dynamical systems without reset. *Proc. of ICML'05* (pp. 980–987). ACM Press.

Nonextensive Entropic Kernels

André F. T. Martins[†‡] AFM@CS.CMU.EDU
Mário A. T. Figueiredo[‡] MARIO.FIGUEIREDO@LX.IT.PT
Pedro M. Q. Aguiar[♯] AGUIAR@ISR.IST.UTL.PT
Noah A. Smith[†] NASMITH@CS.CMU.EDU
Eric P. Xing[†] EPXING@CS.CMU.EDU

[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
[‡]Instituto de Telecomunicações / [♯]Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal

Abstract

Positive definite kernels on probability measures have been recently applied in structured data classification problems. Some of these kernels are related to classic information theoretic quantities, such as mutual information and the Jensen-Shannon divergence. Meanwhile, driven by recent advances in Tsallis statistics, nonextensive generalizations of Shannon's information theory have been proposed. This paper bridges these two trends. We introduce the *Jensen-Tsallis q-difference*, a generalization of the Jensen-Shannon divergence. We then define a new family of nonextensive mutual information kernels, which allow weights to be assigned to their arguments, and which includes the Boolean, Jensen-Shannon, and linear kernels as particular cases. We illustrate the performance of these kernels on text categorization tasks.

1. Introduction

There has been recent interest in kernels on probability distributions, to tackle several classification problems (Moreno *et al.*, 2003; Jebara *et al.*, 2004; Hein & Bousquet, 2005; Lafferty & Lebanon, 2005; Cuturi *et al.*, 2005). By mapping data points to fitted distributions in a parametric family where a kernel is defined, a kernel is automatically induced on the original input space. In text categorization, this appears as an alternative to the Euclidean geometry inherent to the usual bag-of-words vector representations. In fact, approaches that map data to a statistical manifold, where well-motivated non-Euclidean metrics may be defined (Lafferty & Lebanon, 2005), outperform SVM classifiers with linear kernels (Joachims, 1997). Some of these kernels have a natural information theoretic interpretation, creating a bridge between kernel methods and information theory (Cuturi *et al.*, 2005; Hein & Bousquet, 2005).

We reinforce that bridge by introducing a new class of kernels rooted in *nonextensive* (NE) information theory. The Shannon and Rényi entropies (Rényi, 1961) share the *extensivity* property: the joint entropy of a pair of independent random variables equals the sum of the individual entropies. Abandoning this property yields the so-called NE entropies (Havrda & Charvát, 1967; Tsallis, 1988), which have raised great interest among physicists in modeling certain phenomena (*e.g.*, long-range interactions and multifractals) and as generalizations of Boltzmann-Gibbs statistical mechanics (Abe, 2006). NE entropies have also been recently used in signal/image processing (Li *et al.*, 2006) and other areas (Gell-Mann & Tsallis, 2004).

The main contributions of this paper are:

- Based on the new concept of q-convexity and a related q-Jensen inequality, we introduce the *Jensen-Tsallis q-difference*, a NE generalization of the Jensen-Shannon (JS) divergence.
- We propose a broad family of positive definite (pd) kernels, which are interpretable as NE mutual information (MI) kernels. This family ranges from the Boolean to the linear kernels, and also includes the JS kernel (Hein & Bousquet, 2005).
- We extend results of Hein and Bousquet (2005) by proving positive definiteness of kernels based on the unbalanced JS divergence. As a side note, we

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

show that the parametrix approximation of the multinomial diffusion kernel introduced by Lafferty and Lebanon (2005) is *not* pd in general.

Our main purpose is to present new theoretical insights about kernels on measures by unifying some well-known instances into a common parametrized family. This family allows reinterpreting these kernels in light of NE information theory, a connection that to our knowledge had not been presented before. The fact that some members of this family are novel pd kernels leads us to include a set of text categorization experiments that illustrates their effectiveness.

The paper is organized as follows. Sec. 2 reviews NE entropies, while Jensen differences and divergences are discussed in Sec. 3. In Sec. 4, the concepts of q-differences and q-convexity are introduced and used to define the Jensen-Tsallis q-difference. Sec. 5 presents the new family of entropic kernels. Sec. 6 reports experiments on text categorization and Sec. 7 presents concluding remarks and future research directions.

Although, for simplicity, we focus on discrete distributions on finite sets, most results are valid in arbitrary measured spaces, as shown by Martins *et al.* (2008).

2. Nonextensive Information Theory

Let X denote a random variable (rv) taking values in a finite set $\mathcal{X} = \{x_1, \ldots, x_n\}$ according to a probability distribution P_X. An entropy function is said to be *extensive* if it is additive over independent variables. For example, the Shannon entropy (Cover & Thomas, 1991), $H(X) \triangleq -\mathbb{E}[\ln P_X]$, is extensive: if X and Y are independent, then $H(X,Y) = H(X)+H(Y)$. Another example is the family of Rényi entropies (Rényi, 1961), parameterized by $q \geq 0$,

$$R_q(X) \triangleq \frac{1}{1-q} \ln \sum_{i=1}^{n} P_X(x_i)^q, \tag{1}$$

which includes Shannon's entropy as a special case when $q \to 1$.

In classic information theory, extensivity is considered desirable, and is enforced axiomatically (Khinchin, 1957), to express the idea borrowed from thermodynamics that "independent systems add their entropies." In contrast, the *Tsallis entropies* abandon the extensivity requirement (Tsallis, 1988). These NE entropies, denoted $S_q(X)$, are defined as follows:

$$S_q(X) \triangleq -\mathbb{E}_q(\ln_q P_X) = \frac{1}{q-1}\left(1 - \sum_{i=1}^{n} P_X(x_i)^q\right), \tag{2}$$

where $\mathbb{E}_q(f) \triangleq \sum_{i=1}^{n} P(x_i)^q f(x_i)$ is the unnormalized *q-expectation*, and $\ln_q(y) \triangleq (y^{1-q}-1)/(1-q)$ is the so-called *q-logarithm*. It is noteworthy that when $q \to 1$, we get $\mathbb{E}_q \to \mathbb{E}$, $\ln_q \to \ln$, and $S_q \to H$; *i.e.*, the family of Tsallis entropies also includes Shannon's entropy. For the Tsallis family, when X and Y are independent, extensivity no longer holds; instead, we have

$$S_q(X,Y) = S_q(X) + S_q(Y) - (q-1)S_q(X)S_q(Y), \tag{3}$$

where the parameter $q \geq 0$ is called *entropic index*.

While statistical physics has been the main application of Tsallis entropies, some attempts have been made to produce NE generalizations of classic information theory results (Furuichi, 2006). As for the Shannon entropy, the Tsallis *joint* and *conditional* entropies are defined as $S_q(X,Y) \triangleq -\mathbb{E}_q[\ln_q P_{XY}]$ and $S_q(X|Y) \triangleq -\mathbb{E}_q[\ln_q P_{X|Y}]$, respectively, and follow a chain rule $S_q(X,Y) = S_q(X) + S_q(Y|X)$. Similarly, Furuichi (2006) defines the Tsallis MI as

$$I_q(X;Y) \triangleq S_q(X) - S_q(X|Y) = I_q(Y;X), \tag{4}$$

generalizing (for $q > 1$) Shannon's MI. This NE version of the MI underlies one of the central contributions of this paper: the Jensen-Tsallis q-difference (Sec. 4).

For reasons that will become clear in Sec. 5, it is convenient to extend the domain of Tsallis entropies to *unnormalized* measures, *i.e.*, in $\mathbb{R}^n_+ \triangleq \{\boldsymbol{\mu} \in \mathbb{R}^n \mid \forall i\ \mu_i \geq 0\}$, but not necessarily in the probability simplex $\mathbb{P}^{n-1} \triangleq \{\mathbf{p} \in \mathbb{R}^n \mid \sum_{i=1}^{n} p_i = 1,\ \forall i\ p_i \geq 0\}$. The Tsallis entropy of a measure $\boldsymbol{\mu}$ in $\mathbb{R}^n_+$ is[1]

$$S_q(\boldsymbol{\mu}) \triangleq -\sum_{i=1}^{n} \mu_i^q \ln_q \mu_i = \sum_{i=1}^{n} \varphi_q(\mu_i), \tag{5}$$

where $\varphi_q : \mathbb{R}_+ \to \mathbb{R}$ is given by

$$\varphi_q(y) = -y^q \ln_q y = \begin{cases} -y \ln y, & \text{if } q = 1, \\ (y - y^q)/(q-1), & \text{if } q \neq 1. \end{cases} \tag{6}$$

This extension does not add expressive power, since function (5) is completely determined by its values on $\mathbb{P}^{n-1}$, as shown by the following proposition (the proof is straightforward).

Proposition 1 *The following denormalization formula holds for any $c \geq 0$ and $\boldsymbol{\mu} \in \mathbb{R}^n_+$:*

$$S_q(c\boldsymbol{\mu}) = c^q S_q(\boldsymbol{\mu}) + \varphi_q(c)\|\boldsymbol{\mu}\|_1, \tag{7}$$

where $\|\boldsymbol{\mu}\|_1 \triangleq \sum_{i=1}^{n} \mu_i$ is the ℓ_1-norm of $\boldsymbol{\mu}$.

[1] In the following, we represent normalized and unnormalized measures as vectors in $\mathbb{R}^n$, and we use those as arguments of entropy functions, *e.g.*, we write $H(\boldsymbol{\pi})$ to denote $H(X)$ where $X \sim P(X)$, with $\pi_i = P(x_i)$.

This fact will be used in a constructive way in Sec. 5 to devise a family of pd NE entropic kernels.

3. Jensen Differences and Divergences

Jensen's inequality is at the heart of many important results in information theory. Let the rv Z take values on a finite set $\mathcal{Z}$. Jensen's inequality states that if f is a convex function defined on the convex hull of $\mathcal{Z}$, then $f(\mathbb{E}[Z]) \leq \mathbb{E}[f(Z)]$. The nonnegative quantity $\mathbb{E}[f(Z)] - f(\mathbb{E}[Z])$ is known as *Jensen difference* and has been studied by Burbea and Rao (1982) when $-f$ is some form of generalized entropy. Here, we are interested in the case where $Z \in \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m\}$ is a *random measure*, where each $\boldsymbol{\mu}_j \in \mathbb{R}^n_+$, with probabilities $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m) \in \mathbb{P}^{m-1}$. The Jensen difference induced by a (concave) generalized entropy Ψ is

$$\begin{aligned} J^{\boldsymbol{\pi}}_{\Psi}(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m) &\triangleq \Psi\left(\sum_{j=1}^m \pi_j \boldsymbol{\mu}_j\right) - \sum_{j=1}^m \pi_j \Psi(\boldsymbol{\mu}_j) \\ &= \Psi\left(\mathbb{E}[Z]\right) - \mathbb{E}[\Psi(Z)], \end{aligned} \tag{8}$$

Below, we show examples of Jensen differences that have been applied in machine learning. In Sec. 4, we provide a NE generalization of the Jensen difference.

Jensen-Shannon (JS) Divergence Consider a classification problem with m classes, $Y \in \mathcal{Y} = \{1, \ldots, m\}$, with *a priori* probabilities $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m) \in \mathbb{P}^{m-1}$. Let $\mathbf{p}_j = (p_{j1}, \ldots, p_{jn}) \in \mathbb{P}^n$ for $j = 1, \ldots, m$, where $p_{ji} \triangleq P(X = x_i | Y = j)$, be the corresponding class-conditional distributions.

Letting Ψ in (8) be H, the Shannon entropy, the resulting Jensen difference $J^{\boldsymbol{\pi}}_H(\mathbf{p}_1, \ldots, \mathbf{p}_m)$ is known as the JS divergence of $\mathbf{p}_1, \ldots, \mathbf{p}_m$, with weights $\pi_1, \ldots, \pi_m$ (Burbea & Rao, 1982; Lin, 1991). In this instance of the Jensen difference,

$$J^{\boldsymbol{\pi}}_H(\mathbf{p}_1, \ldots, \mathbf{p}_m) = I(X; Y), \tag{9}$$

where $I(X;Y) = H(X) - H(X|Y)$ is the MI between X and Y (Banerjee *et al.*, 2005).

For $m = 2$ and $\boldsymbol{\pi} = (\frac{1}{2}, \frac{1}{2})$, we denote the ensuing $J_H^{(\frac{1}{2}, \frac{1}{2})}(\mathbf{p}_1, \mathbf{p}_2)$ as $JS(\mathbf{p}_1, \mathbf{p}_2)$:

$$JS(\mathbf{p}_1, \mathbf{p}_2) = H((\mathbf{p}_1 + \mathbf{p}_2)/2) - (H(\mathbf{p}_1) + H(\mathbf{p}_2))/2.$$

It can be shown that that $\sqrt{JS}$ satisfies the triangle inequality and is a *Hilbertian metric*[2] (Endres & Schindelin, 2003; Topsøe, 2000), which has motivated its use in kernel-based machine learning.

[2] A metric $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is Hilbertian if there is some Hilbert space $\mathcal{H}$ and an isometry $f : \mathcal{X} \to \mathcal{H}$ such that $d^2(x, y) = \langle f(x) - f(y), f(x) - f(y) \rangle_{\mathcal{H}}$ holds for any $x, y \in \mathcal{X}$ (Hein & Bousquet, 2005).

Jensen-Rényi (JR) Divergence Let $\Psi = R_q$, which is concave for $q \in [0, 1)$; then, (8) becomes

$$J^{\boldsymbol{\pi}}_{R_q}(\mathbf{p}_1, \ldots, \mathbf{p}_m) = R_q\left(\mathbb{E}[\mathbf{p}]\right) - \mathbb{E}[R_q(\mathbf{p})]. \tag{10}$$

We call $J^{\boldsymbol{\pi}}_{R_q}$ the JR divergence. When $m = 2$ and $\boldsymbol{\pi} = (1/2, 1/2)$, we write $J^{\boldsymbol{\pi}}_{R_q}(\mathbf{p}) = JR_q(\mathbf{p}_1, \mathbf{p}_2)$, where

$$JR_q(\mathbf{p}_1, \mathbf{p}_2) = R_q\left(\frac{\mathbf{p}_1 + \mathbf{p}_2}{2}\right) - \frac{R_q(\mathbf{p}_1) + R_q(\mathbf{p}_2)}{2}.$$

The JR divergence has been used in signal processing applications (Karakos *et al.*, 2007). We show in Sect. 5.3 that $\sqrt{JR_q}$ is also an Hilbertian metric.

Jensen-Tsallis (JT) Divergence Divergences of the form (8), with $\Psi = S_q$, are known as JT divergences (Burbea & Rao, 1982) and were recently used in image processing (Hamza, 2006). Unlike the JS divergence, the JT divergence lacks a MI interpretation; in Sec. 4, we introduce an alternative to the JT divergence, which is interpretable as a NE MI in the sense of Furuichi (2006).

4. Jensen q-Differences

We now introduce *Jensen q-differences*, a generalization of Jensen differences. As described shortly, a special case of the Jensen q-difference is the *Jensen-Tsallis q-difference*, which is an NE generalization of the JS divergence, and provides the building block for the NE entropic kernels to be introduced in Sec. 5. We begin by introducing the concept of "q-convexity", which satisfies a Jensen-type inequality.

Definition 1 *Let $q \in \mathbb{R}$ and $\mathcal{X}$ a convex set. A function $f : \mathcal{X} \to \mathbb{R}$ is* q-convex *if, for any $x, y \in \mathcal{X}$ and $\lambda \in [0, 1]$,*

$$f(\lambda x + (1 - \lambda) y) \leq \lambda^q f(x) + (1 - \lambda)^q f(y). \tag{11}$$

f is q-concave *if $-f$ is q-convex.*

Naturally, 1-convexity is the usual convexity. The next proposition states the q-Jensen inequality and is easily proved by induction, like the standard Jensen inequality (Cover & Thomas, 1991). It also states that the property of q-convexity gets stronger as q increases.

Proposition 2 *If $f : \mathcal{X} \to \mathbb{R}$ is q-convex and $f \geq 0$, then, for any $n \in \mathbb{N}$, $x_1, \ldots, x_n \in \mathcal{X}$ and $\boldsymbol{\pi} \in \mathbb{P}^{n-1}$:*

$$f\left(\sum_{i=1}^n \pi_i x_i\right) \leq \sum_{i=1}^n \pi_i^q f(x_i). \tag{12}$$

Moreover, if $q \geq r \geq 0$, we have:

$$f \text{ is } q\text{-convex} \quad \Rightarrow \quad f \text{ is } r\text{-convex} \tag{13}$$

$$f \text{ is } r\text{-concave} \quad \Rightarrow \quad f \text{ is } q\text{-concave.} \tag{14}$$

Based on the q-Jensen inequality, we can now consider *Jensen q-differences* of the form $\mathbb{E}_q[f(Z)] - f(\mathbb{E}[Z])$, which are nonnegative if f is q-convex. As in Sec. 3, we focus on the scenario where Z is a random measure and $-f = \Psi$ is an entropy function, yielding

$$\begin{aligned} T^{\boldsymbol{\pi}}_{q,\Psi}(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m) &\triangleq \Psi\left(\sum_{t=1}^m \pi_t\, \boldsymbol{\mu}_t\right) - \sum_{t=1}^m \pi_t^q \Psi(\boldsymbol{\mu}_t) \\ &= \Psi\left(\mathbb{E}[Z]\right) - \mathbb{E}_q[\Psi(Z)]. \end{aligned} \quad (15)$$

The Jensen q-difference is a deformation of the Jensen 1-difference (8), in which the second expectation is replaced by a q-expectation. We are now ready to introduce the class of Jensen-Tsallis q-differences.

Jensen-Tsallis q-Differences Consider again the classification problem used in the description of the JS divergence, but replacing the Jensen difference with the Jensen q-difference and the Shannon entropy with the Tsallis q-entropy, *i.e.*, letting $\Psi = S_q$ in (15). We obtain (writing $T^{\boldsymbol{\pi}}_{q,S_q}$ simply as $T^{\boldsymbol{\pi}}_q$):

$$T^{\boldsymbol{\pi}}_q(\mathbf{p}_1, \ldots, \mathbf{p}_m) = S_q(X) - S_q(X|Y) = I_q(X;Y), \quad (16)$$

where $S_q(X|Y)$ is the Tsallis conditional q-entropy, and $I_q(X;Y)$ is the Tsallis MI (cf. (4)). Note that (16) is an NE analogue of (9), *i.e.* the Jensen-Tsallis q-differences are NE mutual informations.

We call $T^{\boldsymbol{\pi}}_q(\mathbf{p}_1, \ldots, \mathbf{p}_m)$ the Jensen-Tsallis q-difference of $\mathbf{p}_1, \ldots, \mathbf{p}_m$ with weights $\pi_1, \ldots, \pi_m$.

When $m = 2$ and $\boldsymbol{\pi} = (1/2, 1/2)$, define $T_q \triangleq T^{(1/2,1/2)}_q$,

$$T_q(\mathbf{p}_1, \mathbf{p}_2) = S_q\left(\frac{\mathbf{p}_1 + \mathbf{p}_2}{2}\right) - \frac{S_q(\mathbf{p}_1) + S_q(\mathbf{p}_2)}{2^q}. \quad (17)$$

Three special cases are obtained for $q \in \{0, 1, 2\}$:

$$\begin{aligned} S_0(\mathbf{p}) &= -1 + \|\mathbf{p}\|_0; & T_0(\mathbf{p}_1, \mathbf{p}_2) &= 1 - \|\mathbf{p}_1 \odot \mathbf{p}_2\|_0 \\ S_1(\mathbf{p}) &= H(\mathbf{p}); & T_1(\mathbf{p}_1, \mathbf{p}_2) &= JS(\mathbf{p}_1, \mathbf{p}_2) \\ S_2(\mathbf{p}) &= 1 - \langle \mathbf{p}, \mathbf{p} \rangle; & T_2(\mathbf{p}_1, \mathbf{p}_2) &= \frac{1}{2} - \frac{1}{2}\, \langle \mathbf{p}_1, \mathbf{p}_2 \rangle \end{aligned}$$

where $\|\mathbf{x}\|_0$ is the number of nonzeros in $\mathbf{x}$, $\odot$ denotes the Hadamard-Schur (elementwise) product, and $\langle \cdot, \cdot \rangle$ is the inner product.

The JT q-difference is an NE generalization of the JS divergence, and some of the latter's properties are lost in general. Since Tsallis entropies are 1-concave, Prop. 2 guarantees q-concaveness only for $q \geq 1$. Therefore, nonnegativity is only guaranteed for JT q-differences when $q \geq 1$; for this reason some authors only consider this range of values (Furuichi, 2006). Moreover, unless $q = 1$ (the JS divergence), it is not generally true that $T^{\boldsymbol{\pi}}_q(\mathbf{p}, \ldots, \mathbf{p}) = 0$ or even that $T^{\boldsymbol{\pi}}_q(\mathbf{p}, \ldots, \mathbf{p}, \mathbf{p}') \geq T^{\boldsymbol{\pi}}_q(\mathbf{p}, \ldots, \mathbf{p}, \mathbf{p})$. For example,

$$\operatorname{argmin}_{\mathbf{p}_1 \in \mathbb{P}^{n-1}} T_q(\mathbf{p}_1, \mathbf{p}_2) \quad (18)$$

can be different from $\mathbf{p}_2$, unless $q = 1$. In general, the minimizer is closer to either the uniform distribution (if $q \in [0, 1)$) or a degenerate distribution[3] (for $q \in (1, 2]$). For these reasons, the term "divergence" is misleading and we use the term "difference." Other properties of JT q-differences (convexity, lower/upper bounds) are studied by Martins *et al.* (2008).

5. Nonextensive Entropic Kernels

Using the denormalization formula (7), we now introduce kernels based on the JS divergence and the JT q-difference, which allow weighting their arguments. In this section, $m = 2$ (kernels involve pairs of measures).

5.1. Background on Kernels

We begin with some basic results on kernels (Schölkopf & Smola, 2002). Below, $\mathcal{X}$ denotes a nonempty set; $\mathbb{R}_+$ denote the nonnegative reals, and $\mathbb{R}_{++} \triangleq \mathbb{R}_+ \setminus \{0\}$.

Definition 2 *Let $\varphi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function, i.e., $\varphi(y, x) = \varphi(x, y)$, for all $x, y \in \mathcal{X}$. φ is called a pd kernel if and only if*

$$\sum_{i=1}^n \sum_{j=1}^n c_i\, c_j\, \varphi(x_i, x_j) \geq 0, \quad (19)$$

for any integer n, $x_i, \ldots, x_n \in \mathcal{X}$ and $c_i, \ldots, c_n \in \mathbb{R}$. A symmetric function $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a negative definite *(nd) kernel if and only if*

$$\sum_{i=1}^n \sum_{j=1}^n c_i\, c_j\, \psi(x_i, x_j) \leq 0, \quad (20)$$

for any integer n, $x_i, \ldots, x_n \in \mathcal{X}$ and $c_i, \ldots, c_n \in \mathbb{R}$, satisfying the additional constraint $\sum_i c_i = 0$. In this case, $-\psi$ is called conditionally *pd; obviously, positive definiteness implies conditional positive definiteness.*

Both the sets of pd and nd kernels are closed under pointwise sums/integrations, the former being also closed under pointwise products; moreover, both sets are closed under pointwise convergence. While pd kernels correspond to inner products via embedding in a Hilbert space, nd kernels that vanish on the diagonal and are positive anywhere else, correspond to squared Hilbertian distances. These facts, and the following ones, are shown by Berg *et al.* (1984).

[3] Notice that $T_2(\mathbf{p}_1, \mathbf{p}_2) = \frac{1}{2} - \frac{1}{2}\langle \mathbf{p}_1, \mathbf{p}_2 \rangle$; in this case, (18) becomes a linear program, and the solution is $\mathbf{p}_1^* = \mathbf{e}_j$, where $j = \operatorname{argmax}_i p_{2i}$.

Proposition 3 *Let $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a symmetric function, and $x_0 \in \mathcal{X}$. Let $\varphi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be*

$$\varphi(x, y) = \psi(x, x_0) + \psi(y, x_0) - \psi(x, y) - \psi(x_0, x_0).$$

Then, φ is pd if and only if ψ is nd.

Proposition 4 *The function $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a nd kernel if and only if $\exp(-t\psi)$ is pd for all $t > 0$.*

Proposition 5 *The function $\psi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is a nd kernel if and only if $(t + \psi)^{-1}$ is pd for all $t > 0$.*

Proposition 6 *If ψ is nd and $\psi(x, x) \geq 0$, for all $x \in \mathcal{X}$, then so are ψ^α, for $\alpha \in [0,\, 1]$, and $\ln(1 + \psi)$.*

Proposition 7 *If $f : \mathcal{X} \to \mathbb{R}$ satisfies $f \geq 0$, then, for $\alpha \in [1, 2]$, the function $-(f(x) + f(y))^\alpha$ is a nd kernel.*

5.2. Jensen-Shannon and Tsallis Kernels

The basic result that allows deriving pd kernels based on the JS divergence and, more generally, on the JT q-difference, is the fact that the denormalized Tsallis q-entropies are nd functions[4] on $\mathbb{R}^n_+$, for $q \in [0, 2]$. Of course, this includes the denormalized Shannon entropy as a particular case, corresponding to $q = 1$. Partial proofs are given by Berg *et al.* (1984), Topsøe (2000), and Cuturi *et al.* (2005); we present here a complete proof.

Proposition 8 *For $q \in [0, 2]$, the denormalized Tsallis q-entropy S_q is an nd function on $\mathbb{R}^n_+$.*

Proof: Since nd kernels are closed under pointwise summation, it suffices to prove that φ_q (see (6)) is nd on $\mathbb{R}_+$. For $q \neq 1$, $\varphi_q(y) = (q-1)^{-1}(y - y^q)$. If $q \in [0, 1)$, φ_q equals $-\iota + \iota^q$ times a positive constant, where ι is the identity ($\iota(y) = y$) on $\mathbb{R}_+$. Since the set of nd functions is closed under sums, we only need to show that both $-\iota$ and ι^q are nd, which is easily seen from the definition; besides, since ι is nd and nonnegative, Prop. 6 implies that ι^q is also nd. For $q \in (1, 2]$, φ_q equals $\iota - \iota^q$ times a positive constant. It remains to show that $-\iota^q$ is nd for $q \in (1, 2]$; since $k(x, y) = -(x+y)^q$ is nd (Prop. 7), so is ι^q. For $q = 1$, since the set of nd functions is closed under limits,

$$\varphi_1(x) = \varphi_H(x) = -x \ln x = \lim_{q \to 1} -x^q \ln_q x = \lim_{q \to 1} \varphi_q(x),$$

it follows that φ_1 is nd. □

The following proposition, proved by Berg *et al.* (1984), will also be used below.

[4] A function $f : \mathcal{X} \to \mathbb{R}$ is called pd (resp. nd) if $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, defined as $k(x, y) = f(x + y)$, is a pd (resp. nd) kernel (Berg *et al.*, 1984).

Proposition 9 *The function $\zeta_q : \mathbb{R}_{++} \to \mathbb{R}$, defined as $\zeta_q(y) = y^{-q}$ is pd, for $q \in [0, 1]$.*

We now present the main contribution of this section, the family of *weighted JT kernels*, generalizing the JS divergence kernels in two ways: **(i)** they apply to unnormalized measures (equivalently, they allow weighting the arguments differently); **(ii)** they extend the MI nature of the JS divergence kernel to the NE case.

Definition 3 (weighted Jensen-Tsallis kernels) *The kernel $\widetilde{k}_q : (\mathbb{R}^n_+)^2 \to \mathbb{R}$ is defined as*

$$\begin{aligned} \widetilde{k}_q(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) &= \widetilde{k}_q(\omega_1 \mathbf{p}_1, \omega_2 \mathbf{p}_2) \\ &\triangleq \left[S_q(\boldsymbol{\pi}) - T_q^{\boldsymbol{\pi}}(\mathbf{p}_1, \mathbf{p}_2)\right](\omega_1 + \omega_2)^q, \end{aligned}$$

where $\mathbf{p}_1 = \boldsymbol{\mu}_1/\omega_1$ and $\mathbf{p}_2 = \boldsymbol{\mu}_2/\omega_2$ are the normalized counterparts of $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, with corresponding weights $\omega_1, \omega_2 \in \mathbb{R}_+$, and $\boldsymbol{\pi} = (\omega_1/(\omega_1 + \omega_2), \omega_2/(\omega_1 + \omega_2))$.

The kernel $k_q : (\mathbb{R}^n_{++})^2 \to \mathbb{R}$ is defined as

$$k_q(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = k_q(\omega_1 \mathbf{p}_1, \omega_2 \mathbf{p}_2) \triangleq S_q(\boldsymbol{\pi}) - T_q^{\boldsymbol{\pi}}(\mathbf{p}_1, \mathbf{p}_2).$$

Recalling (16), notice $S_q(Y) - I_q(X; Y) = S_q(Y|X)$ can be interpreted as the *Tsallis posterior conditional entropy*. Hence, k_q can be seen (in Bayesian classification terms) as a NE expected measure of uncertainty in correctly identifying the class given the prior $\boldsymbol{\pi} = (\pi_1, \pi_2)$ and a random sample from the mixture distribution $\pi_1 \mathbf{p}_1 + \pi_2 \mathbf{p}_2$. The more similar the two distributions are, the greater this uncertainty.

Proposition 10 *The kernel $\widetilde{k}_q$ is pd, for $q \in [0, 2]$. The kernel k_q is pd, for $q \in [0, 1]$.*

Proof: With $\boldsymbol{\mu}_1 = \omega_1 \mathbf{p}_1$ and $\boldsymbol{\mu}_2 = \omega_2 \mathbf{p}_2$ and using the denormalization formula (7), we obtain $\widetilde{k}_q(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = -S_q(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) + S_q(\boldsymbol{\mu}_1) + S_q(\boldsymbol{\mu}_2)$. Now invoke Prop. 3 with $\psi = S_q$ (which is nd by Prop. 8), $x = \boldsymbol{\mu}_1$, $y = \boldsymbol{\mu}_2$, and $x_0 = \mathbf{0}$ (the null measure). Observe now that $k_q(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = \widetilde{k}_q(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2)(\omega_1 + \omega_2)^{-q}$. Since the product of two pd kernels is a pd kernel and (Prop. 9) $(\omega_1 + \omega_2)^{-q}$ is a pd kernel, for $q \in [0, 1]$, k_q is pd. □

As we can see, the weighted JT kernels have two inherent properties: they are parameterized by the entropic index q and they allow their arguments to be unbalanced, *i.e.*, to have different weights ω_i. We now mention some instances of kernels where each of these degrees of freedom is suppressed.

Weighted JS Kernel Setting $q = 1$, we obtain an extensive subfamily that contains unbalanced versions of the JS kernel (Hein & Bousquet, 2005). Namely, we

get the pd kernels:

$$\begin{aligned}\widetilde{k}_1(\boldsymbol{\mu}_1,\boldsymbol{\mu}_2) &= [H(\boldsymbol{\pi})-J^{\boldsymbol{\pi}}(\mathbf{p}_1,\mathbf{p}_2)]\,(\omega_1+\omega_2),\\ k_1(\boldsymbol{\mu}_1,\boldsymbol{\mu}_2) &= H(\boldsymbol{\pi})-J^{\boldsymbol{\pi}}(\mathbf{p}_1,\mathbf{p}_2).\end{aligned} \quad (21)$$

Exponentiated Weighted JS Kernel Using Prop. 4, we have that exponentiated weighted JS kernel $k_{EWJS}:\mathbb{R}^n_+\to\mathbb{R}$,

$$\begin{aligned}k_{EWJS}(\boldsymbol{\mu}_1,\boldsymbol{\mu}_2) &\triangleq \exp[t\,k_1(\boldsymbol{\mu}_1,\boldsymbol{\mu}_2)]\\ &= \exp(t\,H(\boldsymbol{\pi}))\exp\left[-tJ^{\boldsymbol{\pi}}(\mathbf{p}_1,\mathbf{p}_2)\right]\end{aligned} \quad (22)$$

is also pd for any $t>0$. This generalizes the exponentiated JS kernel $k_{EJS}(\mathbf{p}_1,\mathbf{p}_2)\triangleq\exp\left[-t\,JS\left(\mathbf{p}_1,\mathbf{p}_2\right)\right]$ (Cuturi *et al.*, 2005).

We now keep $q\in[0,2]$ but consider the weighted JT kernel family restricted to normalized measures, $k_q|_{(\mathbb{P}^{n-1})^2}$. This corresponds to setting uniform weights ($\omega_1=\omega_2=1/2$); note that in this case $\widetilde{k}_q$ and k_q collapse into the same kernel,

$$\widetilde{k}_q(\mathbf{p}_1,\mathbf{p}_2)=k_q(\mathbf{p}_1,\mathbf{p}_2)=\ln_q(2)-T_q(\mathbf{p}_1,\mathbf{p}_2). \quad (23)$$

Prop. 10 tells us that these kernels are pd for $q\in[0,2]$. Remarkably, we recover three well-known particular cases for $q\in\{0,1,2\}$.

Jensen-Shannon kernel (JSK) For $q=1$, we obtain the JS kernel, $k_{JS}:(\mathbb{P}^{n-1})^2\to\mathbb{R}$,

$$k_{JS}(\mathbf{p}_1,\mathbf{p}_2)=\ln(2)-JS(\mathbf{p}_1,\mathbf{p}_2), \quad (24)$$

introduced and shown pd by Hein and Bousquet (2005).

Boolean kernel For $q=0$, we obtain the kernel $k_0=k_{Bool}:(\mathbb{P}^{n-1})^2\to\mathbb{R}$,

$$k_{Bool}(\mathbf{p}_1,\mathbf{p}_2)=\|\mathbf{p}_1\odot\mathbf{p}_2\|_0. \quad (25)$$

Linear kernel For $q=2$, we obtain the kernel $k_2=k_{lin}:(\mathbb{P}^{n-1})^2\to\mathbb{R}$,

$$k_{lin}(\mathbf{p}_1,\mathbf{p}_2)=\frac{1}{2}\,\langle\mathbf{p}_1,\mathbf{p}_2\rangle. \quad (26)$$

Summarizing, Boolean, JS, and linear kernels, are members of the much wider family of Tsallis kernels, continuously parameterized by $q\in[0,2]$. Furthermore, Tsallis kernels are a particular subfamily of the even wider set of weighted Tsallis kernels.

A key feature of our generalization is that the kernels are defined on unnormalized measures. This is relevant for empirical measures (*e.g.*, term counts, image histograms); instead of the usual normalization (Hein & Bousquet, 2005), these empirical measures may be left unnormalized, allowing objects of different sizes to have different weights. Another possibility is the explicit inclusion of weights (ω_i): given an input set of normalized measures, each can be multiplied by an arbitrary (positive) weight before computing the kernel.

5.3. Other Kernels Based on Jensen Differences

Other pd kernels may be devised inspired by Jensen-Rényi and Jensen-Tsallis divergences (Section 3). For example, it is a direct consequence of Prop. 6 that, for $q\in[0,1]$, $(\mathbf{p}_1,\mathbf{p}_2)\mapsto R_q\big(\frac{\mathbf{p}_1+\mathbf{p}_2}{2}\big)$, and therefore JR_q, are nd kernels on $(\mathbb{P}^{n-1})^2$. We can then make use of Prop. 4 to derive pd kernels via exponentiation; for example, the *exponentiated Jensen-Rényi kernel* (pd for $q\in[0,1]$ and $t\geq 0$):

$$\begin{aligned}k_{EJR}(\mathbf{p}_1,\mathbf{p}_2) &\triangleq \exp(-t\,J_{R_q}(\mathbf{p}_1,\mathbf{p}_2))\\ &= \left(\frac{\sum_i\left(\frac{p_{1i}+p_{2i}}{2}\right)^q}{\sqrt{\sum_i p_{1i}^q\sum_i p_{2i}^q}}\right)^{-\frac{t}{1-q}}.\end{aligned} \quad (27)$$

However, these kernels are no longer interpretable as MIs, and arbitrary weights are not allowed. Martins *et al.* (2008) also show that a related family of pd kernels for probability measures introduced by Hein and Bousquet (2005) can be written as differences between JT-type divergences.

5.4. The Heat Kernel Approximation

The diffusion kernel for statistical manifolds, recently proposed by Lafferty and Lebanon (2005), is grounded in information geometry. It models the diffusion of "information" over the manifold through the heat equation. Since in the case of the multinomial manifold the diffusion kernel has no closed form, the authors adopt the so-called "first-order parametrix expansion," which resembles the Gaussian kernel replacing the Euclidean distance by the geodesic distance induced by the Fisher information metric. The resulting heat kernel approximation is

$$k_{\,heat}(\mathbf{p}_1,\mathbf{p}_2)=(4\pi\tau)^{-\frac{n}{2}}\exp\left(-\frac{1}{4t}\,d_g^2(\mathbf{p}_1,\mathbf{p}_2)\right), \quad (28)$$

where $\tau>0$ and $d_g(\mathbf{p}_1,\mathbf{p}_2)=2\arccos\left(\sum_i\sqrt{p_{1i}p_{2i}}\right)$. Whether $k_{\,heat}$ is pd has been an open problem (Hein *et al.*, 2004; Zhang *et al.*, 2005).

Proposition 11 *Let $n\geq 2$. For sufficiently large τ, the kernel k_{heat} is* not *pd.*

Proof: From Prop. 4, k_{heat} is pd, for all $\tau > 0$, if and only if d_g^2 is nd. We provide a counterexample, using the following four points in $\mathbb{P}^2$: $\mathbf{p}_1 = (1, 0, 0)$, $\mathbf{p}_2 = (0, 1, 0)$, $\mathbf{p}_3 = (0, 0, 1)$ and $\mathbf{p}_4 = (1/2, 1/2, 0)$. The squared distance matrix $[D_{ij}] = [d_g^2(\mathbf{p}_i, \mathbf{p}_j)]$ is

$$D = \frac{\pi^2}{4} \cdot \begin{bmatrix} 0 & 4 & 4 & 1 \\ 4 & 0 & 4 & 1 \\ 4 & 4 & 0 & 4 \\ 1 & 1 & 4 & 0 \end{bmatrix}. \tag{29}$$

Taking $\mathbf{c} = (-4, -4, 1, 7)$ we have $\mathbf{c}^T D \mathbf{c} = 2\pi^2 > 0$, showing that D is not nd. Although $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ lie on the boundary of $\mathbb{P}^2$, continuity of d_g^2 implies that it is not nd. The case $n > 2$ follows easily, by appending zeros to the four vectors above. □

6. Experiments

We illustrate the performance of the proposed NE kernels, in comparison with common kernels, for SVM text classification. We performed experiments in two standard datasets: *Reuters-21578* and *WebKB*.[5] Since our objective was to evaluate the kernels, we considered a simple binary classification task that tries to discriminate among the two largest categories of each dataset; this led us to the *earn-vs-acq* classification task for the first dataset, and *stud-vs-fac* (student *vs.* faculty webpages) in the second dataset.

After the usual preprocessing steps of stemming and stop-word removal, we mapped text documents into probability distributions over words using the bag-of-words model and maximum likelihood estimation (which corresponds to normalizing term frequency using the ℓ_1-norm), which we denote by *tf*. We also used the *tf-idf* measure, which penalizes terms that occur in many documents. To weight the documents for the Tsallis kernels, we tried four strategies: uniform weighting, word counts, square root of the word counts, and one plus the logarithm of the word counts; however, for both tasks, uniform weighting revealed the best strategy, which may be due to the fact that documents in both collections are usually short and do not differ much in size.

As baselines, we used the linear kernel with ℓ_2 normalization, commonly used for this task, and the heat kernel approximation (28) (Lafferty & Lebanon, 2005), which is known to outperform the former, albeit not being guaranteed to be pd for an arbitrary choice of τ (see 28), as shown above. This parameter and the SVM C parameter were tuned with cross-validation over the training set. The SVM-Light package (`http://svmlight.joachims.org/`) was used to solve the SVM quadratic optimization problem.

Figs. 1–2 summarize the results. We report the performance of the Tsallis kernels as a function of the entropic index. For comparison, we also plot the performance of an instance of a Tsallis kernel with q tuned through cross-validation. For the first task, this kernel and the two baselines exhibit similar performance for both the *tf* and the *tf-idf* representations; differences are not statiscally significant. In the second task, the Tsallis kernel outperformed the ℓ_2-normalized linear kernel for both representations, and the heat kernel for *tf-idf*; the differences are statistically significant (using the unpaired t test at the 0.05 level). Regarding the influence of the entropic index, we observe that in both tasks, the optimum value of q is usually higher for *tf-idf* than for *tf*.

The results on these two problems are representative of the typical relative performance of the kernels considered: in almost all tested cases, both the heat kernel and the Tsallis kernels (for a suitable value of q) outperform the ℓ_2-normalized linear kernel; the Tsallis kernels are competitive with the heat kernel.

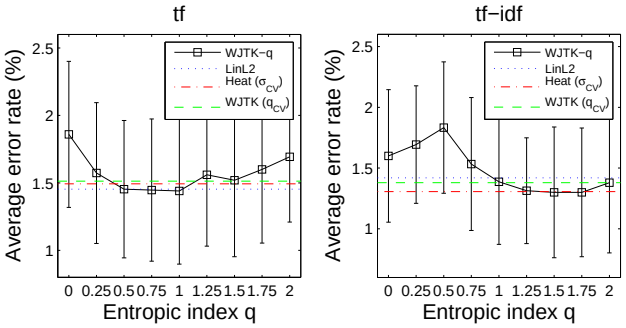


Figure 1. Results for *earn-vs-acq* using *tf* and *tf-idf* representations. The error bars represent ±1 standard deviation on 30 runs. Training (resp. testing) with 200 (resp. 250) samples per class.

7. Conclusion

We have introduced a new family of positive definite kernels between measures, which contains some well-known kernels as particular cases. These kernels are defined on unnormalized measures, which makes them suitable for use on empirical measures (*e.g.*, word counts or pixel intensity histograms), allowing objects of different sizes to be weighted differently. The family is parameterized by the entropic index, a key concept in Tsallis statistics, and includes as extreme cases the Boolean and the linear kernels. The new kernels, and

[5] Available at `http://www.daviddlewis.com/resources/testcollections` and `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data`, respectively.

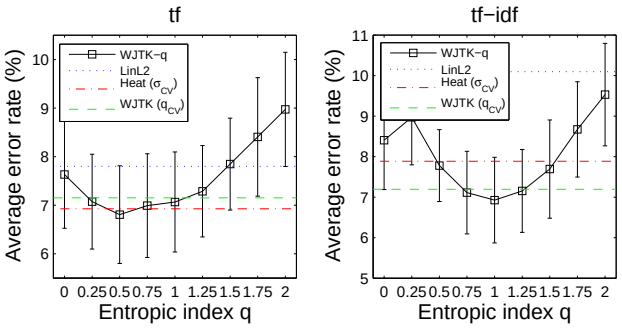


Figure 2. Results for *stud-vs-fac*.

the proofs of positive definiteness, are supported by the other contributions of this paper: the new concept of q-convexity, the underlying Jensen q-inequality, and the concept of *Jensen-Tsallis q-difference*, a nonextensive generalization of the Jensen-Shannon divergence. Experimentally, kernels in this family outperformed the linear kernel in the task of text classification and achieved similar results to the first-order approximation of the multinomial diffusion kernel. They have the advantage, however, of being pd, which fails to happen with the latter kernel, as also shown in this paper.

Future research will concern applying Jensen-Tsallis q-differences to other learning problems, like clustering, possibly exploiting the fact that they accept more than two arguments.

Acknowledgments

The authors thank the reviewers for helpful comments and Guy Lebanon for fruitful discussions on the heat kernel. This work was partially supported by *Fundação para a Ciência e Tecnologia* (FCT), Portugal, grant PTDC/EEA-TEL/72572/2006. A.M. was supported by a grant from FCT through the CMU-Portugal Program and the Information and Communications Technologies Institute (ICTI) at CMU. N.S. was supported by NSF IIS-0713265 and DARPA HR00110110013. E.X. was supported by NSF DBI-0546594, DBI-0640543, and IIS-0713379.

References

Abe, S. (2006). Foundations of nonextensive statistical mechanics. *Chaos, Nonlinearity, Complexity*. Springer.

Banerjee, A., Merugu., S., Dhillon, I. S. & Ghosh, J. (2005). Clustering with Bregman divergences. *JMLR*, *6*, 1705–1749.

Berg, C., Christensen, J., & Ressel, P. (1984). *Harmonic analysis on semigroups*. Springer.

Burbea, J., & Rao, C. (1982). On the convexity of some divergence measures based on entropy functions. *IEEE Trans. Inf. Theory*, *28*, 489–495.

Cover, T., & Thomas, J. (1991). *Elements of information theory*. Wiley.

Cuturi, M., Fukumizu, K., & Vert, J.-P. (2005). Semigroup kernels on measures. *JMLR*, *6*, 1169–1198.

Endres, D., & Schindelin, J. (2003). A new metric for probability distributions. *IEEE Trans. Inf. Theory*, *49*, 1858–1860.

Fuglede, B. (2005). Spirals in Hilbert space. *Expositiones Mathematicae*, *25*, 23–46.

Furuichi, S. (2006). Information theoretical properties of Tsallis entropies. *J. Math. Physics*, *47*, no. 2.

Gell-Mann, M., & Tsallis, C. (2004). *Nonextensive entropy: interdisciplinary applications*. Oxford University Press.

Hamza, A. (2006). A nonextensive information-theoretic measure for image edge detection. *Journal of Electronic Imaging*, *15-1*, 13011.1–13011.8.

Havrda, M., & Charvát, F. (1967). Quantification method of classification processes: concept of structural α-entropy. *Kybernetika*, *3*, 30–35.

Hein, M., & Bousquet, O. (2005). Hilbertian metrics and positive definite kernels on probability measures. *Proc. 10th AISTATS*.

Hein, M., Lal, T., & Bousquet, O. (2004). Hilbertian metrics on probability measures and their application in SVMs. *DAGM-Symposium*, 270-277.

Jebara, T., Kondor, R., & Howard, A. (2004). Probability product kernels. *JMLR*, *5*, 819–844.

Joachims, T. (1997). Text categorization with support vector machines: Learning with many relevant features (T.R.). Universität Dortmund.

Karakos, D., Khudanpur, S., Eisner, J., & Priebe, C. (2007). Iterative denoising using Jensen-Rényi divergences with an application to unsupervised document categorization. *Proc. IEEE ICASSP*.

Khinchin, A. (1957). *Mathematical foundations of information theory*. Dover.

Lafferty, J., & Lebanon, G. (2005). Diffusion kernels on statistical manifolds. *JMLR*, *6*, 129–163.

Li, Y., Fan, X., & Li, G. (2006). Image segmentation based on Tsallis-entropy and Rényi-entropy and their comparison. *IEEE Intern. Conf. on Industrial Informatics* (pp. 943–948).

Lin, J. (1991). Divergence measures based on Shannon entropy. *IEEE Trans. Inf. Theory*, *37*, 145–151.

Martins, A.F.T., Figueiredo, M.A.T., Aguiar, P.M.Q., Smith, N.A., Xing, E.P. (2008). Nonextensive entropic kernels. T.R. CMU-ML-08-106.

Moreno, P., Ho, P., & Vasconcelos, N. (2003). A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. *NIPS*.

Rényi, A. (1961). On measures of entropy and information. *Proc. 4th Berkeley Symp. Math. Statist. and Prob.* (pp. 547–561). Univ. Calif. Press.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. MIT Press.

Topsøe, F. (2000). Some inequalities for information divergence and related measures of discrimination. *IEEE Trans. Inf. Theory*, *46*, 1602–1609.

Tsallis, C. (1988). Possible generalization of Boltzmann-Gibbs statistics. *J. Stats. Physics*, *52*, 479–487.

Zhang, D., Chen, X., & Lee, W. (2005). Text classification with kernels on the multinomial manifold. *Proc. ACM SIGIR*.

Automatic Discovery and Transfer of MAXQ Hierarchies

Neville Mehta MEHTANE@EECS.OREGONSTATE.EDU
Soumya Ray SRAY@EECS.OREGONSTATE.EDU
Prasad Tadepalli TADEPALL@EECS.OREGONSTATE.EDU
Thomas Dietterich TGD@EECS.OREGONSTATE.EDU
Oregon State University, Corvallis OR 97331, USA

Abstract

We present an algorithm, HI-MAT (Hierarchy Induction via Models And Trajectories), that discovers MAXQ task hierarchies by applying dynamic Bayesian network models to a successful trajectory from a source reinforcement learning task. HI-MAT discovers subtasks by analyzing the causal and temporal relationships among the actions in the trajectory. Under appropriate assumptions, HI-MAT induces hierarchies that are consistent with the observed trajectory and have compact value-function tables employing safe state abstractions. We demonstrate empirically that HI-MAT constructs compact hierarchies that are comparable to manually-engineered hierarchies and facilitate significant speedup in learning when transferred to a target task.

1. Introduction

Scaling up reinforcement learning (RL) to large domains requires leveraging the structure in these domains. Hierarchical reinforcement learning (HRL) provides mechanisms through which domain structure can be exploited to constrain the value function and policy space of the learner, and hence speed up learning (Sutton et al., 1999; Dietterich, 2000; Andre & Russell, 2002). In the MAXQ framework, a task hierarchy is defined (along with relevant state variables) for representing the value function of the overall task. This allows for decomposed subtask-specific value functions that are easier to learn than the global value function.

Automated discovery of such task hierarchies is compelling for at least two reasons. First, it avoids the significant human effort in engineering the task-subtask structural decomposition, along with the associated state abstractions and subtask goals. Second, if the same hierarchy is useful in multiple domains, it leads to significant transfer of learned structural knowledge from one domain to the other. The cost of learning can be amortized over several domains. Several researchers have focused on the problem of automatically inducing temporally extended actions and task hierarchies (Thrun & Schwartz, 1995; McGovern & Barto, 2001; Menache et al., 2001; Pickett & Barto, 2002; Hengst, 2002; Şimşek & Barto, 2004; Jonsson & Barto, 2006).

In this paper, we focus on the *asymmetric knowledge transfer* setting where we are given access to solved source RL problems. The objective is to derive useful biases from these solutions that could speed up learning in target problems. We present and evaluate our approach, HI-MAT, for learning MAXQ hierarchies from a solved RL problem. HI-MAT applies dynamic Bayesian network (DBN) models to a single successful trajectory from the source problem to construct a *causally annotated trajectory* (CAT). Guided by the causal and temporal associations between actions in the CAT, HI-MAT recursively parses it and defines MAXQ subtasks based on each discovered partition of the CAT.

We analyze our approach both theoretically and empirically. Our theoretical results show that, under appropriate conditions, the task hierarchies induced by HI-MAT are consistent with the observed trajectory, and possess compact value-function tables that are safe with respect to state abstraction. Empirically, we show that (1) using a successful trajectory can result in more compact task decompositions than when using only DBNs, (2) our induced hierarchies are comparable to manually-engineered hierarchies on target RL tasks, and MAXQ-learning converges significantly faster than flat Q-learning on those tasks, and

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

(3) transferring hierarchical structure from a source task can speed up learning in target RL tasks where transferring value functions cannot.

2. Background and Related Work

We briefly review the MAXQ framework (Dietterich, 2000). This framework facilitates learning separate value functions for subtasks which can be composed to compute the value function for the overall semi-Markov Decision Process (SMDP) with state space $\mathcal{S}$ and action space $\mathcal{A}$. The task hierarchy $\mathcal{H}$ is represented as a directed acyclic graph called the *task graph*, and reflects the task-subtask relationships. Leaf nodes are the primitive subtasks corresponding to $\mathcal{A}$. Each composite subtask T_i defines an SMDP with parameters $\langle X_i, S_i, G_i, C_i \rangle$, where X_i is the set of relevant state variables, $S_i \subseteq \mathcal{S}$ is the set of admissible states, G_i is the termination/goal predicate, and C_i is the set of child tasks of T_i. T_0 represents the root task. T_i can be invoked in any state $s \in S_i$, it terminates when $s' \in G_i$, and (s, a) is called an *exit* if $\Pr(s'|s, a) > 0$.

The set S_i is defined using a projection function that maps a world state to an abstract state defined by a subset of the state variables. A *safe* abstraction function only merges world states that have identical values. The local policy for a subtask T_i is a mapping $\pi_i : S_i \mapsto C_i$. A hierarchical policy π for the overall task is an assignment of a local policy to each T_i. A *hierarchically optimal policy* for a given MAXQ graph is a hierarchical policy that has the best possible expected total reward. A hierarchical policy is *recursively optimal* if the local policy for each subtask is optimal given that all its child tasks are in turn recursively optimal.

HEXQ (Hengst, 2002) and VISA (Jonsson & Barto, 2006) are two existing approaches to learning task hierarchies. These methods define subtasks based on the changing values of state variables. HEXQ employs a heuristic that orders state variables based on the frequencies of change in their values to induce an *exit-option* hierarchy. The most frequently-changing variable is associated with the lowest-level subtask, and the least frequently-changing variable with the root. VISA uses DBNs to analyze the influence of state variables on one another. The variables are partitioned such that there is an acyclic influence relationship between the variables in different clusters (strongly-connected components). Here, state variables that influence others are associated with lower-level subtasks. VISA provides a more principled rationale for HEXQ's heuristic – a variable used to satisfy a precondition for setting another variable through an action typically changes more frequently than the other variable. A key difference between VISA and HI-MAT is the use of a successful trajectory in addition to the DBNs. In Section 5.1, we provide empirical evidence that this allows HI-MAT to learn hierarchies that are exponentially more compact than those of VISA.

The algorithm developed by Marthi et al. (2007) takes a search-based approach to generating hierarchies. Flat Q-value functions are learned for the source domain, and are used to sample trajectories. A greedy top-down search is conducted for the best-scoring hierarchy that fits the trajectories. The set of relevant state variables for each task is determined through statistical tests on the Q values of different states with differing values of the variables. In contrast to this approach, HI-MAT relies less on direct search through the hierarchy space, and more on the causal analysis of a trajectory based on DBN models.

3. Discovering MAXQ Hierarchies

In this work, we consider MDPs where the agent is solving a known conjunctive goal. This is a subset of the class of stochastic shortest-path MDPs. In such MDPs, there is a goal state (or a set of goal states), and the optimal policy for the agent is to reach such a state as quickly as possible. We assume that we are given factored DBN models for the source MDP where the conditional probability distributions are represented as trees (CPTs). Further, we are given a successful trajectory that reaches the goal in the source MDP. With this in hand, our objective is to automatically induce a MAXQ hierarchy that can suitably constrain the policy space when solving a related target problem, and therefore achieve faster convergence in the target problem. This is achieved via recursive partitioning of the given trajectory into subtasks using a top-down parse guided by backward chaining from the goal. We use the DBNs along with the trajectory to define the termination predicate, the set of subtasks, and the relevant abstraction for each MAXQ subtask.

We use the Taxi domain (Dietterich, 2000) to illustrate our procedure. Here, a taxi has to transport a passenger from a source location to a destination location within a 5×5 grid-world. The *pass.dest* variable is restricted to one of four special locations on the grid denoted by R, G, B, Y; the *pass.loc* could be set to R, G, B, Y or in-taxi; *taxi.loc* could be one of the 25 cells. The goal of *pass.loc* = *pass.dest* is achieved by taking the passenger to its intended destination. Besides the four navigation actions, a successful Pickup changes *pass.loc* to in-taxi, and a successful Putdown changes *pass.loc* from in-taxi to the value of *pass.dest*.

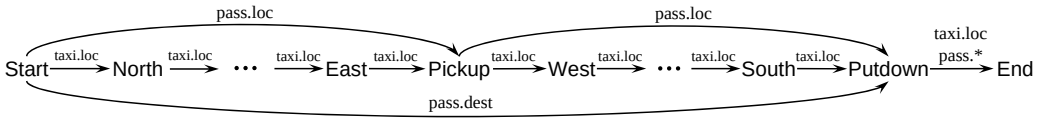


Figure 1. A sample CAT for the Taxi domain.

3.1. Definitions and Notation

We say that a variable v is *relevant* to an action a if the reward and transition dynamics for a either check or change v; it is *irrelevant* otherwise. The set of *trajectory-relevant* (t-relevant) variables of a, a subset of the relevant variables, are the variables that were actually checked or changed when a was executed in the trajectory. A *causal* edge $a \xrightarrow{v} b$ connects a to another action b (b following a in the trajectory) iff v is t-relevant to both a and b, and irrelevant to all actions in between. A *sink* edge, $a \xrightarrow{v} \mathsf{End}$ connects a with a dummy End action iff v is relevant to a and irrelevant to all actions before the final goal state; this holds analogously for a *source* edge $\mathsf{Start} \xrightarrow{v} a$. A *causally annotated trajectory* (CAT) is the original trajectory annotated with all the causal, source, and sink edges. Moreover, the CAT is preprocessed to remove any cycles present in the original trajectory (failed actions, such as an unsuccessful Pickup, introduce cycles of unit length). A sample CAT for Taxi is shown in Figure 1.

Given $a \xrightarrow{v} b$, the phrase "literal on a causal edge" refers to a formula of the form $v = \mathsf{V}$ where V is the value taken by v in the state before b is executed. We define `DBN-closure(`v`)` as the set of variables that influence v recursively as follows. From the action DBNs, add all variables that appear in internal nodes in the CPTs for the dynamics of v. Next, for each added variable u, union `DBN-closure(`u`)` with this set, repeating until no new variables are added. Similarly, the set `DBN-closure(`*reward*`)` contains all variables that influence the reward function of the MDP. The set `DBN-closure(`$fluent$`)` is the union of the `DBN-closure`s of all variables in the fluent. For example, `DBN-closure(`*goal*`)` is the set of all variables that influence the goal fluent. The CAT ignores all variables $v \notin$ `DBN-closure(`*goal*`)`, namely, those variables that never affect the goal conjunction.

3.2. The HI-MAT Algorithm

Given a CAT and the MDP's goal predicate (or recursively, the current subtask's goal predicate), the main loop of the hierarchy induction procedure is illustrated in Algorithm 1. The algorithm first checks if two stopping criteria are satisfied (lines 2 & 4): either the trajectory contains only a single primitive action, or it consists of actions whose relevances are identical. (In the latter case, any further partitioning would yield subtasks with the same abstraction as the parent.) Otherwise, it first initializes the set of "unsolved" goals to the set of literals in the goal conjunction (line 9). It then selects any unsolved goal u, and finds the corresponding subtask (line 12). Algorithm 2 returns indices i, j marking the boundaries of the subtask in the CAT. If this CAT segment is nontrivial (neither just the initial state nor the whole trajectory), it is stored (line 17), and the literals on causal edges that enter it (from earlier in the trajectory) are added to the unsolved goals (line 18). This ensures that the algorithm parses the entire trajectory barring redundant actions. If the trajectory segment is equal to the entire trajectory, this implies that the trajectory achieves only the literal u after the ultimate action. In this case, the trajectory is split into two segments: one segment contains the prefix of the ultimate action a_n with the preconditions of a_n forming the goal literals for this segment (line 14); the other segment contains only the ultimate action a_n (line 15). CAT scanning is repeated until all subgoal literals are accounted for.

The only way trajectory segments can overlap is if they have identical boundaries, and the ultimate action achieves the literals of all these segments. In this case, the segments are merged (line 23). Merging replaces the duplicative segments with one that is assigned a conjunction of the subgoal literals.

The HI-MAT algorithm partitions the CAT into unique segments, each achieving a single literal or a conjunction of literals due to merging. It is called recursively on each element of the partition (line 27). It can be proved that the set of subtasks output by the algorithm is independent of the order in which the literal u is picked (line 11).

3.2.1. Subtask Detection

Given a literal, a subtask is determined by finding the set of temporally contiguous actions that are closed with respect to the causal edges in the CAT such that the final action achieves the literal. The idea is to group all actions that contribute to achieving the specific literal being considered. This procedure is shown in Algorithm 2.

Algorithm 1 HI-MAT
Input: CAT Ω, Goal predicate G.
Output: Task $\langle X, S, G, C\rangle$; X is the set of relevant variables, S is the set of non-terminal states, G is the goal predicate, C is the set of child actions.

1: $n \leftarrow$ Number of actions in Ω excluding Start and End
2: **if** $n = 1$ **then** // Single action
3: **return** $\langle \textsc{RelVars}(\Omega), \mathcal{S}, \text{true}, a_1\rangle$
4: **else if** CheckRelVars(Ω) **then** // Same relevance
5: $S \leftarrow$ All states that reach G via $\textsc{Actions}(\Omega)$
6: **return** $\langle \textsc{RelVars}(\Omega), S, G, \textsc{Actions}(\Omega)\rangle$
7: **end if**
8: $\Psi \leftarrow \emptyset$ // Trajectory segments
9: $U \leftarrow \textsc{Literals}(G)$
10: **while** $U \neq \emptyset$ **do**
11: Pick $u \in U$
12: $(i, j, u) \leftarrow$ CAT-Scan(Ω, u)
13: **if** $i = 1 \wedge j = n$ **then**
14: $\Psi \leftarrow \Psi \cup \{(1, n-1, v) : v \in \textsc{Precondition}(a_n)\}$
15: $\Psi \leftarrow \Psi \cup \{(n, n, \emptyset)\}$
16: **else if** $j > 0$ **then** // Last segment action $\neq$ Start
17: $\Psi \leftarrow \Psi \cup \{(i, j, u)\}$
18: $U \leftarrow U \cup \{v : \exists k < i \; \exists l \; a_k \xrightarrow{v} a_l \in \Omega, i \leq l \leq j\}$
19: **end if**
20: $U \leftarrow U - \{u\}$
21: **end while**
22: **while** $\exists (i, j, u_1), (i, j, u_2) \in \Psi$ **do**
23: $\Psi \leftarrow (\Psi - \{(i, j, u_1), (i, j, u_2)\}) \cup \{(i, j, u_1 \wedge u_2)\}$
24: **end while**
25: $C \leftarrow \emptyset$
26: **for** $t \in \Psi$ **do**
27: $\langle X_t, S_t, G_t, C_t\rangle \leftarrow$ HI-MAT($\textsc{Extract}(\Omega, t_i, t_j), t_u$)
28: $C \leftarrow C \cup \{\langle X_t, S_t, G_t, C_t\rangle\}$
29: **end for**
30: $X \leftarrow \textsc{RelVars}(\Omega) \cup \textsc{Variables}(G)$
31: $S \leftarrow$ All states that reach G via C
32: **return** $\langle X, S, G, C\rangle$

Algorithm 2 CAT-Scan
Input: CAT Ω, literal u.
Output: (i, j, u); i is the start index, j is the end index.

1: Set j such that $a_j \xrightarrow{u}$ End $\in \Omega$
2: $i \leftarrow j - 1$
3: **while** $i > 0$ **and** $\forall v \; \exists k \; a_i \xrightarrow{v} a_k \implies k \leq j$ **do**
4: $i \leftarrow i - 1$
5: **end while**
6: **return** $(i + 1, j, u)$

As before, when considering causal edges in line 3, we can ignore all causal edges that are labeled with variables not in the `DBN-closure` of any variable in the current unsolved goal list. Because of the way we construct the CAT, we can show that this procedure will always stop before adding an action which has a relevant variable that is not relevant to the last action in the partition. Note that the temporal contiguity of the actions we assign to a subtask is required by the MAXQ-style execution of a policy. A hierarchical MAXQ policy cannot interrupt an unterminated subtask, start executing a sibling subtask, and then return to executing the interrupted subtask.

3.2.2. Termination Predicate

After finding the partition that constitutes a subtask, we assign a set of child tasks and a termination predicate to it. To assign the termination condition to a subtask, we consider the relational test(s) t_u in the action and reward DBNs involving the variable u on the causal edge leaving the subtask (line 27 of Algorithm 1). When a subtask's relational termination condition involves other variables not already in the abstraction, these variables are added to the state abstraction (line 30), effectively creating a parameterized subtask. For example, consider the navigation subtask that terminates when $taxi.loc = pass.dest$ in the Taxi domain. The abstraction for this subtask already involves $taxi.loc$. However, $pass.dest$ in the relational test implies that $pass.dest$ behaves like a parameter for this subtask.

3.2.3. Action Generalization

To determine if the set of primitive actions available to any subtask should be expanded, we follow a bottom-up procedure (not shown in Algorithm 1). We start with subtasks that have only primitive actions as children. We create a *merged* DBN structure for such a subtask T using the incorporated primitive actions. The merged DBN represents possible variable effects after any sequence of these primitive actions. Next, for each primitive action that we did not see in this trajectory, we consider the subgraph of its DBN that only involves the variables relevant to T. If this is a subgraph of the merged DBN of T, we add this action to the set of actions available to T. The rationale here is that the added action has similar effects to the actions we observed in the trajectory, and it does not increase the set of relevant variables for T. For example, if the navigation actions used on the observed trajectory consisted only of North and East actions, this procedure would also add South and West to the available actions for this subtask. When considering subtasks that have non-primitive children, we only consider adding actions that have not been added to any of the non-primitive children.

Given the termination predicate and the generalized set of actions, the set of relevant variables for a subtask is the union of the set of relevant variables of the merged DBN (described above) and the variables appearing in the termination predicate (line 30). Computing the relevant variables is similar to explanation-based reinforcement learning (Tadepalli & Dietterich,

1997) except that here we care only about the set of relevant variables and not their values. Moreover, the relevant variables are computed over a set rather than a sequence of actions.

4. Theoretical Analysis

In this section, we establish certain theoretical properties of the hierarchies induced by the HI-MAT algorithm. We consider a factored SMDP state-space $\mathcal{S} = D_{x_1} \times \ldots \times D_{x_k}$, where each D_{x_i} is the domain of variable x_i. We assume that our DBN models have the following property.

Definition 1 *A DBN model is maximally sparse if for any $y \in Y$ where Y is the set of parents of some node x (which represents either a state variable or the reward node), and $Y' = Y - \{y\}$,*

$$\exists y_1, y_2 \in D_y \quad \Pr(x|Y', y = y_1) \neq \Pr(x|Y', y = y_2).$$

Maximal sparseness implies that the parents of a variable have non-trivial influences on it; no parent can be removed without affecting the next-state distribution.

A task hierarchy $\mathcal{H} = \langle V, E \rangle$, is a directed acyclic graph, where V is a set of task nodes, and E represents the task-subtask edges of the graph. Each task node $T_i \in V$ is defined as in Section 2.

A trajectory-task pair $\langle \Omega, T_i \rangle$, where $\Omega = \langle s_1, a_1, \ldots, s_n, a_n, s_{n+1} \rangle$ and $T_i = \langle X_i, S_i, G_i, C_i \rangle$, is consistent with $\mathcal{H}$ if $T_i \in V$, and $\{s_1, \ldots, s_n\} \subseteq S_i$. If T_i is a primitive subtask then $n = 1$, and $C_i = a_1$. If T_i is not primitive then $\{s_1, \ldots, s_n\} \cap G_i = \emptyset$, $s_{n+1} \in G_i$, and there exist trajectory-task pairs $\langle \Omega_j, T_j \rangle$ consistent with $\mathcal{H}$ where Ω is a concatenation of $\Omega_1, \ldots, \Omega_p$ and $T_1, \ldots, T_p \in C_i$.

A trajectory Ω is consistent with a hierarchy $\mathcal{H}$ if $\langle \Omega, T_0 \rangle$ is consistent with $\mathcal{H}$.

Definition 2 *A trajectory $\langle s_1, a_1, \ldots, s_n, a_n, s_{n+1} \rangle$ is non-redundant if no subsequence of the action sequence in the trajectory, $a_1, \ldots, a_n$, can be removed such that the remaining sequence still achieves the goal starting from s_1.*

Theorem 1 *If a trajectory Ω is non-redundant then HI-MAT produces a task hierarchy $\mathcal{H}$ such that Ω is consistent with $\mathcal{H}$.*

Proof sketch: Let $\Omega = \langle s_1, a_1, \ldots, s_n, a_n, s_{n+1} \rangle$ be the trajectory. The algorithm extracts the conjunction of literals that are true in s_{n+1} (and not before), and assigns it to the goal, G_i. Such literals must exist since, otherwise, some suffix of the trajectory can be removed while the rest still achieves the goal, violating the property of non-redundancy. Since the set S_i is set to all states that do not satisfy G_i, the condition that all states $s_1, \ldots, s_n$ are in S_i is satisfied.

Whenever the trajectory is partitioned into a sequence of sub-trajectories, each sub-trajectory is associated with a conjunction of goal literals achieved by that sub-trajectory. Hence, the above argument applies recursively to each such sub-trajectory. □

Definition 3 *A hierarchy $\mathcal{H}$ is safe with respect to the DBN models M if for any trajectory-task pair $\langle \Omega, T_i \rangle$ consistent with $\mathcal{H}$, where $T_i = \langle X_i, S_i, G_i, C_i \rangle$, the total expected reward during the trajectory is only a function of the values of $x \in X_i$ in the starting state of Ω.*

The above definition says that the state variables in each task are sufficient to capture the value of any trajectory consistent with the sub-hierarchy rooted at that task node.

Theorem 2 *If the procedure HI-MAT produces a task hierarchy $\mathcal{H}$ from Ω and the DBN models M then $\mathcal{H}$ is safe with respect to M. Further, if the DBN models are maximally sparse, for any hierarchy $\mathcal{H}'$ which is consistent with Ω and safe with respect to M, and $T_i = \langle X_i, S_i, G_i, C_i \rangle$ in $\mathcal{H}$, there exists $T'_i = \langle X'_i, S'_i, G'_i, C'_i \rangle$ in $\mathcal{H}'$ such that $X_i \subseteq X'_i$.*

Proof sketch: By the construction procedure, in any segment of trajectory Ω composed of primitive actions under a subtask T_i, all primitive actions check or set only the variables in X_i. Thus, changing any other variables in the initial state s of Ω yielding s' does not change the effects of these actions according to the DBN models. Similarly, all immediate rewards in the trajectory are also functions of the variables in X_i. Hence, the total accumulated reward and the probability of the trajectory only depend on X_i, and the hierarchy produced is safe with respect to M.

Suppose that $\mathcal{H}'$ is a consistent hierarchy which is safe with respect to M. Let a_i be the last action in the trajectory Ω_i corresponding to the subtask T_i in $\mathcal{H}$. By consistency, there must be some task T'_i in $\mathcal{H}'$ that matches up with a_i. Recall that X_i includes only those variables checked and set by a_i to achieve the goal G_i. We claim that the abstraction variables X'_i of T'_i must include X_i. If this is not the case then, by maximal sparseness, there is a variable y in $X_i - X'_i$ and some values y_1 and y_2 such that the probabilities of the next state or reward are different based on whether $y = y_1$ or $y = y_2$. Hence, $\mathcal{H}'$ would not be safe, leading to a contradiction. □

Corollary 1 *If the DBN models are maximally sparse then the maximum size of the value function table for any task in the hierarchy produced by HI-MAT is the smallest over all safe hierarchies which are consistent with the trajectory.*

Proof: Direct consequence of part 2 of the previous theorem. □

The significance of the above corollary lies in the fact that the size of the value-function table is exponential in the number of variables $n_i = |X_i|$ in the abstraction of task T_i. If all features are binary and there are t tasks then the total number of values for the value-function tables is $O(t\,2^{n_{\max}})$. Since the hierarchy is a tree with the primitive actions at the leaves, the number of subtasks is bounded by $2l$ where l is the length of the trajectory. Hence, we can claim that the number of parameters needed to fully specify the value-function tables in our hierarchy is at most $O(l)$ times that of the best possible.

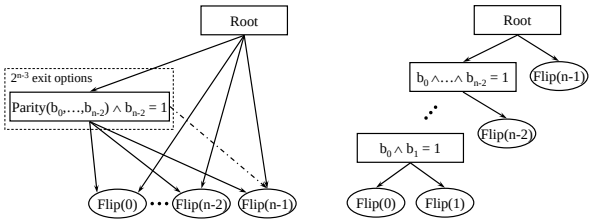
Our analysis does not address state abstractions arising from the so-called *funnel* property of subtasks where many starting states result in a few terminal states. Funnel abstractions permit the parent task to ignore variables that, while relevant inside the child task, do not affect the terminal state. Nevertheless, our analysis captures some of the key properties of our algorithm including consistency with the trajectory, safety, minimality, and sheds some light on its effectiveness.

5. Empirical Evaluation

We test three hypotheses. First, we expect that employing a successful trajectory along with the action models will allow the HI-MAT algorithm to induce task hierarchies that are much more compact than (or at least as compact as) just using the action models. Second, in a transfer setting, we expect that the hierarchies induced by HI-MAT will speed up convergence to the optimal policy in related target problems. Finally, we expect that the HI-MAT hierarchies will be applicable to and speed up learning in RL problems which are different enough from the source problems such that value functions either do not transfer or lead to poor transfer.

5.1. Contribution of the Trajectory

To highlight our first hypothesis, a modified Bitflip domain (Diuk et al., 2006) is designed as follows. The state is represented by n bits, $b_0 b_1 \ldots b_{n-1}$. There are n actions denoted by Flip(i). Flip(i) toggles b_i if both



(a) VISA hierarchy. Task labels are the exit conditions; dash-dot arrows indicate exit options.

(b) HI-MAT hierarchy. Task labels are the termination conditions.

Figure 2. Task hierarchies for the modified Bitflip domain.

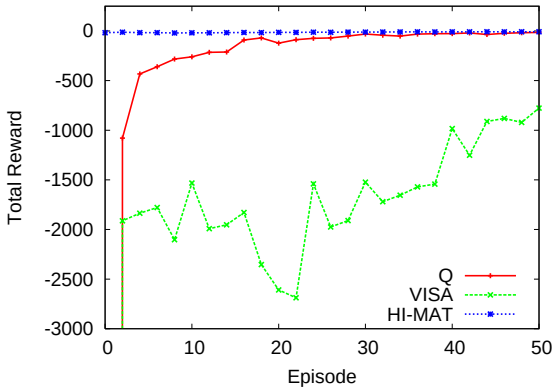


Figure 3. Performance of Q, VISA, and HI-MAT in the 7-bit modified Bitflip domain (averaged over 20 runs).

b_{i-1} is set and the parity across bits $b_0, \ldots, b_{i-1}$ is even when i is even (odd otherwise); if not, it resets the bits $b_0, \ldots, b_i$. All bits are reset at the initial state, and the goal is to set all bits.

We ran both VISA and HI-MAT in this domain with $n = 7$, and compared the induced hierarchies (Figure 2). We observe that VISA constructs an exponentially sized hierarchy even with subtask merging activated within VISA. There are two reasons for this. First, VISA relies on the full action set to construct its causal graph, and does not take advantage of any context-specific independence among its variables that may arise when the agent acts according to certain policies. Specifically, for this domain, the causal graph constructed from DBN analysis has only two strongly connected components (SCCs): one partition has $\{b_0, \ldots, b_{n-2}\}$, and the other has $\{b_{n-1}\}$. This SCC cannot be further decomposed using only information from the DBNs. Second, VISA creates exit options for all strongly connected components that transitively influence the reward function, whereas only a few of these may actually be necessary to solve the problem. Specifically, for this problem, VISA creates

an exit condition for any instantiation that satisfies $\mathsf{parity}(b_0, \ldots, b_{n-2}) \wedge b_{n-2} = 1$, resulting in exponential number of subtasks shown in Figure 2(a). The successful trajectory provided to HI-MAT achieves the goal by setting the bits going from left to right, and results in the hierarchy in Figure 2(b). The performance results are shown in Figure 3. VISA's hierarchy converges even slower than the basic Q learner because the root has $O(2^n)$ children as opposed to $O(n)$.

This domain has been engineered to highlight the case when access to a successful trajectory allows for significantly more compact hierarchies than without. We expect that access to a solved instance will usually improve the compactness of the resulting hierarchy.

5.2. Transfer of the Task Hierarchy

To test our remaining hypotheses, we apply the transfer setting to two domains: Taxi and the real-time strategy game Wargus. The Taxi domain has been described in Section 3. The source and target problems in Taxi differ only in the wall configurations; the passenger sources and destinations are the same. This is engineered to allow value-function transfer to occur. For Wargus, we consider the *resource collection* problem. Here, the agent has units called peasants that can harvest gold and wood from goldmines and forests respectively, and deposit them at a townhall. The goal is to reach a predetermined quota of gold and wood. Since the HI-MAT approach does not currently generalize to termination conditions involving numeric predicates, the state representation of the domain replaces the actual quota variables with Boolean variables that are set when the requisite quotas of gold and wood are met. We consider target problems whose specifications are scaled up from that of the source problems, including the number of peasants, goldmines, forests, and the size of the map. In this domain, coordination does not affect the policy significantly. Thus, in the target maps, we learn a hierarchical policy for the peasants using a shared hierarchy structure without coordination (Mehta & Tadepalli, 2005). In each case, we report the total reward received as a function of the number of episodes, averaged over multiple trials.

We compare three basic approaches: (1) non-hierarchical Q-learning (Q), (2) MAXQ-learning applied to a hierarchy manually engineered for each domain (Manual), and (3) MAXQ-learning applied to the HI-MAT hierarchy induced for each domain (HI-MAT). The HI-MAT algorithm first solves the source problem using flat Q-learning, and generates a successful trajectory from it. In Taxi, we also show the performance of initializing the value-function tables with values learned from the source problem – these curves are suffixed with the phrase "with value". In Wargus, we include the performance of VISA. The results of these experiments are shown in Figures 4 and 5.

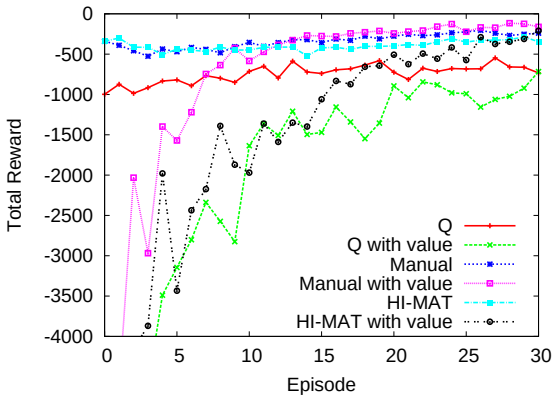


Figure 4. Performance in the Taxi domain (averaged over 20 runs). Source and target problems differ only in the configuration of the grid walls.

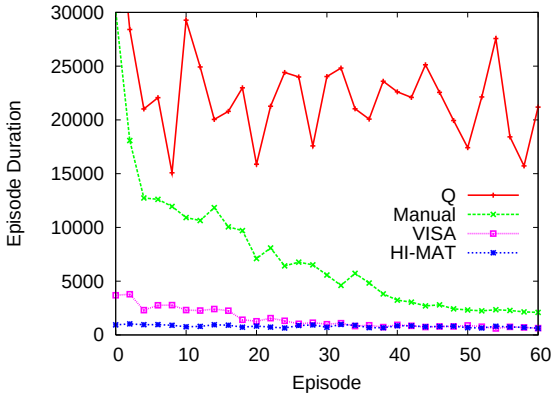


Figure 5. Performance in the Wargus domain (averaged over 10 runs). Source: 25×25 grid, 1 peasant, 2 goldmines, 2 forests, 1 townhall, 100 units of gold, 100 units of wood. Target: 50×50 grid, 3 peasants, 3 goldmines, 3 forests, 1 townhall, 300 units of gold, 300 units of wood.

Although the target problems in Taxi allow value-function transfer to occur, the target problems are still different enough that the agent has to "unlearn" the old policy. This leads to negative transfer evidenced in the fact that transferring value functions leads to worse rates of convergence to the optimal policy than transferring just the hierarchy structure with uninitialized policies. This indicates that transferring structural knowledge via the task-subtask decomposition can be superior to transferring value functions especially when the target problem differs significantly in terms of its optimal policy. In Wargus, the difference

between the source and target problems renders direct value-function transfer impossible even though the hierarchy structure still transfers.

In Taxi, we observe that MAXQ-learning on HI-MAT's hierarchy converges to the optimal policy at a rate comparable to that of the manually-engineered hierarchy. However, in Wargus, HI-MAT's hierarchy is faster to converge than the manually-engineered one because, by analyzing the solved source problem, it is able to find stricter termination conditions for each subtask. Consequently, reducing the policy space in the target problem leads to a greater speed-up in learning than reducing the number of value parameters via subtask sharing as in the manually-engineered hierarchy. The improved rate of convergence is in spite of the fact that HI-MAT does not currently merge subtly different instantiations of the same subtask so there is room for further improvement. VISA's performance suffers initially due to a large branching factor at the root option (which directly includes all the navigation actions).

6. Conclusion

We have presented an approach to automatically inducing MAXQ hierarchies from solved RL problems. Given DBN models and an observed successful trajectory, our method analyzes the causal and temporal relationships between actions, and partitions the trajectory recursively into subtasks as long as the state abstraction improves. We show that the learned hierarchies are consistent, safe, and have compact value-function tables. Our empirical results indicate that using the observed trajectory can allow us to learn more compact hierarchies. Further, in a transfer setting, our hierarchies perform comparably to manually-engineered hierarchies, and improve the rate of convergence where direct policy transfer does not help.

We are currently working on extending the approach to handle disjunctive goals. Further, in order to ensure hierarchical optimality, we may need to deal with non-zero pseudo-rewards. In related work, we are also investigating methods that learn compact DBN models of the MDP. Finally, an important challenge for the future is to investigate the transfer scenario where the induced hierarchy may need to be altered structurally in order to apply effectively to the target problem.

Acknowledgments

We thank Mike Wynkoop and the anonymous reviewers for their input. We gratefully acknowledge the support of the Defense Advanced Research Projects Agency under DARPA grant FA8750-05-2-0249.

References

Andre, D., & Russell, S. (2002). State Abstraction for Programmable Reinforcement Learning Agents. *AAAI* (pp. 119–125).

Şimşek, Ö., & Barto, A. (2004). Using Relative Novelty to Identify Useful Temporal Abstractions in Reinforcement Learning. *ICML* (pp. 751–758).

Dietterich, T. (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303.

Diuk, C., Littman, M., & Strehl, A. (2006). A Hierarchical Approach to Efficient Reinforcement Learning in Deterministic Domains. *AAMAS* (pp. 313–319).

Hengst, B. (2002). Discovering Hierarchy in Reinforcement Learning with HEXQ. *ICML* (pp. 243–250).

Jonsson, A., & Barto, A. (2006). Causal Graph Based Decomposition of Factored MDPs. *Journal of Machine Learning Research*, *7*, 2259–2301.

Marthi, B., Kaelbling, L., & Lozano-Perez, T. (2007). Learning Hierarchical Structure In Policies. *NIPS Hierarchical Organization of Behavior Workshop.*

McGovern, A., & Barto, A. (2001). Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. *ICML* (pp. 361–368).

Mehta, N., & Tadepalli, P. (2005). Multi-Agent Shared Hierarchy Reinforcement Learning. *ICML Rich Representations in Reinforcement Learning Workshop.*

Menache, I., Mannor, S., & Shimkin, N. (2001). Q-Cut - Dynamic Discovery of Sub-Goals in Reinforcement Learning. *ECML* (pp. 295–306).

Pickett, M., & Barto, A. (2002). PolicyBlocks: An Algorithm for Creating Useful Macro-Actions in Reinforcement Learning. *ICML* (pp. 506–513).

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, *112*, 181–211.

Tadepalli, P., & Dietterich, T. (1997). Hierarchical Explanation-Based Reinforcement Learning. *ICML* (pp. 358–366).

Thrun, S., & Schwartz, A. (1995). Finding Structure in Reinforcement Learning. *NIPS* (pp. 385–392).

Rank Minimization via Online Learning

Raghu Meka RAGHU@CS.UTEXAS.EDU
Prateek Jain PJAIN@CS.UTEXAS.EDU
Constantine Caramanis CMCARAM@ECE.UTEXAS.EDU
Inderjit S. Dhillon INDERJIT@CS.UTEXAS.EDU
University of Texas at Austin, Austin, TX 78712

Abstract

Minimum rank problems arise frequently in machine learning applications and are notoriously difficult to solve due to the non-convex nature of the rank objective. In this paper, we present the first online learning approach for the problem of rank minimization of matrices over polyhedral sets. In particular, we present two online learning algorithms for rank minimization - our first algorithm is a multiplicative update method based on a generalized experts framework, while our second algorithm is a novel application of the online convex programming framework (Zinkevich, 2003). In the latter, we flip the role of the decision maker by making the decision maker search over the constraint space instead of feasible points, as is usually the case in online convex programming. A salient feature of our online learning approach is that it allows us to give provable approximation guarantees for the rank minimization problem over polyhedral sets. We demonstrate the effectiveness of our methods on synthetic examples, and on the real-life application of low-rank kernel learning.

1. Introduction

Minimizing the rank of matrices restricted to a convex set is an important problem in the field of optimization with numerous applications in machine learning. For instance, many important problems like low-rank kernel learning, feature efficient linear classification, semi-definite embedding (SDE), non-negative matrix approximation (NNMA), etc., can be viewed as rank minimization problems over a polyhedron with additional convex constraints such as a Frobenius norm constraint and/or a semi-definiteness constraint. Even though there has been extensive work on the specific problems mentioned above, the general problem of rank minimization over polyhedral sets is not well understood. In this paper we address the problem of rank minimization when there are a large number of trace constraints along with a few convex constraints that are relatively "easy" in a precise sense defined below.

We now formulate the rank minimization problem we study. Let $A_1, \ldots, A_m \in \mathbb{R}^{n \times n}$, $b_1, \ldots, b_m \in \mathbb{R}$ and let $\mathcal{C} \subseteq \mathbb{R}^{n \times n}$ be a convex set of matrices. Then, consider the following optimization problem which we refer to as RMP (for Rank Minimization over Polyhedron):

$$\begin{aligned} \min \quad & \text{rank}(X) \\ \text{s.t} \quad & \text{Tr}(A_i X) \geq b_i, \quad 1 \leq i \leq m \\ & X \in \mathcal{C}. \end{aligned} \tag{RMP}$$

The set $\mathcal{C}$ will represent the "easy" constraints in the sense that for such a set $\mathcal{C}$, we assume that RMP with a single trace constraint can be solved efficiently. This holds for many typical convex sets $\mathcal{C}$, e.g., the unit ball under any L_p or Frobenius norm, the semi-definite cone, and the intersection of the unit ball with the p.s.d. cone. Furthermore, low-rank kernel learning, SDE and NNMA can all be seen as instantiations of the above general formulation.

The general RMP problem as stated above is non-convex, NP-hard and, as we prove, cannot be approximated well unless $P = NP$. Due to the computational hardness of the problem, much of the previous work has concentrated on providing heuristics, with no guarantees on the quality of the solution. We remark that the recent trace-norm based approach of (Recht et al., 2007) does guarantee an optimal solution for a simplified instance of RMP where only well-conditioned linear equality constraints are allowed. However, it is not clear how to extend their guarantees to the more general RMP problem.

We now list the main contributions of this paper:

- We show that for the RMP problem, the minimum feasible rank cannot be approximated well unless $P =$

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

NP (see Theorem 3.1). To get over this hurdle we introduce a relaxed notion of approximation, where along with approximating the optimal rank we also allow small violations in the constraints. In practice, this relaxed notion is as meaningful as the standard notion of approximation since almost all real-life problems have noisy measurements.

- We provide an algorithm for RMP based on the Multiplicative Weights Update framework of (Plotkin et al., 1991; Arora et al., 2005b) and under the relaxed notion of approximation, we prove approximation guarantees for the algorithm.
- We provide an algorithm for RMP based on the framework of online convex programming (OCP) introduced by (Zinkevich, 2003). We use the OCP framework in a novel way by changing the role of the decision maker to search over the constraints instead of the feasible points, as is usually the case. We prove that under the relaxed notion of approximation, the algorithm provides approximation guarantees. The guarantees obtained using the OCP framework are better than those obtained using the Multiplicative Weights Update framework by a logarithmic factor.
- For a practical application, we apply our methods to the problem of low-rank kernel learning which can be seen as a specific instance of general RMP.

We empirically evaluate our methods on synthetic instances of RMP, where the constraints are chosen randomly. We compare them with the trace-norm heuristic of (Fazel et al., 2001; Recht et al., 2007) and the log-det heuristic of (Fazel et al., 2003), and our experimental results indicate that our methods are significantly faster and give comparable rank solutions to existing methods. We also evaluate the performance of our methods for low-rank kernel learning on UCI datasets. On all the datasets, our algorithms improve the accuracy of the baseline kernel while also significantly decreasing the rank.

2. Related Work and Background

Most existing methods for rank minimization over convex sets are based on relaxing the non-convex rank function to a convex function, e.g., the trace-norm (Fazel et al., 2001; Recht et al., 2007) or the logarithm of the determinant (Fazel et al., 2003). Unfortunately, these heuristics do not have any guarantees on the quality of the solution in general. A notable exception is the work of (Recht et al., 2007), which extends the techniques of (Candès & Tao, 2005) for compressed sensing to rank minimization. (Recht et al., 2007) show that minimizing the trace-norm guarantees an optimal rank solution to a special class of RMP where only well-conditioned linear equalities are allowed. Thus their approach is limited in its applicability and it is not clear how to extend it to general RMP. We also remark that minimizing the trace-norm is computationally expensive, which further limits its applicability.

(Barvinok, 2002) (Chapter V) describes an approximation algorithm for RMP based on random projections and a generalization of the Johnson-Lindenstrauss Lemma, with an approximation guarantee similar to the one provided by our MW algorithm (Section 4.1). However, this approach works only for a special case of RMP where only linear equalities described by p.s.d. matrices are allowed. Furthermore, this approach needs to solve the relaxed RMP problem without the rank constraint which involves solving a large semi-definite programming problem. This maybe undesirable for various real-world applications such as the low-rank kernel learning problem. In contrast, our approaches can be used for a larger class of convex sets $\mathcal{C}$ and are considerably more scalable.

Several specific instances of the general RMP problem have been widely researched in the machine learning community. Examples include low-rank kernel learning, SDE, sparse PCA and NNMA. Most methods for these problems can be broadly grouped into the following two categories: a) methods which drop the rank constraint and use the top k eigenvectors of the solution to the relaxed optimization problem e.g., (Weinberger et al., 2004); b) methods which factor the matrix X in RMP into AB^T and optimize the resultant non-convex problem e.g., (Lee & Seung, 2000; Kim et al., 2007). However, typically these methods do not have any provable guarantees.

We apply our algorithms for the general RMP problem to the low-rank kernel learning problem(Bach & Jordan, 2005; Kulis et al., 2006). Existing methods for this problem do not provide any provable guarantees on the solution and/or assume that the initial kernel has a small rank to begin with. In contrast, a straight forward application of our general RMP framework gives algorithms with provable guarantees on the rank of the learned kernel. Furthermore, we demonstrate that our algorithms can be used to initialize existing methods to obtain better solutions.

Our approaches to RMP are based on two online learning methods - the generalized experts framework as abstracted in (Arora et al., 2005b) and the online convex programming (Zinkevich, 2003), which we now review briefly.

2.1. Multiplicative Weights Update Algorithm

The Multiplicative Weights Update algorithm (MW algorithm) is an adaptation of the Winnow algorithm (Littlestone & Warmuth, 1989) for a generalized experts framework as described in (Freund & Schapire, 1997). This framework was implicitly used by (Plotkin et al., 1991) for solving several fractional packing and covering problems and was formalized and extended to semi-definite programs

in (Arora et al., 2005a). Throughout this work we will follow the presentation of the generalized experts framework as abstracted in (Arora et al., 2005b).

In the generalized experts (GE) framework there is a set of n experts, a set of events $\mathcal{E}$, and a penalty matrix M such that the i-th expert incurs a penalty of $M(i,j)$ for an event $j \in \mathcal{E}$. The penalties are assumed to be bounded and lie in the interval $[-\rho, \rho]$ for a fixed $\rho > 0$. At each time step $t = 1, 2, \ldots,$ an adversary chooses an event $j^t \in \mathcal{E}$ so that the i-th expert incurs a penalty of $M(i, j^t)$. The goal in the GE framework is to formulate a *prediction algorithm* that chooses a distribution $\mathcal{D}^t = (p_1^t, \ldots, p_n^t)$ on the experts at time step t, so that the total expected loss incurred by the prediction algorithm is not much worse than the total loss incurred by the best expert. Formally, the goal of the prediction algorithm is to minimize

$$\sum_{t=1}^{T}\sum_{l=1}^{n} p_l^t M(l, j^t) - \min_i \sum_{t=1}^{T} M(i, j^t).$$

Note that the distribution in round t, $\mathcal{D}^t$, must be chosen without knowledge of the event j^t chosen at time step t. At every step t, the MW algorithm has a weight w_i^t assigned to expert i, and sets the distribution $\mathcal{D}^t = (p_1^t, \ldots, p_n^t)$, where $p_i^t = w_i^t / \sum_j w_j^t$. The MW algorithm then proceeds analogously to the Winnow algorithm and updates the weights at time step $t+1$ to $w_i^{t+1} = w_i^t(1-\delta)^{M(i,j^t)/\rho}$ if $M(i,j^t) \geq 0$ and $w_i^{t+1} = w_i^t(1+\delta)^{M(i,j^t)/\rho}$ if $M(i,j^t) < 0$, where δ is a parameter provided to the algorithm. For our analysis we will use the following theorem.

Theorem 2.1 (Corollary 4 of (Arora et al., 2005b))**.** *Suppose that for all i and $j \in \mathcal{E}$, $M(i,j) \in [-\rho, \rho]$. Let $\epsilon > 0$ be an error parameter and let $\delta = \min\{\frac{\epsilon}{4\rho}, \frac{1}{2}\}$, and $T = \frac{16\rho^2 \ln n}{\epsilon^2}$. Then, the following bound holds for the average expected loss of the MW algorithm*

$$\frac{\sum_{t=1}^{T}\sum_{l=1}^{n} p_l^t M(l, j^t)}{T} \leq \epsilon + \frac{\sum_t M(k, j^t)}{T}, \ \forall k.$$

2.2. Online Convex Programming

The online convex programming (OCP) framework (Zinkevich, 2003; Kalai & Vempala, 2005; Hazan et al., 2006) models various useful online learning problems like industrial production and network routing. The OCP framework involves a fixed convex set K and a sequence of unknown cost functions $f_1, f_2, \ldots : K \to \mathbb{R}$. At each time step t, a decision maker must choose a point $z_t \in K$ and incurs a cost $f_t(z_t)$. However, the choice of z_t must be made with the knowledge of $z_1, \ldots, z_{t-1}$ and $f_1, \ldots, f_{t-1}$ alone i.e., without knowing f_t. The total cost incurred by the algorithm after T steps equals $\sum_t f_t(z_t)$. The objective in OCP is to minimize the *regret* as defined below:

$$R(T) = \sum_{t=1}^{T} f_t(z_t) - \min_{z \in K} \sum_{t=1}^{T} f_t(z). \tag{1}$$

(Zinkevich, 2003) has shown that in the case when the functions f_t are convex and differentiable with bounded gradient, one can achieve a regret of $O(\sqrt{T})$. Let $\|K\| = \max_{z_1, z_2 \in K} \|z_1 - z_2\|$ and $G = \max_{z \in K, t \in \{1,\ldots\}} \| \bigtriangledown f^t(z)\|$, where $\|\cdot\|$ denotes the Euclidean norm (or Frobenius norm if the set K is defined over matrices). Also, assume that $\bigtriangledown f^t$ can be evaluated efficiently at any given point z. Under the above assumptions (Zinkevich, 2003) proposed a Generalized Infinitesimal Gradient Ascent algorithm which achieves a regret of $O((G^2 + \|K\|^2)\sqrt{T})$. The function GIGA in Algorithm 2 describes a slightly modified version of (Zinkevich, 2003)'s algorithm that achieves the following improved regret bound.

Theorem 2.2 (Adaptation of Theorem 1 of (Zinkevich, 2003))**.** *The following bound holds for the regret of the GIGA sub-routine of Algorithm 2 after T rounds,*

$$R(T) \leq G \cdot \|K\| \sqrt{T} \tag{2}$$

Proof sketch: Using the modified step-size in Algorithm 2, the theorem follows from Zinkevich's original proof. □

3. Computational Complexity

As was mentioned in the introduction, RMP is NP-hard in general. Further, by a reduction to the problem of support minimization over convex sets, and using hardness of approximation results from (Amaldi & Kann, 1998) we prove the following hardness result for RMP. A full proof of the following theorem appears in (Meka et al., 2008).

Theorem 3.1. *There exists no polynomial time algorithm for approximating* RMP *within a logarithmic factor unless $P = NP$. Further, assuming $NP \nsubseteq DTIME(n^{poly \log n})$,* RMP *is not approximable within a factor of $2^{\log^{1-\delta} n}$ for every $\delta > 0$; and* RMP *is not approximable within a factor of $2^{\log^{1-\delta} \Delta}$ for every $\delta > 0$, where $\Delta = \max\{\|A_i\|_F + |b_i| : 1 \leq i \leq m\}$* [1].

In view of the above hardness result we introduce a weaker notion of approximation. We believe the relaxed notion of approximation to be of equal use, if not more, as the standard notion of approximation in practice. For an instance of RMP, let $\mathbb{F}(A_1, \ldots, A_m, \boldsymbol{b}, \mathcal{C})$ denote the feasible region, where $\boldsymbol{b} = (b_1, \ldots, b_m)$:

$$\mathbb{F}(A_1, \ldots, A_m, \boldsymbol{b}, \mathcal{C}) = \{X : X \in \mathcal{C}, \mathrm{Tr}(A_i X) \geq b_i, \forall i\}. \tag{3}$$

[1] This hardness result holds even when $\mathcal{C}$ is fixed to be the unit ball under an L_p or Frobenius norm or many other common sets.

Definition 3.1. *Given a function $c : \mathbb{R} \to \mathbb{R}_+$, we say that a matrix $\overline{X}$ is a $(c(\epsilon), \epsilon)$-approximate solution to* RMP *if the following hold:*

$\bar{X} \in \mathbb{F}(A_1, \ldots, A_m, \boldsymbol{b} - \epsilon\mathbf{1}, \mathcal{C})$

$\textit{rank}(\bar{X}) \leq c(\epsilon) \min\{\textit{rank}(X) : X \in \mathbb{F}(A_1, .., A_m, \boldsymbol{b}, \mathcal{C})\}$.

Further, we say that RMP *is $(c(\epsilon), \epsilon)$-approximable, if there exists a polynomial time algorithm that given inputs $A_1, \ldots, A_m, \boldsymbol{b}, \epsilon$, outputs a $(c(\epsilon), \epsilon)$-approximate solution to* RMP.

Thus, along with approximating the minimum feasible rank we also allow a small violation, quantified by ϵ, of the constraints. Note that for $\epsilon = 0$, we recover the normal notion of approximation with an approximation factor of $c(0)$.

4. Methodology

Our approaches to RMP rely on the fact that even though RMP is hard in general, it is efficiently solvable for certain convex sets $\mathcal{C}$ when there is a single trace constraint. For instance, when $\mathcal{C} = \{X : \|X\|_F \leq 1\}$, a RMP problem with a single trace constraint can be solved efficiently using a singular value decomposition of the constraint matrix.

In our approach, we assume the existence of an oracle $\mathcal{O}$ that solves the following RMP problem with a single trace constraint, and returns an optimal X or declares the problem infeasible:

$$\mathcal{O}: \quad \min \text{ rank}(X) \text{ s.t. } \text{Tr}(AX) \geq b, X \in \mathcal{C}. \quad (4)$$

As discussed above, for certain convex sets $\mathcal{C}$, oracle $\mathcal{O}$ solves a non-convex problem. In both our approaches, we exploit this fact by making several queries to the oracle where the trace constraint $\text{Tr}(AX) \geq b$ is obtained by a weighted combination of the original trace constraints. The trick then is to choose the combinations in such a way that after a small number of iterations, we can find a low-rank X that satisfies all the constraints with at most an ϵ-violation.

Based on the above intuition, we give two approaches to solve the RMP problem - one based on the Multiplicative Weights Update algorithm and the other based on online convex programming.

Before we describe our algorithms, we need to introduce additional notation. For an instance of RMP specified by matrices $A_1, \ldots, A_m$, scalars $b_1, \ldots, b_m$ and convex set $\mathcal{C}$, let $D = \max\{\|X\|_F : X \in \mathcal{C}\}$. We assume, without loss of generality, that $D \geq 1$. Recall that $\mathbb{F}((A_1, \ldots, A_m), \boldsymbol{b}, \mathcal{C})$ and $\mathbb{F}((A_1, \ldots, A_m), \boldsymbol{b} - \epsilon\mathbf{1}, \mathcal{C})$ denote the feasibility sets as defined in (3) and $\Delta = \max\{\|A_i\|_F + |b_i| : 1 \leq i \leq m\}$. Further, let k^* be the rank of the optimal solution to RMP. That is,

$$k^* = \min\{\text{rank}(X) : X \in \mathbb{F}((A_1, \ldots, A_m), \boldsymbol{b}, \mathcal{C})\}.$$

Algorithm 1 RMP-MW (Multiplicative Updates)

Require: Constraints (A_i, b_i), $1 \leq i \leq m$, ϵ
Require: Oracle $\mathcal{O}(A, b)$ which solves $\min \text{ rank}(X) \quad s.t. \quad \text{Tr}(AX) \geq b,\ X \in \mathcal{C}$
Initialize: $w_i^1 = 1,\ \forall i$ and $t = 1$
repeat
  Set $(A^t, b^t) = \sum_i w_i^t (A_i, b_i)$
  if Oracle $\mathcal{O}(A^t, b^t)$ declares infeasibility **then**
    return Problem is infeasible
  else
    Obtain X^t using Oracle $\mathcal{O}(A^t, b^t)$
    Set $M(i, X^t) = \text{Tr}(A_i X^t) - b_i$
    Set $\rho = \max_i M(i, X^t)$
    Set $\boldsymbol{w}^{t+1}$=MultUpdate($\boldsymbol{w}^t, M, \rho, \epsilon$)
  end if
  Set $t = t + 1$
until $t > T$
return $\overline{X} = \sum_t X^t / T$

function $\boldsymbol{w}^{t+1}$=MultUpdate($\boldsymbol{w}^t, M, \rho, \epsilon$)
Set $\delta = \min\{\frac{\epsilon}{4\rho}, \frac{1}{2}\}$
for all $1 \leq i \leq m$ **do**
  if $M(i, X^t) \geq 0$ **then**
    $w_i^{t+1} = w_i^t (1 - \delta)^{M(i, X^t)/\rho}$
  else
    $w_i^{t+1} = w_i^t (1 + \delta)^{-M(i, X^t)/\rho}$
  end if
end for

4.1. Rank Minimization via Multiplicative Weights Update

In this section we present an approach to RMP based on the generalized experts (GE) framework described in Section 2.1. To adapt the GE framework for the RMP problem, we first need to select a set of experts, a set of events and the associated penalties. We associate each RMP constraint $\text{Tr}(A_i X) \geq b_i$ with an expert and let the events correspond to elements of $\mathcal{C}$. The penalty for expert i corresponding to the i-th constraint and event X is then given by $\text{Tr}(A_i X) - b_i$. Note that rather than rewarding a satisfied constraint, we penalize it. This strategy is motivated by the work of (Plotkin et al., 1991; Arora et al., 2005a) and is similar to boosting, where a distribution is skewed towards an example for which the current hypothesis made an incorrect prediction.

We assign weight w_i^t to the i-th expert in the t-th iteration, and initialize the weights $w_i^1 = 1$, for all i. In the t-th iteration we query the oracle $\mathcal{O}$ with $(A^t, b^t) = \sum_i w_i^t (A_i, b_i)$ to obtain a solution $X^{t+1} \in \mathcal{C}$. We then use the Multiplicative Weights Update algorithm as described in function MultUpdate of Algorithm 1 to compute the weights

w_i^{t+1} for the $(t+1)$-st iteration. Algorithm 1 describes our multiplicative update based algorithm for RMP. In the following theorem we prove approximation guarantees for the solution output by Algorithm 1.

Theorem 4.1. *Given the existence of an oracle $\mathcal{O}$ to solve the problem* (4)*, Algorithm 1 outputs an $(O(\frac{\Delta^2 D^2 \log n}{\epsilon^2}), \epsilon)$-approximate solution to* RMP.

Proof. Observe that, if the oracle declares infeasibility at any time step t, the original problem is also infeasible. Hence, we assume that the oracle returns a feasible point X^t at time-step t, for all $1 \leq t \leq T$.

Now, $|\text{Tr}(A_i X) - b_i| \leq \|A_i\|_F \|X\|_F + |b_i| \leq \Delta D$. Thus, the penalties $\text{Tr}(A_i X) - b_i$ lie in the interval $[-\Delta D, \Delta D]$. Since Algorithm 1 uses multiplicative updates to update the weights[2] as in Theorem 2.1, for $T = 16(\Delta D)^2 \log n/\epsilon^2$, we have

$$\frac{\sum_t \sum_j p_j^t [A_j X^t - b_j]}{T} \leq \epsilon + \frac{\sum_t [A_i X^t - b_i]}{T}, \ \forall i,$$

where $p_j^t = w_j^t / \sum_l w_l^t$. Since $\text{Tr}(A^t X^t) \geq b^t), \ \forall t$, the $LHS \geq 0$. Thus, for $\overline{X} = \sum_t X^t/T$ we have

$$\text{Tr}(A_i \overline{X}) \geq b_i - \epsilon, \quad \forall i. \tag{5}$$

We now bound the rank of $\overline{X}$ compared to the optimal value. Let t be such that X_t has the highest rank, say k, among $X_1, \ldots, X_T$. Then, $k^* \geq k$, as for a particular convex combination of (A_i, b_i) the minimum rank possible was k. Thus, $\text{rank}(\overline{X}) \leq kT = O(\frac{(\Delta^2 \cdot D^2 \log n) k^*}{\epsilon^2})$. Using (5) we have that $\overline{X} \in \mathbb{F}((A_1, \ldots, A_m), b - \epsilon \mathbf{1}, \mathcal{C})$. Thus, by Definition 3.1 $\overline{X}$ is an $(O(\frac{\Delta^2 D^2 \log n}{\epsilon^2}), \epsilon)$-approximate solution to RMP. □

The running time of Algorithm 1 is $O(\frac{\Delta^2 D^2 \log n}{\epsilon^2}(T_{\mathcal{O}} + mn^2))$, where $T_{\mathcal{O}}$ denotes the oracle's running time.

4.2. Rank Minimization via OCP

In this section, we present a novel application of online convex programming described in Section 2.2 to obtain an approximate solution to RMP. The intuition behind this approach is similar to that of Section 4.1; in fact this approach can be viewed as a generalization of the approach of Section 4.1.

In the OCP framework one generally associates the convex set K with a feasible region and the cost functions with penalty functions. In our application of OCP to RMP we flip this view and choose K to be the space of convex combinations of the constraints and associate cost functions with feasible points of RMP. In particular, we set $K \subseteq \mathbb{R}^{n \times n} \times \mathbb{R}$ to be the convex hull of $(A_1, b_1), \ldots, (A_m, b_m)$, i.e.,

$$K = \left\{ \sum_i \lambda_i (A_i, b_i) : \sum_i \lambda_i = 1, \lambda_i \geq 0 \ \forall i \right\}.$$

Given a matrix X, we define a cost function $f_X : K \to \mathbb{R}$ by $f_X(A, b) = \text{Tr}(AX) - b$.

We initialize $A^1 = \sum_i A_i/m$ and $b^1 = \sum_i b_i/m$. Given $(A^t, b^t) \in K$ for the t-th iteration, we query the oracle $\mathcal{O}$ with $(A, b) = (A^t, b^t)$ to obtain a solution $X^t \in \mathcal{C}$. We then set the cost function $f^t(A, b) = f_{X^t}(A, b) = \text{Tr}(AX^t) - b$ and use the OCP algorithm (Zinkevich, 2003) as described in function GIGA of Algorithm 2 to compute (A^{t+1}, b^{t+1}) for the $(t+1)$-st iteration. Algorithm 2 describes our OCP based algorithm for RMP. In the following theorem we prove approximation guarantees for the output of Algorithm 2.

Algorithm 2 RMP-OCP (Online Convex Programming)

Require: Constraints (A_i, b_i), $1 \leq i \leq m$, ϵ
Require: Oracle $\mathcal{O}(A, b)$ which solves $\min \text{rank}(X) \quad s.t. \quad \text{Tr}(AX) \geq b, \ X \in \mathcal{C}$
Initialize: $A^1 = \frac{\sum_i A_i}{m}$ and $b^1 = \frac{\sum_i b_i}{m}$, $t = 1$
Set $K = \{\sum_i \lambda_i (A_i, b_i) : \sum_i \lambda_i = 1, \lambda_i \geq 0 \ \forall i\}$
repeat
  if Oracle $O(A^t, b^t)$ declares infeasibility **then**
    return Problem is infeasible
  else
    Obtain X^t using Oracle $\mathcal{O}(A^t, b^t)$
    Define function $f^t(A, b) = \text{Tr}(AX^t) - b$
    Set (A^{t+1}, b^{t+1})=GIGA($(A^t, b^t), f^t(A, b), K, t$)
  end if
  Set $t = t + 1$
until $t > T$
return $\overline{X} = \sum_t X^t/T$

function $\boldsymbol{z^{t+1}}$=GIGA($\boldsymbol{z^t}, f^t(\boldsymbol{z}), K, t$)
Set $\eta_t = \frac{\Delta}{2D\sqrt{t}}$
Set $\boldsymbol{z^{t+1}} = \Pi_K \left(\boldsymbol{z^t} - \eta_t \nabla f^t(\boldsymbol{z^t}) \right)$, where Π_K represents the orthogonal projection onto K

Theorem 4.2. *Given the existence of an oracle $\mathcal{O}$ to solve the problem* (4)*, Algorithm 2 outputs an $(O(\frac{\Delta^2 D^2}{\epsilon^2}), \epsilon)$-approximate solution to* RMP.

Proof. As in Theorem 4.1 we assume that the oracle returns a feasible point at all time steps. Note that using the terminology of Theorem 2.2, $G = \max_{z \in K, t \in \{1, \ldots\}} \| \bigtriangledown$

[2] Our updates are slightly different from those of (Arora et al., 2005b) in that we adaptively choose the width parameter ρ. However, the analysis of (Arora et al., 2005b) is applicable for these updates as well.

$f^t(z)\| \leq D$ and $\|K\| \leq \Delta$. Thus, using Theorem 2.2 we have

$$\sum_{t=1}^{T}(\text{Tr}(A^tX^t)-b^t) \leq \min_{(A,b)\in K}\sum_{t=1}^{T}(\text{Tr}(AX^t)-b)+\Delta D\sqrt{T}.$$

Note that the above LHS ≥ 0 since oracle returns a feasible $X^t,\ \forall t$. Thus, for $T = \Delta^2D^2/\epsilon^2$ and $\overline{X} = \sum_t X^t/T$,

$$\text{Tr}(A\overline{X}) \geq b - \epsilon, \tag{6}$$

for all $(A, b) \in K$. In particular, we have for every i, $\text{Tr}(A_i\overline{X}) \geq b_i - \epsilon$. We now bound the rank of $\overline{X}$ compared to the optimal value. Let t be such that X_t has the highest rank, say k, among $X_1, \ldots, X_T$. Then, we must have $k^* \geq k$, and so we have $\text{rank}(\overline{X}) \leq kT \leq O((\Delta D)^2k^*/\epsilon^2)$. Also, from (6) we have that $\overline{X} \in \mathbb{F}((A_1, \ldots, A_m), b - \epsilon\mathbf{1}, \mathcal{C})$. Thus by Definition 3.1, $\overline{X}$ is a $(O((\Delta^2D^2)/\epsilon^2), \epsilon)$-approximate solution to RMP. □

The running time of Algorithm 2 is $O(\frac{\Delta^2D^2}{\epsilon^2}(T_{\mathcal{O}}+T_{OCP}+mn^2))$, where $T_{\mathcal{O}}$ denotes the running time of the oracle, and T_{OCP} denotes the time taken in each round by the GIGA algorithm of Theorem 2.2.

4.3. Discussion

Oracle: The oracle for solving problem (4) plays a crucial role in both our approaches. As discussed previously, for typical cases of $\mathcal{C}$, like the unit ball under an L_p or Frobenius norm etc., (4) can be solved by the singular value decomposition of A. Further, in the case when the set $\mathcal{C}$ involves a quadratic or ellipsoid constraint we can use the S-procedure (Rockafellar, 1970) to solve (4).

Comparison of the approaches: Our approach to RMP based on Multiplicative Weights Update has a slightly weaker guarantee than the approach based on OCP. This is also confirmed by our experiments where OCP gives better results than the MW approach. However, the MW approach is computationally less intensive as the approach based on OCP involves a projection onto the convex set K. Thus, MW can be used for large scale problems.

Limitations: A drawback of our methods is the dependence on Δ, ϵ in the bounds of Theorems 4.1 and 4.2. This limits the applicability of our methods to problems, such as NNMA, with a large number of non-negativity constraints where the ratio $\frac{\Delta}{\epsilon}$ is typically large. However, our algorithms can be used as a heuristic for such problems and can be used to initialize other methods which require a good low-rank solution for initialization. Also, the lower bounds for the experts framework and boosting suggest that the dependence on Δ, ϵ in our bounds may be optimal for the general RMP problem (Arora et al., 2005b).

5. Low-rank Kernel Learning

In this section we apply both our rank minimization algorithms to the problem of low-rank kernel learning, which involves finding a low-rank positive semi-definite (p.s.d.) matrix that satisfies linear constraints typically derived from labeled data. Due to the rank constraint, this problem is non-convex and is in general hard to solve. As described below, both our online learning approaches can be applied naturally to this problem. We provide provable guarantees on the rank of the obtained kernel.

Formally, the low-rank kernel learning problem can be cast as the following optimization problem:

$$\begin{aligned} \min_K \quad & \|K - K_0\|_F \\ \text{s.t.} \quad & \text{Tr}(S_iK) \leq \ell, \quad \forall\ 1 \leq i \leq |\mathcal{S}|, \\ & \text{Tr}(D_jK) \geq u, \quad \forall\ 1 \leq j \leq |\mathcal{D}|, \\ & \text{rank}(K) \leq r, \quad K \succeq 0, \end{aligned} \tag{7}$$

where $\mathcal{S}$ is a set of pairs of points from the same class that are constrained to have distance less than ℓ. Similarly, $\mathcal{D}$ is a set of pairs of points from different classes that are constrained to have distance greater than u, with $\ell \ll u$. For a similarity constraint matrix S_i, $S_i(i_1, i_1) = S_i(i_2, i_2) = 1$, $S_i(i_1, i_2) = S_i(i_2, i_1) = -1$ and all other entries 0. The dissimilarity constraint matrices D_j can be constructed similarly. Assuming $\|K_0\|_F = 1$, (7) can be reformulated as:

$$\begin{aligned} \min_K \quad & \text{rank}(K) \\ \text{s.t.} \quad & \text{Tr}(S_iK) \leq \ell\ \forall i, \quad \text{Tr}(D_jK) \geq u\ \forall j, \\ & \text{Tr}(KK_0) \geq \beta, \quad \|K\|_F \leq 1, \quad K \succeq 0, \end{aligned} \tag{8}$$

where β is a function of r and can be computed using binary search. Note that (8) is a special case of RMP with the convex set $\mathcal{C}$ being the intersection of the p.s.d. cone and the unit Frobenius ball. Hence, we can use RMP-MW and RMP-OCP to solve (8). Given (A, b) the oracle for both the methods solves:

$$\min_K \text{rank}(K) :\ \text{Tr}(AK) \geq b, \|K\|_F \leq 1,\ K \succeq 0. \tag{9}$$

Let $A = U\Sigma U^T$ be the eigenvalue decomposition of A, and let Λ be a diagonal matrix with just the positive entries of Σ. Then the minimum k s.t. $\sqrt{\sum_{i=1}^{k}\Lambda(i,i)^2} \geq b$ is the solution to (9). This follows from elementary linear algebra. Note that for the oracle solving (9), $T_{\mathcal{O}} = O(n^3)$.

Now, $D = 1$ and $\Delta = O(1 + l^2 + u^2)$ as $\|S_i\|_F = \|D_j\|_F = 2$. Using Theorem 4.1, the RMP-MW algorithm obtains a solution with rank $r \leq O(\frac{1+u^2+l^2}{\epsilon^2}\log n)r^*$ where r^* is the optimal rank. Similarly, RMP-OCP obtains an $\left(O(\frac{1+u^2+l^2}{\epsilon^2}), \epsilon\right)$-approximate solution. In Section 6.2, we present empirical results for RMP-MW and RMP-OCP algorithms on some standard UCI datasets.

6. Experimental Results

We empirically evaluate and compare our algorithms to existing methods for general RMP as well as low-rank kernel learning. For general RMP, we use synthetic examples to compare our methods against the trace-norm heuristic (Recht et al., 2007) and the log-det heuristic (Fazel et al., 2001). The trace-norm heuristic relaxes the rank objective to the trace-norm of the matrix, which is given by the sum of its singular values. Note that the trace-norm of a matrix is a convex function. The log-det heuristic relaxes the rank objective to the log of the determinant of the matrix. For the application of RMP to low-rank kernel learning, we use standard UCI datasets. All the presented results represent the average over 20 runs.

6.1. Synthetic Datasets

First we use synthetic datasets by generating random matrices $A_i \in \mathbb{S}_n$, where $\mathbb{S}_n$ is the set of $n \times n$ symmetric matrices. We also generate a random positive semi-definite matrix $X_0 \in \mathbb{S}_n$ with $\|X_0\|_F \leq 1$, and use the obtained X_0 to generate constraints $\text{Tr}(A_i X) \geq b_i = \text{Tr}(A_i X_0)$. The convex set $\mathcal{C}$ is fixed to be the intersection of the p.s.d cone and the unit ball under the Frobenius norm. We fix the number of constraints to be 200 and the tolerance ϵ for RMP-MW and RMP-OCP to be 5%. We use SeDuMi to implement the trace-norm and log-det heuristics.

In Table 1, we compare the ranks of the solutions obtained by our algorithms against the ones obtained by the trace-norm and log-det heuristics. For small n, both trace-norm and log-det heuristic perform better than RMP-MW and RMP-OCP. Note that since the constraint matrices A_i are random, they satisfy (with high probability) the restricted isometry property used in the analysis of (Recht et al., 2007). However, RMP-OCP outperforms trace-norm heuristic for large n (Table 1, $n = 100$) and RMP-MW performs comparably. We attribute this phenomenon to the Frobenius norm constraint for which the theoretical guarantees of (Recht et al., 2007) are not applicable. Also, both trace-norm and log-det heuristic scale poorly with the problem size and fail to obtain a result in reasonable time even for moderately large n. In contrast, both our algorithms scale well with n, with RMP-MW in particular able to solve problems of sizes up to $n = 5000$.

6.2. Low-rank Kernel Learning

We evaluate the performance of our methods applied to the problem of low-rank kernel learning, as described in Section 5, for k-NN classification on standard UCI datasets. We use two-fold cross validation with $k = 5$. The lower and upper bounds for the similarity and dissimilarity constraints (l, u) are set using the 30-th and 70-th percentiles of the observed distribution of distances between pairs of points. We randomly select a set of $40c^2$ pairs of points for constraints, where c is the number of classes in the dataset. We run both RMP-MW and RMP-OCP for $T = 50$ iterations. Empirically our algorithms significantly outperform the theoretical rank guarantees of Theorems (4.1) and (4.2).

Method\n	50	75	100	200	300
RMP-MW	23.25	11.25	7.3	2	2
RMP-OCP	12.8	7.5	5.3	2	2
Trace-norm	6.8	6.7	6.5	-	-
LogDet	5	4.2	4.0	-	-

Table 1. Rank of the matrices obtained by different RMP methods for varying size of the constraint matrices (n). The number of constraints generated (m) is fixed to be 200. A "-" represents that the method could not find a solution within 3 hours on a 2.6GHz Pentium 4 machine. Note that for large problem sizes, both the trace-norm and the log-det heuristics are not computationally viable. Both our approaches outperform the trace-norm heuristic as the problem size increases.

Dataset\Method	GK	MW	OCP	BK
Musk	80.80 (476)	93.11 (44.1)	**98.15** (61.2)	81.51 (61.2)
Heart	77.44 (267)	91.05 (46.8)	**91.13** (39.5)	83.91 (39.5)
Ionosphere	90.34 (350)	**91.26** (40)	91.17 (27.9)	90.67 (27.9)
Cancer	90.12 (569)	93.14 (82)	91.46 (94)	**93.38** (94)
Scale	66.34 (607)	**73.78** (146)	72.46 (91)	72.11 (91)

Table 2. Accuracies for 5-Nearest Neighbor classification using kernels obtained by different methods. Numbers in parentheses represent the rank of the obtained solution. GK represents Gaussian Kernel ($\sigma = 0.1$), MW represents RMP-MW, OCP represents RMP-OCP and BK represents BurgKernel(Kulis et al., 2006). Overall, RMP-OCP obtains the best accuracy.

Table 2 shows the accuracies achieved by the baseline Gaussian kernel (with $\sigma = 0.1$), RMP-MW, RMP-OCP and the Burg divergence (also called as LogDet divergence) based low-rank kernel learning algorithm (BurgKernel) of (Kulis et al., 2006). It can be seen from the table that both RMP-MW and RMP-OCP obtain a significantly lower rank kernel than the baseline Gaussian kernel. Further, RMP-MW and RMP-OCP achieve a substantially higher accuracy than the Gaussian kernel. Our algorithms also achieve a substantial improvement in accuracy over the BurgKernel method. Note that we iterate our algorithms for fewer iterations compared to the ones suggested by the theoretical bounds, hence few of the constraints maybe unsatisfied. This suggests that these unsatisfied constraints maybe noisy constraints and have small effect on the generalization error. We leave further investigation into gener-

alization error of our methods as a topic for future research.

Note that the BurgKernel method needs to be initialized with a low-rank kernel. Typically, a few top eigenvectors of the baseline kernel are used for this initialization. However, selecting only a few top eigenvectors can lead to a poor initial kernel, especially if the rank of the initial kernel is high. This can further lead to poor accuracy for the BurgKernel method, as indicated by our experiments. Instead, the kernels obtained by our algorithms could be used to *initialize* the BurgKernel algorithm. For example, for the case of the Heart dataset, initialization of BurgKernel algorithm with the low-rank solution obtained by RMP-OCP method achieves an accuracy of 94.29 compared to 83.91 achieved when initialized with the top eigenvectors of the baseline Gaussian kernel. Note that this also improves upon the accuracy achieved by RMP-MW and RMP-OCP.

7. Conclusion

In this paper, we address the general problem of rank minimization over polyhedral sets and in particular the problem of low-rank kernel learning. We show that the problem is hard to approximate within a factor of $2^{\log^{1-\epsilon}\Delta}$ (see Theorem 3.1). Further, we introduce a relaxed notion of approximation and present two novel approaches for solving RMP with provable guarantees. Our first approach is based on the multiplicative weights update framework and provides an $(O(\frac{\Delta^2 D^2}{\epsilon^2}\log n), \epsilon)$-approximate solution. Our second approach is based on online convex programming and provides a tighter bound of $O(\frac{\Delta^2 D^2}{\epsilon^2})$ for the rank of the obtained matrix.

For future work, it would be interesting to see if the hardness of approximation factor of Theorem 3.1 can be improved; we believe it can be improved to $O(\Delta^2)$. Another question of interest is whether the dependence on ϵ in the bounds of Theorems 4.1 and 4.2 can be improved.

The regret bounds of (Zinkevich, 2003) were improved in (Hazan et al., 2006). However, the algorithms of (Hazan et al., 2006) require stronger convexity properties which are not satisfied in our application of OCP to RMP. It would be interesting to see if the linear constraints in RMP can be perturbed to satisfy the strong convexity properties, so that the improved regret bounds of (Hazan et al., 2006) can be used to achieve better bounds for RMP.

Our algorithms to RMP are motivated from an online learning perspective. However, for an optimization problem such as RMP an understanding of the algorithms from an optimization perspective would be highly desirable. In particular, intuitively there seems to be a correspondence between our methods and a primal-dual approach but we were unable to obtain a rigorous connection. We believe that such an understanding would be of importance in obtaining new applications of the online learning approach to solving optimization problems.

References

Amaldi, E., & Kann, V. (1998). On the approximability of minimizing non-zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science, 209*, 237–260.

Arora, S., Hazan, E., & Kale, S. (2005a). Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. *FOCS* (pp. 339–348).

Arora, S., Hazan, E., & Kale, S. (2005b). Multiplicative weights method: a meta-algorithm and its applications. *Survey, http://www.cs.princeton.edu/ arora/pubs/MWsurvey.pdf*.

Bach, F. R., & Jordan, M. I. (2005). Predictive low-rank decomposition for kernel methods. *ICML* (pp. 33–40).

Barvinok, A. (2002). *A course in convexity*. American Mathematical Society.

Candès, E. J., & Tao, T. (2005). Decoding by linear programming. *IEEE Transactions on Information Theory, 51*, 4203–4215.

Fazel, M., Hindi, H., & Boyd, S. (2001). A rank minimization heuristic with application to minimum order system approximation. *American Control Conference, Arlington, Virginia*.

Fazel, M., Hindi, H., & Boyd, S. (2003). Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices. *American Control Conference, Denver, Colorado*.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci., 55*, 119–139.

Hazan, E., Kalai, A., Kale, S., & Agarwal, A. (2006). Logarithmic regret algorithms for online convex optimization. *COLT* (pp. 499–513).

Kalai, A. T., & Vempala, S. (2005). Efficient algorithms for online decision problems. *J. Comput. Syst. Sci., 71*, 291–307.

Kim, D., Sra, S., & Dhillon, I. S. (2007). Fast Newton-type methods for the least squares nonnegative matrix approximation problem. *SDM* (pp. 343–354).

Kulis, B., Sustik, M., & Dhillon, I. S. (2006). Learning low-rank kernel matrices. *ICML* (pp. 505–512).

Lee, D. D., & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. *NIPS* (pp. 556–562).

Littlestone, N., & Warmuth, M. K. (1989). The weighted majority algorithm. *FOCS* (pp. 256–261).

Meka, R., Jain, P., Caramanis, C., & Dhillon, I. S. (2008). *Rank minimization via online learning* (Technical Report TR-08-23). The Univ. of Texas at Austin, Dept. of Comp. Sci.

Plotkin, S. A., Shmoys, D. B., & Tardos, E. (1991). Fast approximation algorithms for fractional packing and covering problems. *IEEE Symposium on Foundations of Computer Science* (pp. 495–504).

Recht, B., Fazel, M., & Parrilo, P. (2007). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *Preprint, http://www.optimization-online.org/DB_HTML/2007/06/1707.html*.

Rockafellar, R. T. (1970). *Convex analysis*. Princeton University Press.

Weinberger, K. Q., Sha, F., & Saul, L. K. (2004). Learning a kernel matrix for nonlinear dimensionality reduction. *ICML*.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *ICML* (pp. 928–936).

An Analysis of Reinforcement Learning with Function Approximation

Francisco S. Melo FMELO@CS.CMU.EDU
Carnegie Mellon University, Pittsburgh, PA 15213, USA

Sean P. Meyn MEYN@CONTROL.CSL.UIUC.EDU
Coordinated Science Lab, Urbana, IL 61801, USA

M. Isabel Ribeiro MIR@ISR.IST.UTL.PT
Institute for Systems and Robotics, 1049-001 Lisboa, Portugal

Abstract

We address the problem of computing the optimal Q-function in Markov decision problems with infinite state-space. We analyze the convergence properties of several variations of Q-learning when combined with function approximation, extending the analysis of TD-learning in (Tsitsiklis & Van Roy, 1996a) to stochastic control settings. We identify conditions under which such approximate methods converge with probability 1. We conclude with a brief discussion on the general applicability of our results and compare them with several related works.

1. Introduction

Convergence of Q-learning with function approximation has been a long standing open question in reinforcement learning (Sutton, 1999). In general, value-based reinforcement learning (RL) methods for optimal control behave poorly when combined with function approximation (Baird, 1995; Tsitsiklis & Van Roy, 1996a). In this paper, we address this problem by analyzing the convergence of Q-learning when combined with linear function approximation. We identify a set of conditions that imply the convergence of this approximation method with probability 1 (w.p.1), when a fixed learning policy is used, and provide an interpretation of the resulting approximation as the fixed point of a Bellman-like operator. This motivates the analysis of several variations of Q-learning when combined with linear function approximation. In particular, we study a variation of Q-learning using importance sampling and an on-policy variant of Q-learning (SARSA).

The paper is organized as follows. We start in Section 2 by describing Markov decision problems. We proceed with our analysis of the Q-learning algorithm and its variants, and produce our main results in Section 3. We also compare our results with other related works in the RL literature. We conclude with some further discussion in Section 4.

2. Markov Decision Problems

Let $(\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be a Markov decision problem (MDP) with a compact state-space $\mathcal{X} \subset \mathbb{R}^p$ and a finite action set $\mathcal{A}$. The action-dependent kernel P_a defines the transition probabilities for the underlying controlled Markov chain $\{X_t\}$ as

$$\mathbb{P}\left[X_{t+1} \in U \mid X_t = x, A_t = a\right] = \mathsf{P}_a(x, U),$$

where U is any measurable subset of $\mathcal{X}$. The $\mathcal{A}$-valued process $\{A_t\}$ represents the control process: A_t is the control action at time instant t.[1] Solving the MDP consists in determining the control process $\{A_t\}$ maximizing the expected total discounted reward

$$V\big(\{A_t\}, x\big) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = x\right],$$

where $0 \leq \gamma < 1$ is a discount-factor and $R(x, a)$ represents a random "reward" received for taking action $a \in \mathcal{A}$ in state $x \in \mathcal{X}$. For simplicity of notation, we consider a bounded deterministic function $r : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \longrightarrow \mathbb{R}$ assigning a reward $r(x, a, y)$ every time a transition from x to y occurs after taking

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] We take the control process $\{A_t\}$ to be adapted to the σ-algebra induced by $\{X_t\}$.

action a. This means that

$$\mathbb{E}\left[R(x,a)\right] = \int_{\mathcal{X}} r(x,a,y)\mathsf{P}_a(x,dy).$$

The *optimal value function* V^* is defined for each state $x \in \mathcal{X}$ as

$$V^*(x) = \max_{\{A_t\}} V\big(\{A_t\}, x\big) = \\ = \max_{\{A_t\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = x\right]$$

and verifies the Bellman optimality equation

$$V^*(x) = \max_{a\in\mathcal{A}} \int_{\mathcal{X}} \big[r(x,a,y) + \gamma V^*(y)\big]\mathsf{P}_a(x,dy). \quad (1)$$

$V^*(x)$ represents the expected total discounted reward received along an optimal trajectory starting at state x. We can also define the optimal Q-function Q^* as

$$Q^*(x,a) = \int_{\mathcal{X}} \big[r(x,a,y) + \gamma V^*(y)\big]\mathsf{P}_a(x,dy), \quad (2)$$

representing the expected total discounted reward along a trajectory starting at state x obtained by choosing a as the first action and following the optimal policy thereafter. The control process $\{A_t\}$ defined as

$$A_t = \arg\max_{a\in\mathcal{A}} Q^*(X_t, a), \quad \forall t,$$

is optimal in the sense that $V(\{A_t\}, x) = V^*(x)$ and defines a mapping $\pi^* : \mathcal{X} \to \mathcal{A}$ known as the *optimal policy*. The optimal policy determines the optimal decision rule for a given MDP.

More generally, a (Markov) *policy* is any mapping π_t defined over $\mathcal{X} \times \mathcal{A}$ generating a control process $\{A_t\}$ verifying, for all t,

$$\mathbb{P}\left[A_t = a \mid X_t = x\right] = \pi_t(x,a), \quad \forall t.$$

We write $V^{\pi_t}(x)$ instead of $V(\{A_t\}, x)$ if the control process $\{A_t\}$ is generated by policy π_t. A policy π_t is *stationary* if it does not depend on t and *deterministic* if it assigns probability 1 to a single action in each state and is thus represented as a map $\pi_t : \mathcal{X} \to \mathcal{A}$ for every t. Notice that the optimal control process can be obtained from the optimal (stationary, deterministic) policy π^*, which can in turn be obtained from Q^*. Therefore, the optimal control problem is solved once the function Q^* is known for all pairs (x,a).

Now given any real function q defined over $\mathcal{X} \times \mathcal{A}$, we define the Bellman operator

$$(\mathbf{H}q)(x,a) = \int_{\mathcal{X}} \big[r(x,a,y) + \gamma \max_{u\in\mathcal{A}} q(y,u)\big]\mathsf{P}_a(x,dy). \quad (3)$$

The function Q^* in (2) is the fixed-point of $\mathbf{H}$ and, since this operator is a contraction in the sup-norm, a fixed-point iteration can be used to determine Q^* (at least theoretically).

2.1. The Q-Learning Algorithm

We previously suggested that a fixed-point iteration could be used to determine the function Q^*. In practice, this requires two important conditions:

- The kernel P and the reward function r are known;
- The successive estimates for Q^* can be represented compactly and stored in a computer with finite memory.

If P and/or r are not known, a fixed-point iteration using $\mathbf{H}$ is not possible. To solve this problem, Watkins proposed in 1989 the *Q-learning algorithm* (Watkins, 1989). Q-learning proceeds as follows: consider a MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ and suppose that $\{x_t\}$ is an infinite sample trajectory of the underlying Markov chain obtained with some policy π_t. The corresponding sample control process is denoted as $\{a_t\}$ and the sequence of obtained rewards as $\{r_t\}$. Given any initial estimate Q_0, Q-learning successively updates this estimate using the rule

$$Q_{t+1}(x,a) = Q_t(x,a) + \alpha_t(x,a)\Delta_t, \quad (4)$$

where $\{\alpha_t\}$ is a step-size sequence and Δ_t is the temporal difference at time t,

$$\Delta_t = r_t + \gamma \max_{b\in\mathcal{A}} Q_t(x_{t+1}, b) - Q_t(x_t, a_t). \quad (5)$$

If both $\mathcal{X}$ and $\mathcal{A}$ are finite sets, each estimate Q_t is simply a $|\mathcal{X}| \times |\mathcal{A}|$ matrix and can be represented explicitly in a computer. In that case, the convergence of Q-learning and several other related algorithms (such as TD(λ) or SARSA) has been thoroughly studied (see, for example, (Bertsekas & Tsitsiklis, 1996) and references therein). However, if either $\mathcal{X}$ or $\mathcal{A}$ are infinite or very large, explicitly representing each Q_t becomes infeasible and some form of compact representation is needed (*e.g.*, using function approximation). In this paper, we address how several RL methods such as Q-learning and SARSA can be combined with function approximation and still retain their main convergence properties.

3. Reinforcement Learning with Linear Function Approximation

In this section, we address the problem of determining the optimal Q-function for MDPs with infinite state-space $\mathcal{X}$. Let $\mathcal{Q} = \{Q_\theta\}$ be a family of real-valued

functions defined in $\mathcal{X} \times \mathcal{A}$. It is assumed that the function class is linearly parameterized, so that $\mathcal{Q}$ can be expressed as the linear span of a *fixed set* of M linearly independent functions $\phi_i : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. For each M-dimensional parameter vector $\theta \in \mathbb{R}^M$, the function $Q_\theta \in \mathcal{Q}$ is defined as,

$$Q_\theta(x,a) = \sum_{i=1}^{M} \phi_i(x,a)\theta(i) = \phi^\top(x,a)\theta,$$

where $^\top$ represents the transpose operator. We will also denote the above function by $Q(\theta)$ to emphasize the dependence on θ over the dependency on (x,a).

Let π be a fixed stochastic, stationary policy and suppose that $\{x_t\}$, $\{a_t\}$ and $\{r_t\}$ are sampled trajectories of states, actions and rewards obtained from the MDP using policy π. In the original Q-learning algorithm, the Q-values are updated according to (4). The temporal difference Δ_t can be interpreted as a 1-step estimation error with respect to the optimal function Q^*. The update rule in Q-learning "moves" the estimates Q_t closer to the desired function Q^*, minimizing the expected value of Δ_t.

In our approximate setting, we apply the same underlying idea to obtain the update rule for approximate Q-learning:

$$\begin{aligned} \theta_{t+1} &= \theta_t + \alpha_t \nabla_\theta Q_\theta(x_t,a_t)\Delta_t \\ &= \theta_t + \alpha_t \phi(x_t,a_t)\Delta_t, \end{aligned} \tag{6}$$

where, as above, Δ_t is the temporal difference at time t defined in (5). Notice that (6) updates θ_t using the temporal difference Δ_t as the error. The gradient $\nabla_\theta Q_\theta$ provides the "direction" in which this update is performed.

To establish convergence of the algorithm (6) we adopt an ODE argument, establishing the trajectories of the algorithm to closely follow those of an associated ODE with a globally asymptotically stable equilibrium point. This will require several regularity properties on the policy π and on its induced Markov chain that will, in turn, motivate the study of the on-policy version of the algorithm. This on-policy algorithm can be seen as an extension of SARSA to infinite settings.

3.1. Convergence of Q-Learning

We now proceed by identifying conditions that ensure the convergence of Q-learning with linear function approximation as described by (6). Due to space limitations, we overlook some of the technical details in the proofs that can easily be filled in.

We start by introducing some notation that will greatly simplify the presentation. Given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ with compact state space $\mathcal{X} \subset \mathbb{R}^p$, let $(\mathcal{X}, \mathsf{P}_\pi)$ be the Markov chain induced by a fixed policy π. We assume the chain $(\mathcal{X}, \mathsf{P}_\pi)$ to be uniformly ergodic with invariant probability measure μ_X and the policy π to verify $\pi(x,a) > 0$ for all $a \in \mathcal{A}$ and μ_X-almost all $x \in \mathcal{X}$. We denote by μ_π the probability measure defined for each measurable set $U \subset \mathcal{X}$ and each action $a \in \mathcal{A}$ as

$$\mu_\pi(U \times \{a\}) = \int_U \pi(x,a)\mu_X(dx).$$

Let now $\{\phi_i, i = 1, \ldots, M\}$ be a set of bounded, linearly independent basis functions to be used in our approximate Q-learning algorithm. We denote by Σ_π the matrix defined as

$$\Sigma_\pi = \mathbb{E}_\pi\left[\phi(x,a)\phi^\top(x,a)\right] = \int_{\mathcal{X}\times\mathcal{A}} \phi\,\phi^\top d\mu_\pi$$

Notice that the above expression is well-defined and independent of the initial distribution for the chain, due to our assumption of uniform ergodicity.

For fixed $\theta \in \mathbb{R}^M$ and $x \in \mathcal{X}$, define the set of maximizing actions at x as

$$\mathcal{A}_x^\theta = \{a^* \in \mathcal{A} \mid \phi^\top(x,a^*)\theta = \max_a \phi^\top(x,a)\theta\}$$

and the greedy policy with respect to θ as any policy that, at each state x, assigns positive probability only to actions in $\mathcal{A}_x^\theta$. Finally, let ϕ_x denote the row-vector $\phi^\top(x,a)$, where a is a random action generated according to the policy π at state x; likewise, let ϕ_x^θ denote the row-vector $\phi^\top(x,a_x^\theta)$, where a_x^θ is now any action in $\mathcal{A}_x^\theta$. We now introduce the θ-*dependent* matrix

$$\Sigma_\pi^*(\theta) = \mathbb{E}_\pi\left[\left(\phi_x^\theta\right)^\top \phi_x^\theta\right].$$

By construction, both Σ_π and Σ_π^* are positive definite, since the functions ϕ_i are assumed linearly independent. Notice also the difference between Σ_π and each Σ_π^*: the actions in the definition of the former are taken according to π while in the latter they are taken greedily with respect to a particular θ.

We are now in position to introduce our first result.

Theorem 1 *Let $\mathcal{M}$, π and $\{\phi_i, i = 1, \ldots, M\}$ be as defined above. If, for all θ,*

$$\Sigma_\pi > \gamma^2 \Sigma_\pi^*(\theta) \tag{7}$$

and the step-size sequence verifies

$$\sum_t \alpha_t = \infty \qquad \sum_t \alpha_t^2 < \infty,$$

then the algorithm in (6) *converges w.p.1 and the limit point* θ^* *verifies the recursive relation*

$$Q(\theta^*) = \Pi_{\mathcal{Q}} \mathbf{H} Q(\theta^*),$$

where $\Pi_{\mathcal{Q}}$ *is the orthogonal projection onto* $\mathcal{Q}$.[2]

PROOF We establish the main statement of the theorem using a standard ODE argument.

The assumptions on the chain $(\mathcal{X}, \mathsf{P}_\pi)$ and basis functions $\{\phi_i, i = 1, \ldots, M\}$ and the fact that $\pi(x, a) > 0$ for all $a \in \mathcal{A}$ and μ_X-almost every $x \in \mathcal{X}$ ensure the applicability of Theorem 17 in page 239 of (Benveniste et al., 1990). Therefore, the convergence of the algorithm can be analyzed in terms of the stability of the equilibrium points of the associated ODE

$$\dot{\theta} = \mathbb{E}_\pi \Big[\phi_x^\top \big(r(x, a, y) + \gamma \phi_y^\theta \theta - \phi_x \theta \big) \Big], \qquad (8)$$

where we omitted the explicit dependence of θ on t to avoid excessively cluttering the expression. If the ODE (8) has a globally asymptotically stable equilibrium point, this implies the algorithm (6) to converge w.p.1 (Benveniste et al., 1990). Let then $\theta_1(t)$ and $\theta_2(t)$ be two trajectories of the ODE starting at different initial conditions, and let $\tilde{\theta}(t) = \theta_1(t) - \theta_2(t)$. From (8), we get

$$\frac{d}{dt} \|\tilde{\theta}\|_2^2 = -2 \tilde{\theta}^\top \Sigma_\pi \tilde{\theta} + 2\gamma \mathbb{E}_\pi \Big[\big(\phi_x \tilde{\theta} \big) \big(\phi_y^{\theta_1} \theta_1 - \phi_y^{\theta_2} \theta_2 \big) \Big].$$

Notice now that, from the definition of $\phi_y^{\theta_1}$ and $\phi_y^{\theta_2}$,

$$\phi_y^{\theta_1} \theta_2 \leq \phi_y^{\theta_2} \theta_2 \qquad \phi_y^{\theta_2} \theta_1 \leq \phi_y^{\theta_1} \theta_1.$$

Taking this into account and defining the sets $S_+ = \{(x, a) \mid \phi^\top(x, a) \tilde{\theta} > 0\}$ and $S_- = \mathcal{X} \times \mathcal{A} - S_+$, the previous expression becomes

$$\frac{d}{dt} \|\tilde{\theta}\|_2^2 \leq -2 \tilde{\theta}^\top \Sigma_\pi \tilde{\theta} + 2\gamma \mathbb{E}_\pi \Big[\big(\phi_x \tilde{\theta} \big) \big(\phi_y^{\theta_1} \tilde{\theta} \big) \mathbb{I}_{S_+} \Big] + 2\gamma \mathbb{E}_\pi \Big[\big(\phi_x \tilde{\theta} \big) \big(\phi_y^{\theta_2} \tilde{\theta} \big) \mathbb{I}_{S_-} \Big],$$

where $\mathbb{I}_S$ represents the indicator function for the set S. Applying Hölder's inequality to each of the expectations above, we get

$$\frac{d}{dt} \|\tilde{\theta}\|_2^2 \leq -2 \tilde{\theta}^\top \Sigma_\pi \tilde{\theta} + 2\gamma \sqrt{\mathbb{E}_\pi \Big[(\phi_x \tilde{\theta})^2 \mathbb{I}_{S_+} \Big] \mathbb{E}_\pi \Big[(\phi_x^{\theta_1} \tilde{\theta})^2 \mathbb{I}_{S_+} \Big]} + 2\gamma \sqrt{\mathbb{E}_\pi \Big[(\phi_x \tilde{\theta})^2 \mathbb{I}_{S_-} \Big] \mathbb{E}_\pi \Big[(\phi_x^{\theta_2} \tilde{\theta})^2 \mathbb{I}_{S_-} \Big]}$$

and a few simple computations finally yield

$$\frac{d}{dt} \|\tilde{\theta}\|_2^2 \leq -2 \tilde{\theta}^\top \Sigma_\pi \tilde{\theta} + 2\gamma \sqrt{\tilde{\theta}^\top \Sigma_\pi \tilde{\theta} \max \big(\tilde{\theta}^\top \Sigma_\pi^*(\theta_1) \tilde{\theta}, \tilde{\theta}^\top \Sigma_\pi^*(\theta_2) \tilde{\theta} \big)}.$$

Since, by assumption, $\Sigma_\pi > \gamma^2 \Sigma_\pi^*(\theta)$, we can conclude from the expression above that

$$\frac{d}{dt} \|\tilde{\theta}\|_2^2 < 0.$$

This means, in particular, that $\tilde{\theta}(t)$ converges asymptotically to the origin, *i.e.*, the ODE (8) is globally asymptotically stable. Since the ODE is autonomous (*i.e.*, time-invariant), there exists one globally asymptotically stable equilibrium point for the ODE, that verifies the recursive relation

$$\theta^* = \Sigma_\pi^{-1} \mathbb{E}_\pi \Big[\phi_x \big(r(x, a, y) + \gamma \phi_y^{\theta^*} \theta^* \big) \Big]. \qquad (9)$$

Since Σ_π is, by construction, positive definite, the inverse in (9) is well-defined. Multiplying (9) by $\phi^\top(x, a)$ on both sides yields the desired result. □

It is now important to observe that condition (7) is quite restrictive: since γ is usually taken close to 1, condition (7) essentially requires that, for every θ,

$$\max_{a \in \mathcal{A}} \phi^\top(x, a) \theta \approx \sum_{a \in \mathcal{A}} \pi(x, a) \phi^\top(x, a) \theta.$$

Therefore, such condition will seldom be met in practice, since it implies that the learning policy π is already close to the policy that the algorithm is meant to compute. In other words, the maximization above yields a policy close to the policy used during learning. And, when this is the case, the algorithm essentially behaves like an *on-policy algorithm.*

On the other hand, the above condition can be ensured by considering only a local maximization around the learning policy π. This is the most interesting aspect of the above result: it explicitly relates how much information the learning policy provides about greedy policies, as a function of γ. To better understand this,

[2] The orthogonal projection is naturally defined in the (infinite-dimensional) Hilbert space containing $\mathcal{Q}$ with inner-product given by

$$\langle f, g \rangle = \int_{\mathcal{X} \times \mathcal{A}} f \, g \, d\mu_\pi.$$

notice that each policy π is associated with a particular invariant measure on the induced chain $(\mathcal{X}, \mathsf{P}_\pi)$. In particular, the measure associated with the learning policy may be very different from the one induced by the greedy/optimal policy. Taking into account the fact that γ measures, in a sense, the "importance of the future", Theorem 1 basically states that:

In problems where the performance of the agent greatly depends on future rewards ($\gamma \approx 1$), the information provided by the learning policy can only be "safely generalized" to nearby greedy policies, in the sense of (7). *In problems where the performance of the agent is less dependent on future rewards ($\gamma \ll 1$), the information provided by the learning policy can be safely generalized to more general greedy policies.*

Suppose then that the maximization in the update equation (6) is to be replaced by a local maximization. In other words, instead of maximizing over all actions in $\mathcal{A}$, the algorithm should maximize over a small neighborhood of the learning policy π (in policy space). The difficulty with this approach is that such maximization can be hard to implement. The use of *importance sampling* can readily overcome such difficulty, by making the maximization in policy-space *implicit*. The algorithm thus obtained, which resembles in many aspects the one proposed in (Precup et al., 2001), is described by the update rule

$$\theta_{t+1} = \theta_t + \alpha_t \phi(x_t, a_t) \hat{\Delta}_t, \tag{10}$$

where the modified temporal difference $\hat{\Delta}_t$ is given by

$$\hat{\Delta}_t = r_t + \gamma \sum_b \frac{\pi_\theta(x_{t+1}, b)}{\pi(x_{t+1}, b)} Q_{\theta_t}(x_{t+1}, b) - Q_{\theta_t}(x_t, a_t), \tag{11}$$

where π_θ is, for example, a θ-dependent ε-greedy policy close to the learning policy π.[3] A possible implementation of such algorithm is sketched in Figure 1. We denoted by π_N the behavior policy at iteration N of the algorithm; in the stopping condition for the algorithm, any adequate policy norm can be used.

Algorithm 1 Modified Q-learning.

Require: Initial policy π_0;
1: Initialize $X_0 = x_0$ and set $N = 0$;
2: **for** $t = 0$ until T **do**
3: Sample $A_t \sim \pi_N(x_t, \cdot)$;
4: Sample next-state $X_{t+1} \sim \mathsf{P}_{a_t}(x_t, \cdot)$;
5: $r_t = r(x_t, a_t, x_{t+1})$;
6: Update θ_t according to (10);
7: **end for**
8: Set $\pi_{N+1}(x, a) = \pi_{\theta^*}(x, a)$;
9: $N = N + 1$;
10: **if** $\|\pi_N - \pi_{N-1}\|$ **then**
11: **return** π_N;
12: **else**
13: Goto 2;
14: **end if**

3.2. SARSA with Linear Function Approximation

The analysis in the previous subsection suggests that on-policy algorithms may potentially yield more reliable convergence properties. Such fact has already been observed in (Tsitsiklis & Van Roy, 1996a; Perkins & Pendrith, 2002). In this subsection we thus focus on on-policy algorithms. We analyze the convergence of SARSA when combined with linear function approximation. In our main result, we recover the essence of the result in (Perkins & Precup, 2003), although in a somewhat different setting. The main differences between our work and that in (Perkins & Precup, 2003) are discussed further ahead.

Once again, we consider a family $\mathcal{Q}$ of real-valued functions, the linear span of a fixed set of M linearly independent functions $\phi_i : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, and derive an on-policy algorithm to compute a parameter vector θ^* such that $\phi^\top(x, a)\theta^*$ approximates the optimal Q-function. To this purpose, and unlike what has been done so far, we now consider a *θ-dependent* learning policy π_θ verifying $\pi_\theta(x, a) > 0$ for all θ. In particular, we consider at each time step a learning policy π_{θ_t} that is ε-greedy with respect to $\phi^\top(x, a)\theta_t$ and Lipschitz continuous with respect to θ, with Lipschitz constant C (with respect to some preferred metric). We further assume that, for every fixed θ, the Markov chain $(\mathcal{X}, \mathsf{P}_\theta)$ induced by such policy is uniformly ergodic.

Let then $\{x_t\}$, $\{a_t\}$ and $\{r_t\}$ be sampled trajectories of states, actions and rewards obtained from the MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ using (at each time-step) the θ-dependent policy π_{θ_t}. The update rule for our approximate SARSA algorithm is:

$$\theta_{t+1} = \theta_t + \alpha_t \phi(x_t, a_t) \Delta_t, \tag{12}$$

where Δ_t is the temporal difference at time t,

$$\Delta_t = r_t + \gamma \phi^\top(x_{t+1}, a_{t+1})\theta_t - \phi^\top(x_t, a_t)\theta_t.$$

In order to use the SARSA algorithm above to approximate the optimal Q-function, it is necessary to slowly decay the exploration rate, ε, to zero, while guaran-

[3] Given $\varepsilon > 0$, a policy π is *ε-greedy* with respect to a function $Q_\theta \in \mathcal{Q}$ if, at each state $x \in \mathcal{X}$, it chooses a random action with probability ε and a greedy action $a \in \mathcal{A}_x^\theta$ with probability $(1 - \varepsilon)$.

teeing the learning policy to verify the necessary regularity conditions (namely, Lipschitz continuous w.r.t. θ). However, as will soon become apparent, decreasing the exploration rate to zero will render our convergent result (and other related results) not applicable.

We are now in position to introduce our main result.

Theorem 2 *Let $\mathcal{M}$, π_{θ_t} and $\{\phi_i, i = 1, \ldots, M\}$ be as defined above. Let C be the Lipschitz constant of the learning policy π_θ with respect to θ. Assume that the step-size sequence verifies*

$$\sum_t \alpha_t = \infty \qquad \sum_t \alpha_t^2 < \infty.$$

Then, there is $C_0 > 0$ such that, if $C < C_0$, the algorithm in (12) *converges w.p.1.*

Proof We again use an ODE argument to establish the statement of the theorem.

As before, the assumptions on $(\mathcal{X}, \mathsf{P}_\theta)$ and basis functions $\{\phi_i, i = 1, \ldots, M\}$ and the fact that the learning policy is Lipschitz continuous with respect to θ and verifies $\pi(x, a) > 0$ ensure the applicability of Theorem 17 in page 239 of (Benveniste et al., 1990). Therefore, the convergence of the algorithm can be analyzed in terms of the stability of the associated ODE:

$$\dot{\theta} = \mathbb{E}_\theta\Big[\phi_x^\top \big(r(x, a, y) + \gamma\phi_y\theta - \phi_x\theta\big)\Big]. \qquad (13)$$

Notice that the expectation is taken with respect to the invariant measure of the chain and learning policy, both θ-dependent. To establish global asymptotic stability, we re-write (13) as

$$\dot{\theta}(t) = \mathbf{A}_\theta\theta(t) + \mathbf{b}_\theta$$

where

$$\mathbf{A}_\theta = \mathbb{E}_\theta\left[\phi_x^\top\big(\gamma\phi_y - \phi_x\big)\right]; \quad \mathbf{b}_\theta = \mathbb{E}_\theta\left[\phi_x^\top r(x, a, y)\right].$$

An equilibrium point of (13) must verify $\theta^* = \mathbf{A}_{\theta^*}^{-1}\mathbf{b}_{\theta^*}$ and the existence of such equilibrium point has been established in (de Farias & Van Roy, 2000) (Theorem 5.1). Let $\tilde{\theta}(t) = \theta(t) - \theta^*$. Then,

$$\frac{d}{dt}\|\tilde{\theta}\|_2^2 = 2\tilde{\theta}^\top\big(\mathbf{A}_\theta\theta + \mathbf{b}_\theta\big) = \\ = 2\tilde{\theta}^\top\big(\mathbf{A}_\theta\theta - \mathbf{A}_{\theta^*}\theta^* + \mathbf{b}_\theta - \mathbf{b}_{\theta^*}\big).$$

Let

$$\lambda_A = \sup_\theta \|\mathbf{A}_\theta - \mathbf{A}_{\theta^*}\|_2 \quad \lambda_b = \sup_{\theta\,\neq\theta^*} \frac{\|\mathbf{b}_\theta - \mathbf{b}_{\theta^*}\|_2}{\|\theta - \theta^*\|_2},$$

where the norm in the definition of λ_A is the induced operator norm and the one in the definition of λ_b is the regular Euclidian norm. The previous expression thus becomes

$$\frac{d}{dt}\|\tilde{\theta}\|_2^2 = \\ = 2\tilde{\theta}^\top\mathbf{A}_{\theta^*}\tilde{\theta} + 2\tilde{\theta}^\top(\mathbf{A}_{\theta^*} - \mathbf{A}_\theta)\theta + 2\tilde{\theta}^\top(\mathbf{b}_\theta - \mathbf{b}_{\theta^*}) \\ \leq 2\tilde{\theta}^\top\mathbf{A}_{\theta^*}\tilde{\theta} + 2(\lambda_A + \lambda_b)\|\tilde{\theta}\|_2^2.$$

Letting $\lambda = \lambda_A + \lambda_b$, the above expression can be written as

$$\frac{d}{dt}\|\tilde{\theta}\|_2^2 \leq \tilde{\theta}^\top\big(\mathbf{A}_{\theta^*} + \lambda\mathbf{I}\big)\tilde{\theta}.$$

The fact that the learning policy is assumed Lipschitz w.r.t. θ and the uniform ergodicity of the corresponding induced chain implies that $\mathbf{A}_\theta$ and $\mathbf{b}_\theta$ are also Lipschitz w.r.t. θ (with a different constant). This means that λ goes to zero with C and, therefore, for C sufficiently small, $(A + \lambda\mathbf{I})$ is a negative definite matrix.[4] Therefore, the ODE (13) is globally asymptotically stable and the conclusion of the theorem follows. □

Several remarks are now in order. First of all, Theorem 2 basically states that, for fixed ε if the dependence of the learning policy π_θ can be made sufficiently "smooth", then SARSA converges w.p.1. This result is similar to the result in (Perkins & Precup, 2003), although the algorithms are not exactly similar: we consider a continuing task, while the algorithm featured in (Perkins & Precup, 2003) is implemented in an episodic fashion. Furthermore, in our case, convergence was established using an ODE argument, instead of the contraction argument in (Perkins & Precup, 2003). Nevertheless, both methods of proof are, in its essence, equivalent and the results in both papers concordant.

A second remark is related with the implementation of SARSA with a decaying exploration policy. The analysis of one such algorithm could be conducted using, once again, an ODE argument. In particular, SARSA could be described as a two-time-scale algorithm: the iterations of the main algorithm (corresponding to (12)) would develop on a faster time-scale and the decaying exploration rate would develop at a slower time-scale. The analysis in (Borkar, 1997) could then be replicated. However, it is well-known that, as ε approaches zero, the learning policy will approach the greedy policy w.r.t. θ which is, in general, discontinuous. Therefore, there is little hope that the smoothness

[4] The fact that $\mathbf{A}_\theta$ is negative definite has been established in several works. See, for example, Lemma 3 in (Perkins & Precup, 2003) or, in a slightly different setting (easily extendable to our setting) the proof of Theorem 1 in (Tsitsiklis & Van Roy, 1996a).

condition in Theorem 2 (or its equivalent in (Perkins & Precup, 2003)) can be met as ε approaches to zero.

4. Discussion

We now briefly discuss some of the assumptions in the above theorems.

We start by emphasizing that all stated conditions are only *sufficient*, meaning that it is possible that convergence may occur even if some (or all) fail to hold. We also discuss the relation between our results and other related works from the literature.

Seconsly, uniform ergodicity of a Markov chain essentially means that the chain quickly converges to a stationary behavior uniformly over the state-space and we can study any properties of the stationary chain by direct sampling.[5] This property and the requirement that $\pi(x,a) > 0$ for all $a \in \mathcal{A}$ and μ_X-almost all $x \in \mathcal{X}$ can be interpreted as a continuous counterpart to the usual condition that all state-action pairs are visited infinitely often. In fact, uniform ergodicity implies that all the regions of the state-space with positive μ_X measure are "sufficiently" visited (Meyn & Tweedie, 1993), and the condition $\pi(x,a) > 0$ ensures that, at each state, every action is "sufficiently" tried. It appears to be a standard requirement in this continuous scenario, as it has also been used in other works (Tsitsiklis & Van Roy, 1996a; Singh et al., 1994; Perkins & Precup, 2003).[6] The requirement that $\pi(x,a) > 0$ for all $a \in \mathcal{A}$ and μ_X-almost all $x \in \mathcal{X}$ also corresponds to the concept of *fair control* as introduced in (Borkar, 2000).

We also remark that the divergence example in (Gordon, 1996) is due to the fact that the learning policy fails to verify the Lipschitz continuity condition stated in Theorem 2 (as discussed in (Perkins & Pendrith, 2002)).

4.1. Related Work

In this paper, we analyzed how RL algorithms can be combined with linear function approximation to approximate the optimal Q-function in MDPs with infinite state-spaces. In the last decade or so, several authors have addressed this same problem from different perspectives. We now briefly discuss several such approaches and their relation to the results in this paper.

One possible approach is to rely on *soft-state aggregation* (Singh et al., 1994; Gordon, 1995; Tsitsiklis & Van Roy, 1996b), partitioning the state-space into "soft" regions. Treating the soft-regions as "hyper-states", these methods then use standard learning methods (such as Q-learning or SARSA) to approximate the optimal Q-function. The main differences between such methods and those using linear function approximation (such as the ones portrayed here) are that, in the former, only one component of the parameter vector is updated at each iteration and the basis functions are, by construction, restrained to be positive and to add to one at each point of the state-space.

Sample-based methods (Ormoneit & Sen, 2002; Szepesvári & Smart, 2004) further generalize the applicability of soft-state aggregation methods by using *spreading functions/kernels* (Ribeiro & Szepesvári, 1996). Sample-based methods thus exhibit superior convergence rate when compared with simple soft-state aggregation methods, although under somewhat more restrictive conditions.

Finally, RL with general linear function approximation was thorougly studied in (Tsitsiklis & Van Roy, 1996a; Tadić, 2001). Posterior works extended the applicability of such results. In Precup01icml, an off-policy convergent algorithm was proposed that uses an importance-sampling principle similar to the one described in Section 3. In (Perkins & Precup, 2003), the authors establish the convergence of SARSA with linear function approximation.

4.2. Concluding Remarks

We conclude by observing that all methods analyzed here as well as those surveyed above experience a degradation in performance as the distance between the target function and the chosen linear space increases. If the functions in the chosen linear space provide only a poor approximation of the desired function, there are no practical guarantees on the usefulness of such approximation. The error bounds derived in (Tsitsiklis & Van Roy, 1996a) are reassuring in that they state that the performance of approximate TD "gracefully" degrades as the distance between the target function and the chosen linear space increases. Although we have not addressed such topic in our analysis, we expect the error bounds in (Tsitsiklis & Van Roy, 1996a) to carry with little changes to our setting. Finally, we make no use of eligibility traces in our algorithms. However, it is just expectable that the methods described herein can easily be adapted to ac-

[5] Explicit bounds on the rate of convergence to stationarity are available in the literature (Meyn & Tweedie, 1994; Diaconis & Saloff-Coste, 1996; Rosenthal, 2002). However, for general chains, such bounds tend to be loose.

[6] Most of these works make use of geometric ergodicity which, since we admit a compact state-space, is a consequence of uniform ergodicity.

commodate for eligibility traces, this eventually yielding better approximations (with tighter error bounds) (Tsitsiklis & Van Roy, 1996a)

Acknowledgements

The authors would like to acknowledge the many useful comments from Doina Precup and the unknown reviewers. Francisco S. Melo acknowledges the support from the ICTI and the Portuguese FCT, under the Carnegie Mellon-Portugal Program. Sean Meyn gratefully acknowledges the financial support from the National Science Foundation (ECS-0523620). Isabel Ribeiro was supported by the FCT in the frame of ISR-Lisboa pluriannual funding. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. *Proc. 12th Int. Conf. Machine Learning* (pp. 30–37).

Benveniste, A., Métivier, M., & Priouret, P. (1990). *Adaptive algorithms and stochastic approximations*, vol. 22. Springer-Verlag.

Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming*. Athena Scientific.

Borkar, V. (1997). Stochastic approximation with two time scales. *Systems & Control Letters*, *29*, 291–294.

Borkar, V. (2000). A learning algorithm for discrete-time stochastic control. *Probability in the Engineering and Informational Sciences*, *14*, 243–258.

de Farias, D., & Van Roy, B. (2000). On the existence of fixed points for approximate value iteration and temporal-difference learning. *Journal of Optimization Theory and Applications*, *105*, 589–608.

Diaconis, P., & Saloff-Coste, L. (1996). Logarithmic Sobolev inequalities for finite Markov chains. *Annals of Applied Probability*, *6*, 695–750.

Gordon, G. (1995). *Stable function approximation in dynamic programming* (Technical Report CMU-CS-95-103). School of Computer Science, Carnegie Mellon University.

Gordon, G. (1996). *Chattering in SARSA(λ)*. (Technical Report). CMU Learning Lab Internal Report.

Meyn, S., & Tweedie, R. (1993). *Markov chains and stochastic stability*. Springer-Verlag.

Meyn, S., & Tweedie, R. (1994). Computable bounds for geometric convergence rates of Markov chains. *Annals of Applied Probability*, *4*, 981–1011.

Ormoneit, D., & Sen, S. (2002). Kernel-based reinforcement learning. *Machine Learning*, *49*, 161–178.

Perkins, T., & Pendrith, M. (2002). On the existence of fixed-points for Q-learning and SARSA in partially observable domains. *Proc. 19th Int. Conf. Machine Learning* (pp. 490–497).

Perkins, T., & Precup, D. (2003). A convergent form of approximate policy iteration. *Adv. Neural Information Proc. Systems* (pp. 1595–1602).

Precup, D., Sutton, R., & Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. *Proc. 18th Int. Conf. Machine Learning* (pp. 417–424).

Ribeiro, C., & Szepesvári, C. (1996). Q-learning combined with spreading: Convergence and results. *Proc. ISRF-IEE Int. Conf. Intelligent and Cognitive Systems* (pp. 32–36).

Rosenthal, J. (2002). Quantitative convergence rates of Markov chains: A simple account. *Electronic Communications in Probability*, *7*, 123–128.

Singh, S., Jaakkola, T., & Jordan, M. (1994). Reinforcement learning with soft state aggregation. *Adv. Neural Information Proc. Systems* (pp. 361–368).

Sutton, R. (1999). Open theoretical questions in reinforcement learning. *Lecture Notes in Computer Science*, *1572*, 11–17.

Szepesvári, C., & Smart, W. (2004). Interpolation-based Q-learning. *Proc. 21st Int. Conf. Machine learning* (pp. 100–107).

Tadić, V. (2001). On the convergence of temporal-difference learning with linear function approximation. *Machine Learning*, *42*, 241–267.

Tsitsiklis, J., & Van Roy, B. (1996a). An analysis of temporal-difference learning with function approximation. *IEEE Trans. Automatic Control*, *42*, 674–690.

Tsitsiklis, J., & Van Roy, B. (1996b). Feature-based methods for large scale dynamic programming. *Machine Learning*, *22*, 59–94.

Watkins, C. (1989). *Learning from delayed rewards*. Doctoral dissertation, King's College, University of Cambridge.

Empirical Bernstein Stopping

Volodymyr Mnih MNIH@CS.UALBERTA.CA
Csaba Szepesvári SZEPESVA@CS.UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8 Canada

Jean-Yves Audibert AUDIBERT@CERTIS.ENPC.FR
Certis - Ecole des Ponts, 6 avenue Blaise Pascal, Cité Descartes, 77455 Marne-la-Vallée France
Willow - ENS / INRIA, 45 rue d'Ulm, 75005 Paris, France

Abstract

Sampling is a popular way of scaling up machine learning algorithms to large datasets. The question often is how many samples are needed. Adaptive stopping algorithms monitor the performance in an online fashion and they can stop early, saving valuable resources. We consider problems where probabilistic guarantees are desired and demonstrate how recently-introduced empirical Bernstein bounds can be used to design stopping rules that are efficient. We provide upper bounds on the sample complexity of the new rules, as well as empirical results on model selection and boosting in the filtering setting.

1. Introduction

Being able to handle large datasets and streaming data is crucial to scaling up machine learning algorithms to many-real world settings. When making even a single pass through the data is prohibitive, sampling may offer a good solution. In order for the resulting algorithms to be theoretically sound, sampling techniques that come with probabilistic guarantees are desirable. For example, when estimating the error of a classifier on a large dataset one may want to sample until the estimated error is within some small number ϵ of the true error with probability at least $1 - \delta$. The key problem is one of *stopping* or determining the required number of samples. Taking too many samples will result in inefficient algorithms, while taking too few may not be enough to achieve the desired guarantees.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Finite sample bounds, such as Hoeffding's inequality (Hoeffding, 1963), are the key technique used by recent, non-parametric stopping algorithms with probabilistic guarantees. While these stopping algorithms have proved to be effective for scaling up machine learning algorithms (Bradley & Schapire, 2008), (Domingos & Hulten, 2001), they can be significantly improved by incorporating variance information in a principled manner. We show how to employ the recently introduced empirical Bernstein bounds (Audibert et al., 2007a) to improve stopping algorithms and provide sample complexity bounds and empirical results to demonstrate the effect of incorporating variance information.

Before proceeding, we identify two classes of stopping problems that will be examined. The first class concerns problems where some unknown quantities have to be measured either up to some prespecified level of accuracy or to support making a binary decision. Examples in this class include stopping with a fixed relative or absolute accuracy, with applications in hypothesis testing such as deciding on the sign of the mean, independence tests, and change detection. In problems belonging to the second group, the task is to pick the best option from a finite pool while measuring their performance using samples. Some notable examples include various versions of bandit problems, Hoeffding Races (Maron & Moore, 1993), and the general framework for scaling up learning algorithms proposed by Domingos (2001).

The paper is organized as follows. In Section 2 we examine Hoeffding's inequality and introduce the empirical Bernstein bound. In Section 3, we introduce a new stopping algorithm for stopping with a predefined relative accuracy and show that it is more efficient than previous algorithms. Section 4 demonstrates how a simple application of the empirical Bernstein bound can result in substantial improvements for problems

from the second class. Conclusions and future work directions are presented in Section 5.

2. Hoeffding Bounds vs. Empirical Bernstein Bounds

Let $X_1, \ldots, X_t$ real-valued *i.i.d.* random variables with range R and, mean μ, and let $\overline{X}_t = 1/t \sum_{i=1}^{t} X_i$. Hoeffding's inequality (Hoeffding, 1963) states that with probability at least $1 - \delta$

$$|\overline{X}_t - \mu| \leq R\sqrt{\frac{\log(2/\delta)}{2t}}.$$

Due to its generality, Hoeffding's inequality has been widely used in online learning scenarios. A drawback of the bound is that it scales linearly with the range R and does not scale with the variance of X_i. If a bound on the variance is known, Bernstein's inequality can be used instead, which can yield significant improvements when the variance bound is small relative to the range. Since useful a priori bounds on the variance are rarely available, this approach is not practical.

An approach that is more suitable to online scenarios is to apply Bernstein's inequality to the sum of $X_1, \ldots, X_t$, as well as the sum of the squares to obtain a single bound on the mean of $X_1, \ldots, X_t$. The resulting bound, which we will refer to as the *empirical Bernstein bound* (Audibert et al., 2007a) states that with probability at least $1 - \delta$

$$|\overline{X}_t - \mu| \leq \overline{\sigma}_t\sqrt{\frac{2\log(3/\delta)}{t}} + \frac{3R\log(3/\delta)}{t},$$

where $\overline{\sigma}_t$ is the empirical standard deviation of $X_1, \ldots, X_t$: $\overline{\sigma}_t^2 = \frac{1}{t}\sum_{i=1}^{t}(X_i - \overline{X}_t)^2$. The term involving the range R decreases at the rate of t^{-1} and quickly becomes negligible when the variance is large, while the square root term depends on $\overline{\sigma}_t$ instead of R. Hence, when $\overline{\sigma}_t \ll R$ the empirical Bernstein bound quickly becomes much tighter than Hoeffding's inequality.

3. Stopping Rules

Let $X_1, X_2, \ldots$ be i.i.d. random variables with mean μ and variance σ^2. We will refer to an algorithm as a stopping rule if at time t it observes X_t and based on past observations decides whether to stop or continue sampling. If a stopping rule $\mathcal{S}$ returns $\hat{\mu}$ that satisfies

$$\mathbb{P}\left[|\hat{\mu} - \mu| \leq \epsilon|\mu|\right] \geq 1 - \delta, \tag{1}$$

then $\mathcal{S}$ is a (ϵ, δ)-stopping rule and $\hat{\mu}$ is an (ϵ, δ)-approximation of μ. In this section, we develop an (ϵ, δ)-stopping rule for bounded X_i.

Algorithms proposed for this problem include the Nonmonotonic Adaptive Sampling (NAS) algorithm, shown as Algorithm 1, due to Domingo et al. (2000a). The general idea is to first use Hoeffding's inequality to construct a sequence $\{\alpha_t\}$ such that the event $\mathcal{E} = \left\{|\overline{X}_t - \mu| \leq \alpha_t,\, t \in \mathbb{N}^+\right\}$ occurs with probability at least $1 - \delta$, and then use this sequence to design a stopping criterion that stops only if $\left|\overline{X}_t - \mu\right| \leq \epsilon|\mu|$ given that $\mathcal{E}$ holds.

Algorithm 1 Algorithm NAS

```
t ← 0
repeat
  t ← t + 1
  Obtain X_t
  α ← sqrt((1/2t) log(t(t+1)/δ))
until |X̄_t| < α(1 + 1/ε)
return X̄_t
```

Domingo et al. (2000a) argue that if T is the number of samples after which NAS stops, and $|\mu| > 0$, then there exists a constant C such that with probability at least $1 - \delta$

$$T < C \cdot \frac{R^2}{\epsilon^2\mu^2}\Big(\log\frac{2}{\delta} + \log\frac{R}{\epsilon\mu}\Big). \tag{2}$$

The assumption that $|\mu| > 0$ is necessary for guaranteeing that the algorithm will indeed stop, and will be assumed for the rest of the section.

Dagum et al. (2000) introduced the $\mathcal{AA}$ algorithm for the case of bounded and nonnegative X_i. The $\mathcal{AA}$ algorithm is a three step procedure. In the first step, an $(\max(\sqrt{\epsilon}, 1/2), \delta/3)$-approximation of μ, $\tilde{\mu}$, is obtained. In the second step $\tilde{\mu}$ is used to determine the number of samples necessary to produce an estimate $\tilde{\sigma}^2$ of σ^2 such that $\max(\tilde{\sigma}^2, \epsilon\tilde{\mu})$ is a high probability upper bound on $\max(\sigma^2, \epsilon\mu)/2$. In the last step, $c\max(\tilde{\sigma}^2, \epsilon\tilde{\mu})\frac{\log(1/\delta)}{\epsilon^2\tilde{\mu}^2}$ samples are drawn and their average is returned as $\hat{\mu}$, where c is a universal constant.

Dagum et al. (2000) prove that $\hat{\mu}$ is indeed an (ϵ, δ)-approximation of μ and that that if T is the number of samples taken by $\mathcal{AA}$, then there exists a constant C such that with probability at least $1 - \delta$

$$T \leq C \cdot \max(\sigma^2, \epsilon\mu) \cdot \frac{1}{\epsilon^2\mu^2}\log\frac{2}{\delta}. \tag{3}$$

In addition, Dagum et al. prove that if T is the number of samples taken by any (ϵ, δ)-stopping rule, then there exists a constant C' such that with probability at least $1 - \delta$

$$T \geq C' \cdot \max(\sigma^2, \epsilon\mu) \cdot \frac{1}{\epsilon^2\mu^2}\log\frac{2}{\delta}.$$

Hence, for bounded X_i, the $\mathcal{AA}$ algorithm requires a number of samples that is at most a multiplicative constant larger than that required by any other (ϵ, δ)-stopping rule. In this sense the algorithm achieves "optimal" efficiency, up to a multiplicative constant.

While the $\mathcal{AA}$ algorithm is able to take advantage of variance, it requires the random variables to be nonnegative. Any trivial extension of the $\mathcal{AA}$ algorithm to the case of signed random variables seems unlikely since the rule heavily relies on the monotonicity of partial sums that is present in the nonnegative case. On the other hand, Equation (2) suggests that the NAS algorithm is not able to take advantage of variance.

As the first demonstration of how the empirical Bernstein bound can be used to design improved stopping algorithms, we propose a new algorithm, EBStop, which uses empirical Bernstein Bounds to achieve nearly the same scaling properties as the $\mathcal{AA}$ algorithm and, like the NAS algorithm, only requires the random variables to be bounded.

3.1. EBStop

Similarly to the NAS algorithm, EBStop relies on a sequence $\{c_t\}$ with the property that the event $\mathcal{E} = \{|\overline{X}_t - \mu| \leq c_t, t \in \mathbb{N}^+\}$ occurs with probability at least $1-\delta$. Let d_t be a positive sequence satisfying $\sum_{t=1}^{\infty} d_t \leq \delta$ and let

$$c_t = \overline{\sigma}_t \sqrt{\frac{2\log(3/d_t)}{t}} + \frac{3R\log(3/d_t)}{t}.$$

Since $\{d_t\}$ sums to at most δ and $(\overline{X}_t - c_t, \overline{X}_t + c_t)$ is a $1-d_t$ confidence interval for μ obtained from the empirical Bernstein bound, by a union bound argument, the event $\mathcal{E}$ indeed occurs with probability at least $1-\delta$. In our work, we use $d_t = c/t^p$ and $c = \delta(p-1)/p$.

The pseudocode for EBStop is shown as Algorithm 2, but the general idea is as follows. After drawing t samples, we set LB to $\max(0, \max_{1\leq s\leq t} |\overline{X}_s| - c_s)$ and UB to $\min_{1\leq s\leq t}(|\overline{X}_s| + c_s)$. EBStop terminates as soon as $(1+\epsilon)\text{LB} \geq (1-\epsilon)\text{UB}$ and returns $\hat{\mu} = \text{sgn}(\overline{X}_t)\cdot 1/2\cdot [(1+\epsilon)\text{LB} + (1-\epsilon)\text{ UB}]$.

To see why $\hat{\mu}$ is an (ϵ, δ)-approximation, suppose the stopping condition has been satisfied and event $\mathcal{E}$ holds. Then

$$|\hat{\mu}| \leq 1/2 \cdot [(1+\epsilon)\text{LB} + (1+\epsilon)\text{LB}] \leq (1+\epsilon)|\mu|,$$

and similarly $(1-\epsilon)|\mu| \leq |\hat{\mu}|$. From the definition of LB, it also follows that $|\overline{X}_t| > c_t \geq |\overline{X}_t - \mu|$ which implies that $\text{sgn}(\hat{\mu}) = \text{sgn}(\mu)$. Since event $\mathcal{E}$ holds with probability at least $1-\delta$, $\hat{\mu}$ is indeed an (ϵ, δ)-approximation of μ.

Algorithm 2 Algorithm EBStop

```
LB ← 0
UB ← ∞
t ← 1
Obtain X_1
while (1 + ε)LB < (1 − ε)UB do
  t ← t + 1
  Obtain X_t
  LB ← max(LB, |X̄_t| − c_t)
  UB ← min(UB, |X̄_t| + c_t)
end while
return sgn(X̄_t) · 1/2· [(1 + ε)LB +(1 − ε) UB]
```

While we omit the proof due to space constraints[1], we note that if X is a random variable distributed with range R, and if T is the number of samples taken by EBStop on X, then there exists a constant C such that with probability at least $1-\delta$

$$T < C \cdot \max\left(\frac{\sigma^2}{\epsilon^2\mu^2}, \frac{R}{\epsilon|\mu|}\right)\left(\log\frac{1}{\delta} + \log\frac{R}{\epsilon|\mu|}\right). \quad (4)$$

This bound is very similar to the upper bound for the stopping time of the $\mathcal{AA}$ algorithm, with the only difference being the extra $\log\frac{R}{\epsilon|\mu|}$ term. This term comes from constructing a confidence interval at each t and is not an artifact of our proof techniques. However, this extra term can be reduced to $\log\log\frac{1}{\epsilon|\mu|}$ by applying a geometric grid as we will see in the next section. Since EBStop does not require the variables to be non-negative, we can say that EBStop combines the best properties of NAS and $\mathcal{AA}$ algorithms for signed random variables.

3.2. Improving EBStop

While EBStop has the desired scaling properties, we make two simple improvements in order to make it more efficient in practice.

The first improvement is based on the idea that if the algorithm is not close to stopping, there is no point in checking the stopping condition at every point. We incorporate this idea into EBStop by adopting a *geometric* sampling schedule, also used by Domingo and Watanabe (2000a). Instead of testing the stopping criterion after each sample, we perform the kth test after $\lceil\beta^k\rceil$ samples have been taken for some $\beta > 1$. Under this sampling strategy, when EBStop constructs a $1-d$ confidence interval after t samples, d is of the order $1/(\log_\beta t)^p$, which is much larger than $1/t^p$. Since

[1] A version of the paper containing the proofs will be made available as a technical report.

this results in tighter confidence intervals, LB and UB will approach each other faster and the stopping condition will be satisfied after fewer samples.

While geometric sampling can often reduce the number of required samples, it can also lead to taking roughly β times too many samples, because testing is only done at the ends of intervals. Nevertheless, the following result due to Audibert et al. (2007b) can be used to test the stopping condition after each sample without sacrificing the advantages of geometric sampling. Let $t_1 \leq t_2$ for $t_1, t_2 \in \mathbb{N}$ and let $\alpha \geq t_2/t_1$. Then with probability at least $1-d$, for all $t \in \{t_1, \ldots, t_2\}$

$$|\overline{X}_t - \mu| \leq \overline{\sigma}_t \sqrt{2\alpha \log(3/d)/t} + 3\alpha \log(3/d)/t. \quad (5)$$

We use Equation (5) with $t_1 = \lfloor \beta^k \rfloor + 1$, $t_2 = \lfloor \beta^{k+1} \rfloor$, and $d = c/k^p$ to construct c_t for each $t \in \{t_1, \ldots, t_2\}$. This allows us to test the stopping condition after each sample, and use d that is on the order of $1/(\log_\beta t)^{p\alpha}$ after t samples. A variant of EBStop that incorporates these two improvements is shown as Algorithm 3.

Algorithm 3 Algorithm EBGStop

```
LB ← 0
UB ← ∞
t ← 1
k ← 0
Obtain X_1
while (1 + ε)LB < (1 − ε)UB do
  t ← t + 1
  Obtain X_t
  if t > floor(β^k) then
    k ← k + 1
    α ← floor(β^k)/floor(β^{k−1})
    x ← −α log d_k/3
  end if
  c_t ← σ̄_t sqrt(2x/t) + 3Rx/t
  LB ← max(LB, |X̄_t| − c_t)
  UB ← min(UB, |X̄_t| + c_t)
end while
return sgn(X̄_t) · 1/2· [(1 + ε)LB +(1 − ε) UB]
```

One can show that as the result of adding geometric sampling to EBStop reduces the $\log \frac{1}{\epsilon|\mu|}$ term in inequality (4) to $\log\log \frac{1}{\epsilon|\mu|}$. It should be noted that from the arguments of Dagum et al. (2000), no stopping rule can achieve a better bound than (3) for the case of bounded non-negative random variables. Hence, EBGStop is very close to being "optimal" in this sense. Where it would loose (for non-negative random variables) to $\mathcal{AA}$ is when ϵ, μ and δ are such that $\log(R/(\epsilon\mu))$ becomes significantly larger than $1/\delta$. We do not expect to see this happening in practice for not too large values of δ. For example, for $\delta = 0.05$, $R = 1$, the condition is $\epsilon\mu < e^{-20}$.

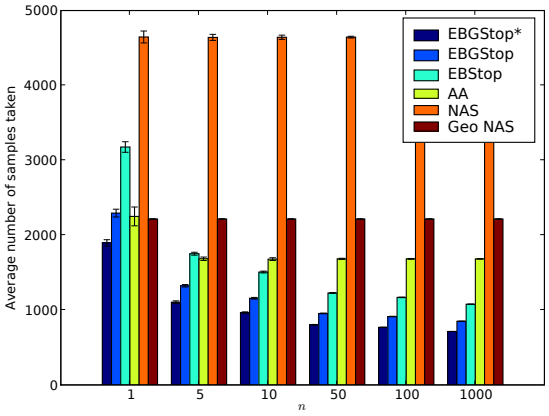


Figure 1. Comparison of stopping rules on averages of uniform random variables with varying variances. Error bars are at one standard deviation.

3.3. Results: Synthetic Data

In this section we evaluate the stopping rules $\mathcal{AA}$, NAS, geometric NAS, EBStop, and EBGStop on the problem of estimating means of various random variables. To make the comparison fair, the geometric version of the NAS algorithm and EBGStop both grew intervals by a factor of 1.5, as this value worked best in previous experiments (Domingo & Watanabe, 2000b). We also used $d_t = (t(t+1))^{-1}$ in EBStop and EBGStop since this is the sequence implicitly used in NAS for constructing confidence intervals at time t. Since this put EBGStop at a slight disadvantage, we also include results for EBGStop, denoted by EBGStop*, with our default choice of $d_t = c/(\log_\beta t)^p$, $p = 1.1$, and $\beta = 1.1$. In all the experiments we used $\epsilon = \delta = 0.1$. We use only non-negative valued random variables as they allow comparison to $\mathcal{AA}$. Finally, we only compare the number of samples taken because none of the algorithms produced any estimates with relative error greater than ϵ in any of our experiments.

The first set of experiments was meant to test how well the various stopping rules are able to exploit the variance when it is small. Let the average of n uniform $[a, b]$ random variables be denoted by $U(a, b, n)$. Note that the expected value and variance of $U(a, b, n)$ are $(a + b)/2$ and $(b - a)^2/(12n)$, respectively. For this comparison we fixed a to 0, b to 1, and varied n to control the variance for a fixed mean. Figure 1 shows the results of running each stopping rule 100 times on $U(0, 1, n)$ random variables for $n = 1, 5, 10, 50, 100, 1000$. Not surprisingly, NAS and geometric NAS fail to make use of the variance

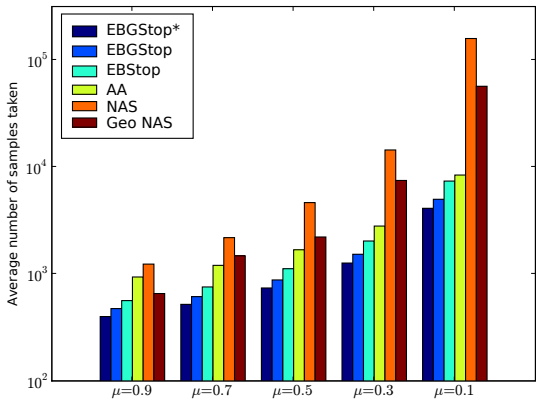


Figure 2. Comparison of stopping rules on averages of uniform random variables with varying means. The number of samples is shown in log scale.

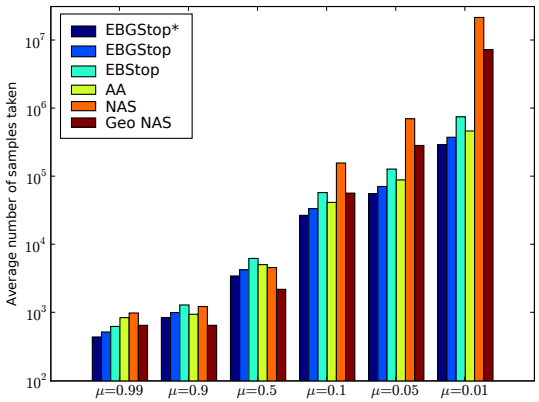


Figure 3. Comparison of stopping rules on Bernoulli random variables. The number of samples is shown in log scale.

and take roughly the same numbers of samples for all values of n. Variants of EBStop improve when the variance decreases, with EBGStop* performing especially well, beating all the other algorithms for all the scenarios tested. $\mathcal{AA}$ initially improves with the decreasing variance, but the effect is not as large as with EBGStop* because of the multi-phase structure of $\mathcal{AA}$.

In the second set of experiments we fix n at 10 and $b - a$ at 0.2, keeping the variance fixed, and vary the mean. The variance is small enough that EBStop, its variants, and $\mathcal{AA}$ should take a number of samples in the order of $R/(\epsilon\mu)$. The results are presented in Figure 2 and suggest that both variants of NAS require $1/\mu$ times more samples than the variance-adaptive methods. Note that Figure 2 shows the number of samples taken by each method in log scale.

It may be surprising that in both experiments the $\mathcal{AA}$ algorithm did not outperform EBStop and EBGStop even though $\mathcal{AA}$ offers better guarantees on sample complexity. We believe that EBStop is able make better use of the data because it uses all samples in its stopping criterion, while $\mathcal{AA}$ wastes some samples on intermediate quantities. However, this difference should be reflected in the hidden constants. As discussed earlier, for really small values of μ and ϵ the $\mathcal{AA}$ algorithm should stop earlier than EBStop.

Finally, we include a comparison of the stopping rules on Bernoulli random variables. Since Bernoulli random variables have maximal variance of all bounded random variables, the advantage of variance estimation should be diminished. However, inequality (4) suggests that in the case of Bernoulli random variables EBStop requires $O(1/(\epsilon^2\mu))$ samples since $\sigma^2 = \mu(1-\mu)$. Similarly, inequality (2) suggests that NAS requires $O(1/(\epsilon^2\mu^2))$ samples.

Figure 3 shows the results of running each stopping rule on Bernoulli random variables with means 0.99, 0.9, 0.5, 0.1, 0.05, and 0.01, averaged over 100 runs. As in the previous set of experiments, the variance-adaptive methods seem to require $1/\mu$ times fewer samples to stop. It should also be noted that the geometric version of the NAS algorithm does outperform EBStop for some intermediate values of μ, where the variance is the largest. However the performance difference is not large, and so we think the price paid for the unboundedly better performance of EBStop for small or large values of μ is not large.

3.4. Results: FilterBoost

Boosting by filtering (Bradley & Schapire, 2008) is a framework for scaling up boosting algorithms to large or streaming datasets. Instead of working with the entire training set, all steps, such as finding a weak learner that has classification accuracy of at least 0.5, are done through sampling that employs stopping algorithms. Bradley and Schapire showed that such an approach can lead to a drastic speedup over a batch boosting algorithm.

We evaluated the suitability of EBGStop and both variants of the NAS algorithm for the boosting by filtering setting by plugging them into the FilterBoost algorithm (Bradley & Schapire, 2008). The $\mathcal{AA}$ algorithm was not included because it cannot deal with signed random variables.

Following Bradley and Schapire, the Adult and Covertype datasets from the UCI machine learning repository (Asuncion & Newman, 2007) were used. The covertype dataset was converted into a binary classi-

fication problem by taking "Lodgepole Pine" as one class and merging the other classes. In setting up boosting we followed the procedure of Domingo and Watanabe (2000b) who also considered the use of stopping rules in the same context. Accordingly, we used decisions stumps as weak learners and we discretized all continuous attributes by binning their values into five equal bins. The results for the Adult dataset were averaged over 10 runs on the training set, while 10-fold cross-validation was used for the Covertype dataset.

As shown in Figure 4, EBGStop required fewer samples and offered lower variance in stopping times than either variant of the NAS algorithm on both datasets. At the same time, the resulting classification accuracies were within 0.2% of each other on the Adult dataset and within 0.04% of each other on the Covertype dataset.

4. Racing Algorithms

In this section we demonstrate how a general stopping algorithm that makes use of finite sample deviation bounds can be improved with the use of empirical Bernstein bounds. We consider the Hoeffding races algorithm (Maron & Moore, 1993) since it is representative of the class of general stopping algorithms.

Racing algorithms aim to reduce the computational burden of performing tasks such model selection using a hold-out set by discarding poor models quickly (Maron & Moore, 1993; Ortiz & Kaelbling, 2000). The context of racing algorithms is the one of multi-armed bandit problems. Formally, consider M options. When option m is chosen the t^{th} time, it gives a random value $X_{m,t}$ from an unknown distribution ν_m. The samples $\{X_{m,t}\}_{t\geq 1}$ are independent of each other. Let $\mu_m = \int x\nu_m(dx)$ be the mean reward obtained of option m. The goal is to find the options with the highest mean reward.

Let $\delta > 0$ be the confidence level parameter and N be the maximal amount of time allowed for deciding which option leads to the best expected reward. A racing algorithm either terminates when it runs out of time (i.e. at the end of the N-th round) or when it can say that with probability at least $1-\delta$, it has found the best option, i.e. an option m^* with $\mu_{m^*} = \max_{m\in\{1,\ldots,M\}} \mu_m$.

The Hoeffding race is an algorithm based on discarding options which are likely to have smaller mean than the optimal one until only one option remains. Precisely, for each time step and each distribution, $\delta/(MN)$ confidence intervals are constructed for the mean. Options with upper confidence smaller than the lower confidence bound of another option are discarded. The algorithm samples one by one all the options that have not been discarded yet.

We assume that the rewards have a bounded range R. If $\overline{X}_{m,t}$ denotes the sample mean for option m after seeing t samples of this option then according to Hoeffding's inequality, a $\delta/(MN)$ confidence interval for the mean of option m is

$$\left[\overline{X}_{m,t} - R\sqrt{\frac{\log(2MN/\delta)}{2t}}, \overline{X}_{m,t} + R\sqrt{\frac{\log(2MN/\delta)}{2t}}\right]$$

The Hoeffding race has been introduced and studied in (Maron & Moore, 1993; Maron & Moore, 1997) in a slightly different viewpoint since there the target was to find an option with mean at most ϵ below the optimal mean $\max_{m\in\{1,\ldots,M\}} \mu_m$, where ϵ is a given positive parameter. The same problem was also studied by (Even-Dar et al., 2002, Theorem 3) in the infinite horizon setting.

By substituting Hoeffding's inequality with the empirical Bernstein bound we obtain a new algorithm, which we will refer to as the empirical Bernstein race.

4.1. Analysis of Racing Algorithms

For the analysis we are interested in the expected number of samples taken by the Hoeffding race and the empirical Bernstein race. Due to space limitations, we omit the proofs of the following theorems.

Let $\Delta_m = \mu_{m^*} - \mu_m$, where option m^* still denotes an optimal option: $\mu_{m^*} = \max_{m\in\{1,\ldots,M\}} \mu_m$. Let $\lceil u \rceil$ denote the smallest integer larger or equal to u, and let $\lfloor u \rfloor$ denote the largest integer smaller or equal to u.

Theorem 1 (Hoeffding Race)**.** *Let* $n_H(m) = \left\lceil \frac{8R^2\log(2MN/\delta)}{\Delta_m^2} \right\rceil$. *Without loss of generality, assume that* $n_H(1) \leq n_H(2) \leq \ldots \leq n_H(m)$. *The number of samples, T, taken by the Hoeffding race is bounded by*

$$2\sum_{\mu_m<\mu_{m*}} n_H(m).$$

The probability that no optimal option is returned is bounded by δ. *If the algorithm runs out of time, then with probability at least* $1-\delta$, *(i) the number of discarded options is at least d, where d is the largest integer such that* $2\sum_{m=1}^{d} n_H(m) \leq N$, *and (ii) the non-discarded options satisfy*

$$\mu_m \geq \mu_{m^*} - 4R\sqrt{\frac{\log(2MN/\delta)}{2\lfloor N/M \rfloor}}.$$

We recall that the principle of the empirical Bernstein race algorithm is the same as the Hoeffding's one. We

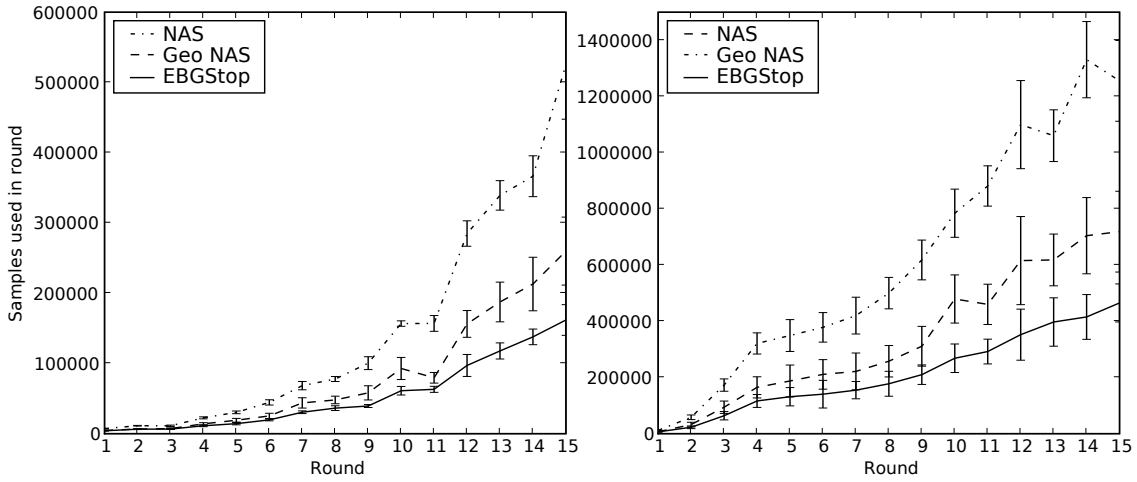


Figure 4. Comparison of the number of samples required by different stopping rules in FilterBoost. Parameters ϵ, δ were set to 0.1 for both methods while τ was set to 0.25. Error bars are at 1 standard deviation. a) Results on the Adult dataset b) Results on the covertype dataset.

sample one by one all the distributions that has not been discarded yet. The algorithm discards an option as soon as the upper bound on its mean reward is smaller than at least one of the lower bound on the mean of any other option.

Theorem 2 (Empirical Bernstein Race)**.** *Let σ_m denote the standard deviation of ν_m. Introduce $\Sigma_m = \sigma_{m^*} + \sigma_m$ and*

$$n(m) = \left\lceil \frac{8\Sigma_m^2 + 18R\Delta_m}{\Delta_m^2} \log(4MN/\delta) \right\rceil.$$

Without loss of generality, assume that $n(1) \leq n(2) \leq \ldots \leq n(m)$. The number of samples taken by the empirical Bernstein race is bounded by

$$2 \sum_{\mu_m < \mu_{m^*}} n(m).$$

The probability that no optimal option is returned is bounded by δ. If the algorithm runs out of time, then with probability at least $1 - \delta$, (i) the number of discarded options is at least d, where d is the largest integer such that $2\sum_{m=1}^{d} n_H(m) \leq N$, and (ii) the non-discarded options satisfy

$$\mu_m \geq \mu_{m^*} - \Sigma_m \sqrt{\frac{8\log(4MN/\delta)}{\lfloor N/M \rfloor}} - \frac{9R\log(4MN/\delta)}{\lfloor N/M \rfloor}.$$

As can be seen from the bounds, the result of incorporating the variance estimates is similar to what was observed in Section 3: The dependence of the number of required samples on R^2 is reduced to a dependence on R and the variance. Similar results can be expected when applying the empirical Bernstein bound to other situations.

4.2. Results

Following the procedure of Maron and Moore (1997), we evaluated the Hoeffding and empirical Bernstein races on the task of selecting the best k for k-nearest neighbor regression and classification through leave-one-out cross-validation.[2] Three datasets of different types were used for the comparison. The SARCOS data presents a regression problem which involves predicting the torques at 7 joints of a robot arm based on the positions, velocities and accelerations at those joints. We only considered predicting the torque at the first joint. The Covertype2 dataset consists of 50,000 points sampled from the Covertype dataset from Section 3.4 and is a binary classification task. The Local dataset presents a regression problem that was created by sampling 10,000 points from a noisy piecewise-linear function defined on the unit interval and having a range of 1.

The value of the range parameter R was set to 1 for the Covertype2 and Local datasets. For the SARCOS dataset, R was set to the range of the target values in the dataset. This differs from the approach of setting R separately for each option to several times the standard deviation in the samples observed, suggested by Maron and Moore (1997). We do not follow this approach because it invalidates the use of Hoeffding's inequality.

[2]Since leave-one-out cross-validation creates dependencies between the samples, the analysis does not apply to this case. However, our experiments gave similar results when we used a separate hold-out set. We decided to present results for leave-one-out cross-validation to facilitate comparison with the original papers.

Table 1. Percentage of work saved / number of options left after termination.

Data set	Hoeffding	EB
SARCOS	0.0% / 11	44.9% / 4
Covertype2	14.9% / 8	29.3% / 5
Local	6.0% / 9	33.1% / 6

All methods were given the options $k = 2^0, 2^1, 2^2, 2^3, \ldots, 2^{10}$ to begin with. The results are presented in Table 1. The table shows the percentage of work saved by each method ($1-$ number of samples taken by method / MN), as well as the number of options remaining after termination.

The empirical Bernstein racing algorithm, which is denoted by EB, significantly outperforms the Hoeffding racing algorithm on all three datasets. The advantage of incorporating variance estimates is the smallest on the Covertype2 classification dataset. This is expected because the samples come from Bernoulli distributions which have the largest possible variance for a bounded random variable. The advantage of variance estimation is the largest on the SARCOS dataset, where R is much larger than the variance. While one may argue that the Hoeffding racing algorithm would do much better if R was set to a smaller value based on the standard deviation, the empirical Bernstein algorithm would also benefit. However, tweaking R this way is merely an unprincipled way of incorporating variance estimates into a racing algorithm.

5. Conclusions and Future Work

We showed how variance information can be exploited in stopping problems in a principled manner. Most notably, we presented a near-optimal stopping rule for relative error estimation on bounded random variables, significantly extending the results of Domingo and Watanabe, and Dagum et al.. We also provided empirical and theoretical results on the effect that can be expected from incorporating variance estimates into existing stopping algorithms.

One interesting question that should be addressed is if the bound achieved by the $\mathcal{AA}$ algorithm in the non-negative case, which is known to be optimal, can be achieved without the non-negativity condition.

Acknowledgements

This work was supported in part by Agence Nationale de la Recherche project "Modèles Graphiques et Applications", the Alberta Ingenuity Fund, iCore, the Computer and Automation Research Institute of the Hungarian Academy of Sciences, and NSERC.

References

Asuncion, A., & Newman, D. (2007). UCI machine learning repository.

Audibert, J. Y., Munos, R., & Szepesvári, C. (2007a). Tuning bandit algorithms in stochastic environments. *ALT* (pp. 150–165).

Audibert, J.-Y., Munos, R., & Szepesvári, C. (2007b). *Variance estimates and exploration function in multi-armed bandit* (Technical Report 07-31). Certis - Ecole des Ponts. `http://certis.enpc.fr/~audibert/RR0731.pdf`.

Bradley, J. K., & Schapire, R. (2008). Filterboost: Regression and classification on large datasets. *NIPS-20* (pp. 185–192).

Dagum, P., Karp, R., Luby, M., & Ross, S. (2000). An optimal algorithm for Monte Carlo estimation. *SIAM Journal on Computing*, *29*, 1484–1496.

Domingo, C., & Watanabe, O. (2000a). MadaBoost: A modification of AdaBoost. *COLT'00* (pp. 180–189).

Domingo, C., & Watanabe, O. (2000b). Scaling up a boosting-based learner via adaptive sampling. *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 317–328).

Domingos, P., & Hulten, G. (2001). A general method for scaling up machine learning algorithms and its application to clustering. *ICML* (pp. 106–113).

Even-Dar, E., Mannor, S., & Mansour, Y. (2002). PAC bounds for multi-armed bandit and Markov decision processes. *COLT'02* (pp. 255–270).

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*, 13–30.

Maron, O., & Moore, A. (1993). Hoeffding races: Accelerating model selection search for classification and function approximation. *NIPS 6* (pp. 59–66).

Maron, O., & Moore, A. W. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, *11*, 193–225.

Ortiz, L. E., & Kaelbling, L. P. (2000). Sampling methods for action selection in influence diagrams. *AAAI/IAAI* (pp. 378–385).

Efficiently Solving Convex Relaxations for MAP Estimation

M. Pawan Kumar PAWAN@ROBOTS.OX.AC.UK
Department of Engineering Science, University of Oxford

P.H.S. Torr PHILIPTORR@BROOKES.AC.UK
Department of Computing, Oxford Brookes University

Abstract

The problem of obtaining the maximum *a posteriori* (MAP) estimate of a discrete random field is of fundamental importance in many areas of Computer Science. In this work, we build on the tree reweighted message passing (TRW) framework of (Kolmogorov, 2006; Wainwright et al., 2005). TRW iteratively optimizes the Lagrangian dual of a linear programming relaxation for MAP estimation. We show how the dual formulation of TRW can be extended to include cycle inequalities (Barahona & Mahjoub, 1986) and some recently proposed second order cone (SOC) constraints (Kumar et al., 2007). We propose efficient iterative algorithms for solving the resulting duals. Similar to the method described in (Kolmogorov, 2006), these algorithms are guaranteed to converge. We test our approach on a large set of synthetic data, as well as real data. Our experiments show that the additional constraints (i.e. cycle inequalities and SOC constraints) provide better results in cases where the TRW framework fails (namely MAP estimation for non-submodular energy functions).

1. Introduction

The problem of obtaining the maximum *a posteriori* (MAP) estimate of a discrete random field plays a central role in various applications, e.g. stereo reconstruction (Szeliski et al., 2006) and protein side-chain prediction (Sontag & Jaakkola, 2007). Furthermore, it is closely related to many important combinatorial optimization problems such as MAXCUT (Goemans & Williamson, 1995) and 0-extension (Karzanov, 1998). It is therefore not surprising that a number of approximate MAP estimation approaches exist in the literature. One such class of approaches which provides a good approximation, both in theory and in practice, is based on convex relaxations (e.g. see (Kumar et al., 2007) for an overview). In this work, we focus on the issue of solving these relaxations efficiently with the goal of handling a large number of random variables, e.g. variables corresponding to pixels in an image.

A discrete random field is defined over random variables $\mathbf{v} = \{v_0, \cdots, v_{n-1}\}$, each of which can take a label from the set $\mathbf{l} = \{l_0, \cdots, l_{h-1}\}$. Throughout this paper, we will assume a conditional random field (CRF) while noting that all our results are applicable to the Markov random field framework. A CRF describes a neighbourhood relationship $\mathcal{E}$ between the variables such that $(a, b) \in \mathcal{E}$ if, and only if, v_a and v_b are neighbours. A labelling of the CRF is specified by a function $f : \{0, \cdots, n-1\} \longrightarrow \{0, \cdots, h-1\}$ (i.e. variable v_a takes label $l_{f(a)}$). Given data $\mathbf{D}$, the energy of the labelling is given by

$$Q(f; \mathbf{D}, \boldsymbol{\theta}) = \sum_{v_a \in \mathbf{v}} \theta^1_{a;f(a)} + \sum_{(a,b) \in \mathcal{E}} \theta^2_{ab;f(a)f(b)}, \quad (1)$$

where $\theta^1_{a;f(a)}$ and $\theta^2_{ab;f(a)f(b)}$ are the data-dependent unary and pairwise potentials respectively, and $\boldsymbol{\theta}$ denotes the parameter of the CRF. The problem of MAP estimation is to obtain the labelling f^* with the minimum energy (or equivalently the maximum posterior probability), i.e. $f^* = \arg\min_f Q(f; \mathbf{D}, \boldsymbol{\theta})$.

Related Work: We build upon the linear programming (LP) relaxation of (Wainwright et al., 2005), which we call LP-S (since it was first proposed by (Schlesinger, 1976) for the special case of hard constraint pairwise potentials). Although the LP-S relaxation can be solved in polynomial time using Interior Point algorithms, the state of the art softwares can only handle up to a few hundred variables due to their large memory requirements. To overcome this prob-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

lem, two iterative algorithms were proposed by (Wainwright et al., 2005) for solving the dual of the LP-S relaxation. Similar to min-sum belief propagation (BP), these algorithms are not guaranteed to converge. The work of (Kolmogorov, 2006) addressed this problem by proposing a convergent sequential tree-reweighted message passing (TRW-S) algorithm for solving the dual.

Despite its strong theoretical foundation, it was observed that TRW-S yields labellings with very high energies when the energy function contains non-submodular terms (Kolmogorov, 2006). This is not surprising since the LP-S relaxation provides an inaccurate approximation in such cases (e.g. see (Kumar et al., 2007)). In this work, we address this deficiency of TRW-S by appending the LP-S relaxation with some useful constraints.

Our Results: We show how the dual formulation of the LP-S relaxation can be extended to include linear cycle inequalities (Barahona & Mahjoub, 1986) (section 3). Furthermore, we incorporate the recently proposed second order cone (SOC) constraints of (Kumar et al., 2007) within this framework (section 4). Note that although the importance of cycle inequalities and SOC constraints is well-recognized, their use has been limited to a small number of random variables due to the lack of efficient algorithms (Sontag & Jaakkola, 2007). Our results on including these constraints within the TRW formulation allow us to develop efficient convergent algorithms for solving the resulting duals. We successfully apply these algorithms to several synthetic and real problems containing a large number of variables which could not be handled by previous approaches (section 5). Our experiments indicate that incorporating these constraints provides a much better approximation for the MAP estimation problem within reasonable computational times compared to several state of the art algorithms. Additional experimental results and proofs are provided in (Kumar & Torr, 2008).

2. Preliminaries

We begin by introducing some notation which would allow us to describe our results concisely.

Optimal Energy and Min-Marginals: The energy of the optimal labelling and the min-marginals of random variables and neighbouring random variables is given by the following equations respectively:

$$q(\boldsymbol{\theta}) = \min_{f} Q(f : \mathbf{D}), \tag{2}$$

$$q_{a;i}(\boldsymbol{\theta}) = \min_{f, f(a)=i} Q(f; \mathbf{D}, \boldsymbol{\theta}), \tag{3}$$

$$q_{ab;ij}(\boldsymbol{\theta}) = \min_{f, f(a)=i, f(b)=j} Q(f; \mathbf{D}, \boldsymbol{\theta}), \tag{4}$$

where the term $\mathbf{D}$ is dropped from the LHS to make the notation less cluttered.

Reparameterization: A parameter $\overline{\boldsymbol{\theta}}$ is called a reparameterization of the parameter $\boldsymbol{\theta}$ (denoted by $\overline{\boldsymbol{\theta}} \equiv \boldsymbol{\theta}$) if, and only if,

$$Q(f; \mathbf{D}, \overline{\boldsymbol{\theta}}) = Q(f; \mathbf{D}, \boldsymbol{\theta}), \forall f. \tag{5}$$

Over-complete Representations: A labelling f can be represented using an over-complete set of boolean variables $\mathbf{y}$ defined as

$$y_{a;i} = \begin{cases} 1 & \text{if } f(a) = i, \\ 0 & \text{otherwise.} \end{cases}, \quad y_{ab;ij} = y_{a;i} y_{b;j}. \tag{6}$$

We also define variables $(\mathbf{x}, \mathbf{X})$ such that

$$x_{a;i} = 2y_{a;i} - 1, \;\; X_{ab;ij} = 4y_{ab;ij} - 2y_{a;i} - 2y_{b;j} + 1. \tag{7}$$

We will sometimes specify the additional constraints (i.e. cycle inequalities and SOC constraints) using variables $(\mathbf{x}, \mathbf{X})$, since they will allow us to write these constraints concisely.

The LP-S Relaxation: The LP-S relaxation of the MAP estimation problem is given by

$$\mathbf{y}^* = \arg\min_{\mathbf{y} \in LOCAL(\mathbf{v}, \mathcal{E})} \mathbf{y}^\top \boldsymbol{\theta},$$

$$LOCAL(\mathbf{v}, \mathcal{E}) = \begin{cases} y_{a;i} \in [0,1], y_{ab;ij} \in [0,1], \\ \sum_{l_i \in \mathbf{l}} y_{a;i} = 1, \\ \sum_{l_j \in \mathbf{l}} y_{ab;ij} = y_{a;i}. \end{cases} \tag{8}$$

The term $LOCAL(\mathbf{v}, \mathcal{E})$ stands for *local consistency polytope* (Wainwright et al., 2005) and denotes the feasibility region of the LP-S relaxation (specified by the above constraints for all $v_a \in \mathbf{v}, (a,b) \in \mathcal{E}, l_i, l_j \in \mathbf{l}$).

Dual of the LP-S Relaxation: Let $\mathcal{T}$ denote a set of tree-structured CRFs defined over subsets of the given random variables. For a CRF $T \in \mathcal{T}$, we denote its random variables by $\mathbf{v}_T$, its neighbourhood relationship by $\mathcal{E}_T$ and its parameter as $\boldsymbol{\theta}^T$. The parameter $\boldsymbol{\theta}^T$ consists of unary potentials $\theta^{T1}_{a;i}$ and pairwise potentials $\theta^{T2}_{ab;ij}$. Let $\boldsymbol{\rho} = \{\rho(T), T \in \mathcal{T}\}$ be a set of non-negative real numbers which sum to one. Using the above notation, the dual of the LP-S relaxation can be written as follows (Kolmogorov, 2006; Wainwright et al., 2005):

$$\max_{\sum_{T \in \mathcal{T}} \rho(T) \boldsymbol{\theta}^T \equiv \boldsymbol{\theta}} \sum_T \rho(T) q(\boldsymbol{\theta}^T). \tag{9}$$

The TRW-S Algorithm: Table 1 describes the TRW-S algorithm (Kolmogorov, 2006) which attempts to solve the dual of the LP-S relaxation. In other words, it solves for the set of parameters $\boldsymbol{\theta}^T$, $T \in \mathcal{T}$, which maximize the dual (9). There are two main steps: (i)

reparameterization, which involves running one pass of BP on the tree structured CRFs T; and (ii) averaging operation. TRW-S is guaranteed not to decrease the value of the dual (9) at each iteration. Further, it can be shown that it converges to a solution which satisfies the *weak tree agreement* (WTA) (Kolmogorov, 2006).

Initialization

1. For every $\omega \in \mathbf{v} \bigcup \mathcal{E}$, find all trees $\mathcal{T}_\omega \subseteq \mathcal{T}$ which contains ω.
2. Initialize $\boldsymbol{\theta}^T$ such that $\sum_T \rho(T)\boldsymbol{\theta}^T \equiv \boldsymbol{\theta}$.
   Typically, we set $\rho(T) = \frac{1}{|\mathcal{T}|}$ for all $T \in \mathcal{T}$.
   Then we can initialize $\theta^{T1}_{a;i} = \theta^1_{a;i} \frac{|\mathcal{T}|}{|\mathcal{T}_{v_a}|}$ for all $T \in \mathcal{T}_{v_a}$.
   Similarly, $\theta^{T2}_{ab;ij} = \theta^2_{ab;ij} \frac{|\mathcal{T}|}{|\mathcal{T}_{(a,b)}|}$ for all $T \in \mathcal{T}_{(a,b)}$.

Iterative Steps

3. Pick an element $\omega \in \mathbf{v} \bigcup \mathcal{E}$.
4. For all $T \in \mathcal{T}_\omega$, reparameterize $\boldsymbol{\theta}^T$ to $\overline{\boldsymbol{\theta}}^T$ such that
   (i) $\overline{\theta}^{T1}_{a;i} = q_{a;i}(\boldsymbol{\theta}^T)$, if $\omega = v_a \in \mathbf{v}$,
   (ii) $\overline{\theta}^{T1}_{a;i} + \overline{\theta}^{T1}_{b;j} + \overline{\theta}^{T2}_{ab;ij} = q_{ab;ij}(\boldsymbol{\theta}^T)$, if $\omega = (a,b) \in \mathcal{E}$.
   This step involves running one iteration of BP for T.
5. Averaging operation:
   (i) If $\omega = v_a \in \mathbf{v}$,
   (a) Compute $\nu_{a;i} = \frac{1}{\rho_a} \sum_{T \in \mathcal{T}_a} \rho(T) \overline{\theta}^{T1}_{a;i}$.
   (b) Set $\overline{\theta}^{T1}_{a;i} = \nu_{a;i}$, for all $T \in \mathcal{T}_{v_a}$.
   (ii) If $\omega = (a,b) \in \mathcal{T}$,
   (a) Compute $\nu_{ab;ij} = \frac{1}{\rho_{ab}} \sum_{T \in \mathcal{T}_{(a,b)}} \rho(T)(\overline{\theta}^{T1}_{a;i} + \overline{\theta}^{T1}_{b;j} + \overline{\theta}^{T2}_{ab;ij})$.
   (b) Set $\overline{\theta}^{T1}_{a;i} + \overline{\theta}^{T1}_{b;j} + \overline{\theta}^{T2}_{ab;ij} = \nu_{ab;ij}$, for all $T \in \mathcal{T}_{(a,b)}$.
6. Repeat steps 3, 4 and 5 till convergence.

Table 1. The TRW-S algorithm. Recall that $\theta^{T1}_{a;i}$ and $\theta^{T2}_{ab;ij}$ are the unary and pairwise potentials for the parameter $\boldsymbol{\theta}^T$. Similarly, $\overline{\theta}^{T1}_{a;i}$ and $\overline{\theta}^{T2}_{ab;ij}$ are the unary and pairwise potentials defined by the parameter $\overline{\boldsymbol{\theta}}$. The terms $\rho_a = \sum_{T, v_a \in \mathbf{v}_T} \rho(T)$ and $\rho_{ab} = \sum_{T,(a,b) \in \mathcal{E}_T} \rho(T)$ are the variable and edge appearance terms for $v_a \in \mathbf{v}$ and $(a,b) \in \mathcal{E}$ respectively. In step 3, the value of the dual (9) remains unchanged. Step 4, i.e. the averaging operation, ensures that the value of the dual does not decrease. TRW-S converges to a solution which satisfies the WTA condition.

3. Adding Linear Constraints

We now show how the results of (Kolmogorov, 2006; Wainwright et al., 2005) can be extended to include an arbitrary number of linear cycle inequalities (Barahona & Mahjoub, 1986; Kumar et al., 2007). This requires us to incorporate cycle inequalities into the dual (11).

We begin by briefly describing cycle inequalities. Consider a cycle of length c in the graphical model of the given CRF, which is specified over a set of random variables $\mathbf{v}_C = \{v_b, b = a_1, a_2, \cdots, a_c\}$ such that $\mathcal{E}_C = \{(a_1,a_2),(a_2,a_3),\cdots,(a_n,a_1)\} \subseteq \mathcal{E}$. Further, let $\mathcal{E}_F \subseteq \mathcal{E}_C$ such that $|\mathcal{E}_F|$ (i.e. the cardinality of $\mathcal{E}_F$) is odd. Using these sets of edges, a cycle inequality can be specified as

$$\sum_{(a_k,a_m) \in \mathcal{E}_F} X_{a_k a_m; i_k i_m} - \sum_{(a_k,a_m) \in \mathcal{E}_C - \mathcal{E}_F} X_{a_k a_m; i_k i_m} \geq 2 - c, \tag{10}$$

where $l_{i_k}, l_{i_m} \in \mathbf{l}$[1]. The variables $X_{a_k a_m; i_k i_m}$ are defined in equation (7). It can be shown that adding cycle inequalities to LP-S, i.e. problem (8), provides a better relaxation than LP-S alone. Their importance is reflected in their wide use in recent literature such as (Sontag & Jaakkola, 2007; Zwick, 1999).

In general, a set of N_C cycle inequalities defined on a cycle $C = (\mathbf{v}_C, \mathcal{E}_C)$ (using different labels l_{i_k} for variables $v_{a_k} \in \mathbf{v}_C$) can be written as $\mathbf{A}^C \mathbf{y} \geq \mathbf{b}^C$. In other words, for every cycle we can define up to h^c cycle inequalities (where $h = |\mathbf{l}|$), i.e. $N_C \in \{0, 1, \cdots, h^c\}$. Let $\mathcal{C}$ be a set of cycles in the given CRF. Theorem 1 (given below) provides us with the dual of the LP relaxation obtained by appending problem (8) with cycle inequalities (defined over cycles in the set $\mathcal{C}$). We refer to the resulting relaxation as LP-C (where C denotes cycles).

Theorem 1: The following problem is the dual of problem (8) appended with a set of cycle inequalities $\mathbf{A}^C \mathbf{y} \geq \mathbf{b}^C$, for all $C \in \mathcal{C}$ (hereby referred to as the LP-C relaxation):

$$\begin{aligned} \max \quad & \textstyle\sum_T \rho(T) q(\boldsymbol{\theta}^T) + \sum_C \rho'(C) (\mathbf{b}^C)^\top \mathbf{u}^C, \\ \text{s.t.} \quad & \textstyle\sum_T \rho(T) \boldsymbol{\theta}^T + \sum_C \rho'(C) (\mathbf{A}^C)^\top \mathbf{u}^C \equiv \boldsymbol{\theta}, \\ & u^C_k \geq 0, \forall k \in \{1, 2, \cdots, N_C\}, C \in \mathcal{C}. \end{aligned} \tag{11}$$

Here $\boldsymbol{\rho}' = \{\rho'(C), C \in \mathcal{C}\}$ is some (fixed) set of non-negative real numbers which sum to one, and $\mathbf{u}^C = \{u^C_k, k = 1, \cdots, N_C\}$ are some non-negative slack variables.

Similar to the dual (9), the above problem cannot be solved using standard software for a large number of variables $\mathbf{v}$. In order to overcome this deficiency we propose a convergent algorithm (similar to TRW-S) to approximately solve problem (11). We call our approach the TRW-S(LP-C) algorithm. In order to describe TRW-S(LP-C), we need the following definitions.

We say that a tree structured random field $T = (\mathbf{v}_T, \mathcal{E}_T) \in \mathcal{T}$ *belongs to* a cycle $C = (\mathbf{v}_C, \mathcal{E}_C) \in \mathcal{C}$ (denoted by $T \in C$) if, and only if, there exists an edge $(a,b) \in \mathcal{E}_T$ such that $(a,b) \in \mathcal{E}_C$. In other words, $T \in C$ if they share a common pair of neighbouring random variables $(a,b) \in \mathcal{E}$. We also define the following problem:

$$\max \quad \textstyle\sum_{T \in C} \rho(T) q(\boldsymbol{\theta}^T) + \rho'(C) (\mathbf{b}^C)^\top \mathbf{u}^C,$$

[1]Note that using the variable $\mathbf{y}$ would result in a less compact representation of cycle inequalities.

$$\text{s.t.} \quad \sum_{T \in C} \rho(T)\boldsymbol{\theta}^T + \rho'(C)(\mathbf{A}^C)^\top \mathbf{u}^C = \boldsymbol{\theta}^C, \quad u_k^C \geq 0, \forall k \in \{1, 2, \cdots, N_C\}, \tag{12}$$

for some parameter $\boldsymbol{\theta}^C$. The variables of the above problem are restricted to u_k^C, $\theta_{a;i}^{T1}$ and $\theta_{ab;ij}^{T2}$ where $(a, b) \in \mathcal{E}_T \bigcap \mathcal{E}_C$ for some $T \in C$. In other words, problem (12) has fewer variables and constraints than dual (11) and can be solved easily using standard Interior Point algorithms for small cycles C. As will be seen, even using cycles of size 3 or 4 results in much better approximations of the MAP estimation problem for non-submodular energy functions.

Table 2 describes the convergent TRW-S(LP-C) algorithm for approximately solving the dual (11). The algorithm consists of two main steps : (i) solving problem (12) for a cycle; and (ii) running steps 4 and 5 of the TRW-S algorithm. Note that our approach is different from other generalizations of TRW, e.g. (Wiegernick, 2005) which computes marginals. Specifically, we do not cluster random variables but include additional constraints to reduce the feasibility region of the relaxation. Our experiments in section 5 show that, unlike (Wiegernick, 2005), we always outperform BP. The properties of the TRW-S(LP-C) algorithm are summarized below.

Initialization
1. Choose a set of tree structured random fields $\mathcal{T}$. Choose a set of cycles $\mathcal{C}$. For example, if the 4-neighbourhood is employed, $\mathcal{C}$ can be the set of all cycles of size 4.
2. Initialize $\boldsymbol{\theta}^T$ such that $\sum_T \rho(T)\boldsymbol{\theta}^T \equiv \boldsymbol{\theta}$. Initialize $u_k^C = 0$ for all C and k.
Iterative Steps
3. Pick an element $\omega \in \mathbf{v} \bigcup \mathcal{C}$. Find all cycles $\mathcal{C}_\omega \subseteq \mathcal{C}$ which contains ω.
4. For a cycle $C \in \mathcal{C}_\omega$, compute $\boldsymbol{\theta}^C = \sum_{T \in C} \rho(T)\boldsymbol{\theta}^T + \rho'(C)(\mathbf{A}^C)^\top \mathbf{u}^C$ using the values of $\boldsymbol{\theta}^T$ and $\mathbf{u}^C$ obtained in the previous iteration. Solve problem (12) using an Interior Point method. Update the values of $\boldsymbol{\theta}^T$ and $\mathbf{u}^C$.
5. For all trees $T \in \mathcal{T}$ which contain ω, run steps 4 and 5 of the TRW-S algorithm.
6. Repeat steps 3 and 4 for all cycles $C \in \mathcal{C}_\omega$.
7. Repeat steps 3 to 5 for all elements ω till convergence.

Table 2. The TRW-S(LP-C) algorithm.

3.1. Properties of the TRW-S(LP-C) Algorithm.

Property 1: *At each step of the algorithm, the reparameterization constraint is satisfied, i.e.*

$$\sum_T \rho(T)\boldsymbol{\theta}^T + \sum_C \rho'(C)(\mathbf{A}^C)^\top \mathbf{u}^C \equiv \boldsymbol{\theta}. \tag{13}$$

The constraint in problem (12) ensures that parameter vector $\boldsymbol{\theta}^C$ of cycle C remains unchanged. Hence, after step 4 of the TRW-S(LP-C) algorithm, the reparameterization constraint is satisfied. It was also shown that step 5 (i.e. running TRW-S) provides a reparameterization of $\boldsymbol{\theta}$ (see Lemma 3.3 of (Kolmogorov, 2006) for details). This proves Property 1.

Property 2: *At each step of the algorithm, the value of the dual (11) never decreases.* Clearly, step 4 of the TRW-S(LP-C) algorithm does not decrease the value of the dual (11) (since the objective function of problem (12) is part of the objective function of dual (11)). The work of (Kolmogorov, 2006) showed that step 5 (i.e. TRW-S) also does not decrease this value. Note that the LP-C relaxation is guaranteed to be bounded since it dominates the LP-S relaxation (Kumar et al., 2007), which itself is bounded (Kolmogorov, 2006). Therefore, by the Bolzano-Weierstrass theorem (Fitzpatrick, 2006), it follows that TRW-S(LP-C) will converge.

Property 3: *Like* TRW-S*, the necessary condition for convergence of* TRW-S(LP-C) *is that the parameter vectors $\boldsymbol{\theta}^T$ of the trees $T \in \mathcal{T}$ satisfy* WTA. This follows from the fact that TRW-S increases the value of the dual in a finite number of steps as long as the set of parameters $\boldsymbol{\theta}^T$, $T \in \mathcal{T}$, do not satisfy WTA (see (Kolmogorov, 2006) for details).

Property 4: *Unlike* TRW-S, WTA *is not the sufficient condition for convergence.* One of the main drawbacks of the TRW-S algorithm is that it converges as soon as the WTA condition is satisfied. Experiments in (Kolmogorov, 2006) indicate that this results in high energy solutions for the MAP estimation problem when the energy function is non-submodular. Using a counter-example, it can be shown that WTA is not the sufficient condition for the convergence of TRW-S(LP-C) (Kumar & Torr, 2008).

Obtaining the Labelling: Similar to the TRW-S algorithm, TRW-S(LP-C) solves the dual (11) and not the primal problem. In other words, it does not directly provide a labelling of the random variables. In order to obtain a labelling, we use the same scheme as the one suggested in (Kolmogorov, 2006) for the TRW-S algorithm. Briefly, we assign labels to the variables $\mathbf{v} = \{v_0, v_1, \cdots, v_{n-1}\}$ in increasing order (i.e. we label variable v_0, followed by variable v_1 and so on). Let $\boldsymbol{\theta}^{\mathcal{T}} = \sum_T \rho(T)\boldsymbol{\theta}^T$. At each stage, a variable v_a is assigned the label $l_{f(a)}$ such that

$$f(a) = \arg\min_{i, l_i \in \mathbf{l}} \left(\theta_{a;i}^{\mathcal{T}1} + \sum_{b<a, (a,b) \in \mathcal{E}} \theta_{ab;i,f(b))}^{\mathcal{T}2} \right), \tag{14}$$

where $\theta_{a;i}^{\mathcal{T}1}$ and $\theta_{ab;i,f(b)}^{\mathcal{T}2}$ are the unary and pairwise potentials corresponding to the parameter $\boldsymbol{\theta}^{\mathcal{T}}$ respec-

tively. It can be shown that under certain conditions the above procedure provides the optimal labelling (Meltzer et al., 2005).

4. Adding SOC Constraints

We now show how second order cone (SOC) constraints can be added to the dual (9). Specifically, we consider the two SOC constraints proposed in (Kumar et al., 2007) which result in the SOCP-C and SOCP-Q relaxations described below.

The SOCP-C Relaxation: Consider a set of random variables $\mathbf{v}_C = \{v_b, b = a_1, \cdots, a_c\} \subseteq \mathbf{v}$ such that $\mathcal{E}_C = \{(a_1, a_2), (a_2, a_3), (a_c, a_1)\} \subseteq \mathcal{E}$ (i.e. $\mathbf{v}_C$ forms a cycle of length c). We define a vector $\mathbf{x}_C$ whose k^{th} element is given by $x_{a_k;i_k}$ and a matrix $\mathbf{X}_C$ whose $(k,m)^{th}$ element is given by $X_{a_k a_m;i_k i_m}$ (where $l_{i_k}, l_{i_m} \in \mathbf{l}$). SOCP-C specifies constraints $||\mathbf{U}^\top \mathbf{x}_C||^2 \leq \mathbf{C} \bullet \mathbf{X}_C$ where $\mathbf{C} = \mathbf{D}_c + \lambda_c I = \mathbf{U}\mathbf{U}^\top$ and $(\bullet)$ represents the Frobenius inner product. The $c \times c$ matrix $\mathbf{D}_c$ is given by

$$D_c(i,j) = \begin{cases} (-1)^{c-1} & \text{if} \quad |i-j| = c-1, \\ 1 & \text{if} \quad |i-j| = 1, \\ 0 & \text{otherwise}, \end{cases} \tag{15}$$

and λ_c is the absolute value of the smallest eigenvalue of $\mathbf{D}_c$.

The SOCP-Q Relaxation: Consider a set of random variables $\mathbf{v}_C = \{v_b, b = a_1, \cdots, a_c\} \subseteq \mathbf{v}$ such that $\mathcal{E}_C = \{(a_i, a_j), i, j = 1, \cdots, c\} \subseteq \mathcal{E}$ (i.e. $\mathbf{v}_C$ form a clique of size c). SOCP-Q specifies constraints of the form $||\mathbf{U}^\top \mathbf{x}_C||^2 \leq \mathbf{C} \bullet \mathbf{X}_C$ where $\mathbf{C}$ is a matrix whose elements are all 1.

In general, a set of N_C SOC constraints on a cycle/clique can be defined as

$$||\mathbf{A}_k^C \mathbf{y} + \mathbf{b}_k^C|| \leq \mathbf{y}^\top \mathbf{c}_k^C + d_k^C, k \in \{1, 2, \cdots, N_C\}. \tag{16}$$

Let $\mathcal{C}$ be a set of cycles/cliques in the graphical model of the given random field. The following theorem provides us with the dual of the SOCP relaxation obtained by appending problem (8) with SOC constraints defined over the set $\mathcal{C}$.

Theorem 2: The following problem is the dual of problem (8) appended with a set of SOC constraints $||\mathbf{A}_k^C \mathbf{y} + \mathbf{b}_k^C|| \leq \mathbf{y}^\top \mathbf{c}_k^C + d_k^C$ for $k \in \{1, 2, \cdots, N_C\}$ and $C \in \mathcal{C}$.

$$\begin{aligned} \max \quad & \textstyle\sum_T \rho(T) q(\boldsymbol{\theta}^T) - \sum_C \rho'(C) \sum_k p_k^C, \\ \text{s.t.} \quad & \textstyle\sum_T \rho(T) \boldsymbol{\theta}^T + \sum_C \rho'(C) \sum_k q_k^C \equiv \boldsymbol{\theta}, \\ & ||\mathbf{u}_k^C|| \leq v_k^C, \forall k \in \{1, 2, \cdots, N_C\}, C \in \mathcal{C}. \end{aligned} \tag{17}$$

where

$$p_k^C = (\mathbf{b}_k^C)^\top \mathbf{u}_k^C + d_k^C v_k^C, \tag{18}$$

$$q_k^C = (\mathbf{A}_k^C)^\top \mathbf{u}_k^C + \mathbf{c}_k^C v_k^C. \tag{19}$$

Here $\mathbf{u}_k^C$ and v_k^C are some slack variables.

We can define up to h^c SOC constraints for a cycle/clique, where c is the size of the cycle/clique (i.e. $N_C \in \{0, 1, \cdots, h^c\}$). Before proceeding further, we also define the following problem:

$$\begin{aligned} \max \quad & \textstyle\sum_{T \in C} \rho(T) q(\boldsymbol{\theta}^T) - \rho'(C) \sum_k p_k^C, \\ \text{s.t.} \quad & \textstyle\sum_{T \in C} \rho(T) \boldsymbol{\theta}^T + \rho'(C) \sum_k q_k^C = \boldsymbol{\theta}^C, \\ & ||\mathbf{u}_k^C|| \leq v_k^C, \forall k \in \{1, 2, \cdots, N_C\}, \end{aligned} \tag{20}$$

where $\boldsymbol{\theta}^C$ is some parameter vector. The variables of the above problem are restricted to $\mathbf{u}_k^C$, v_k^C, $\theta_{a;i}^{T1}$ and $\theta_{ab;ij}^{T2}$ where $(a,b) \in \mathcal{E}_T \bigcap \mathcal{E}_C$. Like problem (12), we can solve problem (20) using standard Interior Point algorithms for small cycles/cliques C.

Similar to TRW-S(LP-C), a convergent algorithm can now be described for solving the dual (17). This algorithm differs from TRW-S(LP-C) in only step 4, where it solves problem (20) for a cycle/clique C instead of problem (12). We refer to this algorithm as either TRW-S(SOCP-C) or TRW-S(SOCP-Q) depending upon the SOCP relaxation that we are solving. When using the TRW-S(SOCP-Q) algorithm, we include all slack variables corresponding to the cycle inequalities defined over the cycles in clique C. It can easily be shown that both TRW-S(SOCP-C) and TRW-S(SOCP-Q) satisfy all the properties given in § 3.1. Note that, like TRW-S and TRW-S(LP-C), these algorithms do not directly provide a labelling for the random variables of the CRF. Instead we use the procedure described in § 3.1 to obtain the final solution.

5. Experiments

We tested the approaches described in this paper using both synthetic and real data. For synthetic data experiments, we closely follow the setup of (Kolmogorov, 2006). We show that our algorithms overcome a well-known deficiency of TRW-S, namely that it does not provide good MAP estimates for non-submodular energy functions. Next, we consider the problem of segmentation using real data and show favourable comparison between our methods and several other standard MAP estimation techniques.

5.1. Synthetic Data

Datasets: We conducted two sets of experiments using binary grid CRFs (i.e. $h = |\mathbf{l}| = 2$) of size 30×30. In the first experiment the edges of the graphical model, i.e. $\mathcal{E}$, were defined using a 4-neighbourhood system while the second experiment used an 8-neighbourhood system. Similar to (Kolmogorov, 2006), the unary potentials $\theta_{a;0}^1$ and $\theta_{a;1}^1$ were generated using the normal distribution $\mathcal{N}(0,1)$. The pairwise potentials $\theta_{ab;00}^2$ and

$\theta^2_{ab;11}$ were set to 0 while $\theta^2_{ab;01}$ and $\theta^2_{ab;10}$ were generated using $\mathcal{N}(0, \sigma^2)$. For both experiments, 50 CRFs were generated using the method described above. All the CRFs defined non-submodular energy functions (i.e. there exists an $(a, b) \in \mathcal{E}$ such that $\theta_{ab;01} + \theta_{ab;10} < 0$) which are in general NP-hard to minimize. As noted in (Kolmogorov, 2006), TRW-S performs considerably worse than BP on such examples.

Implementation Details: We tested the LP-C and the SOCP-C relaxations in the first experiment. Constraints were defined on all cycles of size 4. The LP-C and SOCP-Q relaxation were tested in the second experiment. Cycles inequalities were defined on all cycles of size 3. In addition, for SOCP-Q, SOC constraints were defined on all cliques of size 4. In both the experiments, our algorithms were tested using trees defined by individual edges of the graphical model for ease of implementation. In other words, a tree $T = (\mathbf{v}_T, \mathcal{E}_T) \in \mathcal{T}$ such that $\mathbf{v}_T = \{v_a, v_b\}$ and $\mathcal{E}_T = \{(a, b)\} \subseteq \mathcal{E}$. However, we note here that our algorithms are general and can be applied for any choice of trees. Although our current set of trees are quite restrictive, the results show that they outperform several state of the art algorithms. The TRW-S algorithm, as well as other standard approaches, was tested using the publically available code which uses monotonic chains as trees.

The terms $\rho(T)$ and $\rho'(C)$ were set to $1/|\mathcal{T}|$ and $1/|\mathcal{C}|$ respectively for all $T \in \mathcal{T}$ and $C \in \mathcal{C}$. We found it sufficient to define one cycle inequality per cycle C using a set of labels $\{l_{i_1}, l_{i_2}, \cdots, l_{i_c}\}$ which satisfies

$$\sum_{(a_k,a_m)\in\mathcal{E}_F} \theta_{a_k a_m; i_k i_m} - \sum_{(a_k,a_m)\in\mathcal{E}_C-\mathcal{E}_F} \theta^2_{a_k a_m; i_k i_m} \geq \sum_{(a_k,a_m)\in\mathcal{E}_F} \theta_{a_k a_m; j_k j_m} - \sum_{(a_k,a_m)\in\mathcal{E}_C-\mathcal{E}_F} \theta^2_{a_k a_m; j_k j_m},$$

for all sets of labels $\{l_{j_1}, \cdots, l_{j_c}\}$. Here $\mathcal{E}_C = \{(a_1, a_2), \cdots, (a_n, a_1)\}$ and $\mathcal{E}_F \subseteq \mathcal{E}_C$ such that $|\mathcal{E}_F| = 3$. As proposed in (Kumar et al., 2007), we also define only one SOC constraint per cycle/clique when considering the SOCP-C and the SOCP-Q relaxations. At each iteration, problems (12) and (20) were solved using the MOSEK software (available at http://www.mosek.com).

Results: Figure 1 (a) shows the results obtained for the first experiment using $\sigma = \frac{10}{\sqrt{d}}$ (where d is the degree of the variables in the graphical model). Note that since the energy functions are non-submodular, TRW-S provides labellings with higher energies than BP as observed in (Kolmogorov, 2006). However, the additional constraints in the LP-C and SOCP-C algorithm enable us to obtain labelling with lower energies than BP. Further, unlike BP, they also provide us with the value of the dual at each iteration. This value allows us to find out how close we are to the global optimum (since the energy of the optimal labelling cannot be less than the value of the dual). Also note that the value of the LP-C dual is greater than the value of the SOCP-C dual. This provides empirical evidence that LP-C dominates SOCP-C as conjectured in (Kumar et al., 2007).

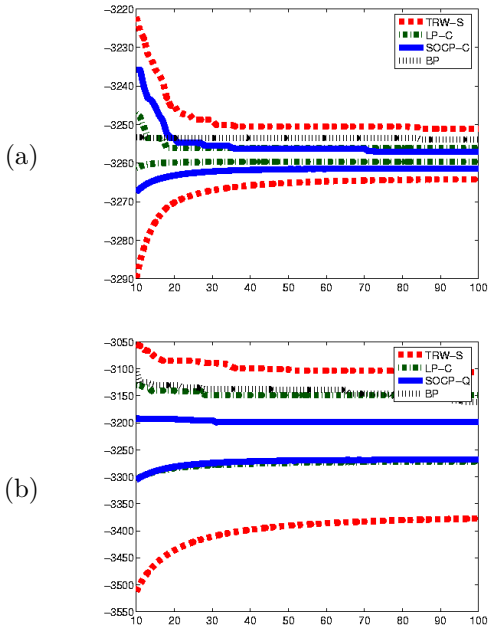


Figure 1. Results of the synthetic data experiment. **(a)** First experiment. The x-axis shows the iteration number. The lower curves show the average value of the dual at each iteration over 50 random CRFs while the upper curves show the average energy of the best labelling found till that iteration. The additional constraints in the LP-C and SOCP-C relaxations enable us to obtain labellings with lower energy compared to TRW-S and BP. Cycle inequalities provide a better approximation than the SOC constraint of the SOCP-C relaxation. **(b)** Second experiment. Note that the value of the dual obtained using SOCP-Q is greater than the value of the dual of the LP-C relaxation.

The results of the second experiment are shown in Figure 1 (b) using $\sigma = \frac{10}{\sqrt{d}}$. Again, BP outperforms TRW-S, while LP-C and SOCP-Q provide better approximations. The SOC constraints defined over cliques in SOCP-Q provide a greater value of the dual compared to the LP-C relaxation. The complexity and timings for all the algorithms are given in tables 3 and 4.

5.2. Real Data - Segmentation

We now present the results of our method on interactive segmentation (Boykov & Jolly, 2001) where, given some seed pixels for all the segments present in an image, we wish to obtain the segmentation of the image.

Problem Formulation: The problem of obtaining the segmentation of an image can be cast within the CRF framework. Specifically, we define a CRF over random variables $\mathbf{v} = \{v_0, \cdots, v_{n-1}\}$, where each variable

Algorithm	No. of Var.	No. of Cons.	Time(sec)
BP	-	-	0.0018
TRW-S	$nh + \|\mathcal{E}\|h^2$	$n + 2\|\mathcal{E}\|h$	0.0018
LP-C	$nh + \|\mathcal{E}\|h^2$	$2n + 2\|\mathcal{E}\|h$	7.5222
SOCP-C	$nh + \|\mathcal{E}\|h^2$	$2n + 2\|\mathcal{E}\|h$	8.9091

Table 3. Complexity and timings of the algorithms for the first synthetic data experiment with a 4-neighbourhood relationship. Recall that $n = |\mathbf{v}|$ is the number of random variables, $h = |\mathbf{l}|$ is the size of the label set and $\mathcal{E}$ is the neighbourhood relationship defined by the CRF. The second and third columns show the number of variables and constraints in the primal problem respectively. The fourth column shows the average time of the each algorithm for one iteration (in seconds). All timings are reported for a Pentium IV 3.3 GHz processor with 2GB RAM.

Algorithm	No. of Var.	No. of Cons.	Time(sec)
BP	-	-	0.0027
TRW-S	$nh + \|\mathcal{E}\|h^2$	$n + 2\|\mathcal{E}\|h$	0.0027
LP-C	$nh + \|\mathcal{E}\|h^2$	$5n + 2\|\mathcal{E}\|h$	7.7778
SOCP-Q	$nh + \|\mathcal{E}\|h^2$	$6n + 2\|\mathcal{E}\|h$	9.1170

Table 4. Complexity and timings for the second synthetic data experiment with an 8-neighbourhood relationship. Note that SOCP-Q includes all the constraints of LP-C.

corresponds to a pixel of the frame. Each label in the set $\mathbf{l} = \{l_0, \cdots, l_{h-1}\}$ corresponds to a segment (where h is the total number of segments). The unary potential of assigning a variable v_a to segment l_i is specified by the negative log-likelihood of the RGB value of pixel a given the seed pixels of the segment l_i. The pairwise potentials encourage continuous segments whose boundaries lie on image edges. For more details, we refer the reader to (Boykov & Jolly, 2001). The problem of obtaining the segmentation of a frame then boils down to that of finding the MAP estimate of the CRF.

Datasets and Implementation Details: We used the well-known 'Garden' sequence to conduct our experiments (with frame size 120×175). The seed pixels were provided using the ground truth segmentation of a keyframe as shown in Fig. 2.

Similar to the synthetic data experiment, we defined the trees as individual edges of the graphical model of the CRF for our algorithms. Other algorithms were tested using publically available code (including TRW-S which uses monotonic chains as trees). We specified one cycle inequality and one SOC constraint for each cycle/clique (as described in the previous section). The terms $\rho(T)$ and $\rho'(C)$ were set to $1/|\mathcal{T}|$ and $1/|\mathcal{C}|$ respectively for all $T \in \mathcal{T}$ and $C \in \mathcal{C}$. Once again, problems (12) and (20) were solved using MOSEK.

Results: For the first set of experiments, we used a 4-neighbourhood system and tested the following algorithms: TRW-S, LP-C, SOCP-C, $\alpha\beta$-swap, α-expansion and BP. Fig. 3 shows the segmentations (of frames

Figure 2. Segmented keyframe of the 'Garden' sequence. The left image shows the keyframe while the right image shows the corresponding segmentation provided by the user. The four different colours indicate pixels belonging to the four segments namely sky, house, garden and tree.

Algorithm	Avg. Time-1 (s)	Avg. Time-2 (s)
BP	0.1400	0.1740
TRW-S	0.1400	0.1740
$\alpha\beta$-swap	0.1052	0.1201
α-expansion	0.1100	0.1240
LP-C	140.3320	142.2226
SOCP-C/SOCP-Q	143.6365	144.9890

Table 5. Average timings of the algorithms (per iteration) for the first experiment on video segmentation with a 4-neighbourhood relationship (column 2) and the second experiment with an 8-neighbourhood relationship (column 3). Again, all timings are reported for a Pentium IV 3.3 GHz processor with 2GB RAM.

other than the keyframe) and the values of the energy function obtained for all algorithms. Note that, by incorporating additional constraints using all cycles of length 4, LP-C and SOCP-C outperform other methods. Further, the cycle inequalities in LP-C provide better results than the SOC constraints of SOCP-C. Table 5 provides the average time for all algorithms.

The second set of experiments used an 8-neighbourhood system and tested the following algorithms: TRW-S, LP-C, SOCP-Q, $\alpha\beta$-swap, α-expansion and BP. For the LP-C algorithm, cycle inequalities were specified for all cycles of size 3. In addition, the SOCP-Q algorithm specifies SOC constraints on all cliques of size 4. Fig. 4 shows the segmentations and energies obtained for all the algorithms. The average timings per iteration are shown in table 5. Note that, similar to the synthetic data examples, SOCP-Q outperforms LP-C by incorporating additional SOC constraints.

6. Discussion

We extended the LP-S relaxation based approach of (Kolmogorov, 2006; Wainwright et al., 2005) for the MAP estimation problem. Specifically, we showed how cycle inequalities and SOC constraints can be incorporated within the TRW framework. We also proposed convergent algorithms for solving the resulting duals. Our experiments indicate that these additional constraints provide a more accurate approximation for MAP estimation when the energy function is non-submodular. Although our algorithm is much faster than Interior Point methods, it is slower than TRW-S and BP. An interesting direction for future re-

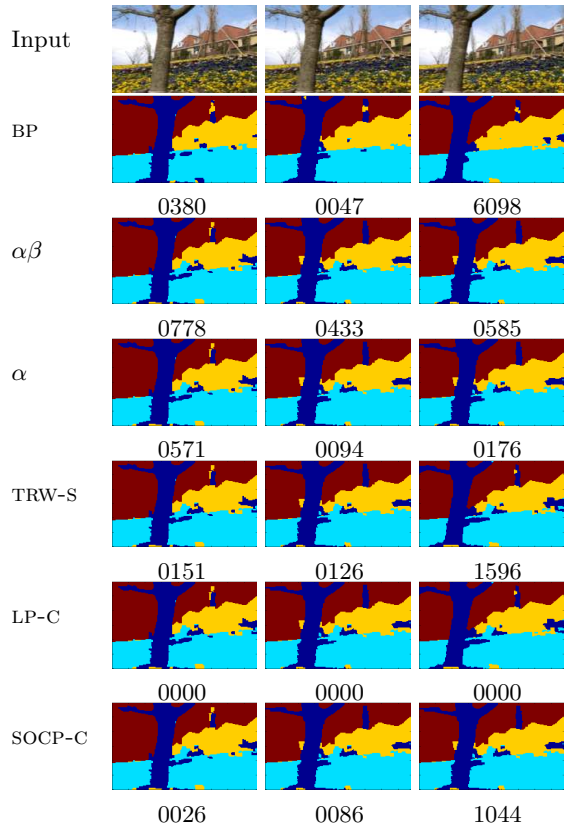


Figure 3. Segmentations obtained for the 'Garden' video sequence using 4-neighbourhood. The corresponding energy values (scaled up to integers for using $\alpha\beta$-swap and α-expansion) of all the algorithms are shown below the segmentation. The following constant terms are subtracted from the energy values of all algorithms for the three frames respectively (to make minimum energy among all algorithms 0): 5139499, 5145234 and 5126941.

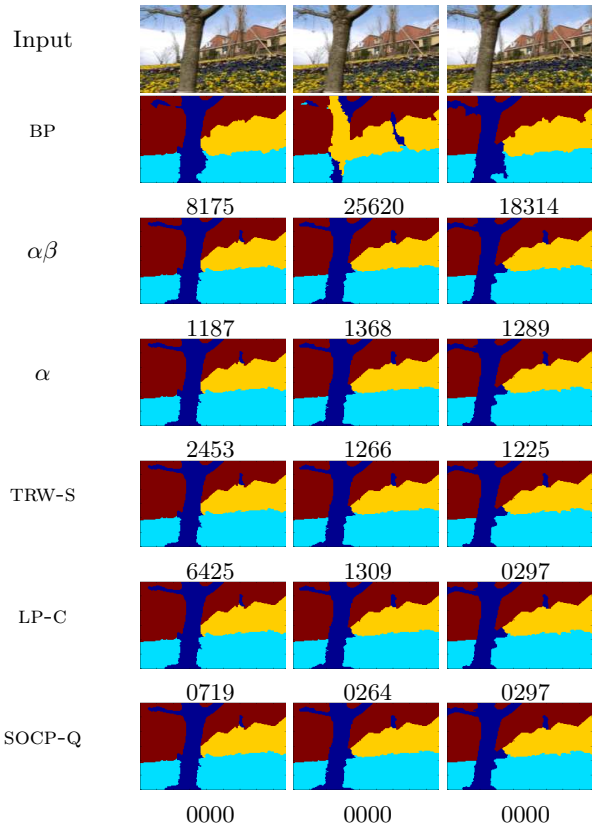


Figure 4. Segmentations obtained for the 'Garden' video sequence using 8-neighbourhood. The corresponding energy values (reduced by 5304466, 5299756 and 5292224 for the three frames respectively) are also shown.

search would be to develop specialized algorithms for solving problems (12) and (20) (which are used in our approach).

References

Barahona, F., & Mahjoub, A. (1986). On the cut polytope. *Mathematical Programming*, *36*, 157–173.

Boykov, Y., & Jolly, M. (2001). Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. *ICCV* (pp. I: 105–112).

Fitzpatrick, P. (2006). *Advanced calculus.* Thompson Brooks/Cole.

Goemans, M., & Williamson, D. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, *42*, 1115–1145.

Karzanov, A. (1998). Minimum 0-extension of graph metrics. *European Journal of Combinatorics*, *19*, 71–101.

Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *PAMI*, *28*, 1568–1583.

Kumar, M. P., Kolmogorov, V., & Torr, P. H. S. (2007). An analysis of convex relaxations for MAP estimation. *NIPS*.

Kumar, M. P., & Torr, P. H. S. (2008). *Efficiently solving convex relaxations for MAP estimation* (Technical Report). Oxford Brookes University.

Meltzer, T., Yanover, C., & Weiss, Y. (2005). Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. *ICCV*.

Schlesinger, M. (1976). Sintaksicheskiy analiz dvumernykh zritelnikh singnalov v usloviyakh pomekh. *Kibernetika*, *4*, 113–130.

Sontag, D., & Jaakkola (2007). New outer bounds on the marginal polytope. *NIPS*.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., & Rother, C. (2006). A comparative study of energy minimization methods for markov random fields. *ECCV* (pp. II: 16–29).

Wainwright, M., Jaakola, T., & Willsky, A. (2005). MAP estimation via agreement on trees: Message passing and linear programming. *IEEE Trans. on Information Theory*, *51*, 3697–3717.

Wiegernick, W. (2005). Approximations with reweighted generalized belief propagation. *AISTATS*.

Zwick, U. (1999). Outward rotations: A tool for rounding solutions of semidefinite relaxations, with applications to MAX CUT and other problems. *STOC* (pp. 679–687).

On the Hardness of Finding Symmetries in Markov Decision Processes

Shravan Matthur Narayanamurthy SHRAVANMN@GMAIL.COM
Yahoo! Labs, Bangalore

Balaraman Ravindran RAVI@CSE.IITM.AC.IN
Dept. of Comp. Sci. and Engg., Indian Institute of Technology Madras, Chennai

Abstract

In this work we address the question of finding symmetries of a given MDP. We show that the problem is *Isomorphism Complete*, that is, the problem is polynomially equivalent to verifying whether two graphs are isomorphic. Apart from the theoretical importance of this result it has an important practical application. The reduction presented can be used together with any off-the-shelf Graph Isomorphism solver, which performs well in the average case, to find symmetries of an MDP. In fact, we present results of using NAutY (the best Graph Isomorphism solver currently available), to find symmetries of MDPs.

1. Introduction

Markov Decision Processes (MDPs) are widely employed to model sequential decision problems. But current solution techniques for MDPs do not scale well with the size of the MDPs, and hence are proving inadequate in solving large real-world problems. While building abstract models of real-world problems, it can be seen that a high degree of redundancy is present which can be exploited to reduce size of the model. This reduction in size could possibly lead to more efficient solution methods.

One such notion of redundancy is a degree of symmetry that is present in any real-world problem. (Amarel, 1968) first looked at exploiting such symmetries in solving a missionaries and cannibals problem. In this work we use the notion of symmetries in MDPs introduced in (Ravindran, 2004). While it is widely believed that finding symmetries in MDPs is a hard problem, no one has investigated before exactly how hard this problem is.

Intuitively this seems harder than finding symmetries in graphs, due to the additional structure introduced by MDPs. In this work we show that finding symmetries in MDPs is no harder than the problem of graph isomorphism. We also show that existing graph isomorphism solvers can be used to find symmetries in MDPs.

We present some notation in the next section, and some related work in Section 3. In Section 4 we formally define the problem, and present a constructive algorithm in Section 5 for showing the equivalence to graph isomorphism. We discuss some results Section 6 and conclude in Section 7.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Homomorphisms and Symmetry Groups

Let B be a partition of a set X. For any $x \in X, [x]_B$ denotes the block of B to which x belongs. Any function f from a set X to a set Y induces a partition (or equivalence relation) on X, with $[x]_f = [x']_f$ if and only if $f(x) = f(x')$ and x, x' are f-equivalent written $x \equiv_f x'$. Let B be a partition of $Z \subseteq X \times Y$, where X and Y are arbitrary sets. The projection of B onto X is the partition $B|X$ of X such that for any $x, x' \in X, [x]_{B|X} = [x']_{B|X}$ if and only if every block containing a pair in which x is a component also contains a pair in which x' is a component or every block containing a pair in which x' is a component also contains a pair in which x is a component.

Definition 1. An *MDP homomorphism* h from an MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ to an MDP $\mathcal{M}' = \langle S', A', \Psi', P', R' \rangle$ is a surjection from Ψ to Ψ', defined by a tuple of surjections $< f, \{g_s | s \in S\} >$, with $h((s,a)) = (f(s), g_s(a))$, where $f : S \rightarrow S'$ and $g_s : A_s \rightarrow A'_{f(s)}$ for $s \in S$, such that: $\forall s, s' \in S, a \in A_s$

$$P'(f(s), g_s(a), f(s')) = \sum_{s'' \in [s']_f} P(s, a, s'') \quad (1)$$

$$R'(f(s), g_s(a)) = R(s, a) \quad (2)$$

We use the shorthand $h(s,a)$ for $h((s,a))$. Often for convenience, we use $< f, \{g_s\} >$ to denote $< f, \{g_s | s \in S\} >$.

Definition 2. Let $\mathcal{M}$' be an image of the MDP $\mathcal{M}$ under homomorphism $h =< f, \{g_s\} >$. For any $s \in S, g_s^{-1}(a')$ denotes the set of actions that have the same image $a' \in A'_{f(s)}$ under g_s. Let π' be a stochastic policy in $\mathcal{M}'$. Then π' lifted to $\mathcal{M}$ is the policy $\pi_{\mathcal{M}'}$ such that for any $a \in g_s^{-1}(a'), \pi'_{\mathcal{M}}(s,a) = \pi'(f(s),a')/|g_s^{-1}(a')|$

Definition 3. An MDP homomorphism $h =< f, \{g_s\} >$ from MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to MDP $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$ is an MDP *isomorphism* from $\mathcal{M}$ to $\mathcal{M}'$ if and only if f and g_s, are bijective. $\mathcal{M}$ is said to be *isomorphic* to $\mathcal{M}'$ and vice versa. An MDP isomorphism from MDP $\mathcal{M}$ to itself is called an *automorphism* of $\mathcal{M}$.

Definition 4. The set of all automorphisms of an MDP $\mathcal{M}$, denoted by $Aut\mathcal{M}$, forms a group under composition of homomorphisms. This group is the *symmetry group* of $\mathcal{M}$.

Let $\mathcal{G}$ be a subgroup of $Aut\mathcal{M}$. The image of $\mathcal{M}$ under $\mathcal{G}$ is called the *$\mathcal{G}$-reduced image* of $\mathcal{M}$.

Definition 5. An MDP $\mathcal{M}'$ is said to be a *reduced model* of an MDP $\mathcal{M}$, iff there exists an MDP homomorphism $h : \mathcal{M} \to \mathcal{M}'$.

3. Related Work

MDP Minimization is a well studied problem. As stated earlier, in the model minimization approach, a reduced MDP that that preserves some key properties as the original MDP is found by combining "equivalent" states. The reduced MDP found depends on the notion of equivalence between states used in the aggregation. The notion of equivalence chosen will be fundamental in designing and analyzing algorithms for reducing MDPs. In (Dean & Givan, 1997) a minimization algorithm is proposed based on the notion of *stochastic bi-simulation homogeneity*. Informally, a partition of the state space for an MDP is said to be homogeneous if for each action, states in the same block have the same probability of transitioning to each other block. They also provide an algorithm for finding the coarsest homogeneous refinement of any partition of the state space of an MDP. The algorithm starts with an initial partition P_0 and iteratively refines it by splitting the blocks until the coarsest homogeneous refinement of P_0 is obtained. A notion of stability of a block with respect to another is defined and unstable blocks are split till all blocks of the partition are stable. The complexity of the algorithm is expressed in terms of the partition manipulation operations. Hence, the actual complexity depends on the underlying partition representation and manipulation algorithms. (Givan et al., 2003) discuss the application of the algorithm to solving factored MDP problems. Enumerating the state space is avoided by describing large blocks of equivalent states in factored form with the block descriptions being inferred directly from the original factored representation.

(Ravindran, 2004) proposes a more generic framework based on the notion of MDP homomorphisms with *state-dependent action recoding* as introduced in Section 2. This allows a greater reduction in problem size and aids in modeling many other notions of equivalence like symmetries. A polynomial time algorithm to find the reduced model under the notion of MDP homomorphisms is also proposed by extending the algorithm proposed by (Givan et al., 2003) and (Lee & Yannakakis, 1992). Again, the algorithm is polynomial in the number of block operations, the stability criterion is modified to suit the equivalence notion and the same process of iterative splitting is used. The notion of stability used is called the *stochastic substitution property*, which is an extension of the *substitution property* for finite state machines (Hartmanis, 1966).

However, literature on MDP minimization using symmetries is sparse. (Zinkevich & Balch, 2001) define symmetries based on state-action equivalence but do not make any connections to group-theoretic concepts or minimization algorithms.

Another dimension to analyze the literature is the approach to symmetry finding. Two main approaches exist:

1. To derive a set of necessary conditions for elements to be symmetric
2. Prove Isomorphism Completeness and use a graph isomorphism finding system

Intuitively symmetries seem easier to identify than homomorphisms and we tried the first approach to find a polynomial time algorithm for symmetry finding, along the lines of the MDP homomorphism finding, with the motivation of finding better algorithms for MDP minimization. The MDP homomorphism definition allows for deriving this easily because, two state action pairs $(s_1, a_1), (s_2, a_2)$ are homomorphically equivalent if

$$\begin{aligned} h(s_1,a_1) &= h(s_2,a_2) \\ P'(f(s_1), g_{s_1}(a_1), f(s')) &= P'(f(s_2), g_{s_2}(a_2), f(s')) \\ T(s_1,a_1,[s']_{B_h|S}) &= T(s_2,a_2,[s']_{B_h|S}) \end{aligned}$$

for all $s' \in S$. This is the stochastic substitution property and it allows us to deal just with blocks without worrying about the actual functions. However, a similar attempt for symmetries still needs the symmetry f in the necessary condition as below:

$$\begin{aligned} h(s_1,a_1) &= (s_2,a_2) \\ P(f(s_1), g_{s_1}(a_1), f(s')) &= P(s_2,a_2,f(s')) \\ P(s_1,a_1,s') &= P(s_2,a_2,f(s')) \end{aligned}$$

(Flener et al., 2002) and (Crawford, 1992) point that symmetry finding for CSPs in general is Isomorphism Com-

plete. However, there also exist results showing that symmetry finding is NP-complete (in case of geometric automorphism of graphs (Manning, 1990)). So we were still unclear whether symmetry finding for MDPs is Isomorphism Complete or NP-complete due to the presence of factorially many action recoding functions. A better understanding of the use of symmetries for abstraction in MDPs is the motivation for this work.

4. Problem Definition

To exploit the power of abstraction using symmetries, we identify them and construct a reduced model by abstracting away the symmetric portions. As the reduced model can be significantly smaller, it can be easier to solve. We use the notion of automorphisms to model symmetries. So formally, given an MDP $\mathcal{M}$,

1. Find the automorphism group, $Aut\mathcal{M}$ and
2. Given the automorphism group, $Aut\mathcal{M}$ find the corresponding reduced model, the $Aut\mathcal{M}$-Reduced Image

5. Finding Symmetries

5.1. Problem Simplification

Let us consider the first part of our problem, i.e., given an MDP $\mathcal{M}$, find the automorphism group of $\mathcal{M}$, $Aut\mathcal{M}$. We know that a group can be specified using its generators. So we simplify the problem to finding the generators of $Aut\mathcal{M}$. Let $AMGEN(\mathcal{M})$ denote the problem of finding the generators of $Aut\mathcal{M}$. We write $A \propto B$ if a problem A is polynomially reducible to B. We say that problems A and B are polynomially equivalent iff $A \propto B$ and $B \propto A$. We denote polynomial equivalence by $\equiv_\propto$.

Definition 6. A problem A is *Isomorphism Complete* iff A is polynomially equivalent to finding whether two graphs are isomorphic.

Let G_1, G_2 be two simple graphs unless otherwise mentioned. The following is a list of relevant *Isomorphism Complete* problems (Booth & Colbourn, 1977) on graphs:

- $ISO(G_1, G_2)$: Isomorphism recognition for G_1 and G_2
- $IMAP(G_1, G_2)$: Isomorphism Map from G_1 to G_2(if it exists),
- $AGEN(G_1)$: Generators of the automorphism group, $AutG_1$
- $DGEN(G)$: Generators of the automorphism group, $AutG$, where G is a weighted digraph

From (Mathon, 1979), (Read & Corneil, 1977), (Miller, 1977) we have,
$DGEN(G) \equiv_\propto AGEN(G) \equiv_\propto IMAP(G_1, G_2) \equiv_\propto ISO(G_1, G_2)$.

We intend to prove that $AMGEN(\mathcal{M})$ is *Isomorphism Complete*. We are done if we prove that $AMGEN(\mathcal{M}) \equiv_\propto DGEN(G_\mathcal{M})$, where $G_\mathcal{M}$ is a weighted graph constructed in polynomial time from $\mathcal{M}$, that is, $AMGEN(\mathcal{M}) \propto DGEN(G_\mathcal{M})$ and $DGEN(G_\mathcal{M}) \propto AMGEN(\mathcal{M})$. It is easy to see that $DGEN(G_\mathcal{M}) \propto AMGEN(\mathcal{M})$ is true because we can always construct a degenerate MDP from a digraph. So we need to prove that $AMGEN(\mathcal{M}) \propto DGEN(G_\mathcal{M})$.

5.2. *Isomorphism Completeness* of the problem

An MDP $\mathcal{M}$ can be considered as a pseudograph with states acting as vertices and actions acting as edges. Since there can be more than one action affecting the transition between 2 states, we need to represent this as a pseudograph. The transition probabilities and rewards can be thought of as weight functions. Next, we formally pose $AMGEN(\mathcal{M})$ as a problem on a weighted pseudograph.

Let $G_\mathcal{M} =< \Sigma_a, V, E, W_P, W_R >$ be the pseudograph corresponding to $\mathcal{M}$, where

Σ_a : Alphabet for labelling corresponding to actions
V : Set of vertices corresponding to states
E : Set of edges, where each edge is a triple (u, a, v) where, $u, v \in V$ and $a \in \Sigma_a$ corresponding to state transitions

W_P : $E \to \mathbb{R}$ corresponding to transition probabilities
W_R : $E \to \mathbb{R}$ corresponding to rewards with $W_R(u, a, v) = W_R(u, a, v')$ $\forall\ (u, a, v), (u, a, v') \in E$

Note, $E = \bigcup_{u,v \in V} E_{uv}$ where, $E_{uv} = \{\ (u', a, v') \in E \mid u' = u \text{ and } v' = v\ \}$

$AMGEN(\mathcal{M})$ can be formulated as finding the generators of the group of bijections $h : V \times \Sigma_a \to V \times \Sigma_a$. h is defined by $h(u, a) = (f(u), g_u(a))$, where

f : $V \to V$ and
g_u : $\Sigma_a \to \Sigma_a$ defined for each $u \in V$ are bijections s. t.

$W_P(f(u), g_u(a), f(v)) = W_P(u, a, v)$ and
$W_R(f(u), g_u(a), f(v)) = W_R(u, a, v)\ \forall\ (u, a, v) \in E$

These two components of each generator can be interpreted as follows:

1. f is a function that permutes the states/vertices
2. The set of functions $\{g_u\}$ defined for each state/vertex permutes the actions/edge labels. These are called the State-Dependent Action Recoding (SDAR) functions.

5.2.1. Set Bijections

Let us assume, for a moment, that we have a procedure that constructs a weighted digraph $WD_{\mathcal{M}}$ from $G_{\mathcal{M}}$. Now, solving $DGEN(WD_{\mathcal{M}})$ gives the generators of $WD_{\mathcal{M}}$. Even if these were somehow same as the fs we are looking for, we still need a way to find the SDAR functions. To achieve this, we define the notion of a *set bijection* which represents a set of bijections very compactly. In the worst case, for each f, there can be factorially many SDAR functions. So a normal explicit representation cannot be used. We also define the operations of intersection between two *set bijections* to find the bijections that are common to both *set bijections*, composition between two *set bijections* and an inverse of a *set bijection*. All these operations can be done in time polynomial of the number of elements in the domain of a bijection belonging to the *set bijection*.

Definition 7. Consider two finite sets A and B. Let $U_A = \{U_{A_1}, U_{A_2}, \ldots, U_{A_k}\}$ and $U_B = \{U_{B_1}, U_{B_2}, \ldots, U_{B_k}\}$ be partitions of A and B respectively. U_A and U_B are said to be *similar* iff $|U_A| = |U_B|$ and for each $U_{A_i} \in U_A$ there exists a unique $U_{B_j} \in U_B$ such that $|U_{A_i}| = |U_{B_j}|$. We denote it by $U_A \sim U_B$.

Note that, by definition the sets A and B will be of the same size.

Definition 8. Let A and B be two finite sets and $U_A = \{U_{A_1}, U_{A_2}, \ldots, U_{A_k}\}$ and $U_B = \{U_{B_1}, U_{B_2}, \ldots, U_{B_k}\}$ be partitions of A and B respectively such that $U_A \sim U_B$. A bijective map $X : U_A \to U_B$ where $X(U_{A_i}) = U_{B_j}$ implies $|U_{A_i}| = |U_{B_j}|$ for all $U_{A_i} \in U_A$ is called a *set bijection*.

Informally, a *set bijection* can be interpreted as representing a set of bijections from A to B. $X(U_{A_i}) = U_{B_j}$ represents all possible bijective mappings from elements in U_{A_i} to elements in U_{B_j}. A bijection from A to B in the set of bijections that represent the *set bijection*, can be formed by collating mappings from each $X(U_{A_i}) = U_{B_j}$. The *set bijection* represents all mappings that can be formed by such collations. To formalize this notion, we define the *interpretation function* next.

Let $X_{AB} \triangleq \{$ all bijections $X : U_A \to U_B$ such that U_A and U_B are similar partitions of A and B respectively $\}$ be the set of all *set bijections*. Let $2^{S_{|V|}}$ be the powerset set of all permutations from $A \to B$. Define, $\hat{I} : X_{AB} \to 2^{S_{|V|}}$ such that $\hat{I}(X : U_A \to U_B) = \{$ all bijections $l : A \to B \mid l(x \in U_{A_i}) \in X(U_{A_i})\ \forall U_{A_i} \in U_A\}$. Evidently, $\hat{I}$ is only injective and not surjective as there exist sets of $2^{S_{|V|}}$ that cannot be represented by a *set bijection*. For example, consider the set of bijections, between $\{a, b, c\}$ and $\{1, 2, 3\}$, $L = \{(a \to 1, b \to 2, c \to 3), (a \to 2, b \to 1, c \to 3), (a \to 2, b \to 3, c \to 1)\}$. Clearly there does not exist an $X : U_A \to U_B$ such that $\hat{I}(X) = L$. All we can say is that there exists an X such that $L \subset \hat{I}(X)$. To get a bijective interpretation function, we define, $I : X_{AB} \to image(\hat{I})$ such that $I(X : U_A \to U_B) = \hat{I}(X : U_A \to U_B)$. Clearly I is a bijection and we call this the interpretation function.

Definition 9. Let A be a finite set and let $U^1_A = \{U^1_{A_1}, U^1_{A_2}, \ldots, U^1_{A_k}\}$, $U^2_A = \{U^2_{A_1}, U^2_{A_2}, \ldots, U^2_{A_k}\}$ be two partitions of A such that, $U^1_A \sim U^2_A$. We define the intersection of two similar partitions of a finite set as $U^1_A \cap U^2_A = \{U^1_{A_i} \cap U^2_{A_j} \mid U^1_{A_i} \in U^1_A, U^2_{A_j} \in U^2_A \text{ and } U^1_{A_i} \cap U^2_{A_j} \neq \emptyset\}$.

Definition 10. Let A and B be two finite sets and $U^1_A = \{U^1_{A_1}, U^1_{A_2}, \ldots, U^1_{A_k}\}$, $U^2_A = \{U^2_{A_1}, U^2_{A_2}, \ldots, U^2_{A_k}\}$ be two partitions of A and $U^1_B = \{U^1_{B_1}, U^1_{B_2}, \ldots, U^1_{B_k}\}$, $U^2_B = \{U^2_{B_1}, U^2_{B_2}, \ldots, U^2_{B_k}\}$ be two partitions of B. Also let $U^1_A \sim U^1_B$ and $U^2_A \sim U^2_B$. Let two *set bijections* X_1 and X_2 be defined from U^1_A to U^1_B and from U^2_A to U^2_B respectively. If $(U^1_A \cap U^2_A) \sim (U^1_B \cap U^2_B)$, we define the intersection between the two set bijections $X = X_1 \cap X_2$ as follows: $\forall U^1_{A_i} \in U^1_A, U^2_{A_j} \in U^2_A$ such that $U^1_{A_i} \cap U^2_{A_j} \neq \emptyset$, $X(U^1_{A_i} \cap U^2_{A_j}) = X_1(U^1_{A_i}) \cap X_2(U^2_{A_j})$. Note that, $X : U^1_A \cap U^2_A \to U^1_B \cap U^2_B$ and it can be shown that $I(X) = I(X_1) \cap I(X_2)$.

Definition 11. Let A be a finite set. Let $U^1_A = \{U^1_{A_1}, U^1_{A_2}, \ldots, U^1_{A_k}\}$, $U^2_A = \{U^2_{A_1}, U^2_{A_2}, \ldots, U^2_{A_k}\}$ be two *similar* partitions of A. Let X be a *set bijection* defined from U^1_A to U^2_A. We define the *inverse* of X as $X^{-1} : U^2_A \to U^1_A$ such that $X^{-1}(U^2_{A_i}) = U^1_{A_j}$ iff $X(U^1_{A_j}) = U^2_{A_i}$.

Definition 12. Let A, B and C be three finite sets and $U_A = \{U_{A_1}, U_{A_2}, \ldots, U_{A_k}\}$, $U_B = \{U_{B_1}, U_{B_2}, \ldots, U_{B_k}\}$ and $U_C = \{U_{C_1}, U_{C_2}, \ldots, U_{C_k}\}$ be partitions of A, B and C respectively . Also let U_A, U_B and U_C be pairwise similar to each other. Let two *set bijections* X_1 and X_2 be defined from U_B to U_C and U_A to U_B respectively. We define the composition of X_1 and X_2, $X = X_1 \odot X_2$ as the *set bijection* from U_A to U_C defined by $X(U_{A_i}) = X_1(X_2(U_{A_i}))$, for each $U_{A_i} \in U_A$. It can be shown that for each $l \in I(X)$ there exist, $l_1 \in I(X_1)$ and $l_2 \in I(X_2)$ such that $l = l_1 \circ l_2$ where $\circ$ denotes normal function composition.

5.2.2. Vector-Weighted Digraph

We assume that Σ_a can be ordered and let O be such an ordering.

Without loss of generality, we can assume that $|E_{uv}| =$

$k, \forall u, v \in V$ because, we can always take $max_{u,v \in V}|E_{uv}| = k$ and if $\exists u, v \in V$ such that $(u, a, v) \in E$ for some $a \in \Sigma_a$ and $|E_{uv}| < k$, then add dummy labels (chosen from the remaining labels in Σ_a) and zero weights to make $|E_{uv}| = k$. This corresponds to the general assumption in MDPs that $|A_s| = k, \forall s \in S$.

Let $< a_1, a_2, \ldots, a_k >$ ordered as per O be the k-tuple representing the label of each edge in E_{uv}. This being the same for all edges, we leave out labeling from the graph definition.

Now we define the vector-weighted digraph corresponding to $\mathcal{M}$, $VWG_{\mathcal{M}} =< V, E', \mathbf{W_P}, \mathbf{W_R} >$, as follows:

$$
\begin{aligned}
E' &= \{(u,v) \mid \exists a \in \Sigma_a \text{ and } (u,a,v) \in E\} \\
\mathbf{W_P} &: E' \to \mathbb{R}^k \text{ defined by} \\
&\quad \mathbf{W_P}(\mathbf{u},\mathbf{v}) =< W_P(u,a_1,v), \ldots, W_P(u,a_k,v) > \\
\mathbf{W_R} &: E' \to \mathbb{R}^k \text{ defined by} \\
&\quad \mathbf{W_R}(\mathbf{u},\mathbf{v}) =< W_R(u,a_1,v), \ldots, W_R(u,a_k,v) >
\end{aligned}
$$

where $a_1, a_2, \ldots, a_k$ are ordered as per O.

5.2.3. Sorted Vector-Weighted Digraph

We define the sorted vector-weighted digraph, $SVWG_{\mathcal{M}} =< V, E', \mathbf{W_{P_s}}, \mathbf{W_{R_s}} >$, as follows:

$$
\begin{gathered}
\mathbf{W_{P_s}} : E' \to \mathbb{R}^k \text{ defined by} \\
\mathbf{W_{P_s}}(\mathbf{u},\mathbf{v}) =< W_P(u, p_{uv}(1), v), \ldots, W_P(u, p_{uv}(k), v) > \\
\text{where, } p_{uv} : \mathbb{N}_k \to \Sigma_a \text{ such that} \\
W_P(u, p_{uv}(1), v) \leq \ldots \leq W_P(u, p_{uv}(k), v) \\
\mathbf{W_{R_s}} : E' \to \mathbb{R}^k \text{ defined by} \\
\mathbf{W_{R_s}}(\mathbf{u},\mathbf{v}) =< W_R(u, r_{uv}(1), v), \ldots, W_R(u, r_{uv}(k), v) > \\
\text{where, } r_{uv} : \mathbb{N}_k \to \Sigma_a \text{ such that} \\
W_R(u, r_{uv}(1), v) \leq \ldots \leq W_R(u, r_{uv}(k), v)
\end{gathered}
$$

Note that, p_{uv} and r_{uv} are not unique. So we choose them such that the order O is preserved.

5.2.4. *Set Bijections* that sort the vector-weights

Here we show that there exists a *set bijection* whose interpretation is the set of permutations that sort the vector-weights. Let $\mathbb{N}_k$ be the set of first k natural numbers. Let $D^P_{uv} \triangleq \{$ all permutations $l : \mathbb{N}_k \to \Sigma_a \mid l$ sorts $\mathbf{W_P}(\mathbf{u},\mathbf{v})\}$ be defined for each $(u,v) \in E'$. So, $\mathbf{W_{P_s}}(\mathbf{u},\mathbf{v}) =< W_P(u,l(1),v), \ldots, W_P(u,l(k),v) >$ and $W_P(u,l(1),v) \leq W_P(u,l(2),v) \leq \ldots \leq W_P(u,l(k),v)$. Clearly, $\mathbb{N}_k$ can be partitioned into $U^{uv}_{\mathbb{N}_k} = \{\mathbb{N}^1_k, \mathbb{N}^2_k, \ldots, \mathbb{N}^j_k\}$ such that, $\forall t \in \mathbb{N}^y_k$, $W_P(u,l(t),v)$ has the same value for each $y = 1, 2, \ldots, j$ and if $t \in \mathbb{N}^y_k$ and $t' \in \mathbb{N}^{y+1}_k$ then $W_P(u,l(t),v) < W_P(u,l(t'),v)$. This partition induces a corresponding partition $U^{uv}_{\Sigma_a} = \{\Sigma^1_a, \Sigma^2_a, \ldots, \Sigma^j_a\}$ where $\Sigma^i_a = \{l(t) \mid t \in \mathbb{N}^i_k\}$. Since, each l sorts $\mathbf{W_P}(\mathbf{u},\mathbf{v})$, they satisfy the property that $l(x \in \mathbb{N}^i_k) \in \Sigma^i_a$. Therefore, there exists a set bijection $Q^P_{uv} : U^{uv}_{\mathbb{N}_k} \to U^{uv}_{\Sigma_a}$ such that, $I(Q^P_{uv}) = D^P_{uv}$.

Using a similar procedure, we can show that there exists set bijection $Q^R_{uv} : U^{uv}_{\mathbb{N}_k} \to U^{uv}_{\Sigma_a}$ whose interpretation is the set of permutations that sort $\mathbf{W_R}(\mathbf{u},\mathbf{v})$.

Let $Q_{uv} = Q^P_{uv} \cap Q^R_{uv}$. If $Q_{uv} = \emptyset$, then there doesn't exist an automorphism for the MDP $\mathcal{M}$.

5.2.5. Weighted Digraph

Now we define the weighted digraph $WG_{\mathcal{M}} =< V, E', W' >$ as follows:

$$
\begin{aligned}
W' &: E' \to \mathbb{R} \text{ such that } W'(u,v) = m(\mathbf{W_{P_s}}(u,v). \\
&\quad \mathbf{W_{R_s}}(u,v)) \text{ where m is a bijection from } \mathbb{R}^{2k} \to \mathbb{R} \\
&\quad \text{and . denotes concatenation}
\end{aligned}
$$

Algorithm 1 Construction

1: Given $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$
2: Let *SOLN* be an empty set
3: Construct the pseudograph $G_{\mathcal{M}} =< \Sigma_a, V, E, W_P, W_R >$ as defined in Section 5.2
4: Construct the vector-weighted digraph $VWG_{\mathcal{M}} =< V, E', \mathbf{W_P}, \mathbf{W_R} >$ as defined in Section 5.2.2
5: Construct the sorted vector-weighted digraph $SVWG_{\mathcal{M}} =< V, E', \mathbf{W_{P_s}}, \mathbf{W_{R_s}} >$ as defined in Section 5.2.3
6: **for** each $(u,v) \in E'$ **do**
7: Compute Q^P_{uv} and Q^R_{uv} by finding the partition of $\mathbb{N}_k$ as described in Section 5.2.4
8: $Q_{uv} \leftarrow Q^P_{uv} \cap Q^R_{uv}$
9: **if** $Q^P_{uv} \cap Q^R_{uv}$ does not exist **then**
10: **exit**
11: **end if**
12: **end for**
13: Construct the weighted digraph $WG_{\mathcal{M}} =< V, E', W' >$ using m as described in Section 5.2.5
14: $F \leftarrow DGEN(WG_{\mathcal{M}})$ where F is the set of generators of $AutWG_{\mathcal{M}}$
15: **for** each $f \in F$ **do**
16: **for** each $(u,v) \in E'$ **do**
17: $G_{uv} \leftarrow Q_{f(u)f(v)} \odot Q^{-1}_{uv}$
18: **end for**
19: Let $\hat{H^f}$ be an empty set
20: **for** each $u \in V$ **do**
21: $G_u \leftarrow G_{uv}$ from some $v \in V$
22: **for** each $v \in V$ **do**
23: $G_u \leftarrow G_u \cap G_{uv}$
24: **end for**
25: Add G_u to $\hat{H^f}$
26: **end for**
27: Add $< f, \hat{H^f} >$ to *SOLN*
28: **end for**

5.2.6. CONSTRUCTION

The procedure for finding symmetries of an MDP $\mathcal{M}$ is given in Algorithm 1.

The complexity of the algorithm is as follows. The construction steps in lines 3 to 5, are at most polynomial in $|E|$. Using a constant access time data structure like a hashtable, Q^P_{uv} and Q^R_{uv} can be constructed in $O(|E_{uv}|)$ time.The intersection takes $O(|E_{uv}|^2)$ time. Since this runs for $|E'|$ iterations, computation of Q_{uv} is at most polynomial in $|E|$. Since m is known, the construction of weighted digraph in line 13, is polynomial in $|E|$. With the use of procedures that return at most $|V|$ automorphisms of $AutWG_{\mathcal{M}}$ (Mathon, 1979), the construction of G_u for each f, from lines 15 to 26, runs for at most $|V|$ iterations.

The most expensive part of the loop from lines 20 to 26 is the computation of $|V|^2$ intersections. But this is still polynomial in $|V||E|$ time. Hence the algorithm takes polynomially more time than the solution time of *DGEN*. Also to extract a solution from *SOLN*, we need to extract $|V|$ SDAR functions from $\hat{H^f}$ for each f, which takes $|E_{uv}|$ time if we use a constant access time data structure. So extraction of a solution takes $O(|V|^2|E|)$ which is still polynomial in $|V||E|$. While one can intuitively see that the reduction is indeed polynomial time, the proof is presented in an associated technical report (Narayanamurthy & Ravindran, 2008), due to lack of space.

5.3. Significance

The above result is significant both theoretically and practically. Practically speaking, the reduction to Graph Isomorphism allows us to use any of the numerous off-the-shelf Graph Isomorphism solvers to find symmetries on MDPs. In fact, we use NAutY - No Automorphisms, Yes?, the best Graph Isomorphism solver currently available (Skiena, 1997) to find out symmetries in MDPs. NAutY solves *AGEN*(*G*). It uses backtracking and a refinement procedure to find the canonical labeling. If two different labelings lead to the same graph, then an automorphism can be found using these labelings (McKay, 1981). In the worst case it can take exponential time. So it allows the use of a variety of vertex invariants, which act like heuristics, to solve harder problems. However, for random graphs with n vertices and edge probability 0.5, average execution times for large n are about n^2 nanosecs.We use NAutY in the fourteenth line in the construction, where we need to solve *DGEN*(*G*). We first convert the weighted digraph into an unweighted digraph using standard procedure. We then use NAutY to find the generators of the automorphism group of the so found digraph. From these we extract generators of *AutWG* as per the above procedure. We present some results in Section 6.

6. Results

The experiments were run on the following two domains. We describe results per domain.

6.1. Probabilistic GridWorld

The domain is an $N \times N$ GridWorld with four probabilistic actions of going UP, DOWN, RIGHT and LEFT having a 90% success probability. The initial state was (0,0) and the goal states were $\{(0, N-1), (N-1, 0)\}$. We used Algorithm 1 to find the symmetries with NAutY being used as the *DGEN* solver. We then used the symmetries to find the partition of Ψ. We were able to find the partition corresponding to the symmetry group, that is, for a grid of size $M \times N$, states (x,y), (y,x), (M-1-x,N-1-y) and (N-1,M-1-x) are equivalent. We present the time taken by the algorithm for GridWorlds of different sizes.

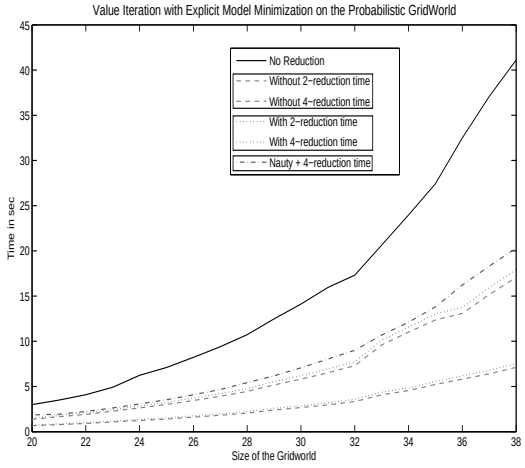


Figure 1. Average running times of the value iteration algorithm with explicit model minimization on Probabilistic GridWorld vs size of the GridWorld. Each of the 3 sets should be compared with the graph for no reduction. Curves in a set represent different degrees of symmetry. Each set shows the time reduction with reduced model usage. First one discounts the time taken to find symmetries and for reduction. The next set includes the time for reduction but discounts time taken to find symmetries. The last one includes both the time taken to find symmetries using NAutY and time for reduction.

To complete the end-to-end approach, we ran the $\mathcal{G}$-reduced image algorithm, presented in (Ravindran, 2004), to find the reduced image and ran the Value Iteration algorithm on the reduced image. To show the efficiency of reduction, we show the time taken for reduction and solution separately. We also present the case of a handcrafted 2-folded symmetry which is used with the $\mathcal{G}$-reduced image algorithm and reduced model is used with Value Iteration.

From Figure 1 it is evident that the reduced model con-

struction is efficient and adds little overhead. However, the results of the end-to-end approach show a significant overhead due to symmetry finding. It cuts the saving by almost half. Still the results are significant because they double the size of the largest GridWorld that can be solved in some given time.

6.2. GridWorld Soccer

The domain is a soccer-inspired grid domain. It is a slightly modified version of that described in (Bowling, 2003). We first describe the original version of the domain and then state the modification.

It is an $M \times N$ grid with two agents. One is denoted the attacker (**A**) who holds the ball and the other as the defender (**B**) who tries to snatch the ball from the attacker. The center lines/grids(depending on whether M and N are even or odd) for both x-axis and y-axis are chosen naturally. The state is defined by the non-identical positions of the attacker and the defender. This defines the state space with $(MN)^2 - (MN)$ states. The actions are movements in the four compass directions: **N**, **E**, **W**, **S** and the hold action **H**. It is a single player game, in that, only the attacker chooses actions deliberately while the defender executes random actions. The action chosen by the attacker and the random action of the defender are executed in random order, which determines the next state. However if the defender tries to move into the attacker's location then the state is unchanged and if the attacker tries to move into the defender's location, the game is reset to the initial state which is shown in Figure 2. The right hand section of the grid is the attacker's half and the left hand section that of the defender. The goal is chosen to be situated beyond the first column of grids occupying one grid on each side of the y-axis central line/grid. A **W** action from the squares in front of the goal state leads to a goal with a reward of 1 and to the end of an episode. Everywhere else the reward is 0. A 5×4 domain is shown in Figure 2.

Intuitively, the domain seems symmetric around the y-axis center line. However, the results of using Algorithm 1 on this domain showed us that the domain is not symmetric due to the existence of the reset action when the attacker tries to move into the defender's position. So we modified the domain to have symmetric reset, that is, reset happens to the initial state and its symmetric state around the y-axis center line with equal probability. This makes the domain symmetric as per intuition, which the algorithm confirms.

Interestingly, the algorithm also finds that the existence of the hold action adds further symmetry. The grids along the border of the domain act as walls. For example, the northern wall stops the **N** action leaving the state unchanged which is the same result if the agent were to execute a **H** action. These additional symmetries which we did not think

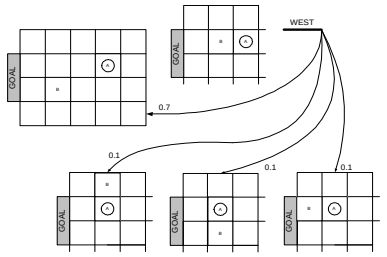


Figure 2. Single Player grid soccer where agent **B** selects it actions randomly. The initial state is shown on the left and an example of transitions and associated probabilities are given for a particular state and action on the right. Notice that fifty percent of the time **A**'s actions are executed first causing it to lose the ball and the game reset to the initial state. In addition, if **B** selects **H** or **E** it does not move and so **A** still loses the ball and returns to the initial state. The other outcomes are equiprobable.

of before running algorithm were found by the algorithm. This suggests that there might exist complicated symmetries that will be discovered by the algorithm, which are hard to find, even upon a close examination. Also in many cases, symmetries are size invariant. So we can use the algorithm on a relatively smaller version of the domain and find symmetries which might still hold on the larger version.

We present the time taken by the algorithm for different sizes. An increment of one here means an increase of one along both axes. The presence of two agents, blows up the state space very rapidly and we hit the limit on the order of the graph imposed by NAutY very soon (for a 11×10 grid).To present similar graphs as in the probabilistic GridWorld case, we use the explicit model minimization approach with Value Iteration. The results are presented in Figure 3.

In this case, we find that the overheads due to the construction and the $\mathcal{G}$-reduced image algorithm is negligible. Though efficiency of the $\mathcal{G}$-reduced image algorithm is expected, the efficiency of the construction can be possibly because of the structure of the domain yielding an easy graph to find automorphisms on.

7. Conclusion

In this work, we have provided a constructive proof for the *Isomorphism Completeness* of the problem of finding symmetries. We have also proposed the use of this constructive proof along with an efficient minimization algorithm to solve an MDP using symmetries and demonstrated it empirically. As part of future work, we are looking at adapting approximation algorithms for finding graph isomorphisms to finding approximate symmetries in MDPs.

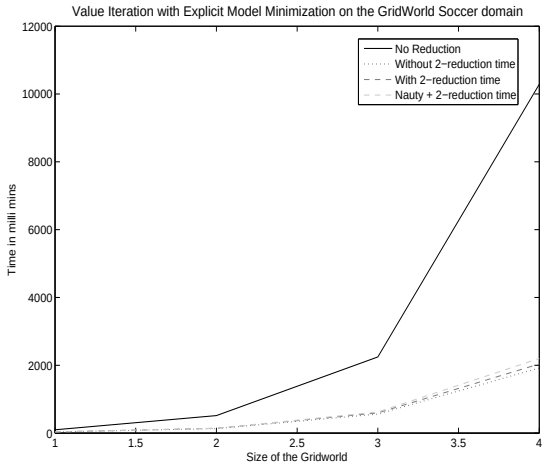


Figure 3. Average running times of the value iteration algorithm with explicit model minimization on GridWorld Soccer domain vs size of the domain. Size of one represents the 5×4 grid. Thereafter an increment of one means an increment of one along both axes. Each graph should be compared with the graph for no reduction. The other graphs show the time reduction with reduced model usage. First one discounts the time taken to find symmetries and for reduction. The next one includes the time for reduction but discounts time taken to find symmetries. The last one includes both the time taken to find symmetries using NAutY and time for reduction.

Acknowledgements

We would like to thank the reviewers for their valuable comments and inputs. We would also like to thank Google Research for supporting Dr. Ravindran's participation in the conference.

References

Amarel, S. (1968). On representations of problems of reasoning about actions. In D. Michie (Ed.), *Machine intelligence 3*, vol. 3, 131–171. Amsterdam, London, New York: Elsevier/North-Holland. Amarel, S.

Booth, K. S., & Colbourn, C. J. (1977). *Problems polynomially equivalent to graph isomorphism* (Technical Report). University of Waterloo.

Bowling, M. (2003). *Multiagent learning in the presence of agents with limitations*. Doctoral dissertation, Carnegie Mellon University.

Crawford, J. (1992). A theoretical analysis of reasoning by symmetry in first-order logic.

Dean, T., & Givan, R. (1997). Model minimization in markov decision processes. *AAAI/IAAI* (pp. 106–111).

Flener, P., Frisch, A., Hnich, B., Kiziltan, Z., Miguel, I., Pearson, J., & Walsh, T. (2002). Breaking row and column symmetries in matrix models.

Givan, R., Dean, T., & Greig, M. (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, *147*, 163–223.

Hartmanis, J. (1966). *Algebraic structure theory of sequential machines (prentice-hall international series in applied mathematics)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Lee, D., & Yannakakis, M. (1992). Online minimization of transition systems (extended abstract). *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing* (pp. 264–274). New York, NY, USA: ACM Press.

Manning, J. B. (1990). *Geometric symmetry in graphs*. Doctoral dissertation, Purdue University.

Mathon, R. (1979). A note on the graph isomorphism counting problem. *Information Processing Letters*, *8*, 131–132.

McKay, B. D. (1981). Practical graph isomorphism. *Congressus Numerantium*, *30*, 45–87.

Miller, G. L. (1977). Graph isomorphism, general remarks. *STOC '77: Proceedings of the ninth annual ACM symposium on Theory of computing* (pp. 143–150). New York, NY, USA: ACM Press.

Narayanamurthy, S. M., & Ravindran, B. (2008). *On the hardness of finding symmetries in markov decision processes* (Technical Report IITMCSETR-01-08). Indian Institute of Technology, Madras.

Ravindran, B. (2004). *An algebraic approach to abstraction in reinforcement learning*. Doctoral dissertation, Department of Computer Science, University of Massachusetts Amherst.

Read, R. C., & Corneil, D. G. (1977). The graph isomorphism disease. *Journal of Graph Theory I*, 339–363.

Skiena, S. (1997). The stony brook algorithm repository.

Zinkevich, M., & Balch, T. (2001). Symmetry in markov decision processes and its implications for single agent and multiagent learning. *Proceedings of the ICML-01* (pp. 632–640). Morgan Kaufmann.

Bayes Optimal Classification for Decision Trees

Siegfried Nijssen SIEGFRIED.NIJSSEN@CS.KULEUVEN.BE
K.U. Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract

We present an algorithm for exact Bayes optimal classification from a hypothesis space of decision trees satisfying leaf constraints. Our contribution is that we reduce this classification problem to the problem of finding a rule-based classifier with appropriate weights. We show that these rules and weights can be computed in linear time from the output of a modified frequent itemset mining algorithm, which means that we can compute the classifier in practice, despite the exponential worst-case complexity. In experiments we compare the Bayes optimal predictions with those of the maximum a posteriori hypothesis.

1. Introduction

We study the problem of Bayes optimal classification for density estimation trees. A density estimation tree in this context is a decision tree which has a probability density for a class attribute in each of its leaves. One can distinguish two Bayesian approaches to density estimation using a space of such trees.

In the first approach a single *maximum a posteriori* (MAP) density estimation tree is identified first:

$$T = \arg\max_T P(T|\mathbf{X}, \vec{y}),$$

where $\mathbf{X}$ and $\vec{y}$ together constitute the training data. The posterior probability $P(T|\mathbf{X}, \vec{y})$ of a hypothesis T is usually the product of a prior and a likelihood. The MAP hypothesis can then be used to classify a test example x' using the densities in the leaves.

The second approach is to marginalize over all possible trees, instead of preferring a single one:

$$\arg\max_c P(c|x', \mathbf{X}, \vec{y}) = \arg\max_c \sum_T P(c|x', T)P(T|\mathbf{X}, \vec{y}).$$

Predictions that are performed using this second approach are called *Bayes optimal predictions*. It has been claimed that "no single tree classifier using the same prior knowledge as an optimal Bayesian classifier can obtain better performance on average" (Mitchell, 1997). The Bayesian point of view is that Bayesian averaging cancels out the effects of overfitted models (Buntine, 1990), and "solves" overfitting problems.

This claim was challenged by Domingos (2000). Domingos demonstrated experimentally that an ensemble of decision trees that are weighted according to posterior probabilities performs worse than uniformly weighted hypotheses. It was found that one overfitting tree usually dominates an ensemble.

However, these results were obtained by sampling from the hypothesis space. Even though Domingos argued that similar issues should also occur in the truly optimal approach, this claim could not be checked in practice as the exact computation of Bayes optimal predictions was considered to be impractical. Indeed, in (Chipman et al., 1998) it was already claimed that "exhaustive evaluation ... over all trees will not be feasible, except in trivially small problems, because of the sheer number of trees". Similar claims were made in other papers studying Bayesian tree induction (Buntine, 1992; Chipman et al., 1998; Angelopoulos & Cussens, 2005; Oliver & Hand, 1995), and have led to the use of sampling techniques such as Markov Chain Monte Carlo sampling.

In this paper we present an algorithm that can be used to evaluate Domingos' claim in a reasonable number of non-trivial settings. Our algorithm allows us to exactly compute the Bayes optimal predictions given priors that assign non-zero probability to trees that satisfy certain constraints. An example of a constraint is that every leaf covers a significant number of examples; this constraint has been used very often in the literature (Buntine, 1992; Quinlan, 1993; Chipman et al., 1998; Angelopoulos & Cussens, 2005; Oliver & Hand, 1995).

Our algorithm is an extension of our earlier work, in

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

which we developed the DL8 algorithm for determining one tree that maximizes accuracy (Nijssen & Fromont, 2007). DL8 is based on dynamic programming on a pre-computed lattice of itemsets, and scans these itemsets decreasing in size. Its time complexity is linear in the size of the lattice. In this paper we extend this algorithm to a Bayesian setting. From a technical point of view, the main contribution is that we prove that a different pass over the lattice allows us to perform Bayes optimal predictions without increasing the asymptotic complexity of building the lattice.

The task that our algorithm addresses is similar to the task addressed in (Cleary & Trigg, 1998). Compared to this earlier work, we study the more common Dirichlet priors also considered in (Chipman et al., 1998; Angelopoulos & Cussens, 2005); furthermore, by exploiting the link to itemset mining, our algorithm is more efficient, and its results are more interpretable.

The paper is organized as follows. Notation and concepts are introduced in Section 2. Bayes optimal classification is formalized in Section 3. We show how to map this problem to the problem of finding itemsets and building a classifier with weighted rules in Section 4. Experiments are performed in Section 5.

2. Preliminaries

Before we are ready to formalize our problem and our proposed solution, we require some notation. We restrict ourselves to binary data; we assume that data is converted in this form in a preprocessing step. The data is stored in binary matrix $\mathbf{X}$, of which each row $\vec{x}_k$ corresponds to one example. Every example $\vec{x}_k$ has a class label y_k out of a total number of C class labels. Class labels are collected in a vector $\vec{y}$.

We assume that the reader is familiar with the concept of decision trees (see (Breiman et al., 1984; Quinlan, 1993) for details). Essential in our work is a link between decision trees and *itemsets*. Itemsets are a concept that was introduced in the data mining literature (Agrawal et al., 1996). If $\mathcal{I}$ is a domain of items, $I \subseteq \mathcal{I}$ is an itemset. In our case, we assume that we have *two* types of items: for every attribute there is a positive item i that represents a positive value, and a negative item $\neg i$ that represents a negative value. An example $\vec{x}$ can be represented as an itemset

$$\{i | x_i = 1\} \cup \{\neg i | x_i = 0\}.$$

Thus, for a data matrix with n columns, we have that $\mathcal{I} = \mathcal{I}_{pos} \cup \mathcal{I}_{neg}$, where $\mathcal{I}_{pos} = \{1, 2, \ldots n\}$ and $\mathcal{I}_{neg} = \{\neg 1, \neg 2, \ldots \neg n\}$. We overload the use of the $\subseteq$ operator: when I is an itemset, and $\vec{x}$ is an example, we use $I \subseteq \vec{x}$ to denote that I is a subset of $\vec{x}$ after translating $\vec{x}$ into an itemset.

Every sequence of test outcomes in a decision tree, starting from the root of the tree to an arbitrary node deeper down the tree, can be represented as an itemset. For instance, a decision tree with B in the root, and A in its right-hand branch can be represented by:

$$T = \{\emptyset, \{B\}, \{\neg B\}, \{\neg B, A\}, \{\neg B, \neg A\}\}.$$

Every itemset in T corresponds to one node in the tree. By $\mathcal{T}$ we denote all subsets of $2^{\mathcal{I}}$ that represent decision trees. A decision tree structure is an element $T \in \mathcal{T}$. Consequently, when T is a decision tree we can write $I \in T$ to determine if the itemset I corresponds to a path occurring in the tree.

An itemset is an unordered set: given an itemset in a tree, we cannot derive from this itemset in which order its tests appear in the tree. This order can only be determined by considering all itemsets in a tree T.

We are not always interested in all nodes of a tree. The subset of itemsets that correspond to the leaves of a tree T will be denoted by $leaves(T)$; in our example,

$$leaves(T) = \{\{B\}, \{\neg B, A\}, \{\neg B, \neg A\}\}.$$

The most common example of a decision tree is the *classification tree*, in which every leaf is labeled with a single class. In a *density estimation tree*, on the other hand, we attach a class *distribution* to each leaf, represented by a vector $\vec{\theta}_I$; for each class c this vector contains the probability θ_{Ic} that examples $\vec{x} \supseteq I$ belong to class c. All the parameters of the leaves of a tree are denoted by $\mathbf{\Theta}$. The vectors in $\mathbf{\Theta}$ are thus indexed by the itemsets representing leaves of the tree.

For the evaluation of a tree T on a binary matrix $\mathbf{X}$, it is useful to have a shorthand notation for the number of examples covered by a leaf:

$$f(I, \mathbf{X}) = |\{\vec{x}_k | \vec{x}_k \supseteq I\}|;$$

usually we omit the matrix $\mathbf{X}$ in our notation, as we assume the training data to be fixed. We call $f(I, \mathbf{X})$ the *frequency* of I. Class-based frequency is given by:

$$f_c(I, \mathbf{X}, \vec{y}) = |\{\vec{x}_k | \vec{x}_k \supseteq I, y_k = c\}|.$$

The *frequent itemset mining* problem is the problem of finding all $I \subseteq \mathcal{I}$ such that $f(I) \geq \gamma$, for a given threshold γ. Many algorithms for computing this set exist (Agrawal et al., 1996; Goethals & Zaki, 2003). They are based on the property that the frequency constraint is *anti-monotonic*. A binary constraint p on itemsets is called anti-monotonic iff $\forall I' \subseteq I : p(I) =$

$true \implies p(I') = true$. Consequently, these algorithms do not need to search through all supersets $I' \supseteq I$ of an itemset I that is found to be infrequent.

One application of itemsets is in the construction of rule-based classifiers (CMAR (Li et al., 2001) is an example). Many rule-based classifiers traverse rules sequentially when predicting examples. Here, we study a rule-based classifier that derives a prediction from all rules through voting. Such a classifier can be seen as a subset $\mathcal{P} \subseteq 2^{\mathcal{I}}$ of itemsets, each of which has a weight vector $\vec{w}(I)$. We predict an example $\vec{x}$ by computing

$$\arg\max_c \sum_{\{I \in \mathcal{P} | I \subseteq \vec{x}\}} w_c(I),$$

where we thus pick the class that gets most votes of all rules in the ruleset; each rule votes with a certain weight on each class. The aim of this paper is to show that we can derive a set of itemsets $\mathcal{P}$ and weights $\vec{w}(I)$ for all $I \in \mathcal{P}$ such that the predictions of the rule-based classifier equal those of a Bayes optimal classifier. The rules in $\mathcal{P}$ represent all paths that can occur in trees in the hypothesis space.

3. Problem Specification

In this section we formalize the problem of Bayes optimal classification for a hypotheses space of decision trees. Central in the Bayesian approach is that we first define the probability of the data given a tree structure T and parameters $\mathbf{\Theta}$:

$$P(\vec{y}|\mathbf{X}, T, \mathbf{\Theta}) = \prod_{I \in leaves(T)} \prod_{c=1}^{C} (\theta_{Ic})^{f_c(I)}$$

In Bayes optimal classification we are interested in finding for a particular example $\vec{x}'$ the class y' which maximizes the probability

$$\begin{aligned} y' &= \arg\max_c P(c|\vec{x}', \mathbf{X}, \vec{y}) \qquad (1) \\ &= \arg\max_c \sum_{T \in \mathcal{T}} \int_{\mathbf{\Theta}} P(c|\vec{x}', T, \mathbf{\Theta}) P(T, \mathbf{\Theta}|\mathbf{X}, \vec{y}) d\mathbf{\Theta}, \end{aligned}$$

where we sum over the space of all decision trees and integrate over all possible distributions in the leaves of each tree. Applying Bayes' rule on the second term, and observing that $\mathbf{\Theta}$ is dependent on the tree T, we can rewrite this into

$$\sum_{T \in \mathcal{T}} \int_{\mathbf{\Theta}} P(c|\vec{x}', T, \mathbf{\Theta}) P(\vec{y}|T, \mathbf{\Theta}, \mathbf{X}) P(\mathbf{\Theta}|T, \mathbf{X}) P(T|\mathbf{X}) d\mathbf{\Theta}; \qquad (2)$$

in this formula $P(T|\mathbf{X})$ is the probability of a tree given that we have seen all data except the class labels. Our method is based on the idea that we can constrain the space of decision trees by manipulating this term.

A first possibility is that we set $P(T|\mathbf{X}) = 0$ if there is a leaf $I \in leaves(T)$ such that $f(I) < \gamma$, for a frequency threshold γ. We call such leaves *small leaves*. The class estimates of a small leaf are often unreliable, and it is therefore common in many algorithms to consider only large leaves.

Additionally, we can set $P(T|\mathbf{X}) = 0$ if the depth of the decision tree exceeds a predefined threshold.

Both limitations impose *hard constraints* on the trees that are considered to be feasible estimators. We denote trees in $\mathcal{T}$ that satisfy all hard constraints by $\mathcal{L}$.

In the simplest case we can assume a uniform distribution on the trees that satisfy the hard constraints. Effectively, this would mean that we set $P(\mathbf{\Theta}|T, \mathbf{X}) = 1$ in Equation 2 for all $T \in \mathcal{L}$. However, we will study a more sophisticated prior in this paper to show the power of our method. The aim of this prior, which was proposed in (Chipman et al., 1998), is to give more weight to smaller trees; it can be seen as a *soft constraint*. This prior is defined as follows.

$$P(T|\mathbf{X}) = \prod_{I \in T} P_{node}(I, T, \mathbf{X})$$

Here, the term $P_{node}(I, T, \mathbf{X})$ is defined as follows.

$$P_{node}(I, T, \mathbf{X}) = \begin{cases} P_{leaf}(I, \mathbf{X}), & \text{if } I \text{ is a leaf in } T; \\ P_{intern}(I, \mathbf{X}), & \text{if } I \text{ is internal in } T; \end{cases}$$

where

$$P_{leaf}(I, \mathbf{X}) = \begin{cases} 0, & \text{if } f(I) < \gamma \text{ or } |I| > \delta; \\ 1, & \text{else if } |I| = \delta \text{ or } e(I) = 0; \\ 1 - \alpha(1 + |I|)^{-\beta}, & \text{otherwise;} \end{cases}$$

and

$$P_{intern}(I, \mathbf{X}) = \begin{cases} 0, & \text{if } f(I) < \gamma \text{ or } |I| \geq \delta \\ & \text{or } e(I) = 0; \\ \alpha(1 + |I|)^{-\beta}/e(I), & \text{otherwise;} \end{cases}$$

Here $e(I)$ is the size of the set $\{i \in \mathcal{I}_{pos} | f(I \cup i) \geq \gamma \wedge f(I \cup \neg i) \geq \gamma\}$, which consists of all possible tests that can still be performed to split the examples covered by itemset I.

The term $\alpha(1 + |I|)^{-\beta}$ makes it less likely that nodes at a higher depth are split. The term $e(I)$ determines how many tests are still available if a test is to be performed. We assume that tests are apriori equally likely, independent of the order in which the previous tests on the path have been performed. An alternative could be to give more likelihood to tests that are well-balanced.

Note that P_{leaf} and P_{intern} are computed for an itemset I almost independently from the tree T: we only need to know if I is a leaf or not.

As common in Bayesian approaches, we assume that the parameters in every leaf of the tree are Dirichlet distributed with the same parameter vector $\vec{\alpha}$, i.e.

$$P(\mathbf{\Theta}|T,\mathbf{X}) = P(\mathbf{\Theta}|T) = \prod_{I \in leaves(T)} Dir(\vec{\theta}_I|\vec{\alpha}),$$

where

$$Dir(\vec{\theta}_I|\vec{\alpha}) = \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \prod_c {\theta_{Ic}}^{\alpha_c - 1},$$

and Γ is the gamma function.

Finally, it can be seen that

$$P(c|\vec{x}', T, \mathbf{\Theta}) = \theta_{I(T,\vec{x}')c},$$

where $I(T,\vec{x}')$ is the leaf of T for which $I \subseteq \vec{x}'$.

We now have formalized all terms of Equation 2.

4. Solution Strategy

An essential step in our solution strategy is the construction of the set

$$\mathcal{P} = \{I | T \in \mathcal{L}, I \in T\},$$

which consists of all itemsets in trees that satisfy the hard constraints. Only these paths are needed when we wish to compute the posterior distribution over class labels, and are used as rules in our rule-based classifier. The weights of these rules are obtained by rewriting the Bayesian optimization criterion for a test example $\vec{x}'$ (Equation 1) as

$$\arg\max_c \sum_{\{I \in \mathcal{P} | I \subseteq \vec{x}'\}} w_c(I)$$

where

$$w_c(I) = \sum_{T \in \mathcal{L},\ \text{has leaf } I} \left(\prod_{I \in T} P_{node}(I,T,\mathbf{X}) \right) \left(\int_{\mathbf{\Theta}} \theta_{Ic} \prod_{I' \in leaves(T)} Dir(\vec{\theta}'_I|\vec{\alpha}) \prod_c \theta_{I'_c}^{f_c(I')} d\mathbf{\Theta} \right). \quad (3)$$

The idea behind this rewrite is that the set of all trees in $\mathcal{L}$ can be partitioned by considering in which leaf a test example ends up. An example ends in exactly one leaf in every tree, and thus every tree belongs to one partition as determined by that leaf. We sum first over all possible leaves that can contain the example, and then over all trees having that leaf. The weights of the rules in our classifier consist of the terms $w_c(I)$, and will be computed from the training data in the training phase; the sum of the weights $w_c(I)$ is computed for a test example in the classification phase.

This rewrite shows that in the training phase we need to compute weights for all itemsets that are in $\mathcal{P}$. We will discuss now how to compute these.

In the formulation above we multiply over all leaves, including the leaf that we assumed the example ended up in. Taking this special leaf apart we obtain:

$$w_c(I) = W_c(I) \sum_{T \in \mathcal{L},\ \text{has leaf } I} \prod_{I' \in T, I' \neq I} V(I',T); \quad (4)$$

where

$$W_c(I) = P_{leaf}(I,\mathbf{X}) \int_{\vec{\theta}_I} \theta_{Ic} Dir(\vec{\theta}_I|\vec{\alpha}) \prod_c \theta_{Ic}^{f_c(I)} d\vec{\theta}_I$$

and

$$V(I,T) = \begin{cases} P_{intern}(I,\mathbf{X}) & \text{if } I \text{ is internal in } T; \\ P_{leaf}(I,\mathbf{X}) \int_{\vec{\theta}_I} Dir(\vec{\theta}_I|\vec{\alpha}) \prod_c \theta_{Ic}^{f_c(I)} d\vec{\theta}_I, & \text{otherwise.} \end{cases}$$

This rewrite is correct due to the fact that we can move the integral of Equation 3 within the product over the leaves: the parameters of the leaves are independent from each other.

Let us write the integrals in closed form. First consider $W_c(I)$. As the Dirichlet distribution is the conjugate prior of the binomial distribution, we have

$$W_c(I) = P_{leaf}(I,\mathbf{X}) \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \int_{\vec{\theta}_I} \theta_{Ic} \prod_c \theta_{Ic}^{\alpha_c - 1 + f_c(I)} d\vec{\theta}_I = P_{leaf}(I,\mathbf{X}) \frac{\Gamma(\sum_c \alpha_c)}{\prod_c \Gamma(\alpha_c)} \frac{\prod_c \Gamma(\alpha_c + f'_c(I))}{\Gamma(\sum_c \alpha_c + f'_c(I))}$$

Here $f'_{c'}(I) = f_{c'}(I)$ if $c \neq c'$, else $f'_{c'}(I) = f_{c'}(I) + 1$.

Similarly, we can compute $V(I,T)$ as follows.

$$V(I,T) = \begin{cases} V_{intern}(I) = P_{intern}(I,\mathbf{X}), \\ \qquad \text{if } I \text{ is internal in } T; \\ V_{leaf}(I) = \\ \quad P_{leaf}(I,\mathbf{X}) \frac{\Gamma(\sum_c \alpha_c) \prod_c \Gamma(\alpha_c + f_c(I))}{\prod_c \Gamma(\alpha_c) \Gamma(\sum_c \alpha_c + f_c(I))}, \\ \qquad \text{otherwise.} \end{cases}$$

The remaining question is now how to avoid summing all trees of Equation 4 explicitly. In the following, we will derive a dynamic programming algorithm to

implicitly compute this sum. We use a variable that is defined as follows.

$$u(I) = \sum_{T \in \mathcal{L}(I)} \prod_{I' \in T} V(I', T); \tag{5}$$

Here we define $\mathcal{L}(I)$ as follows:

$$\mathcal{L}(I) = \{\{I' \in T | I' \supseteq I\} | \text{ all } T \in \mathcal{L} \text{ for which } I \in T\};$$

thus, $\mathcal{L}(I)$ consists of all subtrees that can be put below an itemset I while satisfying the hard constraints. As usual, we represent a subtree by listing all its paths.

For this variable we will first prove the following.

Theorem 1. *The following recursive relation holds for $u(I)$:*

$$u(I) = V_{leaf}(I) + \sum_{i \in \mathcal{I}_{pos} \text{ s.t. } I \cup i, I \cup \neg i \in \mathcal{P}} V_{intern}(I) u(I \cup i) u(I \cup \neg i).$$

Proof. We prove this by induction. Assume that for all itemsets $|I| > k$ our definition holds. Let us fill in our definition in the recursive formula, then we get:

$$u(I) = V_{leaf}(I) + \sum_{i \in \mathcal{I}_{pos}, \text{ s.t. } I \cup i, I \cup \neg i \in \mathcal{P}} \sum_{T \in \mathcal{L}(I \cup i)} \sum_{T' \in \mathcal{L}(I \cup \neg i)} V_{intern}(I) \prod_{I' \in T} V(I', T) \prod_{I' \in T'} V(I', T');$$

This can be written as Equation 5 to prove our claim: the term for V_{leaf} corresponds to the possibility that I is a leaf, the first sum passes over all possible tests if the node is internal, the second and third sum traverse all possible left-hand and right-hand subtrees; the product within the three sums is over all nodes in each resulting tree. □

We can use this formula to write $w_c(I)$ as follows.

Theorem 2. *The formula $w_c(I)$ can be written as:*

$$w_c(I) = W_c(I) \sum_{\pi \in \Pi(I)} \prod_{i=1}^{|I|} V_{intern}(\{\pi_1, \ldots, \pi_{i-1}\}) u(\{\pi_1, \ldots, \pi_{i-1}, \neg \pi_i\}).$$

Here, $\Pi(I)$ contains all permutations $(\pi_1, \ldots, \pi_n)$ of the items in I for which it holds that $\forall 1 \leq i \leq n : \{\pi_1, \ldots, \pi_i\}, \{\pi_1, \ldots, \pi_{i-1}, \neg \pi_i\} \in \mathcal{P}$.

Proof. The set of permutations $\Pi(I)$ consists of all (ordered) paths that can be constructed from the items in I and that fulfill the constraints on size and frequency. Each tree $T \in \mathcal{L}$ with $I \in T$ must have exactly one of these paths. Given one such path, Equation 4 requires us to sum over all trees that contain this path. Each tree in this sum consists of a particular choice of subtrees for each sidebranch of the path. Every node in a tree $T \in \mathcal{L}$ with $I \in T$ is either (1) part of the path to node I or (2) part of a sidebranch; this means that we can decompose the product $\prod_{I' \in T, I' \neq I} V(I', T)$, which is part of Equation 4, into a product for nodes in sidebranches, and a product for nodes on the path to I. The term for nodes on the path is computed by

$$W_c(I) \prod_{i=1}^{|I|} V_{intern}(\{\pi_1, \ldots, \pi_{i-1}\});$$

considering the side branches, $u(I)$ sums over all possible subtrees below sidebranches of the path $\{\pi_1, \ldots, \pi_n\}$; using the product-of-sums rule that $\prod_{i=1}^{n} \sum_{j=1}^{m_i} \alpha_{ij} = \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} x_{1\alpha_1} \cdots \alpha_{ni_n}$, where $\sum_{j=1}^{m_i} \alpha_{ij}$ corresponds to a u-value of a sidebranch, we can deduce that the product $\prod_{i=1}^{|I_k|} u(\{\pi_1, \ldots, \pi_{i-1}, \neg \pi_i\})$ sums over all possible combinations of side branches. □

Given their potentially exponential number it is undesirable to enumerate all permutations of item orders for every itemset. To avoid this let us define

$$v(I) = \sum_{\pi \in \Pi(I)} \prod_{k=1}^{|I|} V_{intern}(\{\pi_1, \ldots, \pi_{k-1}\}) u(\{\pi_1, \ldots, \pi_{k-1}, \neg \pi_k\}),$$

such that $w_c(I) = W_c(I) v(I)$.

Theorem 3. *The following recursive relation holds.*

$$v(I) = \begin{cases} 1, & \text{if } I = \emptyset, \\ \sum_{i \in I \text{ s.t. } I - i \cup \neg i \in \mathcal{P}} V_{intern}(I - i) \\ \quad u(I - i \cup \neg i) v(I - i), & \text{otherwise.} \end{cases}$$

Proof. This can be shown by induction: if we fill in our definition of $v(I)$ in the recursive formula we get

$$\sum_{i \in I \text{ s.t. } I - i \cup \neg i \in \mathcal{P}} V_{intern}(I - i) u(I - i \cup \neg i) \sum_{\pi \in \Pi(I - i)} \prod_{k=1}^{|I|-1} V_{intern}(\{\pi_1, \ldots, \pi_{k-1}\}) u(\{\pi_1, \ldots, \neg \pi_k\})$$

Both sums together sum exactly over all possible permutations of the items; the product is exactly over all terms of every permutation. □

Algorithm 1 Compute Bayes Optimal Weights

input The set of itemsets $\mathcal{P}$ and for all $I \in \mathcal{P} : \vec{f}(I)$
output The weight vectors $\vec{w}(I)$ for all $I \in \mathcal{P}$

```
% Bottom-up Phase
Let n be the size of the largest itemset in P
for k := n downto 0 do
  for all I ∈ P s.t. |I| = k do
    u[I] := V_leaf(I)
    for all i ∈ I_pos s.t. I ∪ i, I ∪ ¬i ∈ P do
      u[I] := u[I] + V_intern(I)u[I ∪ i]u[I ∪ ¬i]
    end for
  end for
end for
% Top-down Phase
v[∅] := 1
for k := 1 to n do
  for all I ∈ P s.t. |I| = k do
    v[I] := 0
    for all i ∈ I s.t. I − i ∪ ¬i ∈ P do
      v[I] := v[I] + V_intern(I − i)u[I − i ∪ ¬i]v[I − i]
    end for
    for c := 1 to C do
      w_c[I] := W_c(I)v[I]
    end for
  end for
end for
```

A summary of our algorithm is given in Algorithm 1. The main idea is to apply the recursive formulas for $u(I)$ and $v(I)$ to perform dynamic programming in two phases: one bottom-up phase to compute the $u(I)$ values, and one top-down phase to compute the $v(I)$ values. Given appropriate data structures to perform the look-up of sub- and supersets of itemsets I, this procedure has complexity $\mathcal{O}(|\mathcal{P}|\delta C)$. As $|\mathcal{P}| = \mathcal{O}(n2^m)$, where n is the number of examples in the training data and m the number of attributes, this algorithm is exponential in the number of attributes.

After the run of this algorithm, for a test example we can compute $q_c(\vec{x}') = \sum_{I \subseteq \vec{x}'} w_c(I)$ for every c. We can easily compute the exact class probability estimates from this: $P(y' = c|\vec{x}', \mathbf{X}, \vec{y}) = \frac{q_c(\vec{x}')}{\sum_{c'} q_{c'}(\vec{x}')}$.

To compute the set $\mathcal{P}$ of paths in feasible trees, we can modify a frequent itemset miner (Goethals & Zaki, 2003), as indicated in our earlier work (Nijssen & Fromont, 2007). We replace the itemset lattice post-processing method of (Nijssen & Fromont, 2007) by the algorithm for computing Bayes optimal weights.

Compared to the OB1 algorithm of Cleary & Trigg (1998), the main advantage of our method is its clear link to frequent itemset mining. OB1 is based on the use of option trees, which have a worst case complexity of $O(nm!)$ instead of $O(n2^m)$. Cleary et al. suggest that sharing subtrees in option trees could improve performance; this exactly what our approach achieves in a fundamental way. The link between weighted rule-based and Bayes optimal classification was also not made by Cleary et al., making the classification phase either more time or space complex. We can interpret predictions by our approach by listing the (maximal) itemsets that contribute most weight to a prediction.

5. Experiments

We do not perform a feasibility study here, as we did such a study in earlier work (Nijssen & Fromont, 2007).

We performed several experiments to determine the importance of the α and β parameters of the size prior. We found that the differences between values $\alpha, \beta \in \{0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$ were often not significant and choose $\alpha = 0.80$ and $\beta = 0.80$ as defaults. We also experimented with a uniform prior. We choose $\vec{\alpha} = (1.0, \ldots, 1.0)$ as default for the Dirichlet prior. This setting is common in the literature.

All comparisons were tested using a corrected, two-tailed, paired $t-$test with a 95% confidence interval.

Artificial Data In our first set of experiments we use generated data. We use this data to confirm the influence of priors and the ability of the Bayes optimal classifier to recognize that data can best be represented by an ensemble of multiple trees.

A common approach is to generate data from a model and to compute how well a learning algorithm recovers this original model. In our setting this approach is however far from trivial, as it is hard to generate a realistic lattice of itemsets: Calders (2007) showed that it is NP-hard to decide if a set of itemset frequencies can occur at all in data. Hence we used an alternative approach. The main idea is that we wish to generate data such that different trees perform best on different parts of the data. We proceed as follows: we first generate n tree structures (in our experiments, all trees are complete trees of depth 7; the trees do not yet have class labels in their leaves); from these n trees we randomly generate a database of given size (4000 examples with 15 binary attributes in our experiments, without class labels). We make sure that every leaf in every tree has at least γ examples (3% of the training data in our experiments). Next, we iterate in a fixed order over these trees to assign classes to the examples in one leaf of each tree; in each tree we pick the leaf which has the largest number of examples without class, and assign a class to these examples, taking care that two adjacent leaves get different majority classes. We aim for pure leaves, but these are less likely for higher numbers of generating trees.

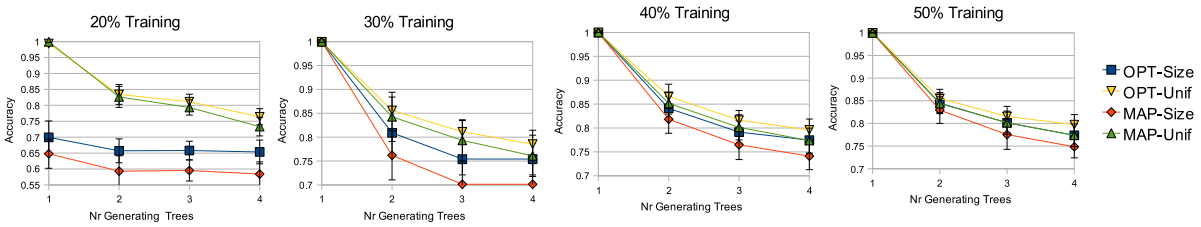


Figure 1. Results on artificial data.

The results of our experiments are reported in Figure 1. The accuracies in these experiments are computed for 20 randomly generated datasets. Each figure represents a different fraction of examples used as training data; remaining examples were as test data. The learners were run using the same depth and support constraints as used to generate the data.

We can learn the following from these experiments.

As all our datasets were created from trees with maximal height, the prior which prefers small trees performs worse than the one which assigns equal weight to all trees. If the amount of training data is small, the size prior forces the learner to prefer trees which are not 100% accurate for data created from one tree.

In all cases, the Bayes optimal approach is significantly more accurate than the corresponding MAP approach, except if the data was created using a single tree; in this case we observe that a single (correct) tree is dominating the trees in the ensembles.

The more training data we provide, the smaller the differences between the approaches are. For the correct prior the optimal approach has a better learning curve.

Additional experiments (not reported here) for other tree depths, dataset sizes and less pure leaves confirm the results above, although sometimes less pronounced.

UCI Data In our next set of experiments we determine the performance of our algorithm on common benchmark data, using ten-fold cross validation.

The frequency and depth constraints in our prior influence the efficiency of the search; too low frequency or too high depth constraints can make the search infeasible. Default values for δ that we considered were 4, 6 and ∞; for γ we considered 2, 15 and 50. We relaxed the constraints as much as was computationally possible; experiments (not reported here) show that this usually does not worsen accuracy.

As our algorithm requires binary data, numeric attributes were discretized in equifrequency bins. Only a relatively small number of 4 bins was feasible in all experiments; we used this value in all datasets to avoid drawing conclusions after parameter overfitting. Where feasible within the range of parameters used, we added results for other numbers of bins to investigate the influence of discretization.

The experiments reported in Figure 1 help to provide more insight in the following questions:

(Q1) Is a single tree dominating a Bayes optimal classifier in practice?

(Q2) Are there significant differences between a uniform and a size-based prior in practice?

(Q3) Is the optimal approach overfitting more in practice than the traditional approach, in this case Weka's implementation of C4.5?

(Q4) What is the influence of the 4-bin discretization?

To get an indication about (Q1) we compare the optimal and MAP predictions. We underlined those cases where there is a significant difference between optimal and MAP predictions. We found that in many cases there is indeed no significant difference between these two settings; in particular when hard constraints impose a high bias, such as in the Segment and Vote data, most predictions turn out to be equal. If there is a significant difference, the optimal approach is always the most accurate.

To answer (Q2) we highlighted in bold for each dataset the system that performs significantly better than all other systems. In many cases, the differences between the most accurate settings are not significant; however, our results indicate that a uniform prior performs slightly better than a size prior in the Bayes optimal case; the situation is less clear in the MAP setting.

Answering (Q3), we found not many significant differences between J48's and Bayes optimal predictions in those cases where we did not have to enforce very hard constraints to turn the search feasible. This supports the claim of Domingos (2000) that Bayes optimal predictions are not really much better. However, our results also indicate that there is no higher risk of overfitting either. The optimal learner does not perform as well as J48 in those cases where the search is only feasible for high frequency or low depth constraints, and

Dataset	γ	δ	Bins	Accuracy Opt - Size	MAP - Size	Opt - Unif	MAP - Unif	J48
Anneal	2	6	4	0.81±0.02	0.80±0.01	0.82±0.03	0.81±0.03	0.82±0.04
Anneal	15	6	10	0.86±0.04	0.86±0.04	0.86±0.04	0.85±0.04	**0.89**±0.03
Anneal	2	4	10	0.81±0.02	0.81±0.01	0.81±0.01	0.81±0.01	**0.89**±0.03
Balance	2	∞	4	0.81±0.04	0.76±0.06	**0.84**±0.03	0.83±0.03	0.76±0.06
Balance	2	∞	10	0.80±0.03	0.74±0.06	**0.85**±0.03	0.79±0.03	0.78±0.03
Heart	2	6	4	0.82±0.07	0.79±0.05	**0.84**±0.05	0.73±0.08	0.78±0.06
Heart	2	4	10	0.81±0.06	0.79±0.05	0.81±0.06	0.78±0.04	0.79±0.05
Vote	15	4	–	0.95±0.03	0.96±0.02	0.95±0.03	0.94±0.03	0.96±0.02
Segment	15	4	4	0.78±0.02	0.78±0.02	0.78±0.02	0.78±0.02	**0.95**±0.02
P-Tumor	2	∞	–	0.40±0.05	0.37±0.05	0.43±0.05	0.37±0.05	0.40±0.05
Yeast	2	6	4	0.52±0.03	0.52±0.03	0.53±0.03	0.52±0.03	0.54±0.05
Yeast	2	4	10	0.51±0.03	0.50±0.03	0.49±0.03	0.49±0.03	**0.58**±0.03
Diabetes	2	6	4	0.75±0.06	0.74±0.06	0.75±0.05	0.71±0.05	0.74±0.06
Diabetes	2	4	10	0.76±0.05	0.75±0.04	0.77±0.05	0.75±0.05	0.74±0.06
Ionosphere	15	4	4	0.87±0.06	0.87±0.06	0.87±0.06	0.87±0.05	0.86±0.07
Ionosphere	15	4	10	0.91±0.04	0.91±0.04	0.90±0.03	0.88±0.03	0.92±0.03
Vowel	50	6	4	0.42±0.04	0.40±0.04	0.41±0.07	0.38±0.05	**0.78**±0.04
Vehicle	50	6	4	0.67±0.03	0.66±0.03	0.66±0.03	0.65±0.03	0.70±0.04

Table 1. Experimental results on UCI data. A result is highlighted if it is the best in its row; significant winners of comparisons between MAP and Opt settings are underlined. Bins are not indicated for datasets without numeric attributes.

thus quite unrealistic priors; in (Nijssen & Fromont, 2007) we found that under the same hard constraints J48 is not able to find accurate trees either, and often finds even worse trees in terms of accuracy.

To provide more insight in (Q4), we have added results for different discretizations. In the datasets where we used harder constraints to make the search feasible, a negative effect on accuracy is observed compared to J48. Where the same hard constraints can be used we observe similar accuracies as in J48. The experiments do not indicate that a higher number of bins leads to increased risks of overfitting.

6. Conclusions

Our results indicate that instead of constructing the optimal MAP hypothesis, it is always preferable to use the Bayes optimal setting; even though we found many cases in which the claim of Domingos (2000) is confirmed and a single tree performs equally well, in those cases where there is a significant difference, the comparison is always in favor of the optimal setting. The computation of both kinds of hypothesis remains challenging if no hard constraints are applied, while incorrect constraints can have a negative impact.

Acknowledgments S. Nijssen was supported by the EU FET IST project "IQ", contract number FP6-516169.

References

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. I. (1996). Fast discovery of association rules. In *Advances in knowledge discovery and data mining*, 307–328.

Angelopoulos, N., & Cussens, J. (2005). Exploiting informative priors for Bayesian classification and regression trees. *Proceedings of IJCAI* (pp. 641–646).

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Statistics/Probability Series. Belmont, California, U.S.A.

Buntine, W. (1990). *A theory of learning classification rules*. Doctoral dissertation, Sydney.

Buntine, W. (1992). Learning classification trees. *Statistics and Computing 2* (pp. 63–73).

Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian CART model search. *Journal of the American Statistical Association*, *93*, 935–947.

Cleary, J. G., & Trigg, L. E. (1998). *Experiences with OB1, an optimal Bayes decision tree learner* (Technical Report). University of Waikato.

Domingos, P. (2000). Bayesian averaging of classifiers and the overfitting problem. *Proceedings of ICML* (pp. 223–230).

Goethals, B., & Zaki, M. J. (Eds.). (2003). *Proceedings of the ICDM 2003 FIMI workshop*, vol. 90 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. *Proceedings of ICDM* (pp. 369–376).

Nijssen, S., & Fromont, E. (2007). Mining optimal decision trees from itemset lattices. *Proceedings of KDD* (pp. 530–539).

Mitchell, T. (1997). *Machine learning*. New York: McGraw-Hill.

Oliver, J. J., & Hand, D. J. (1995). On pruning and averaging decision trees. *Proceedings of ICML* (pp. 430–437).

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.

A Decoupled Approach to Exemplar-based Unsupervised Learning

Sebastian Nowozin SEBASTIAN.NOWOZIN@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, Spemannstrasse 38, 72076 Tübingen, Germany

Gökhan Bakır GHB@GOOGLE.COM
Google GmbH, Brandschenkestrasse 110, 8002 Zurich, Switzerland

Abstract

A recent trend in exemplar based unsupervised learning is to formulate the learning problem as a convex optimization problem. Convexity is achieved by restricting the set of possible prototypes to training exemplars. In particular, this has been done for clustering, vector quantization and mixture model density estimation. In this paper we propose a novel algorithm that is theoretically and practically superior to these convex formulations. This is possible by posing the unsupervised learning problem as a single convex "master problem" with non-convex subproblems. We show that for the above learning tasks the subproblems are extremely well-behaved and can be solved efficiently.

1. Introduction

Methods for unsupervised learning aim at recovering underlying structure from data. In this paper, we are concerned with *exemplar based models* in which this structure is represented by a weighted set of points in input space. Depending on the used model, these points can be interpreted as *clusters*, *codebook vectors* or *mixture components*.

Although the representation is done by a finite point set, the structure being represented – such as a density – is defined on the entire input space by expanding a *smoothing kernel function* around each representing point. In this setting learning simply becomes deciding on the number of points and their weights, as well as their location in input space by means of a suitable *objective*. In EM-learning of mixture models and in k-means clustering one fixes the number of points and adjusts their position by performing descent steps on the objective function starting from a random initialization. This leads to well-behaved but usually non-convex learning problems. Recently, a number of *convex* approaches have been proposed. Our goal in this paper is to improve on these approaches.

In section 2 we review convex formulations for unsupervised learning tasks and discuss two recent methods. We show how convexity is achieved and derive a small experiment whose result suggests a way to improve on the established models. We describe our model in section 3 together with an algorithm and a theoretical justification. The model is validated experimentally in section 4 and we conclude in section 5.

2. Review of Convex Approaches

We now discuss two convex approaches to unsupervised learning from the literature. We will denote the training set as $X = \{\boldsymbol{x}_i\}_{i=1,\ldots,N}$, with $\boldsymbol{x}_i \in \mathcal{X}$ and usually $\mathcal{X} = \mathbb{R}^d$.

Kernel Vector Quantization (Tipping & Schölkopf, 2001) learns a small set of codebook vectors such that the minimum distance from any training sample to its nearest codebook vector is bounded above by a given *maximum distortion* h. In (Tipping & Schölkopf, 2001), this is done by formulating a linear programming problem, of which the following problem is an equivalent reformulation.[1]

$$\begin{aligned} \max_{\boldsymbol{q},\rho} \quad & \rho \\ \text{sb.t.} \quad & K\boldsymbol{q} \geq \rho\mathbf{1}, \\ & \|\boldsymbol{q}\|_1 = 1, \\ & \boldsymbol{q} \geq \mathbf{0}. \end{aligned} \tag{1}$$

Here K is a (N, N) matrix with $K_{i,j} = I(\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \leq h)$, where $I(\cdot)$ evaluates to one if the predicate is true and to zero otherwise, therefore, $K_{i,j}$ is one if a ball of radius h centered on $\boldsymbol{x}_j$ contains $\boldsymbol{x}_i$. In the solution of (1) the balls selected by $q_j > 0$ form a sparse covering of the training set and the distance of each sample to its closest covering ball is bounded by h.

Convex Clustering (Lashkari & Golland, 2007) was re-

[1] Subject to rescaling of $\boldsymbol{q}$.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

cently proposed for clustering. In Lashkari and Golland's model, a mixture model is fit to an observed training set, such that a candidate mixture component is centered around each training set exemplar. Using the framework of Bregman clustering (Banerjee et al., 2005), their objective maximizes the log-likelihood subject to the constraint that the resulting model is a proper mixture model. In the optimum solution of the model, a sparse set of exemplars is selected, allowing the interpretation as clusters.

Formally, Lashkari and Golland maximize $\frac{1}{N}\sum_{i=1}^{N}\log\left[\sum_{j=1}^{N} q_j e^{-\beta d_\phi(\boldsymbol{x}_i,\boldsymbol{x}_j)}\right]$ over the mixture parameters $q_j \geq 0,\ j = 1,\ldots,N$ with $\sum_{j=1}^{N} q_j = 1$. The model allows all exponential family distributions with a corresponding Bregman divergence d_ϕ (Banerjee et al., 2005). For the maximization, a multiplicative update is used, which leads to slow convergence once elements of $\boldsymbol{q}$ approach zero. We reformulate the above objective function by introducing a new set of variables γ_i, with $i = 1,\ldots,N$ as follows.

$$\max_{\boldsymbol{q},\boldsymbol{\gamma}} \quad \frac{1}{N}\sum_{i=1}^{N}\log\gamma_i \tag{2}$$

$$\text{sb.t.} \quad K\boldsymbol{q} = \boldsymbol{\gamma}, \tag{3}$$
$$\|q_j\|_1 = 1,$$
$$q_j \geq 0, \qquad j = 1,\ldots,N,$$

where K is a (N,N) matrix and $K_{i,j} = e^{-\beta d_\phi(\boldsymbol{x}_i,\boldsymbol{x}_j)}$. Clearly, problem (2) is equivalent to the previous one because constraints (3) only serve to evaluate the likelihood γ_i for each sample $\boldsymbol{x}_i$.

2.1. Where does Convexity come from?

Models as proposed in (Tipping & Schölkopf, 2001) and (Lashkari & Golland, 2007) achieve convexity by changing the problem parametrization. Instead of learning the coordinates of a fixed number of exemplars $\boldsymbol{z}_j,\ j = 1,\ldots,M$, there is now a larger set of possible candidate exemplars with fixed coordinates. Learning is performed by optimizing over indicator variables, selecting a sparse subset of the candidates.

This reparametrization makes the problem convex but also changes the regularization: whereas usually the number of exemplars M is the main regularization parameter, it is now an implicit guarantee on the quality of the solution. In (Tipping & Schölkopf, 2001) this is the maximum distortion h, whereas in (Lashkari & Golland, 2007) the regularization parameter β controls the smoothness of the density.

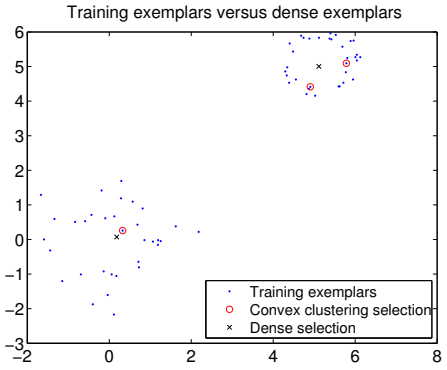


Figure 1. Exemplar selection within the training set versus the finest dense set of 900 exemplars on a regular grid. In this toy example, there are 66 data points.

2.2. Motivating Experiment: More Exemplars

Restricting the set of possible prototype candidates to the training set might result in a suboptimal solution if there is no exemplar close to the true mean of a cluster. If the data is low-dimensional, normal-distributed within each cluster, has low noise and there are enough training examples, this effect is small and can be ignored. But in high dimensions the true mean might be far away from any exemplar.

To demonstrate the effect of restricting the prototype candidate set we perform an experiment. A simple two-dimensional data set is created by sampling from an isotropic Gaussian and a ring of uniform density, forming two well-separated clusters, see Figure 1. We compare convex clustering (Lashkari & Golland, 2007) with a modified model where the objective is changed to $\frac{1}{N}\sum_{i=1}^{N}\log\left[\sum_{j=1}^{M} q_j e^{-\beta d_\phi(\boldsymbol{x}_i,\boldsymbol{z}_j)}\right]$, with N training samples $\boldsymbol{x}_i$ and M cluster center candidates $\boldsymbol{z}_j$. This convex objective still represents the log-likelihood of the training samples under a mixture model. We generate $\boldsymbol{z}_j$ by densely discretizing the $[-2;7]^2$ box on a regular grid. Our hope is that a fine discretization will increase the chance that $\{\boldsymbol{z}_j\}_{j=1,\ldots,M}$ contains exemplars close to the true center of each cluster. For both models we use an isotropic multivariate normal distribution with covariance matrix $\Sigma = \sigma^2 I$, $\sigma = 2.5$.

The clustering result is shown in Figure 1. For the cluster around the origin there is indeed a training set exemplar close to the mean of the generating Gaussian and the difference between the convex clustering and dense selection is small. However, for the ring-like structure, the training set exemplars cannot represent the cluster center adequately. This causes convex clustering to select two exemplars, while in the dense set a single good candidate is selected. A slight perturba-

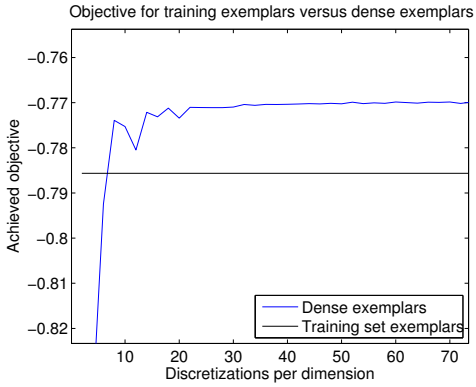


Figure 2. Training set vs. dense set log-likelihood.

tion in the training data would lead to a different selection by the convex clustering method, as all samples bordering to the interior of the ring are roughly equally bad. For this data set, the solution produced by convex clustering is not only qualitatively disappointing but also unstable. The achieved objectives are shown in Figure 2, where the convex clustering objective is drawn as horizontal line and the dense exemplar model forms a curve as the discretization becomes finer and finer. At around eight discretizations per dimension our modified model surpasses the log-likelihood of the convex clustering model. At around 30 discretizations per dimension the log-likelihood levels out and adding more cluster candidates does not improve the solution.

This experiment suggests that a larger set of candidate clusters can lead to higher quality results which are also more robust. While dense discretization is only feasible in case the input space is low-dimensional, ideally we would like to use an infinitely fine discretization and thus use the set of *all* possible input points as candidates. This idea will be the basis for our method.

3. A Decoupled Model

We now introduce our model for unsupervised learning together with an efficient solution algorithm. Essential to the solution is the ability to solve a certain subproblem which we analyze in detail.

3.1. Model

Our model for unsupervised learning generalizes convex clustering (Lashkari & Golland, 2007) and kernel vector quantization (Tipping & Schölkopf, 2001). Let $k_{\boldsymbol{z}}(\cdot)$ be a non-negative smoothing kernel centered at $\boldsymbol{z} \in \mathcal{Z}$, with $\mathcal{Z} \subseteq \mathcal{X}$. Let $\{\boldsymbol{x}_i\}_{i=1,\dots,N}$, $\boldsymbol{x}_i \in \mathcal{X}$ denote the training set. The following semi-infinite convex programming problem learns a convex combination of response functions such that an objective is minimized.

$$\min_{\boldsymbol{q},\gamma,\rho} \quad \Omega(\boldsymbol{\gamma}, \rho) \tag{4}$$

$$\text{sb.t.} \quad \int_{\mathcal{Z}} q_{\boldsymbol{z}} k_{\boldsymbol{z}}(\boldsymbol{x}_i)\, d\boldsymbol{z} = \gamma_i : \alpha_i,\ i = 1, \dots, N \tag{5}$$

$$\rho \le \gamma_i : \omega_i, \quad i = 1, \dots, N, \tag{6}$$

$$q_{\boldsymbol{z}} \ge 0 : \mu_{\boldsymbol{z}}, \quad \forall \boldsymbol{z} \in \mathcal{Z}, \tag{7}$$

$$\int_{\mathcal{Z}} q_{\boldsymbol{z}}\, d\boldsymbol{z} = 1 : \sigma, \tag{8}$$

where $\boldsymbol{\alpha}$, $\boldsymbol{\omega}$, $\boldsymbol{\mu}$ and σ are the Lagrange multipliers for the respective constraints. Before discussing the choice of objective function Ω, let us discuss the purpose of the constraints.

- Constraint (5) evaluates a convex combination of responses for each sample. γ_i contains the combined response for sample $\boldsymbol{x}_i$.
- Constraint (6) identifies – if $\nabla_\rho \Omega(\boldsymbol{\gamma}, \rho) < 0$ – the lowest response among all samples. The value of the lowest combined response is ρ.
- Constraints (7) and (8) define the combination simplex of the response functions.

For the special case where $\mathcal{Z}$ is a finite set of points in $\mathcal{X}$, we can replace the integrals and infinite constraints with a finite sum and finite set of constraints, respectively. Constraints (5) can then be compactly written as $K\boldsymbol{q} = \boldsymbol{\gamma}$, where K is a $(N, |\mathcal{Z}|)$ matrix storing the kernel responses. The dual problem of (4) can be derived from the conjugate function $\Omega^*(\boldsymbol{\alpha}, \sigma, \boldsymbol{\omega}, \boldsymbol{\mu})$ and its respective domain (Boyd & Vandenberghe, 2004, result (5.11)). The dual problem is

$$\max_{\boldsymbol{\alpha},\sigma,\boldsymbol{\omega},\boldsymbol{\mu}} \quad -\Omega^*(\boldsymbol{\alpha}, \sigma, \boldsymbol{\omega}, \boldsymbol{\mu}) - \sigma \tag{9}$$

$$\text{sb.t.} \quad (\boldsymbol{\alpha}, \sigma, \boldsymbol{\omega}, \boldsymbol{\mu}) \in \mathbf{dom}(\Omega^*),$$

$$\sum_{i=1}^{N} \alpha_i k_{\boldsymbol{z}}(\boldsymbol{x}_i) \ge \mu_{\boldsymbol{z}} - \sigma, \quad \forall \boldsymbol{z} \in \mathcal{Z}$$

$$\boldsymbol{\omega} \ge \mathbf{0} \tag{10}$$

$$\mu_{\boldsymbol{z}} \ge \mathbf{0}, \quad \forall \boldsymbol{z} \in \mathcal{Z}$$

We propose the following choices of convex objective functions $\Omega(\boldsymbol{\gamma}, \rho)$.

1. $\Omega(\boldsymbol{\gamma}, \rho) = -\rho$

   The objective states that the lowest response among all samples is to be maximized. All other samples have equal or higher responses but are ignored by this objective, hence a single exemplar can have a large influence on the overall objective. The KVQ problem (1) corresponds to this

objective with K chosen as discussed in section 2. The conjugate is $\Omega^*(\boldsymbol{\alpha}, \sigma, \boldsymbol{\omega}, \boldsymbol{\mu}) = 0$ and domain $\mathbf{dom}(\Omega^*) = \{(\boldsymbol{\alpha}, \sigma, \boldsymbol{\omega}, \boldsymbol{\mu}) : \boldsymbol{\omega} + \boldsymbol{\alpha} \leq \mathbf{0}, \boldsymbol{\omega}^\top \mathbf{1} = 1\}$. With (10) we have $\boldsymbol{\alpha} \leq \mathbf{0}$.

2. $\Omega(\boldsymbol{\gamma}, \rho) = -\frac{1}{N} \sum_{i=1}^N \log(\gamma_i)$

   This objective maximizes $\prod_{i=1}^N \gamma_i$. For the special case where the columns of K correspond to evaluations of probability density functions at the training samples this objective maximizes the log-likelihood of the samples under a mixture model, resulting in convex clustering (2). A single exemplar can have a significant effect on the overall objective, but all sample responses are considered, contrasting the previous objective function. The conjugate is $\Omega^*(\boldsymbol{\alpha}, \sigma, \boldsymbol{\omega}, \boldsymbol{\mu}) = -\frac{1}{N} \sum_{i=1}^N \log(-\alpha_i) + \log(N)$ with domain $\mathbf{dom}(\Omega^*) = \{(\boldsymbol{\alpha}, \sigma, \boldsymbol{\omega}, \boldsymbol{\mu}) : \boldsymbol{\alpha} < \mathbf{0}, \boldsymbol{\omega} = \mathbf{0}\}$.

3. $\Omega(\boldsymbol{\gamma}, \rho) = -\rho + \frac{C}{N} \sum_{i=1}^N (\gamma_i - \rho)^2$

   The objective maximizes the margin ρ while penalizing large deviations from the margin, where the penalty strength is determined by $C \geq 0$. The objective may prefer a smaller margin if the corresponding choice of $\boldsymbol{q}$ leads to a more uniform γ_i. This *margin-minus-variance* (MMV) objective was first proposed in (Rückert & Kramer, 2006) for supervised learning.

4. $\Omega(\boldsymbol{\gamma}, \rho) = -\frac{1}{N} \sum_{i=1}^N \gamma_i + \frac{C}{N} \sum_{i=1}^N (\gamma_i - \frac{1}{N} \sum_{i=1}^N \gamma_i)^2$

   The objective maximizes the mean response while penalizing large deviations from it, where the penalty strength is determined by $C \geq 0$. This maximizes the *mean-minus-variance* popular in applications such as portfolio optimization, see for example (Cornuejols & Tütüncü, 2007).

In order to be able to compare our method with established methods from the literature we only use the first two objectives in the experiments.

3.1.1. Relation to existing methods.

Most relevant for our approach is Boosting Density Estimation (Rosset & Segal, 2002). We note the following differences, i) our model includes different objectives, ii) in our solution algorithm, we will use totally-corrective weight updates[2] instead of a simple line-search procedure, and iii) we identify each *weak learner* uniquely with a point in input space. Also related is the hard-margin case of 1-class Boosting (Rätsch et al., 2001). With exemplar-based weak learners it is a special case of our model with the first objective.

[2]Totally-corrective steps update all weights individually in each iteration, leading to faster convergence.

Algorithm 1 Infinite Exemplar Column Generation

$(Z, \boldsymbol{q}) = \textsc{Infex}(X, \epsilon, k, Z_0)$
Input:
  Sample set $X = \{\boldsymbol{x}_i\}_{i=1,\dots,N}$, $\boldsymbol{x}_i \in \mathcal{X}$
  Convergence tolerance $\epsilon > 0$
  Non-negative smoothing kernel $k_{\boldsymbol{z}} : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$
  Initial exemplar set $Z_0 = \{\boldsymbol{z}_j\}_{j=1,\dots,|Z_0|}$, $\boldsymbol{z}_j \in \mathcal{Z}$
Output:
  Column exemplar set $Z = \{\boldsymbol{z}_j\}_{j=1,\dots,R}$, $\boldsymbol{z}_j \in \mathcal{Z}$
  Weightings $q_{\boldsymbol{z}_j} \in \mathbb{R}_+$, $j = 1, \dots, R$
Algorithm:
$\boldsymbol{\alpha} \leftarrow -\frac{1}{N}\mathbf{1}$, $Z \leftarrow Z_0$, $R \leftarrow |Z_0| + 1$, $\delta \leftarrow \infty$, $\sigma^* \leftarrow 0$
loop
  $\boldsymbol{z}_R \leftarrow \operatorname{argmax}_{\boldsymbol{z} \in \mathcal{Z}} - \sum_{i=1}^N \alpha_i k_{\boldsymbol{z}}(\boldsymbol{x}_i)$ {(SP)}
  $\delta \leftarrow \sigma^* - \sum_{i=1}^N \alpha_i k_{\boldsymbol{z}_R}(\boldsymbol{x}_i)$ {Compute $\nabla_{\boldsymbol{z}_R}$}
  if $\delta < \epsilon$ **then**
    break {convergence to tolerance}
  end if
  $Z \leftarrow Z \cup \{\boldsymbol{z}_R\}$
  $K \leftarrow \left[k_{\boldsymbol{z}_j}(\boldsymbol{x}_i)\right]_{i=1,\dots,N,\ j=1,\dots,R}$ {response matrix}
  p_R^*, $(\boldsymbol{q}_R^*, \boldsymbol{\gamma}^*, \rho^*)$, $(\boldsymbol{\alpha}^*, \boldsymbol{\omega}^*, \boldsymbol{\mu}_R^*, \sigma^*) \leftarrow$
    objective value, primal- and dual-solution to problem (4) with finite (N, R) matrix K.
  $R \leftarrow R + 1$
end loop

3.2. Algorithm

To solve problem (4), we propose Algorithm 1 (INFEX), a delayed column generation algorithm. The algorithm works with a finite and usually small set of candidate prototypes $\boldsymbol{z}_j$. This set is iteratively enlarged by adding good candidates. Selecting the candidates to add in each iteration becomes a subproblem, which we define now.

Problem 1 (Subproblem (SP)) *Given a set of samples $\boldsymbol{x}_i \in \mathcal{X}$, $i = 1, \dots, N$, a corresponding non-positive sample weighting $\alpha_i \leq 0$, $i = 1, \dots, N$ and a non-negative smoothing kernel $k_{\boldsymbol{z}}(\boldsymbol{x}) : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}_+$, obtain $\boldsymbol{z}^*$ as the solution of*

$$\boldsymbol{z}^* = argmax_{\boldsymbol{z} \in \mathcal{Z}} - \sum_{i=1}^N \alpha_i k_{\boldsymbol{z}}(\boldsymbol{x}_i).$$

The solution to this subproblem provides a candidate $\boldsymbol{z}^*$ that, when added to the set of considered candidates, will reduce the global objective.[3] We will now rigorously derive the subproblem from global optimality conditions of problem (4).

[3]In the optimization literature such columns are referred to as having *negative reduced cost*. The overall decoupled solution approach is closely related to the generalized Benders decomposition (Geoffrion, 1972).

Theorem 1 *Assume that the subproblem (SP) can be solved exactly in each iteration. Then Algorithm 1 solves problem (4) globally to the desired accuracy ϵ.*

Proof. Consider a slightly modified version of problem (4), where a part of the constraints (7) is replaced by equality constraints. We replace (7) by the following constraint set, parametrized by a finite set of points $Z_R = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{|Z_R|}\}$.

$$q_{\boldsymbol{z}} \geq 0 : \mu_{\boldsymbol{z}}, \qquad \forall \boldsymbol{z} \in Z_R, \tag{11}$$
$$q_{\boldsymbol{z}} = v_{\boldsymbol{z}} : \mu_{\boldsymbol{z}}, \qquad \forall \boldsymbol{z} \in \mathcal{Z} \setminus Z_R, \tag{12}$$

where $v_{\boldsymbol{z}} = 0$ is constant for all $\boldsymbol{z} \in \mathcal{Z} \setminus Z_R$. Together, constraints (11) and (12) restrict problem (4) such that only a finite subset of the variables $\boldsymbol{q}$ are used.

For a given finite Z_R, we can obtain an optimal primal $(\boldsymbol{q}^*, \boldsymbol{\gamma}^*, \rho^*)$, and dual $(\boldsymbol{\alpha}^*, \boldsymbol{\omega}^*, \boldsymbol{\mu}^*, \sigma^*)$ solution to the modified problem by solving a finite problem in the restricted set of variables $\{q_{\boldsymbol{z}} : \boldsymbol{z} \in Z_R\}$. Let the optimal function value of this solution be denoted by $p(\boldsymbol{v})$. Because the optimal solution must be feasible, we have $q^*_{\boldsymbol{z}} = v_{\boldsymbol{z}} = 0$ for all $\boldsymbol{z} \in \mathcal{Z} \setminus Z_R$. How would the objective function value $p(\boldsymbol{v})$ change if we force a $q^*_{\boldsymbol{z}}$ to become non-zero? That is, if we increase $v_{\boldsymbol{z}}$ by a very small amount can we improve the solution? The sensitivity theorem (Bertsekas, 1999, Proposition 3.3.3) provides a definite answer, namely we have for all $\boldsymbol{z} \in \mathcal{Z} \setminus Z_R$ the following.

$$\nabla_{v_{\boldsymbol{z}}} p(\boldsymbol{v}) = -\mu^*_{\boldsymbol{z}}.$$

If we have for all $\boldsymbol{z} \in \mathcal{Z} \setminus Z_R$ that $\nabla_{v_{\boldsymbol{z}}} p(\boldsymbol{v}) \geq 0$, then this implies that we can not decrease $p(\boldsymbol{v})$ by making $q_{\boldsymbol{z}} > 0$. Conversely, this observation provides us with a *global optimality condition*: if and only if Z_R contains all relevant (positive $q_{\boldsymbol{z}}$) exemplars, we have $\forall \boldsymbol{z} \in \mathcal{Z} \setminus Z_R : \mu^*_z \leq 0$. Given Z_R and a primal-dual optimal solution we can find an alternative expression for $\mu^*_{\boldsymbol{z}}$. Consider the Lagrangian of the modified problem.

$$\begin{aligned}\mathcal{L}(\boldsymbol{q}, \boldsymbol{\gamma}, \rho, \boldsymbol{\alpha}, \boldsymbol{\omega}, \boldsymbol{\mu}, \sigma) &= \Omega(\boldsymbol{\gamma}, \rho) \\ &+ \sum_{i=1}^{N} \alpha_i \left(\int_{\mathcal{Z}} q_{\boldsymbol{z}} k_{\boldsymbol{z}}(\boldsymbol{x}_i)\, d\boldsymbol{z} - \gamma_i \right) + \boldsymbol{\omega}^\top (\rho \mathbf{1} - \boldsymbol{\gamma}) \\ &- \sum_{\boldsymbol{z} \in Z_R} \mu_{\boldsymbol{z}} q_{\boldsymbol{z}} + \int_{\mathcal{Z} \setminus Z_R} \mu_{\boldsymbol{z}} q_{\boldsymbol{z}}\, d\boldsymbol{z} \\ &+ \sigma\Big(\sum_{\boldsymbol{z} \in Z_R} q_{\boldsymbol{z}} + \int_{\mathcal{Z} \setminus Z_R} q_{\boldsymbol{z}}\, d\boldsymbol{z} - 1 \Big)\end{aligned}$$

Because of optimality of the solution, it must satisfy the Karush-Kuhn-Tucker necessary conditions (Bertsekas, 1999), therefore we must have a zero gradient with respect to the primal variables. Specifically, for all $\boldsymbol{z} \in \mathcal{Z} \setminus Z_R$ we must have $\nabla_{q_{\boldsymbol{z}}} \mathcal{L}(\boldsymbol{q}^*, \boldsymbol{\gamma}^*, \rho^*, \boldsymbol{\alpha}^*, \boldsymbol{\omega}^*, \boldsymbol{\mu}^*, \sigma^*) = \sum_{i=1}^{N} \alpha^*_i k_{\boldsymbol{z}}(\boldsymbol{x}_i) + \mu^*_{\boldsymbol{z}} + \sigma^* = 0$. This allows us to express $\mu^*_{\boldsymbol{z}}$ as

$$\mu^*_{\boldsymbol{z}} = \sigma^* - \sum_{i=1}^{N} \alpha^*_i k_{\boldsymbol{z}}(\boldsymbol{x}_i). \tag{13}$$

Therefore, if for all $\boldsymbol{z} \in \mathcal{Z} \setminus Z_R$ we have dual feasible $\mu^*_z \leq 0$, then the current solution is optimal, despite the restrictions imposed by constraints (12). If we satisfy the optimality condition, then replacing (12) with constraints (11), does not change the solution, which remains optimal in the original problem (4).

What remains to be shown is that Algorithm 1 makes progress in each iteration and thus in the limit will satisfy the optimality condition. Consider the case where the above optimality condition is violated for one or more $\boldsymbol{z} \in \mathcal{Z} \setminus Z_R$. Then, let $\boldsymbol{z}^* = \operatorname{argmax}_{\boldsymbol{z} \in \mathcal{Z} \setminus Z_R} \left(\sigma^* - \sum_{i=1}^{N} \alpha^*_i k_{\boldsymbol{z}}(\boldsymbol{x}_i) \right)$ be the sample corresponding to the most negative partial derivative $\nabla_{v_{\boldsymbol{z}^*}} p(v) < 0$. Because of the sensitivity theorem, adding $\boldsymbol{z}^*$ to Z_R – making $q_{\boldsymbol{z}^*}$ a free variable – and re-solving (4) will reduce the objective value. Therefore, *either* no $\boldsymbol{z}^*$ with $\nabla_{v_{\boldsymbol{z}^*}} p(v) < -\epsilon$ is found and convergence to the tolerance is established, *or* a strict decrease in the objective is obtained. □

Note that in practice, we can add multiple exemplars in each iteration. Suppose during solving the subproblem (SP) we obtain a number of good local maximizers. Then, we can add all these local maximizers in order to obtain a faster convergence. Adding redundant exemplars with $\nabla_{v_{\boldsymbol{z}}} p(v) > 0$ does not have an effect as they will receive a zero weight $q_{\boldsymbol{z}} = 0$.

3.3. On the Nature of the Subproblem

The subproblem (SP) is completely determined by the negative weighting of the training set and the shape of the smoothing kernel function. For further discussion let us define $\eta_i = -\alpha_i$ and rewrite the subproblem as $\operatorname{argmax}_{\boldsymbol{z} \in \mathcal{Z}} \sum_{i=1}^{N} \eta_i k(\boldsymbol{x}_i, \boldsymbol{z})$. From the definition it follows that all η_i are non-negative. Clearly, this problem is non-concave whenever k is non-concave in $\boldsymbol{z}$ which is true for all smoothing functions we consider.

However, for kernel functions of the form $k_{\boldsymbol{z}}(\boldsymbol{x}) = k(\|\boldsymbol{x} - \boldsymbol{z}\|)$, the optima of the subproblem, thus the new candidates, are located at the *modes* of the expansion $\sum_{i=1}^{N} \eta_i k_{\boldsymbol{z}}(\boldsymbol{x}_i)$. It is this fact that can be exploited to efficiently solve the subproblem by standard hill-climbing algorithms. Such algorithms start at a point $\boldsymbol{z}^{(0)}$ in input space and generate iteratively better candidates such that $\sum_{i=1}^{N} \eta_i k_{\boldsymbol{z}^{(t+1)}}(\boldsymbol{x}_i) > \sum_{i=1}^{N} \eta_i k_{\boldsymbol{z}^{(t)}}(\boldsymbol{x}_i)$. In this paper, we use the *weighted*

mean shift procedure which was introduced by (Fukunaga & Hostetler, 1975; Cheng, 1995) and gained popularity due to (Comaniciu & Meer, 2002). Given an initial starting point $\boldsymbol{z}^{(0)}$ the iterates are produced by

$$\boldsymbol{z}^{(t+1)} = \frac{\sum_{i=1}^{N} \alpha_i g\left(\left\|\frac{\boldsymbol{z}^{(t)}-\boldsymbol{x}_i}{h}\right\|^2\right)\boldsymbol{x}_i}{\sum_{i=1}^{N} \alpha_i g\left(\left\|\frac{\boldsymbol{z}^{(t)}-\boldsymbol{x}_i}{h}\right\|^2\right)}, \qquad (14)$$

where $g : \mathbb{R}_+ \to \mathbb{R}_+$ is the negative derivative of the so called *kernel profile*. If for a continuous kernel the function g is convex and non-increasing, then the mean shift procedure is guaranteed to converge to a local maxima (Comaniciu & Meer, 2002). For each of the common continuous smoothing kernels, a unique function g exists and some popular kernels and their profile derivatives are discussed in section 4. For the Gaussian kernel, g is a scaled version of the original kernel profile and thus particularly easy to maximize.[4] Mean shift is popular in computer vision, where specialized procedures have been developed to efficiently find globally good modes, for example the *annealed mean shift* procedure (Shen et al., 2007).

If the smoothing kernel function is a reproducing Hilbert kernel (Schölkopf & Smola, 2002), then problem (SP) is known as the *pre-image problem* (Schölkopf et al., 1999). An important difference which simplifies our subproblem considerably is that all our weights $\boldsymbol{\alpha}$ are of the same sign. In the general pre-image problem the sign is not fixed and procedures such as the one of (Schölkopf et al., 1999) can be unstable and do not have a convergence guarantee.

3.4. Optimality Bound

The proof of global optimality of the solution obtained by Algorithm 1 was based on the assumption that the subproblem (SP) can be solved globally. We now show that even without this assumption, the method can be no worse than methods using a fixed exemplar set.

Theorem 2 *Given $\Omega(\boldsymbol{\gamma}, \rho)$, a set $X = \{\boldsymbol{x}_i\}_{i=1,\dots,N}$, $\boldsymbol{x}_i \in \mathcal{X}$ and a finite set of exemplars $Z_F = \{\boldsymbol{z}_j\}_{j=1,\dots,M}$, the solution obtained by solving problem (4) with $\mathcal{Z} = Z_F$ can not achieve a better objective than the solution obtained by Algorithm 1 with $\mathcal{Z} = \mathcal{X}$, $Z_0 = Z_F$.*

[4]The Gaussian kernel has received special attention in the literature. In (Carreira-Perpiñán, 2000) it was conjectured that the number of modes in a Gaussian mixture is bounded above by the number of components. While this is true in the univariate case, this has been proven wrong in general in (Carreira-Perpiñán & Williams, 2003). See also the counter-example at `http://www.inference.phy.cam.ac.uk/mackay/gaussians/`.

Proof. Let Algorithm 1 be called with $Z_0 = Z_F$. In the first iteration of Algorithm 1, the solved problem is identical to problem (4) with $\mathcal{Z} = Z_F$. Therefore, after the first iteration, the objective of Algorithm 1 is equal to the one obtained by solving problem (4). In all later iterations, the objective can only improve. □

4. Experiments and Results

For the following experiments, we solve the restricted master problem (4) using IpOpt (Wächter & Biegler, 2006), a modern primal-dual interior point solver for non-linear programming available as open-source. For each master problem, we obtain accurate convergence in a few dozen solver iterations. We use tolerances 10^{-10} for the restricted master problem and 10^{-7} for the subproblems for all experiments.[5]

As smoothing kernels we use the unnormalized Gaussian, the unnormalized Epanechnikov, and a simple uniform disc kernel. All are parametrized by a bandwidth parameter h. The following are the kernel functions k and profiles g used in the mean shift procedure.

1. Gaussian, bandwidth h
$$k_{\boldsymbol{z}}(\boldsymbol{x}) = e^{-\frac{1}{2}\left\|\frac{\boldsymbol{x}-\boldsymbol{z}}{h}\right\|^2}, \quad g(y) = \frac{1}{2}e^{-\frac{1}{2}y}$$
2. Epanechnikov, bandwidth h
$$k_{\boldsymbol{z}}(\boldsymbol{x}) = \begin{cases} 1 - \left\|\frac{\boldsymbol{x}-\boldsymbol{z}}{h}\right\|^2 & \left\|\frac{\boldsymbol{x}-\boldsymbol{z}}{h}\right\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$
$$g(y) = \begin{cases} 1 & 0 \le y \le 1 \\ 0 & y > 1 \end{cases}$$
3. Uniform disc, maximum distortion h
$$k_{\boldsymbol{z}}(\boldsymbol{x}) = \begin{cases} 1 & \left\|\frac{\boldsymbol{x}-\boldsymbol{z}}{h}\right\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$

The first two kernels are common in non-parametric density estimation, whereas the last one is used by (Tipping & Schölkopf, 2001) for vector quantization. We use the mean shift procedure (14) started from all training samples to solve the subproblem (SP) for the Gaussian and Epanechnikov kernels. We collect the result of each run and add the set of unique local maximizers to the restricted master problem.

However, mean shift cannot be used to solve subproblem (SP) for the non-continuous uniform disc kernel. Instead, when using the uniform disc kernel, we find new codebook candidates by solving the subproblem with the Epanechnikov kernel instead. This is a reasonable approximation as the Epanechnikov kernel response lower bounds the uniform disc kernel response and its maximum lies in the center of the disc.

[5]Our implementation is available at `http://www.kyb.mpg.de/bs/people/nowozin/infex/`.

4.1. Comparison with KVQ

In the first experiment we compare the original Kernel Vector Quantization formulation (1) with all training exemplars as possible prototypes with our Algorithm 1, where the initial set is empty, $Z_R = \emptyset$. We use the first objective $\Omega(\boldsymbol{\gamma}, \rho) = -\rho$ and the uniform disc kernel. As dataset we use a subset of 1100 exemplars from the USPS digit machine learning dataset, with all labels removed and each class sampled equally such that there are 110 exemplars from each class. We evaluate by selecting the maximum allowed distortion h from $\{800, 1000, 1200, 1400, 1600, 1800, 2000\}$, where $\approx$ 2000 is the mean inter-class L_2-distance in the dataset. We compare the achieved margin $\rho^*_{\text{KVQ}}(h)$ with $\rho^*_{\text{INFEX}}(h)$, and the number of codebook vectors $\|\boldsymbol{q}^*_{\text{KVQ}}\|_0$ with $\|\boldsymbol{q}^*_{\text{INFEX}}\|_0$. Figures 3 and 4 show these as the maximum allowed distortion is varied.

The proposed method outperforms KVQ, selecting a smaller number of codebook vectors and achieving a better objective value. Especially for larger allowed distortions, the benefit of selecting an arbitrary point in input space is substantial as due to the high dimensionality of the data set all input samples are relatively far away from each other. Because we use $Z_R = \emptyset$ to initialize our method, the results show that our subproblem approximation using the Epanechnikov kernel is an effective way to find good codebook candidates.

4.2. Comparison with Gaussian Mixture EM

In the second experiment we consider mixture model density estimation and compare our method with Convex Clustering and a homoscedastic Gaussian mixture ($\Sigma = \sigma^2 I$) learned with Expectation Maximization (EM).[6] The log-likelihood objective and the same USPS dataset as before is used. The experimental protocol is as follows. For a range of bandwidths our model and convex clustering are run once per bandwidth. For each run, the number of components of our model is used to fix the number of components in the Gaussian mixture model, which is trained by EM starting 20 times from random initial sample points.

The results are shown in Table 1. Clearly, a single run of our model is consistently the best. The best EM run is always close to our result and Convex Clustering is always the worst. (Lashkari & Golland, 2007) mention that their solution "can be improved in practice with a few extra steps of the EM algorithm". From Table 1, we conclude that the results of convex clustering are qualitatively inferior to plain EM and such *refitting* is actually essential for obtaining good results.

[6] A similar experiment is in (Lashkari & Golland, 2007).

4.3. Subproblem Modes

In the last experiment we show the qualitative behavior of our model with the Epanechnikov kernel with $h = 1500$ and the log-likelihood objective. Because the Epanechnikov kernel has finite support, if we start with $Z_0 = \emptyset$ we could have some samples $\boldsymbol{x}_i$ which have zero response because $k_{\boldsymbol{z}_j}(\boldsymbol{x}_i) = 0$ for all j. Then, the restricted variables $\boldsymbol{q}_j$ are too few and problem (4) would be infeasible. Thus, in order to ensure feasibility of the initial master problems, we use $Z_0 = X$. Some subproblem modes are shown in Figure 5. The modes approximate the "natural" clusters well except for classes such as 3, 8 and 9, which seem to be explained by one joint region with many local modes in it, for example in the first and second row.

5. Discussion and Conclusion

We presented a unifying perspective on existing exemplar based methods that aim at density estimation, clustering and vector quantization. Existing methods were either non-convex or achieved convexity by severe restrictions. In contrast, our approach – although still non-convex as a whole – is provable better than all existing methods. This is achieved by isolating a non-convex but still efficient solvable subproblem. The non-convex subproblem is embedded into a convex master problem steering towards an optimal solution.

One limitation of our model is that one cannot fix $\|\boldsymbol{q}^*\|_0$, the number of components. For problems where guarantees such as maximum distortion or smoothness are more natural constraints, this is not an issue.

There are open questions that result from our work:

1. Does there exists a response function k that is useful for unsupervised learning and at the same time yields a globally solvable subproblem?
2. What is the relation between objective Ω, kernel k and number of components $\|\boldsymbol{q}^*\|_0$?

Table 1. Achieved log-likelihoods. CC is Convex Clustering; for EM the best and mean of 20 runs are shown.

σ	CC	INFEX	EM BEST	EM MEAN
440	−6.3356	**-5.1370**	−5.1442	−5.1485
460	−6.1269	**-4.7424**	−4.7486	−4.7503
480	−5.8705	**-4.3796**	−4.3823	−4.3834
500	−5.5813	**-4.0499**	−4.0507	−4.0520
520	−5.2780	**-3.7499**	−3.7502	−3.7512
540	−4.9779	**-3.4788**	−3.4789	−3.4795

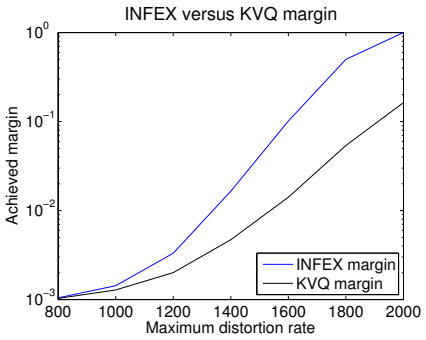


Figure 3. Optimal margin ρ^* as a function of the maximum allowed distortion. Note the log-scale.

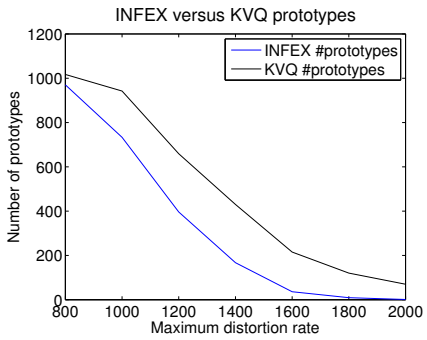


Figure 4. The number of selected prototypes as a function of the maximum allowed distortion.

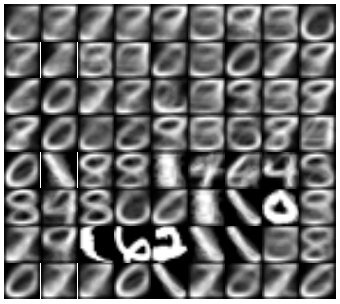


Figure 5. Subproblem modes found in different iterations.

3. Can a decomposition similar to ours yield a training scheme for supervised learning of RBF networks in the line of (Bengio et al., 2005)?

Acknowledgments

This work is funded in part by the EU CLASS project, IST 027978. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

Banerjee, A., Merugu, S., Dhillon, I. S., & Ghosh, J. (2005). Clustering with bregman divergences. *Journal of Machine Learning Research*, *6*, 1705–1749.

Bengio, Y., Roux, N. L., Vincent, P., Delalleau, O., & Marcotte, P. (2005). Convex neural networks. *NIPS*.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena Scientific. 2nd edition.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Carreira-Perpiñán, M. Á. (2000). Mode-finding for mixtures of gaussian distributions. *IEEE Trans. Pattern Anal. Mach. Intell*, *22*, 1318–1323.

Carreira-Perpiñán, M. Á., & Williams, C. K. I. (2003). An isotropic gaussian mixture can have more modes than components.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Analysis and Machine Intelligence*, *17*, 790–799.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell*, *24*, 603–619.

Cornuejols, G., & Tütüncü, R. (2007). *Optimization methods in finance*. Mathematics, Finance and Risk.

Fukunaga, K., & Hostetler, L. D. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Information Theory*, *21*, 32–40.

Geoffrion, A. M. (1972). Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, *10*, 237–260.

Lashkari, D., & Golland, P. (2007). Convex clustering with exemplar-based models. *NIPS*.

Rätsch, G., Schölkopf, B., & Mika, S. (2001). SVM and boosting: One class.

Rosset, S., & Segal, E. (2002). Boosting density estimation. *NIPS* (pp. 641–648). MIT Press.

Rückert, U., & Kramer, S. (2006). A statistical approach to rule learning. *ICML* (pp. 785–792).

Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Mueller, K.-R., Raetsch, G., & Smola, A. J. (1999). Input Space versus Feature Space in Kernel-Based Methods. *IEEE-NN*, *10*, 1000.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. MIT Press.

Shen, C., Brooks, M. J., & van den Hengel, A. (2007). Fast global kernel density mode seeking: Applications to localization and tracking. *IEEE Transactions on Image Processing*, *16*, 1457–1469.

Tipping, M., & Schölkopf, B. (2001). A kernel approach for vector quantization with guaranteed distortion bounds. *AISTATS*.

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*, 25–57.

Cost-sensitive Multi-class Classification from Probability Estimates

Deirdre B. O'Brien DEIRDRE@GOOGLE.COM
Google Inc., Mountain View, CA

Maya R. Gupta GUPTA@EE.WASHINGTON.EDU
University Of Washington, Seattle, WA

Robert M. Gray RMGRAY@STANFORD.EDU
Stanford University, Stanford, CA

Abstract

For two-class classification, it is common to classify by setting a threshold on class probability estimates, where the threshold is determined by ROC curve analysis. An analog for multi-class classification is learning a new class partitioning of the multiclass probability simplex to minimize empirical misclassification costs. We analyze the interplay between systematic errors in the class probability estimates and cost matrices for multi-class classification. We explore the effect on the class partitioning of five different transformations of the cost matrix. Experiments on benchmark datasets with naive Bayes and quadratic discriminant analysis show the effectiveness of learning a new partition matrix compared to previously proposed methods.

1. Introduction

Many classifiers first estimate class probabilities $\hat{p}_k(x)$ for each class $k \in \{1, \ldots, K\}$, then classify a test sample x as the class $\hat{y}(x)$ that minimizes the expected misclassification costs:

$$\hat{y}(x) = \arg \min_{i=1,\ldots,K} \sum_{j=1}^{K} c_{i|j} \hat{p}_j(x) \doteq g(\hat{p}(x); c), \qquad (1)$$

where $c_{i|j}$ is the i^{th} row, j^{th} column element of the cost matrix c, and the cost of classifying as class i when the true class is j. We define the function g to use as short-hand for this minimization.

Such probability-based classifiers can be interpreted as mapping each test sample to a point on the $\hat{p}$-simplex, where each corner of the simplex has $\hat{p}_j = 1$ for some j, and $\hat{p}_i = 0$ for all $i \neq j$; and that the cost matrix c induces a partitioning of the $\hat{p}$-simplex into regions assigned to each of the K classes. However, the probability estimation can suffer from systematic errors, e.g. oversmoothing the estimate towards class prior probabilities. The main contribution of this paper is an analytic and experimental investigation of how changing the partitioning of the $\hat{p}$-simplex can reduce the effect of such systematic errors on classification loss, analogous to ROC analysis for two-class classification.

First, we discuss systematic probability estimation errors and show how these errors can cause classification errors. Then in Section 3 we review methods to reduce the effect of such errors. In Section 4 we establish properties that describe how changing c affects the class-partitioning of the $\hat{p}$-simplex. In Section 5, we propose learning a partitioning of the $\hat{p}$-simplex that seeks to minimize the empirical misclassification costs for the given c, and we provide experimental evidence of the effectiveness of our approach in Section 6.

2. Systematic Error in Multi-class Probability Estimation

Friedman uses the term *oversmoothing* for cases where the probability estimates are systematically smoothed towards the class prior probabilities, and *undersmoothing* for cases where the class probability estimates produced are too confident, such as 1-NN (Friedman, 1997). Other systematic errors in the probability estimates can occur; Niculescu-Mizil and Caruana have documented the systematic errors introduced by various methods of probability estimation for two-class classification (Niculescu-Mizil & Caruana, 2005). Here we provide illustrative examples of over- and under-smoothing in multiclass tasks.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

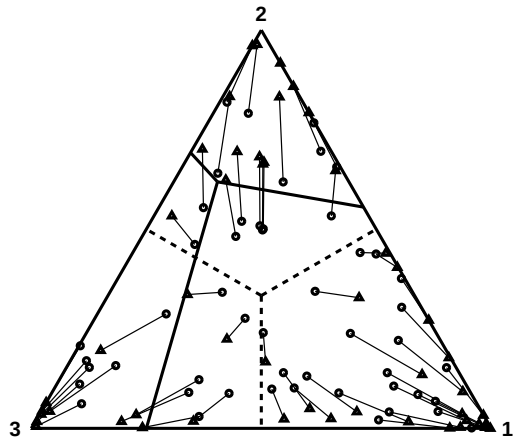
2.1. A Naive Bayes' Example

Consider a three-class problem with discrete features. We estimate class probabilities for test samples using naive Bayes (Hastie et al., 2001). The three classes are equally likely, and the feature vector consists of two identical copies of the same feature. Thus the naive Bayes' assumption of independent features is clearly violated. Figure 1(a) shows pairings of a true class probability (marked with an attached circle) and the associated naive Bayes' estimated probability (marked with a triangle). The incorrect feature-independence assumption *undersmooths* the probability estimates, pushing them towards the edges of the simplex.

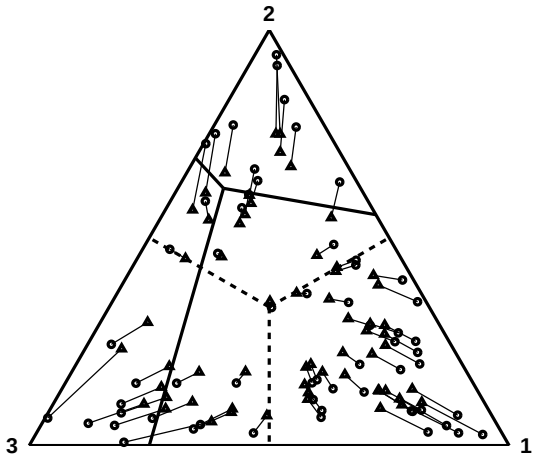
When an estimated probability and the corresponding true probability fall in the same class partition, the undersmoothing does not cause any classification error. When the line attaching a triangle to a circle crosses a class partition line, a classification error occurs. One sees that undersmoothing does not cause errors given the 0/1 cost matrix (dashed lines), but causes many errors given an asymmetric cost matrix (solid lines).

2.2. A k-NN Example

Let N be the number of training samples. Then for the k-NN classifier as the number of nearest neighbors $k \rightarrow N$, the probability estimates are smoothed towards the class prior probabilities. Figure 1(b) illustrates an extreme example: $k = 2000$, $N = 3000$, and the samples are drawn iid and with equal probability from one of three class-conditional normal distributions. The oversmoothing does not cause errors given the 0/1 cost matrix (dashed lines), but causes many errors given an asymmetric cost matrix (solid lines).



(a) Naive Bayes example



(b) k-NN example

Figure 1. Circles mark the true probabilities, triangles mark the estimated probabilities, and each line connects a true probability to the corresponding estimate. The dashed lines mark the class partitioning of the $\hat{p}$-simplex induced by the 0-1 cost matrix, and the solid lines mark the class partitioning induced by an asymmetric cost matrix.

3. Related Work

Approaches to deal with the systematic errors in probability estimation can be analyzed in terms of the classification rule given in (1). Such approaches generally either change the partitioning of the $\hat{p}$-simplex, or change the probability estimates.

3.1. Related Work in Two-class Classification

For the two-class case, the $\hat{p}$-simplex is a line segment from $\hat{p}_1(x) = 0$ to $\hat{p}_1(x) = 1$, and a scalar threshold t partitions the two class regions. The optimal threshold $t^\star$ derived with respect to (1) is,

$$t^\star = \frac{c_{1|2} - c_{2|2}}{c_{1|2} + c_{2|1} - c_{1|1} - c_{2|2}}. \qquad (2)$$

Classification errors can be reduced by changing the class-partitioning by specifying a threshold t that reduces the effect of systematic errors of the class probability estimates. The most common approach uses the receiver operating characteristic (ROC) curves (Egan, 1975; Hanley & McNeil, 1982). An ROC curve plots estimates of the probabilities $P_{\hat{Y}|Y}(2|2)$ versus $P_{\hat{Y}|Y}(2|1)$ for thresholds $t, 0 \leq t \leq 1$, where the estimates are derived from training or validation data. For a given cost matrix the desired point on the ROC curve is chosen and the associated threshold t is used for the classifier (Noe, 1983; Provost & Fawcett, 2001).

Other methods fix the threshold at the theoretical optimal $t^\star$ given by (2), and seek to improve classification by improving the probability estimates. Friedman considered adding a scalar a to the probability estimates

for class 1 (Friedman, 1997); it is easy to show that this method is equivalent to using a threshold $\tilde{t} = t^\star - a$.

Zadrozny and Elkan use monotonic functions of the probability estimate $\hat{p}_1$ to give a *calibrated* estimate and show great improvements in cost-sensitive classification when the calibrated probability estimates are used in place of the original estimates (Zadrozny & Elkan, 2001; Zadrozny & Elkan, 2002). One of their approaches builds on Platt's earlier work to transform support vector machine (SVM) scores into probability estimates using a sigmoid function (Platt, 2000). The same approach can be applied to probability estimates rather than SVM scores. Zadrozny and Elkan propose two other approaches to perform the calibration: *binning* and *pair-adjacent violators*.

Binning takes the probability estimates obtained using cross-validation, orders these values and then groups them into B bins so that there are an equal number of samples in each bin (Zadrozny & Elkan, 2001). The upper and lower boundaries of each bin are determined, and for any test sample with a probability estimate falling in bin b, the updated probability estimate for class 1 is given by the fraction of validation samples in bin b that belong to class 1.

Pair-adjacent violators (PAV) monotonically transform the probability estimates using isotonic regression (Ayer et al., 1955). It has been shown that applying threshold $t^\star$ from (2) to the calibrated probability estimates obtained using PAV is equivalent to using a threshold chosen by ROC analysis on the original probabilities (O'Brien, 2006).

3.2. Related Work in Multi-class Classification

Zadrozny and Elkan extended their two-class solutions to multi-class problems by breaking the classification task into a number of binary classification tasks and using error correcting output codes (ECOC) to obtain multi-class probability estimates (Zadrozny & Elkan, 2002).

Other methods seek to extend the ROC thresholding approach to K-class classification. Instead of choosing a scalar threshold t, a partition of the $(K-1)$-dimensional $\hat{p}$-simplex must be specified. Mossman proposed a method for three class tasks using a very restrictive partitioning of the simplex (Mossman, 1999):

$$\hat{y}(x) = \begin{cases} 1 & \text{if} \quad \hat{p}_3(x) \leq \delta_1 \text{ and } \hat{p}_2(x) - \hat{p}_1(x) \leq \delta_2 \\ 2 & \text{if} \quad \hat{p}_3(x) \leq \delta_1 \text{ and } \hat{p}_2(x) - \hat{p}_1(x) > \delta_2 \\ 3 & \text{if} \quad \hat{p}_3(x) > \delta_1. \end{cases} \tag{3}$$

Lachiche and Flach proposed an alternative to the minimum expected misclassification cost assignment of (1) (Lachiche & Flach, 2003):

$$\hat{y}(x) = \arg\max_i w_i^* \hat{p}_i, \tag{4}$$

where the w_i^* are chosen by minimizing costs on the training set:

$$w^* = \arg\min_w \sum_{n=1}^{N} c_{(\arg\max_i w_i \hat{p}_i(x_n))|y_n}, \tag{5}$$

and x_n, y_n are the nth training sample and its associated class label. We refer to (4) and (5) as the *LF method*. Mossman's method and the LF method can both be viewed as learning a new partitioning for the $\hat{p}$-simplex. In Section 5, we show how these partitions can be achieved by using different cost matrices in equation (1).

MetaCost is a wrapper method that can be used with any classification algorithm and reduces the variance of the probability estimates by bootstrapping (Domingos, 1999). MetaCost reduces the variance of probability estimates, but is not designed to overcome systematic probability estimation errors.

4. The Effect of the Cost Matrix on the Class-Decision Boundaries

In this section we establish how different changes in the cost matrix affect the class partitioning of the $\hat{p}$-simplex enacted by (1). In Section 5 we use these properties to propose a new method to reduce the effect of systematic errors in probability estimation.

For a particular cost matrix c, and any two classes i, k that are adjacent in the partition of the $\hat{p}$-simplex, the partition-boundary between them is described by the hyperplane,

$$\sum_{j=1}^{K} c_{i|j} \hat{p}_j = \sum_{j=1}^{K} c_{k|j} \hat{p}_j. \tag{6}$$

We restrict attention to cost matrices where the cost of correct assignment is always less than the cost of incorrect assignment, that is, $c_{j|j} < c_{i|j}$, $\forall i \neq j$.

Property 1: *For any $\hat{p}$, the assigned class is the same for cost matrices c and αc for any scalar α.*

Proof: The minimization in (1) is unaffected by replacing $c_{i|j}$ by $\alpha c_{i|j}$, $\forall i, j$.

Property 2: *If the cost matrix c is full rank, then there is a point ζ^e where all two class boundaries as described by (6) intersect, and ζ^e may occur outside*

the probability simplex. We term ζ^e the "equal risk point".

Proof: If c is full rank then the solution is $\zeta^e = c^{-1}\mathbf{1}/\|c^{-1}\mathbf{1}\|_1$ that will solve (6) for all classes. However, it can happen that $\|c^{-1}\mathbf{1}\|_1 = 0$, in which case the equal risk point can be said to be at infinity.

If c is not full rank there may still be an unique equal risk point. In general, ζ^e must solve $\left[\begin{array}{c|c} c & \mathbf{-1} \\ \hline \mathbf{1}^T & 0 \end{array}\right] \left[\begin{array}{c} \zeta^e \\ \hline \gamma \end{array}\right] = \left[\begin{array}{c} \mathbf{0} \\ \hline 1 \end{array}\right]$ for some $\gamma \in \mathbb{R}$. If the above matrix is full rank then there is a unique point solution for ζ^e, otherwise the above system is underdetermined and there is a hyperplane of solutions, or the above system can be overdetermined and there is no solution.

Property 3: *Adding a constant to all costs for a particular true class (equivalently, adding a constant to each term in any column of the cost matrix) does not affect the assignment.*

Proof: The minimization in (1) is unaffected by changing $c_{i|j}$ to $c_{i|j} + \alpha_j$, $\forall i$.

Property 4: *The class-partitioning produced by any cost matrix c can equivalently be produced by some cost matrix $\tilde{c}$ where $\tilde{c}_{i|i} = 0$ and $\tilde{c}_{i|j} > 0$ for $i \neq j$.*

Proof: Follows directly from Property 3.

Property 5: (See Fig. 2b) *Adding a constant α to the cost of assignment to class i irrespective of the true class (that is, adding α to each term in row i of a cost matrix), produces a new class-partitioning with partition boundaries parallel to those of the original.*

Proof: This change only affects the two-class boundary equations specified by (6) for class i and each class k:

$$\sum_{j=1}^{K}(c_{i|j} + \alpha)\hat{p}_j = \sum_{j=1}^{K} c_{k|j}\hat{p}_j = \sum_{j=1}^{K} c_{i|j}\hat{p}_j + \alpha.$$

Thus the new boundary between class i and k is parallel to the original boundary between class i and k. To maintain $c_{j|j} < c_{i|j}$, $\forall i \neq j$ requires $\max_{k\neq i}(c_{k|k} - c_{i|k}) < \alpha < \min_{k\neq i}(c_{k|i} - c_{i|i})$.

Property 6: (See Fig. 2c) *Scaling all costs where the true class is ℓ by a positive constant α (that is, multiplying column ℓ of c by α) moves an equal risk point along the line joining it to the corner of the simplex where $\hat{p}_\ell = 1$. The intersections of the class boundaries with the $\hat{p}_\ell = 0$ plane are unchanged.*

Proof: To prove that the new equal risk point $\tilde{\zeta}^e$ is on the line joining the original equal risk point ζ^e and the corner of the simplex where $\hat{p}_\ell = 1$, we show that there exists a constant β such that

$$\tilde{\zeta}^e_j = \beta\zeta^e_j + (1-\beta)\mathcal{I}_{(j=\ell)}, \tag{7}$$

for all j, and where $\mathcal{I}$ is the indicator function.

From (6), multiplying column ℓ by α, and requiring equal risk for classes ℓ and k:

$$\sum_{j=1, j\neq\ell}^{K}(c_{i|j} - c_{k|j})\tilde{\zeta}^e_j + (c_{i|\ell} - c_{k|\ell})(\alpha\tilde{\zeta}^e_\ell) = 0. \tag{8}$$

First note that if $\zeta^e_\ell = 0$, then ζ^e remains an equal risk point for the transformed cost matrix. Otherwise, for any ζ^e, we write $\zeta^e_j = s_j\zeta^e_\ell$. Comparing (6) and (8), there exists $\tilde{\zeta}$ such that $\tilde{\zeta}^e_j = \alpha(s_j\tilde{\zeta}^e_\ell)$, $\forall j \neq \ell$. Thus,

$$\tilde{\zeta}^e_j = \left(\frac{\alpha\tilde{\zeta}^e_\ell}{\zeta^e_\ell}\right)\zeta^e_j. \tag{9}$$

Let $\beta = \alpha\tilde{\zeta}^e_\ell/\zeta^e_\ell$, then (9) establishes (7) $\forall j \neq \ell$. Also,

$$\sum_{j\neq\ell}\tilde{\zeta}^e_j = \beta\sum_{j\neq\ell}\zeta^e_j \tag{10}$$

$$\Rightarrow 1 - \tilde{\zeta}^e_\ell = \beta(1-\zeta^e_\ell), \tag{11}$$

where (11) follows from (10) since components of $\tilde{\zeta}^e$ and ζ^e both sum to 1. This establishes (7) for ℓ.

Lastly, by setting $\hat{p}_\ell = 0$ in equation (6), it is evident that changes in $c_{i|\ell}$ and $c_{j|\ell}$ will not effect the intersections of the class boundaries with the $\hat{p}_\ell = 0$ plane.

Property 7: (See Fig. 2d) *Scaling all costs where the assigned class is i by a positive constant α (that is, multiplying all elements in row i by α) moves an equal risk point ζ^e along the hyper-plane where all classes but class i have equal expected misclassification costs.*

Proof: Let $\tilde{c}_{j|k} = c_{j|k}$ for all $j \neq i$, and let $\tilde{c}_{i|k} = \alpha c_{i|k}$. Then the equal risk point $\tilde{\zeta}^e$ produced by $\tilde{c}$ solves the same set of class boundary equations (6) specifying ζ^e, except

$$\sum_{j=1}^{K}(c_{i|j} - c_{K|j})\zeta^e_j = 0 \quad \Rightarrow \quad \sum_{j=1}^{K}(\alpha c_{i|j} - c_{K|j})\tilde{\zeta}^e_j = 0.$$

Because the constraints specifying that the other classes have equal misclassification costs still apply, the new equal risk point $\tilde{\zeta}^e$ must occur along the hyperplane specified by that subset of the constraints.

5. Learning the Partition Matrix

We propose changing the class partitions so that the class-partitioning corrects for the systematic error in the probability estimates. Changing the partition of the multi-class $\hat{p}$-simplex is analogous to the two-class practice of changing the threshold on $\hat{p}_1(x)$.

We split the cost matrix's role into two separate entities: a partition matrix (which partitions the $\hat{p}$-simplex with linear boundaries), and the misclassification cost matrix that specifies how misclassification errors are to be scored. From now on we will use the term *partition matrix* for the former role, and the term *cost matrix* only for the latter role.

Given a training set $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where x_n has class probability estimate vector $\hat{p}(x_n)$, we propose to use a partition matrix $a^\star$ that solves

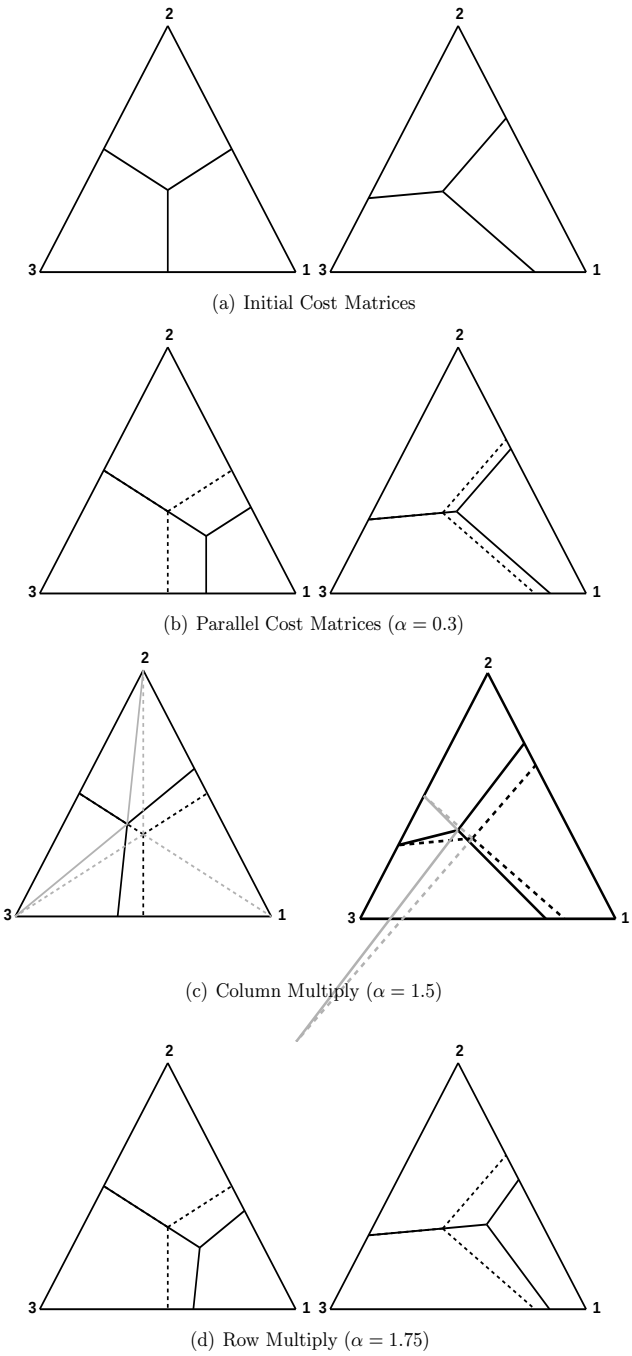
$$a^\star = \arg\min_a \sum_n c_{g(\hat{p}(x_n);a)|y_n}, \tag{12}$$

where the function g in (12) is defined in (1). To avoid the issue of overfitting in learning a partition matrix, we restrict the partition to have linear boundaries that are parallel to the original decision boundaries produced by the cost matrix (see Fig. 2b). We have also considered learning partition matrices with different constraints, including partition matrices with all $K^2 - K$ free parameters, partition matrices that are column-multiply modifications of the original cost matrix (see Fig. 2c), and row-multiply modifications (see Fig. 2d). We found these different constraints resulted in similar performance, with the parallel partitioning working consistently well (O'Brien, 2006).

By Property 4 stated in Section 4, without loss of generality we consider only partition matrices a where $a_{i|i} = 0$ and $a_{i|j} > 0$ for $i \neq j$. From Property 5, adding α_i to row i of the cost matrix will yield a partition matrix with partition boundaries parallel to the partition boundaries induced by c. To maintain the requirement that $a_{i|i} = 0$ we subtract the same constant from column i without affecting the partition boundaries (Property 3). Thus given the cost matrix c, the partition matrix is a where $a_{i|j} = c_{i|j} + \alpha_i - \alpha_j$. This approach requires learning the parameters α_i for $i = 1, \ldots, K$.

Related methods can also be viewed as applications of (12) but with different restrictions on a. Mossman's method for three classes (Mossman, 1999) implicitly requires a to be of the form

$$\begin{bmatrix} 0 & 1 & L - \frac{\delta_2}{1-\delta_2} \\ \frac{1+\delta_2}{1-\delta_2} & 0 & L \\ \frac{\delta_1}{1-\delta_1} L & \frac{\delta_1}{1-\delta_1} L & 0 \end{bmatrix}, \tag{13}$$



(a) Initial Cost Matrices

(b) Parallel Cost Matrices ($\alpha = 0.3$)

(c) Column Multiply ($\alpha = 1.5$)

(d) Row Multiply ($\alpha = 1.75$)

Figure 2. This figure illustrates Properties 5, 6, and 7 described in Section 4 for a three-class classification. The partition produced by the cost matrix is marked by solid lines: a 0/1 cost matrix for the left figures, and an asymmetric cost matrix for the right figures. The corners of the simplex are marked such that $\hat{p}_j = 1$ at corner j, and if a test sample has estimated class probability $\hat{p}$ that falls in the region including corner j, then the estimated class label is j. For each of the figures, manipulations are applied to the class 1 elements of the cost matrix. The dashed lines show the initial cost matrix partitions, the gray lines help to illustrate the properties.

where the δ terms are those used in (3) and $L \gg 1$. The LF method (Lachiche & Flach, 2003) is equivalent to choosing a partition matrix a such that $a_{i|j} = w_j z_{i|j}$ where $z_{i|j}$ is the 0-1 cost matrix and w_j is the weight used in equations (4) and (5). Thus the LF method is equal to a column-multiply of a 0-1 cost matrix (see Property 5 stated in Section 4, and Fig 2c), whereas our proposed method enacts a parallel shift based on the actual cost matrix c.

5.1. Optimizing the Partition Matrix

We learn the new partition matrix by minimizing an empirical loss calculated on a validation set of labeled samples using a greedy approach. The new partition matrix a is initialized to the cost matrix c. Then each free parameter is updated in turn. Let a denote the current partition matrix, and the new partition matrix $\tilde{a}$ will have $\tilde{a}_{i|j} = a_{i|j} + \alpha_i$ and $\tilde{a}_{j|i} = a_{j|i} - \alpha_i$ for all $j \neq i$. Suppose $\alpha_i = -\infty$ and interpret $\tilde{a}$ as a cost matrix – then there would be an infinitely negative cost to assigning a sample as class i, and thus every training sample would be assigned to class i. Suppose one increased α_i from negative infinity. For different values of α_i it would become more cost-effective to classify each of the training samples as a different class, call this classification choice $g^i(\hat{p}(x_n))$, where $g^i(\hat{p}(x_n)) = \arg\min_{k \neq i} \sum_{j=1}^{K} a_{k|j} \hat{p}_j(x)$. Let α_{in} denote the changepoint value – for $\alpha_i < \alpha_{in}$ training sample n would be assigned to class i and for $\alpha_i > \alpha_{in}$ training sample n would be assigned to class $g^i(\hat{p}(x_n))$.

We find these N changepoints α_{in} for $n = 1, \ldots, N$, by solving the N equations,

$$\sum_{j=1}^{K} \left(a_{g^i(\hat{p}(x_n))|j}\right) \hat{p}_j(x_n) = \sum_{j=1}^{K} \left(a_{i|j} + \alpha_{in}\right) \hat{p}_j(x_n).$$

Re-order the training data by their changepoints, so that $\{x_k, y_k\}$ denotes the training point with the kth largest changepoint. Then select N^* where,

$$N^* = \arg \min_{N_0 = 1,2,\ldots,N} \sum_{n < N_0} c_{g^i(\hat{p}(x_n))|y_n} + \sum_{n \geq N_0} c_{i|y_n}. \tag{14}$$

Note that α_{iN^*} to α_{iN^*+1} defines the range of α_i that would yield the empirical cost given in (14); we set the parameter α_i to be the geometric mean of α_{iN^*} and α_{iN^*+1}. Since we require that $\tilde{a}_{j|j} < \tilde{a}_{k|j}$, for all $k \neq j$, it must be that $a_{j|j} < a_{i|j} + \alpha_i$ and $a_{i|i} + \alpha_i < a_{k|i}$, for all $j, k \neq i$, and so α_i is clipped to satisfy these conditions. In addition, if $\alpha_{iN^*} < 0 < \alpha_{iN^*+1}$, then α_i is set equal to 0, or equivalently $\tilde{a}$ is set equal to a.

Each class's partition matrix parameter is adjusted in this manner once, and the parameters are updated in order of class size from most populous to least. Preliminary experiments provided evidence that multiple passes through the parameters did not improve the final classification performance, and that performance was fairly robust to the parameter ordering.

6. Experiments

Experiments with UCI benchmark datasets compare the proposed parallel-partition matrix method to MetaCost (MC) (Domingos, 1999) and to the LF method (Lachiche & Flach, 2003).

For two-class problems the proposed partition matrix methods are equivalent to ROC analysis and therefore only multi-class problems are considered here.

There are two basic variants of MetaCost: the first variant is that the probability estimates are based on training samples not including the sample, while the other variant is that all-inclusive estimates are made. The results reported here are the better of the two variants for each dataset.

Randomized ten-fold cross-validation was done 100 times for each method and each dataset. In the cross-validation, 1/10 of the data was set aside as test data. For MetaCost, 100 resamples were generated using the remaining nine folds. For the proposed methods and the LF method the remaining nine folds were subject to a nine-fold cross-validation so that 8/10 of the data (eight folds) were used to estimate the probabilities for each of the nine folds. Then the partition matrix a and LF parameters were estimated using the nine folds' probability estimates. Finally, the learned cost-sensitive classifier was applied to the withheld 1/10 of the test data.

Experiments were done with two different probability estimation methods. For datasets with discrete features, multinomial naive Bayes was used with Laplacian correction for estimating probabilities. Any continuous features used with naive Bayes were quantized to 21 values. For datasets with continuous features, regularized quadratic discriminant analysis (QDA) (Friedman, 1989) was used. Each class's estimated covariance matrix was regularized as,

$$\hat{\Sigma} = (1 - \gamma - \lambda)\hat{\Sigma}_{ML} + \lambda\bar{\Sigma} + \lambda \frac{\text{trace}\left(\hat{\Sigma}_{ML}\right)}{d} I,$$

where $\hat{\Sigma}_{ML}$ is the maximum likelihood estimate of the full covariance matrix, $\bar{\Sigma}$ is the pooled maximum likelihood estimate, d is the dimensionality of the feature vector, and γ and λ were increased from zero until the condition number of $\hat{\Sigma}$ was less than 10^6.

Experiments were run with two different cost scenarios. In practical situations it is often the rare events that are of greatest interest, and therefore the cost of misclassifying samples from rare classes is higher. To simulate this situation, we set $c_{i|i} = 0$ and,

$$c_{i|j} = \frac{N_i}{N_i + N_j},$$

where N_i is the number of training samples labeled class i. For the second set of experiments, we set $c_{i|i} = 0$, and each element $c_{j|k}$ for $j \neq k$ was drawn randomly and uniformly from $[1, 10]$.

6.1. Discussion of Results

Results are presented in Tables 1 and 2 in terms of the mean increase in performance over the baseline of using the partitioning induced by the cost matrix. The datasets in the results tables are ordered by increasing geometric-mean class size for the training set. The results show that MetaCost did not consistently improve over the baseline. The LF method performed better and usually improved performance, but caused large increases in error in two cases: image segmentation with QDA, and dermatology with naive Bayes. In contrast, learning a new parallel partitioning showed a mean improvement of 10.8% for the rarity-based cost matrix and a mean improvement of 8.9% with the random cost matrix.

Alg.	Dataset	Mean % Improved MC	LF	Par
NB	Bridges 2 (type)	-1	-10	**5**
NB	Bridges 2 (material)	**16**	0	-8
NB	Audiology	-34	**4**	-3
NB	Horse (site)	-7	17	**18**
NB	Bridges 2 (rel-l)	-6	**-3**	-5
NB	Image segmentation	-53	**5**	0
QDA	Image segmentation	-5	-122	**9**
NB	Horse (code)	1	13	**16**
NB	Glass	-4	**18**	16
QDA	Glass	-3	27	**33**
NB	Flag (religion)	2	**7**	6
NB	Horse (type)	1	25	**27**
QDA	Iris	**-1**	-18	-4
NB	Ecoli	-13	**-5**	-7
QDA	Ecoli	-6	-9	**-1**
NB	Dermatology	**-12**	-100	-18
QDA	Wine	1	**64**	56
NB	Horse (subtype)	2	**22**	21
NB	Flare2 (common)	2	**18**	17
NB	Car	-16	10	**18**
NB	Nursery	-16	-8	**31**
	Mean	-7.2	-2.1	**10.8**
	Std. Dev.	14.4	40.4	17.3

Table 1. Performance for the rarity-based cost matrix with $c_{i|i} = 0$ and $c_{i|j} = N_i/(N_i + N_j)$. Largest average improvement for each dataset is in bold.

The LF method and proposed partition-learning method are designed to correct for systematic errors in the probability estimates. Such systematic errors can be interpreted as a bias that can cause the classifier to be wrong in the same way on average over many training sets. Thus, we expected to see a greater increase in performance for the LF method and proposed partition-learning method for larger datasets (further down in the tables) because performance given a large training sets is more likely to suffer from problems of bias than estimation variance. With smaller datasets, estimation variance is generally a larger concern, and the bias reduction offered by the LF and proposed method may not be very helpful. In addition, for small datasets it is harder to learn the systematic error from only a few training samples, and there is an increased risk of overfitting the learned parameters.

The datasets *iris* and *dermatology* had very low misclassification loss for the original probability estimates. For these datasets there was not much improvement possible, and we hypothesize that the methods that learned parameters were likely to overfit to small improvements in the training data.

We used a greedy optimization approach to learn the new partition matrix for our method, but in some cases this leads only to a locally optimal solution. The LF method also uses a greedy search. However, Deng et al.'s results (Deng et al., 2006) show that improving the optimization of the LF objective can lead to an improvement in results. Similarly, we hypothesize that finding a globally optimal solution would also lead to an improvement in costs.

7. Discussion

We analyzed how changes in the cost matrix affect the partitioning of the $\hat{p}$-simplex due to the cost matrix. Based on this analysis, we explored correcting for systematic probability estimation errors by learning a partitioning of the $\hat{p}$-simplex that minimizes empirical misclassification costs on the training set. To reduce overfitting, we only considered partitionings parallel to the original partitioning induced by the cost matrix. Experiments with two standard classifiers showed that this post-processing worked best when the number of training samples per class is relatively large, and when the estimation error with the original cost matrix is large.

Alg.	Dataset	Mean % Improved MC	LF	Par
NB	Bridges 2 (type)	-5	-12	**-3**
NB	Bridges 2 (material)	**12**	-10	2
NB	Audiology	-63	-46	**-9**
NB	Horse(site)	**2**	1	-1
NB	Bridges 2 (rel-l)	**-2**	-5	-5
NB	Image segmentation	-36	**8**	-3
QDA	Image segmentation	-11	-123	**35**
NB	Horse (code)	-2	**5**	1
NB	Glass	**5**	-4	1
QDA	Glass	-4	**7**	1
NB	Flag (religion)	-2	**13**	**13**
NB	Horse(type)	-2	**5**	1
QDA	Iris	**-1**	-7	-3
NB	Ecoli	-11	-17	**-10**
QDA	Ecoli	-3	-2	**3**
NB	Dermatology	-42	-132	**15**
QDA	Wine	-5	68	**73**
NB	Horse(subtype)	-19	**18**	**18**
NB	Flare2 (common)	-5	**35**	22
NB	Car	-22	25	**32**
NB	Nursery	-55	**10**	3
	Mean	-12.9	-7.8	**8.9**
	Std. Dev.	19.9	45.4	19.2

Table 2. Performance for random cost matrix. Largest average improvement for each dataset is in bold.

8. Acknowledgements

The authors thank Jerome Friedman and Richard Olshen for helpful discussions during the course of this research. This research was supported in part by the United States National Science Foundation Grant CCR-0073050.

References

Ayer, M., Brunk, H., Ewing, G., Reid, W., & Silverman, E. (1955). An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, *4*, 641–647.

Deng, K., Bourke, C., Scott, S., & Vinodchandran, N. (2006). New algorithms for optimizing multi-class classifiers with ROC surfaces. *Proc. of the ICML 2006 Workshop on ROC Analysis in Machine Learning*. Pittsburgh, USA.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. *Proc. of 5th International Conference on Knowledge Discovery and Data Mining* (pp. 155–164). San Diego, CA.

Egan, J. (1975). *Signal detection theory and ROC-analysis*. New York: Academic Press.

Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association*, *84*, 165–175.

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*, 55–77.

Hanley, J., & McNeil, B. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*, 29–36.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York: Springer-Verlag.

Lachiche, N., & Flach, P. (2003). Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. *Proc. of 20th International Conference on Machine Learning* (pp. 416–423). Washington DC.

Mossman, D. (1999). Three-way ROCs. *Medical Decision Making*, *19*, 78–98.

Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. *Proc. of 22nd International Conference on Machine Learning*.

Noe, D. (1983). Selecting a diagnostic study's cutoff value by using its receiver operating characteristic curve. *Clinical Chemistry*, *29*, 571–2.

O'Brien, D. B. (2006). *Cost-sensitive performance of probability-estimation based classifiers: analysis and practice*. Doctoral dissertation, Stanford University.

Platt, J. (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers* (pp. 61–74).

Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, *42*, 203 – 231.

Zadrozny, B., & Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naïve Bayesian classifiers. *Proc. of 18th International Conference on Machine Learning* (pp. 609–616). Morgan Kaufmann Publishers, Inc.

Zadrozny, B., & Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. *Proc. of 8th International Conference on Knowledge Discovery and Data Mining* (pp. 694–699). ACM Press.

The Projectron: a Bounded Kernel-Based Perceptron

Francesco Orabona FORABONA@IDIAP.CH
Joseph Keshet JKESHET@IDIAP.CH
Barbara Caputo BCAPUTO@IDIAP.CH
IDIAP Research Institute, Centre du Parc, CH-1920 Martigny, Switzerland
Swiss Federal Institute of Technology (EPFL), CH-1015 Lausanne, Switzerland

Abstract

We present a discriminative online algorithm with a bounded memory growth, which is based on the kernel-based Perceptron. Generally, the required memory of the kernel-based Perceptron for storing the online hypothesis is not bounded. Previous work has been focused on discarding part of the instances in order to keep the memory bounded. In the proposed algorithm the instances are not discarded, but projected onto the space spanned by the previous online hypothesis. We derive a relative mistake bound and compare our algorithm both analytically and empirically to the state-of-the-art Forgetron algorithm (Dekel et al, 2007). The first variant of our algorithm, called Projectron, outperforms the Forgetron. The second variant, called Projectron++, outperforms even the Perceptron.

1. Introduction

One of the most important aspects of online learning methods is their ability to work in an open-ended fashion. Autonomous agents, for example, need to learn continuously from their surroundings, to adapt to the environment and maintain satisfactory performances. A recent stream of work on artificial cognitive systems have signaled the need for life-long learning methods and the promise of discriminative classifiers for this task (Orabona et al., 2007, and references therein).

Kernel-based discriminative online algorithms have been shown to perform very well on binary classification problems (see for example (Kivinen et al., 2004; Crammer et al., 2006)). Most of them can be seen as belonging to the Perceptron algorithm family. They construct their classification function incrementally, keeping a subset of the instances called *support set*. Each time an instance is misclassified it is added to the support set, and the classification function is defined as a kernel combination of the observations in this set. It is clear that if the problem is not linearly separable, they will never stop updating the classification function. This leads eventually to a memory explosion, and it concretely limits the usage of these methods for all those applications where data must be acquired continuously in time.

Several authors tried in the past to address this problem, mainly by bounding a priori the memory requirements. The first algorithm to overcome the unlimited growth of the support set was proposed by Crammer et al. (2003). The algorithm was then refined by Weston et al. (2005). The idea of the algorithm was to discard a vector of the solution, once the maximum dimension has been reached. The strategy was purely heuristic and no mistake bounds were given. A similar strategy has been used also in NORMA (Kivinen et al., 2004) and SILK (Cheng et al., 2007). The very first online algorithm to have a fixed memory "budget" and at the same time to have a relative mistake bound has been the Forgetron (Dekel et al., 2007). A stochastic algorithm that on average achieves similar performances, and with a similar mistake bound has been proposed by Cesa-Bianchi et al. (2006).

In this paper we take a different route. We modify the Perceptron algorithm so that the number of stored samples is always bounded. Instead of fixing a priori the maximum dimension of the solution, we introduce a parameter that can be tuned by the user, to trade accuracy for sparseness, depending on the needs of the task at hand. We call the algorithm, that constitutes the first contribution of this paper, *Projectron*. The Projectron is an online, Perceptron-like method that is bounded in space and in time complexity. We derive for it a mistake bound, and we show experimentally that it outperforms consistently the Forgetron algorithm. The second contribution of this paper is the

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

derivation of a second algorithm, that we call *Projectron++*. It achieves better performances than the Perceptron, retaining all the advantage of the Projectron listed above. Note that this is opposite to previous budget online learning algorithms, delivering performances at most as good as the original Perceptron.

The rest of the paper is organized as follows: in Section 2 we state the problem and we introduce the necessary background theory. Section 3 introduces the Projectron, Section 4 derives its properties and Section 4.1 derives the Projectron++. We report experiments in Section 5, and we conclude the paper with an overall discussion.

2. Problem Setting

The basis of our study is the well known Perceptron algorithm (Rosenblatt, 1958). The Perceptron algorithm learns the mapping $f : \mathcal{X} \to \mathbb{R}$ based on a set of examples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$, where $\mathbf{x}_t \in \mathcal{X}$ is called an *instance* and $y_t \in \{-1, +1\}$ is called a *label*. We denote the prediction of Perceptron as $\text{sign}(f(\mathbf{x}))$ and we interpret $|f(\mathbf{x})|$ as the confidence in the prediction. We call the output f of the Perceptron algorithm a *hypothesis*, and we denote the set of all attainable hypotheses by $\mathcal{H}$. In this paper we assume that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) with a positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ implementing the inner product $\langle \cdot, \cdot \rangle$. The inner product is defined so that it satisfies the reproducing property, $\langle k(\mathbf{x}, \cdot), f(\cdot) \rangle = f(\mathbf{x})$

The Perceptron algorithm is an online algorithm, in which the learning takes place in rounds. At each round a new hypothesis function is estimated, based on the previous one. We denote the hypothesis estimated after the t-th round by f_t. The algorithm starts with the zero hypothesis $f_0 = \mathbf{0}$. On each round t, an instance $\mathbf{x}_t \in \mathcal{X}$ is presented to the algorithm. The algorithm predicts a label $\hat{y}_t \in \{-1, +1\}$ by using the current function, $\hat{y}_t = \text{sign}(f_t(\mathbf{x}_t))$. Then, the correct label y_t is revealed. If the prediction $\hat{y}_t$ differs from the correct label y_t, it updates the hypothesis $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$, otherwise the hypothesis is left intact, $f_t = f_{t-1}$. Practically, the hypothesis f_t can be written as a kernel expansion (Schölkopf et al., 2000),

$$f_t(\mathbf{x}) = \sum_{i \in \mathcal{S}_t} \alpha_i k(\mathbf{x}_i, \mathbf{x}) \ , \tag{1}$$

where $\alpha_i = y_i$ and $\mathcal{S}_t$ is defined to be the set of instance indices for which an update of the hypothesis occurred, i.e., $\mathcal{S}_t = \{0 \le i \le t \,|\, \hat{y}_i \ne y_i\}$. The set $\mathcal{S}_t$ is called the support set. The Perceptron algorithm is summarized in Algorithm 1.

Algorithm 1 Perceptron Algorithm

Initialize: $\mathcal{S}_0 = \emptyset$, $f_0 = \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\mathbf{x}_t$
  Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$
  Receive label y_t
  if $y_t \ne \hat{y}_t$ **then**
    $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$
    $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \{t\}$
  else
    $f_t = f_{t-1}$
    $\mathcal{S}_t = \mathcal{S}_{t-1}$
  end if
end for

Although the Perceptron is a very simple algorithm, it is considered to produce very good results. Our goal is to derive and analyze a new algorithm which attains the same results as the Perceptron but with a minimal size of support set. In the next section we present our Projectron algorithm.

3. The Projectron Algorithm

Let us first consider a finite dimensional RKHS $\mathcal{H}$ induced by a kernel such as the polynomial kernel. Since $\mathcal{H}$ is finite dimensional, there is a finite number of linearly independent hypotheses in this space. Hence, any hypothesis in this space can be expressed using a finite number of examples. We can modify the Perceptron algorithm to use only one set of independent instances as follows. On each round the algorithm receives an instance and predicts its label. On a prediction mistake, if the instance can be spanned by the support set, namely, $\mathbf{x}_t = \sum_{i=1}^{t-1} d_i \mathbf{x}_i$, it is not added to the support set. Instead, the coefficients $\{\alpha_i\}$ in the expansion Eq. (1) are not merely y_i, $i \in \mathcal{S}_{t-1}$, but they are changed to reflect the addition of this instance to the hypothesis, that is, $\alpha_i = y_i + y_t d_i$, $1 \le i \le t-1$. If the instance and the support set are linearly independent, the instance is added to the set with $\alpha_t = y_t$ as before. This technique reduces the size of the support set without changing the hypothesis in any way, and was used by Downs at al. (2001) to simplify Support Vector Machine solutions.

Let us consider now the more elaborate case of an infinite dimensional RKHS $\mathcal{H}$ induced by kernels such as the Gaussian kernel. In this case, it is not possible to find a finite number of linearly independent vectors which span the whole space, and hence there is no guarantee that the hypothesis can be expressed by a finite number of instances. However, we can approximate the concept of linear independence with a

Algorithm 2 Projectron Algorithm

Initialize: $\mathcal{S}_0 = \emptyset$, $f_0 = \mathbf{0}$
for $t = 1, 2, \ldots, T$ **do**
  Receive new instance $\mathbf{x}_t$
  Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$
  Receive label y_t
  if $y_t \neq \hat{y}_t$ **then**
    $f'_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$
    $f''_t = f'_t$ projected onto the space $\mathcal{S}_{t-1}$
    $\delta_t = f''_t - f'_t$
    if $\|\delta_t\| \leq \eta$ **then**
      $f_t = f''_t$
      $\mathcal{S}_t = \mathcal{S}_{t-1}$
    else
      $f_t = f'_t$
      $\mathcal{S}_t = \mathcal{S}_{t-1} \cup \{t\}$
    end if
  else
    $f_t = f_{t-1}$
    $\mathcal{S}_t = \mathcal{S}_{t-1}$
  end if
end for

finite number of vectors (Csató & Opper, 2001; Engel et al., 2002; Orabona et al., 2007). In particular assume that at round t of the algorithm there is a prediction mistake and the mistaken instance $\mathbf{x}_t$ should be added to the support set. Before adding the instance to the support, we construct two hypotheses: a temporal hypothesis f'_t using the function $k(\mathbf{x}_t, \cdot)$, that is, $f'_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$, and a projected hypothesis f''_t, which is the projection of f'_t onto the space spanned by $\mathcal{S}_{t-1}$. That is, the projected hypothesis is a hypothesis from the support set $\mathcal{S}_{t-1}$ which is the closest to the temporal hypothesis. Denote by δ_t the distance between the hypotheses $\delta_t = f''_t - f'_t$. If the norm of distance $\|\delta_t\|$ is below some threshold η, we use the projected hypothesis as our next hypothesis, i.e., $f_t = f''_t$, otherwise we use the temporal hypothesis as our next hypothesis, i.e., $f_t = f'_t$. As we show in the next section, this strategy assures that the maximum size of the support set is always finite, regardless of the dimension of the RKHS $\mathcal{H}$. Guided by these considerations we can design a new Perceptron-like algorithm that projects the solution onto the space spanned by the previous support vectors whenever possible. We call this algorithm Projectron. The algorithm is given in Algorithm 2.

In our algorithm the parameter η plays an important role. If η is equal to zero, we obtain exactly the same solution of the Perceptron algorithm. In this case, however, the Projectron solution can still be sparser when some of the instances are linearly dependent or when the kernel induces a finite dimensional RKHS $\mathcal{H}$. In case η is greater than zero we trade precision for sparseness. Moreover, as shown in the next section, this implies a bounded algorithmic complexity, namely, the memory and time requirements for each step are bounded. We will also derive mistake bounds to analyze the effect of η on the classification accuracy.

We now consider the problem of deriving the projected hypothesis f''_t in a Hilbert space $\mathcal{H}$, induced by a kernel function $k(\cdot, \cdot)$. Denote by $P_{t-1} f_t$ the projection of $f_t \in \mathcal{H}$ onto the subspace $\mathcal{H}_{t-1} \subset \mathcal{H}$ spanned by the set $\mathcal{S}_{t-1}$. The projected hypothesis f''_t is defined as $f''_t = P_{t-1} f'_t$. Expanding f'_t we have

$$f''_t = P_{t-1} f'_t = P_{t-1} \left(f_{t-1} + y_t k(\mathbf{x}_t, \cdot) \right) . \quad (2)$$

The projection is an idempotent ($P^2_{t-1} = P_{t-1}$) and linear operator, hence,

$$f''_t = f_{t-1} + y_t P_{t-1} k(\mathbf{x}_t, \cdot) . \quad (3)$$

Recall that $\delta_t = f''_t - f'_t$. Substitute f''_t from Eq. (3) and f'_t we have

$$\delta_t = f''_t - f'_t = y_t P_{t-1} k(\mathbf{x}_t, \cdot) - y_t k(\mathbf{x}_t, \cdot) . \quad (4)$$

Recall that the projection of $f'_t \in \mathcal{H}$ onto a subspace $\mathcal{H}_{t-1} \subset \mathcal{H}$ is the hypothesis in $\mathcal{H}_{t-1}$ closest to f'_t. Hence, let $\sum_{j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot)$ be an hypothesis in $\mathcal{H}_{t-1}$, where $(d_1, \ldots, d_{t-1})$ is a set of coefficients. The closest hypothesis is the one for which

$$\|\delta_t\|^2 = \min_{(d_1, \ldots, d_{t-1})} \left\| \sum_{j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \cdot) - k(\mathbf{x}_t, \cdot) \right\|^2 . \quad (5)$$

Expanding Eq. (5) we get

$$\|\delta_t\|^2 = \min_{(d_1, \ldots, d_{t-1})} \Bigg(\sum_{i,j \in \mathcal{S}_{t-1}} d_j d_i k(\mathbf{x}_j, \mathbf{x}_i) - 2 \sum_{j \in \mathcal{S}_{t-1}} d_j k(\mathbf{x}_j, \mathbf{x}_t) + k(\mathbf{x}_t, \mathbf{x}_t) \Bigg) . \quad (6)$$

Define $\mathbf{K}_{t-1}$ to be the matrix generated by the instances in the support set $\mathcal{S}_{t-1}$, that is, $\{\mathbf{K}_{t-1}\}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ for every $i, j \in \mathcal{S}_{t-1}$. Define $\mathbf{k}_t$ to be the vector whose i-th element is $\mathbf{k}_{t_i} = k(\mathbf{x}_i, \mathbf{x}_t)$. We have

$$\|\delta_t\|^2 = \min_{\mathbf{d}} \left(\mathbf{d}^T \mathbf{K}_{t-1} \mathbf{d} - 2 \mathbf{d}^T \mathbf{k}_t + k(\mathbf{x}_t, \mathbf{x}_t) \right) , \quad (7)$$

where $\mathbf{d} = (d_1, \ldots, d_{t-1})^T$. Solving Eq. (7), that is, applying the extremum conditions with respect to $\mathbf{d}$, we obtain

$$\mathbf{d}^\star = \mathbf{K}^{-1}_{t-1} \mathbf{k}_t \quad (8)$$

and, by substituting Eq. (8) into Eq. (7),

$$\|\delta_t\|^2 = k(\mathbf{x}_t, \mathbf{x}_t) - \mathbf{k}_t^T \mathbf{d}^\star \ . \tag{9}$$

Furthermore, substituting Eq. (8) into Eq. (3) we get

$$f_t'' = f_{t-1} + y_t \sum_{j \in \mathcal{S}_{t-1}} \mathbf{d}_j^\star \, k(\mathbf{x}_j, \cdot) \ . \tag{10}$$

We have shown how to calculate both the distance δ_t and the projected hypothesis f_t''. In summary, one needs to compute $\mathbf{d}^\star$ according to Eq. (8), plug the result either into Eq. (9) and obtain δ_t or into Eq. (10) and obtain the projected hypothesis.

In order to make the computation more tractable, we introduce an efficient method to calculate the matrix inversion $\mathbf{K}_t^{-1}$ iteratively. This method was first introduced in (Cauwenberghs & Poggio, 2000), and we give it here only for completeness. We would like to note in passing that the matrix $\mathbf{K}_{t-1}$ can be safely inverted since, by incremental construction, it is always full-rank. After the addition of a new sample, $\mathbf{K}_t^{-1}$ becomes

$$\begin{bmatrix} & & & 0 \\ & \mathbf{K}_{t-1}^{-1} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\|\delta_t\|^2} \begin{bmatrix} \mathbf{d}^\star \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{d}^{\star T} & -1 \end{bmatrix} \tag{11}$$

where $\mathbf{d}^\star$ and $\|\delta_t\|^2$ are already evaluated during the previous steps of the algorithm. Thanks to this incremental evaluation, the time complexity of the linear independence check is $O(|\mathcal{S}_{t-1}|^2)$, as one can easily see from Eq. (8).

4. Analysis

In this section we analyze the performance of the Projectron algorithm in the usual framework of online learning with a competitor. First, we present a theorem which states that the size of the support set is bounded.

Theorem 1. *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a continuous Mercer kernel, with $\mathcal{X}$ a compact subset of a Banach space. Then, for any training sequence $(\mathbf{x}_i, y_i), i = 1, \cdots, \infty$ and for any $\eta > 0$, the size of the support set of the Projectron algorithm is finite.*

The proof of this theorem goes along the same lines as the proof of Theorem 3.1 in (Engel et al., 2002), and we omit it for brevity. Note that this theorem guarantees that the size of the support set is bounded, however it does not state that the size of the support set is fixed or can be estimated before training.

The next theorem provides a mistake bound. The main idea is to bound the maximum number of mistakes of the algorithm, relatively to the best hypothesis $g \in \mathcal{H}$ chosen in hindsight. Let us define D_1 as

$$D_1 = \sum_{t=1}^{T} \ell(g(\mathbf{x}_t), y_t) \tag{12}$$

where $\ell(g(\mathbf{x}_t), y_t)$ is the hinge loss suffered by the function g on the example $(\mathbf{x}_t, y_t)$, that is, $\max\{0, 1 - y_t g(\mathbf{x}_t)\}$. With these definitions we can state the following bound for the Projectron Algorithm.

Theorem 2. *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \{-1, +1\}$, and $k(\mathbf{x}_t, \mathbf{x}_t) \leq R$ for all t. Let g be an arbitrary function in $\mathcal{H}$. Assume that the Projectron algorithm is run with $0 \leq \eta < \frac{2-R^2}{2\|g\|}$. Then the number of prediction mistakes the Projectron makes on the sequence is at most*

$$\frac{\|g\|^2 + 2D_1}{2 - R^2 - 2\eta\|g\|}$$

The proof of this theorem is based on the following lemma.

Lemma 1. *Let $(\mathbf{x}, y)$ be an example, with $\mathbf{x} \in \mathcal{X}$ and $y \in \{+1, -1\}$. Denote by f an hypothesis in $\mathcal{H}$, such that $yf(\mathbf{x}) < 1$. Let $f' = f + \tau y q(\cdot)$, where $q(\cdot) \in \mathcal{H}$. Then the following bound holds for any $\tau \geq 0$:*

$$\begin{aligned} \|f - g\|^2 - \|f' - g\|^2 \geq \tau\big(&2\ell(f(\mathbf{x}), y) - 2\ell(g(\mathbf{x}), y) \\ &-\tau\|q(\cdot)\|^2 - 2\langle f, q(\cdot) - k(\mathbf{x}, \cdot)\rangle - 2\|q(\cdot) - k(\mathbf{x}, \cdot)\| \cdot \|g\|\big) \end{aligned}$$

Proof.

$$\begin{aligned} &\|f - g\|^2 - \|f' - g\|^2 = 2\tau y\langle g - f, q(\cdot)\rangle - \tau^2\|q(\cdot)\|^2 \\ &= 2\tau y(g(x) - f(x)) - \tau^2\|q(\cdot)\|^2 \\ &\qquad + 2\tau y\langle g - f, q(\cdot) - k(x, \cdot)\rangle \\ &\geq \tau\big(2\ell(f(\mathbf{x}), y) - 2\ell(g(\mathbf{x}), y) - \tau\|q(\cdot)\|^2 \\ &\qquad - 2y\langle f, q(\cdot) - k(x, \cdot)\rangle - 2\|q(\cdot) - k(x, \cdot)\| \cdot \|g\|\big) \end{aligned}$$

□

With this bound we are ready to prove Thm. 2.

Proof. Define the relative progress in each round as $\Delta_t = \|f_{t-1} - g\|^2 - \|f_t - g\|^2$. We bound the progress from above and below. On rounds in which there is no mistake Δ_t is 0. On rounds in which there is a mistake there are two possible updates: either $f_t = f_{t-1} + y_t P_{t-1} k(\mathbf{x}_t, \cdot)$ or $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$. In the following we bound the progress from below, when the update is of the former type (the same bound can be obtained for the latter type as well, but the derivation is omitted). In particular we set $q(\cdot) = P_{t-1} k(\mathbf{x}_t, \cdot)$ in

Lemma 1 and use $\delta_t = y_t P_{t-1} k(\mathbf{x}_t, \cdot) - y_t k(\mathbf{x}_t, \cdot)$ from Eq. (4)

$$\begin{aligned}\Delta_t &= \|f_{t-1} - g\|^2 - \|f_t - g\|^2 \\ &\geq \tau_t \Big(2\ell(f_{t-1}(\mathbf{x}_t), y_t) - 2\ell(g(\mathbf{x}_t), y_t) \\ &\quad - \tau_t \|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2 - 2\langle f_{t-1}, \delta_t \rangle - 2\|\delta_t\| \|g\| \Big) \ .\end{aligned}$$

Note that $\langle f_{t-1}, \delta_t \rangle = 0$, because f_{t-1} belongs to the space spanned by the functions indexed by $\mathcal{S}_{t-1}$. Moreover, on every projection update $\|\delta_t\| \leq \eta$ and using the theorem assumption $\|P_{t-1} k(\mathbf{x}_t, \cdot)\| \leq R$, we then have

$$\Delta_t \geq \tau_t \Big(2 \left(\ell(f_{t-1}(\mathbf{x}_t), y_t) - \ell(g(\mathbf{x}_t), y_t) \right) - \tau_t R^2 - 2\eta \|g\| \Big) \ .$$

We can further bound Δ_t by noting that on every prediction mistake $\ell(f_{t-1}(\mathbf{x}_t), y_t) \geq 1$. Overall we have

$$\begin{aligned}&\|f_{t-1} - g\|^2 - \|f_t - g\|^2 \geq \\ &\qquad \tau_t \Big(2 \left(1 - \ell(g(\mathbf{x}_t), y_t) \right) - \tau_t R^2 - 2\eta \|g\| \Big) \ .\end{aligned}$$

We sum over t both sides. Let τ_t be an indicator function for a mistake on the t-th round, that is, τ_t is 1 if there is a mistake on round t and 0 otherwise, hence it can be upper bounded by 1. The left hand side of the equation is a telescopic sum, hence it collapses to $\|f_0 - g\|^2 - \|f_T - g\|^2$, which can be upper bounded by $\|g\|^2$, using the fact that $f_0 = \mathbf{0}$ and that $\|f_T - g\|^2$ is non-negative. Finally, we have

$$\|g\|^2 + 2D_1 \geq M \left(2 - R^2 - 2\eta \|g\| \right) \ ,$$

where M is the number of mistakes. □

To compare with other similar algorithms it can be useful to change the formulation of the algorithm in order to use the maximum norm of g as parameter instead of η. Hence we can fix an upper bound, U, on $\|g\|$ and then we set η to have a positive progress. Specifically, on each round we set η to be

$$\frac{1}{2U} \Big(2\ell(f_{t-1}(\mathbf{x}_t), y_t) - \|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2 - 0.5 \Big) \ . \quad (13)$$

The next corollary, based on Thm. 2, provides a mistake bounds in terms of U rather than η.

Corollary 1. *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \{-1, +1\}$, and $k(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t. Let g be an arbitrary function in $\mathcal{H}$, whose norm $\|g\|$ is bounded by U. Assume that the Projectron algorithm is run with a parameter η, which is set in each round according to Eq. (13). Then, the number of prediction mistakes the Projectron makes on the sequence is at most*

$$2\|g\|^2 + 4D_1 \ .$$

Notice that the bound in Corollary 1 is similar to Thm. 5.1 in (Dekel et al., 2007) of the Forgetron algorithm. The difference is in the assumptions made: in the Forgetron, the size of the support set is guaranteed to be less than a fixed size B that depends on U, while in the Projectron we choose the value of η or, equivalently, U, and there is no guarantee on the exact size of the support set. However, the experimental results suggest that, with the same assumptions used in the derivation of the Forgetron bound, the Projectron needs a smaller support set and produces less mistakes.

It is also possible to give yet another bound by slightly changing the proof of Thm. 2. This theorem is a worst-case mistake bound for the Projectron algorithm. We state it here without the proof, leaving it for a long version of this paper.

Theorem 3. *Let $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_T, y_T)$ be a sequence of instance-label pairs where $\mathbf{x}_t \in \mathcal{X}$, $y_t \in \{-1, +1\}$, and $k(\mathbf{x}_t, \mathbf{x}_t) \leq R$ for all t. Let g an arbitrary function in $\mathcal{H}$. Assume that the Projectron algorithm is run with $0 \leq \eta < \frac{1}{\|g\|}$. Then, M, the number of prediction mistakes the Projectron makes on the sequence is at most*

$$\left(\frac{R\|g\| + \sqrt{R^2 \|g\|^2 + 4D_1}}{2(1 - \eta\|g\|)} \right)^2$$

The last theorem suggests that the performance of the Projectron are slightly worse than the Perceptron (Shalev-Shwartz & Singer, 2005). Specifically the degradation in the performance of Projectron compared to the Perceptron are related to $1/(1-\eta\|g\|)^2$. In the next subsection we present a variant to the Projectron algorithm, which attains even better performance.

4.1. Going Beyond the Perceptron

The proof of Thm. 2 and Corollary 1 direct us how to improve the Projectron algorithm to go beyond the performance of the Perceptron algorithm, while maintaining a bounded support set.

Let us start from the algorithm in Corollary 1. We change it so an update takes place not only if there is a prediction mistake, but also when the prediction is correct with a low confidence. We indicate this latter case as a *margin error*, that is, $0 < y_t f_{t-1}(\mathbf{x}_t) < 1$. This strategy improves the classification rate but also increases the size of the support set (Crammer et al., 2006). A possible solution to this obstacle is not to update every round a margin error occurs, but also when the new instance *can be projected* onto the support set. Hence, the update on margin error rounds

would be in the general form

$$f_t = f_{t-1} + y_t \tau_t P_{t-1} k(\mathbf{x}_t, \cdot) \ , \qquad (14)$$

with $0 < \tau_t \leq 1$. The last constraint comes from proofs of Thm. 2 and Corollary 1 in which we upper bound τ_t by 1. Note that setting τ_t to 0 is equivalent to leave the hypothesis unchanged. The bound in Corollary 1 becomes

$$M \leq 2(\|g\|^2 + 2D_1 - \sum_{\{t:0<y_t f_{t-1}(\mathbf{x}_t)<1\}} \beta_t) \ , \qquad (15)$$

where β_t bounds the progress made on margin error round t. In particular it is easy to see from Lemma 1 that β_t is

$$\tau_t \big(2\ell(f_{t-1}(\mathbf{x}_t), y_t) - \tau_t \|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2 - 2U\|\delta_t\|\big) \ , \qquad (16)$$

for $0 < \tau_t \leq 1$, and is 0 when there is no update. Whenever β_t is non-negative the worst-case number of mistakes in Eq. (15) decreases, hopefully along with the classification error rate of the algorithm. Hence, we determine the optimal τ_t which maximizes β_t. In particular, the expression of β_t in Eq. (16) is quadratic in τ_t, and is maximized for $\tau_t = \ell(f_{t-1}(\mathbf{x}_t), y_t)/\|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2$. Constraining τ_t to be less than or equal to 1, we have[1]

$$\tau_t = \min\{\ell(f_{t-1}(\mathbf{x}_t), y_t)/\|P_{t-1} k(\mathbf{x}_t, \cdot)\|^2, 1\} \ . \qquad (17)$$

In summary, at every round t with margin error we calculate τ_t according to Eq. (17), and check that β_t is non-negative. If so we update the hypothesis using Eq. (14), otherwise we leave it untouched.

With this modification we expect better performance, that is, fewer mistakes, but without any increase of the support set size. We can even expect solutions with a smaller support set, since new instances can be added to the support set only if misclassified, hence having less mistakes should result in a smaller support set. We name this variant Projectron++, and in the next section we compare it to the original version.

5. Experimental Results

In this section we present experimental results that demonstrate the effectiveness of the Projectron and the Projectron++. We compare both algorithms to the Perceptron and to the budget algorithms Forgetron (Dekel et al., 2007) and Randomized Budget Perceptron (RBP) (Cesa-Bianchi et al., 2006). For the Forgetron, we choose the state-of-the-art "self-tuned" variant, which outperforms its other variants. We also use two other baseline algorithms: the first one is a Perceptron algorithm which stops updating the solution once the support size has reached some limit, and it is used to verify that the Projectron is better than just stop learning. We name it *Stoptron*. The second baseline algorithm is the PA-I variant of the Passive-Aggressive learning algorithm (Crammer et al., 2006), which gives an upper bound to the classification performance that Projectron++ can reach.

We tested the algorithms with two standard machine learning datasets: *Adults9* and *Vehicle*[2] and a synthetic dataset, all of them with more than 10000 samples. The synthetic dataset is built in the same way as in (Dekel et al., 2007). It is composed with samples taken from two separate bi-dimensional Gaussian distributions. The means of the positive and negative samples are $(1, 1)$ and $(-1, -1)$, respectively, while the covariance matrix for both is diagonal matrix with $(0.2, 2)$ as its diagonal. Then the labels are flipped with a probability of 0.1 to introduce noise.

All the experiments were performed over 5 different permutations of the training set. All algorithms used a Gaussian kernel with σ^2 equals 25, 4, and 0.5 for *Adults9*, *Vehicle*, and the synthetic datasets, respectively. The C parameter of the PA-I was set to 1, to have an update similar to the Perceptron and Projectron. Due to the different nature of our algorithm compared to the budget ones, we cannot select the support set size in hindsight. Hence, we compared them using the proper conditions to obtain the same bounds. That is, we selected the maximum support size B for the Forgetron algorithm, which implies a maximum value U, the norm of g, for its bound to hold. In particular U is equal to $1/4\sqrt{(B+1)/\log(B+1)}$ (Dekel et al., 2007), where B is the budget parameter that sets the maximum size of the support set. We then selected the parameter η in the Projectron in each round according to Eq. (13). Hence the final size of the Projectron solution will depend on U and on the particular classification problem at hand. We have set B on each dataset roughly to 1/2 and 1/4 of the size of the Perceptron support set, for a total of 6 experiments. Note that Projectron can also be used without taking into account the norm of the competitor and considering η just as a parameter. In particular η should be set to trade accuracy for sparseness.

In Tables 1–3 we summarize the results of our experiments. The cumulative number of mistakes as percentage of the training size (mean ± std) and the size of the

[1] This update rule gives $\tau_t = 1$ on rounds in which there is a mistake.

[2] Downloaded from `http://www.sie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

Table 1. Adult9 dataset, 32561 samples.

Algorithm	% Mistakes	Size Support Set
Perceptron	$20.99\% \pm 0.06$	6835.6 ± 20.28
PA-I	$18.11\% \pm 0.10$	12537 ± 36.2
	B=1500	
Projectron	$20.95\% \pm 0.12$	1094.6 ± 16.06
Projectron++	$\mathbf{20.04\% \pm 0.14}$	$\mathbf{992.8 \pm 9.73}$
Forgetron	$21.90\% \pm 0.23$	1500
RBP	$22.05\% \pm 0.21$	1500
Stoptron	$22.73\% \pm 2.82$	1500
	B=3000	
Projectron	$20.97\% \pm 0.13$	1499.6 ± 13.58
Projectron++	$\mathbf{20.16\% \pm 0.11}$	$\mathbf{1364.2 \pm 4.76}$
Forgetron	$21.41\% \pm 0.13$	3000
RBP	$21.49\% \pm 0.11$	3000
Stoptron	$21.04\% \pm 1.54$	3000

Table 2. Vehicle dataset, 78823 samples.

Algorithm	% Mistakes	Size Support Set
Perceptron	$19.58\% \pm 0.09$	15432.0 ± 69.62
PA-I	$15.27\% \pm 0.05$	30131.4 ± 21.07
	B=4000	
Projectron	$19.63\% \pm 0.08$	3496.4 ± 18.39
Projectron++	$\mathbf{18.27\% \pm 0.06}$	$\mathbf{3187.0 \pm 13.64}$
Forgetron	$20.40\% \pm 0.04$	4000
RBP	$20.32\% \pm 0.04$	4000
Stoptron	$19.49\% \pm 3.56$	4000
	B=8000	
Projectron	$19.62\% \pm 0.04$	4668.2 ± 32.88
Projectron++	$\mathbf{18.53\% \pm 0.07}$	$\mathbf{4309.6 \pm 28.67}$
Forgetron	$19.98\% \pm 0.06$	8000
RBP	$19.94\% \pm 0.06$	8000
Stoptron	$20.17\% \pm 2.03$	8000

Table 3. Synthetic dataset, 10000 samples.

Algorithm	% Mistakes	Size Support Set
Perceptron	$18.80\% \pm 0.25$	1880.0 ± 25.12
PA-I	$12.58\% \pm 0.05$	3986.8 ± 42.83
	B=1000	
Projectron	$18.71\% \pm 0.14$	108.6 ± 2.97
Projectron++	$\mathbf{14.09\% \pm 0.10}$	$\mathbf{104.2 \pm 2.39}$
Forgetron	$18.96\% \pm 0.32$	1000
RBP	$18.86\% \pm 0.29$	1000
Stoptron	$17.49\% \pm 1.77$	1000
	B=500	
Projectron	$18.70\% \pm 0.21$	$\mathbf{98.6 \pm 3.05}$
Projectron++	$\mathbf{14.23\% \pm 0.10}$	$\mathbf{98.6 \pm 2.30}$
Forgetron	$19.20\% \pm 0.19$	500
RBP	$19.27\% \pm 0.20$	500
Stoptron	$21.96\% \pm 4.62$	500

support set are reported. In all the experiments both the Projectron and the Projectron++ outperform the Forgetron and the RBP with a smaller support size. Moreover, the Projectron++ always outperforms the Projectron and has smaller support set. Due to its theoretically derived formulation, it achieves better results even if being bounded, and it has better performance than the Perceptron. In particular it gets closer to the classification rate of the PA-I, without paying the price of a large support set. It is interesting to note the performances of the Stoptron: it has an accuracy close to the other bounded algorithms in average, but with much bigger variance. This indicates that all the examined strategies for bounded learning are always better than the simple procedure to stop learning, at least to have stable performances.

Last, we show the behavior of the algorithms over time. In Fig. 1 we show the average online error rate, that is, the total numbers of errors on the examples seen as a function of the number of samples for all algorithms on the Adult9 dataset with $B = 1500$. Note how the Projectron algorithm closely tracks the Perceptron. On the other hand the Forgetron and the RBP stop improving after reaching the support set size B, around 7500 samples. The growth of the support set as a function of the number of samples is depicted in Fig. 2. While for PA-I and Perceptron the growth is clearly linear, it is sub-linear for Projectron and for the Projectron++ and they will reach a maximum size and then they will stop growing (as stated in Thm. 1). In Fig. 3 we show the average online error rate as a function of the size of the support set. It is clear that the Projectron and the Projectron++ outperform the Perceptron with smaller support set.

6. Discussion

This paper presented two different versions of a bounded online learning algorithm. The algorithms depend on a parameter that allows to trade accuracy for sparseness of the solution. The size of the solution is always guaranteed to be bounded, therefore it solves the memory explosion problem of the Perceptron and similar algorithms. Although the size of the support set is guaranteed to be bounded, the actual size of the support set cannot be determined in advance, like in the Forgetron algorithm, and it is not fixed. Practically, the size of the support set of the Projectron algorithms is much smaller than that of the budget algorithms.

Compared to budget algorithms it has the advantage of a bounded support set size without removing or scaling instances in the set. This keeps performance high. We call this algorithm Projectron. Its second variant, the Projectron++, always outperforms the standard Perceptron algorithm, while assuring a bounded solution. Another advantage over budget algorithms is the possibility to obtain bounded batch solutions using standard online-to-batch conversion. In fact using the averaging conversion (Cesa-Bianchi et al., 2004) we get a bounded solution. This is not true for budget algorithms, where more sophisticated techniques have to be used (Dekel & Singer, 2005). A similar approach has been used in (Csató & Opper, 2001) in the framework of the Gaussian Processes. However in that

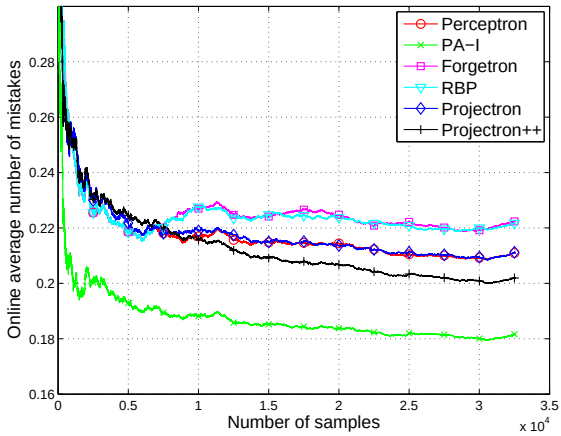


Figure 1. Average online error for the different algorithms on *Adult9* dataset as a function of the number of training samples. B is set to 1500.

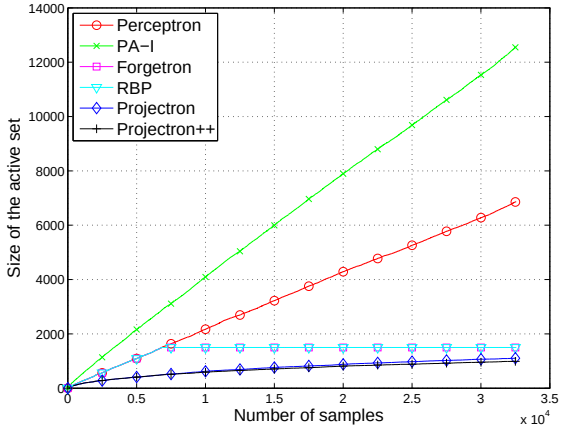


Figure 2. Size of the support set for the different algorithms on *Adult9* dataset as a function of the number of training samples. B is set to 1500.

paper no mistake bounds were derived and the use of the hinge loss allows us to have sparser solution.

Acknowledgments. This work was supported by EU project DIRAC (FP6-0027787).

References

Cauwenberghs, G., & Poggio, T. (2000). Incremental and decremental support vector machine learning. *Advances in Neural Information Processing Systems 14*.

Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2004). On the generalization ability of on-line learning algorithms. *IEEE Trans. on Information Theory*, *50*, 2050–2057.

Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2006). Tracking the best hyperplane with a simple budget Perceptron. *Proc. of the 19th Conference on Learning Theory* (pp. 483–498).

Cheng, L., Vishwanathan, S. V. N., Schuurmans, D., Wang, S., & Caelli, T. (2007). Implicit online learning with kernels. *Advances in Neural Information Processing Systems 19*.

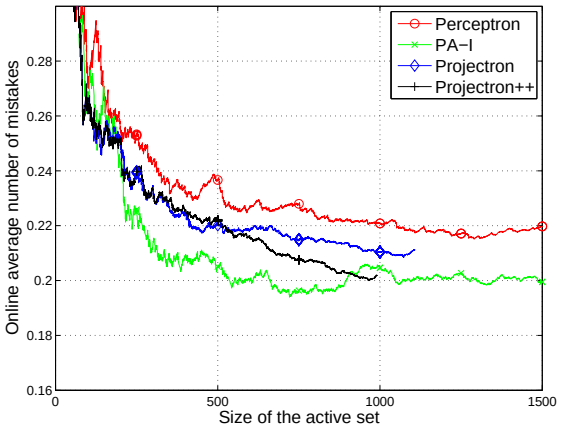


Figure 3. Average online error for the different algorithms on *Adult9* dataset as a function of the size of the support set. B is set to 1500.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, *7*, 551–585.

Crammer, K., Kandola, J., & Singer, Y. (2003). Online classification on a budget. *Advances in Neural Information Processing Systems 16*.

Csató, L., & Opper, M. (2001). Sparse representation for gaussian process models. *Advances in Neural Information Processing Systems 13*.

Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2007). The Forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, *37*, 1342–1372.

Dekel, O., & Singer, Y. (2005). Data-driven online to batch conversions. *Advances in Neural Information Processing Systems 18*.

Downs, T., Gates, K. E., & Masters, A. (2001). Exact simplification of support vectors solutions. *Journal of Machine Learning Research*, *2*, 293–297.

Engel, Y., Mannor, S., & Meir, R. (2002). Sparse online greedy support vector regression. *Proceedings 13th European Conference on Machine Learning*.

Kivinen, J., Smola, A., & Williamson, R. (2004). Online learning with kernels. *IEEE Trans. on Signal Processing*, *52*, 2165–2176.

Orabona, F., Castellini, C., Caputo, B., Luo, J., & Sandini, G. (2007). Indoor place recognition using online independent support vector machines. *Proc. of the British Machine Vision Conference 2007* (pp. 1090–1099).

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–407.

Schölkopf, B., Herbrich, R., Smola, A., & Williamson, R. (2000). A generalized representer theorem. *Proc. of the 13th Conference on Computational Learning Theory*.

Shalev-Shwartz, S., & Singer, Y. (2005). A new perspective on an old perceptron algorithm. *Proc. of the 18th Conference on Learning Theory* (pp. 264–278).

Weston, J., Bordes, A., & Bottou, L. (2005). Online (and offline) on an even tighter budget. *Proceedings of AISTATS 2005* (pp. 413–420).

Learning Dissimilarities by Ranking: From SDP to QP

Hua Ouyang HOUYANG@CC.GATECH.EDU
College of Computing, Georgia Institute of Technology

Alex Gray AGRAY@CC.GATECH.EDU
College of Computing, Georgia Institute of Technology

Abstract

We consider the problem of learning dissimilarities between points via formulations which preserve a specified ordering between points rather than the numerical values of the dissimilarities. *Dissimilarity ranking (d-ranking)* learns from instances like "A is more similar to B than C is to D" or "The distance between E and F is larger than that between G and H". Three formulations of d-ranking problems are presented and new algorithms are presented for two of them, one by semidefinite programming (SDP) and one by quadratic programming (QP). Among the novel capabilities of these approaches are out-of-sample prediction and scalability to large problems.

1. Introduction

Ranking or sometimes referred as *ordinal regression*, is a statistical learning problem which gained much attention recently (Cohen et al., 1998; Herbrich et al., 1999; Joachims, 2002). This problem learns from relative comparisons like "A ranks lower than B" or "C ranks higher than D". The goal is to learn an explicit or implicit function which gives *ranks* over an sampling space $\mathbb{X}$. In most of these tasks, the sampled instances to be ranked are vector-valued data in $\mathbb{R}^D$, while the ranks are real numbers which can be either discrete or continuous. If the problem is to learn a real valued ranking function, it can be stated as: given a set $\mathcal{S}$ of pairs $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}$ (which indicates that the rank of $\mathbf{x}_i$ is lower than $\mathbf{x}_j$), learn a real valued $f : \mathbb{X} \to \mathbb{R}$ that satisfies $f(\mathbf{x}_m) < f(\mathbf{x}_n)$ if the rank of $\mathbf{x}_m$ is lower than $\mathbf{x}_n$.

In this paper we investigate a special ranking problem: *dissimilarity ranking* (*d-ranking*). Unlike ranking, this problem learns from instances like "A is more similar to B than C is to D" or "The distance between E and F is larger than that between G and H". Note that the dissimilarities here are not necessarily distances. Other than real vectors in conventional ranking problems, the data to be ranked here are dissimilates of pairwised data vectors. This problem can be stated as: learning an explicit or implicit function which gives ranks over a space of dissimilarities $d(\mathbb{X}, \mathbb{X}) \in \mathbb{R}$. Based on different requirements of applications, this learning problem can have various formulations. We will present some of them in Section 2.

D-ranking can be regarded as a special instance of dissimilarity learning (or metric learning). Different dissimilarity learning methods have different goals. We highlight some previous work as below.

- In metric learning methods (Hastie & Tibshirani, 1996; Xing et al., 2002), the purpose of learning a proper Mahalanobis distance is to achieve better class/cluster separations.
- In kernel learning methods (Lanckriet et al., 2004; Micchelli & Pontil, 2005), learning a proper kernel is equivalent to learning a good inner-product function which introduces a dissimilarity in the input space. The purpose is to maximize the performance of a kernel-based learning machine.
- Multidimensional scaling (MDS) (Borg & Groenen, 2005) and Isomap (Tenenbaum et al., 2000) can also be regarded as learning an implicity function $f : \mathbb{R}^D \to \mathbb{R}^L$. The purpose of learning an embedding is to preserve distances in a low-dimensional Euclidean space $\mathbb{R}^L$.

In our d-ranking problems, the purpose of learning a proper dissimilarity is to *preserve the ranks* of dissimilarities, not the absolute values of them (which is the case in MDS and Isomap). For example, if we know

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

that "The distance between A and B is smaller than that between C and D", the problem can be formulated as: find a dissimilarity function d, such that $d(A, B) < d(C, D)$.

Unlike conventional learning and ranking problems, d-ranking hasn't received intensive studies in previous research. One of the most important related work is the nonmetric multidimensional scaling (NMDS) (Borg & Groenen, 2005). Given a symmetric proximity (similarity or dissimilarity) matrix $\Delta = [\delta_{mn}]$, NMDS tries to find a low dimensional embedding space $\mathbb{R}^L$ such that $\forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l \in \mathbb{R}^L$, $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 < \|\mathbf{x}_k - \mathbf{x}_l\|_2^2 \Leftrightarrow \delta_{ij} < \delta_{kl}$. NMDS was recently extended to the generalized NMDS (GNMDS) (Agarwal, 2007). GNMDS does not need to know the absolute values of proximities δ_{mn}. Instead it only need a set $\mathcal{S}$ of quadruples $(i, j, k, l) \in \mathcal{S}$, which indicate that $\delta_{ij} < \delta_{kl}$.

Both NMDS and GNMDS learn an embedding space instead of learning an explicit ranking function, thus they are unable to handle out-of-sample problems. Schultz et. al. gave a solution to these problems by proposing to learn a distance metric from relative comparisons (Schultz & Joachims, 2003). They choose to learn a Mahalanobis distance which can preserve ranks of distances. Since the learned distance functions are parameterized, they can be used to handle new samples. The proposed formulation was solved in a similar manner as SVM. Nonetheless, the regularization term was not well justified.

Many applications in biology, computer vision, web search, social science etc. can be put into the framework of d-ranking problems. Take document classification as an instance. Without adequate domain knowledge, it is hard to accurately determine the quantitative dissimilarities between two documents. However, comparing the dissimilarities between every three or four documents can be easily done, either automatically or manually. Generally speaking, d-ranking is especially useful when the quantized dissimilarities are not reliable.

In Section 2, we propose three formulations of d-ranking problems. Section 3 gives the numerical solutions for solving d-ranking by SDP. Section 4 shows how to solve d-ranking by QP. The proposed methods are evaluated in Section 5. Section 6 concludes the paper.

2. Three Formulations of D-Ranking

D-ranking problems can have various formulations depending on specific requirements or settings of applications. Next we will give three formulations.

Formulation 2.1. ***(F1) Inputs**: a set $\mathcal{S}$ of ordered quadruples $(i, j, k, l) \in \mathcal{S}$, indicating that $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq d(\boldsymbol{x}_k, \boldsymbol{x}_l)$, where $d(\cdot, \cdot)$ is a fixed but unknown dissimilarity function; **Outputs**: coefficients of embedded samples $\boldsymbol{x}_i', \boldsymbol{x}_j', \boldsymbol{x}_k', \boldsymbol{x}_l' \in \mathbb{R}^L$; **Criteria**: $(i, j, k, l) \in \mathcal{S} \Leftrightarrow \|\boldsymbol{x}_i' - \boldsymbol{x}_j'\|_2^2 \leq \|\boldsymbol{x}_k' - \boldsymbol{x}_l'\|_2^2$.*

As proposed by Agarwal et. al. (Agarwal, 2007), in **F1** we neither assume any geometry of the input space, nor assume any form of dissimilarities in it. We do not need to know the coefficients of input samples. Only ordering information is provided. Nonetheless we assume a Euclidean metric in the embedding space, which is often of low dimensions (e.g. $L = 2$, or 3). As shown in Section 3, **F1** can be formed as a problem of semidefinite programming (SDP).

Formulation 2.2. ***(F2) Inputs**: a set $\mathcal{S}$ of ordered quadruples $(i, j, k, l) \in \mathcal{S}$, indicating that $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq d(\boldsymbol{x}_k, \boldsymbol{x}_l)$, where $d(\cdot, \cdot)$ is a fixed but unknown dissimilarity function; corresponding coefficients in the input Euclidean space $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{x}_l \in \mathbb{R}^D$; **Outputs**: dissimilarity functions $\hat{d}(\cdot, \cdot) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$; **Criteria**: $(i, j, k, l) \in \mathcal{S} \Leftrightarrow \hat{d}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq \hat{d}(\boldsymbol{x}_k, \boldsymbol{x}_l)$.*

Unlike learning an embedding space as in **F1**, **F2** learns an explicit dissimilarity function $\hat{d}(\cdot, \cdot)$ which preserves the ranks of dissimilarities. We will show in Section 4 that **F2** can be handled in a very similar manner as support vector machines, where the quadratic programming (QP) problem can be solved efficiently by specialized sequential optimization methods. If in some cases we need to find a low dimensional Euclidean embedding of the input samples, we can then use the classical multidimensional scaling (MDS) to preserve the learned dissimilarities.

Formulation 2.3. ***(F3) Inputs**: a set $\mathcal{S}$ of ordered quadruples $(i, j, k, l) \in \mathcal{S}$, indicating that $d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq d(\boldsymbol{x}_k, \boldsymbol{x}_l)$, where $d(\cdot, \cdot)$ is a fixed but unknown dissimilarity function; corresponding coefficients in the input Euclidean space $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{x}_l \in \mathbb{R}^D$; **Outputs**: projection function $f : \mathbb{R}^D \to \mathbb{R}^L$, $\boldsymbol{x}_i', \boldsymbol{x}_j', \boldsymbol{x}_k', \boldsymbol{x}_l' \in \mathbb{R}^L$; **Criteria**: $(i, j, k, l) \in \mathcal{S} \Leftrightarrow \|\boldsymbol{x}_i' - \boldsymbol{x}_j'\|_2^2 \leq \|\boldsymbol{x}_k' - \boldsymbol{x}_l'\|_2^2$.*

Although we formulate **F3** as a function learning problem, currently we have not found any efficient method to solve it. This formulation will remains as our future work.

3. Solving F1 by SDP

F1 was studied by Agarwal et. al. (Agarwal, 2007). The authors proposed GNMDS which can be solved as

a SDP, as shown in Eq.(1).

$$\begin{aligned} &\text{GNMDS:} \\ \min\ & \sum_{ijkl\in\mathcal{S}} \xi_{ijkl} + \lambda \mathrm{tr(K)}, \\ \text{s.t. } & (K_{kk} - 2K_{kl} + K_{ll}) - (K_{ii} - 2K_{ij} + K_{jj}) \\ & + \xi_{ijkl} \geq 1, \\ & \sum_{ab} K_{ab} = 0,\ \xi_{ijkl} \geq 0,\ K \succeq 0, \\ & \text{for all } (i,j,k,l) \in \mathcal{S}. \end{aligned} \tag{1}$$

The main idea of GNMDS is to learn a positive semidefinite Gram matrix $K = X^T X$ which can be eigen-decomposed to recover the embedded samples. The relation between Euclidian distances and the Gram matrix is used:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = K_{ii} - 2K_{ij} + K_{jj}. \tag{2}$$

Nonetheless, the constraints that contain order information of dissimilarities are not sufficient to determine a unique K, since any rotation, translation or scaling can also satisfies these constraints. To reduce these ambiguities, they use $\sum_{ab} K_{ab} = 0$ to center all the embedded samples at the origin.

It is preferable in many applications to find low dimensional embedding spaces, e.g. 2D or 3D Euclidean space. Thus a low-rank K is desired. Unfortunately, minimizing rank(K) subject to linear inequality constraints is NP-Hard (Vandenberghe & Boyd, 1996). Thus the objective function is relaxed heuristically as minimizing trace(K), which is a convex envelope of the rank.

Figure 1 shows the result of GNMDS on a toy problem. The inputs are 990 pairwise ranks of distances between 10 European cities. The outputs are the recovered 2D coefficients. It can be observed that the recovered locations of the cities do not correspond to the true locations. Actually only 789 out of 990 pairs of ranks are preserved by the learned 2D embedding, i.e. 20.3% error rate. Figure 2 shows the 10 sorted eigenvalues of the Gram matrix K. Although the original space is a 2D Euclidean space, the first 2 eigenvalues only account for 49.4% of the total variation.

There are at least two reasons that account for the poor performance of GNMDS. Firstly, there is no guarantee on the quality of the solution of the relaxed problem compared with the original problem. There may exist some higher dimensional spaces which satisfy all the constants while have smaller traces than lower dimensional spaces. Secondly, due to the introduction of

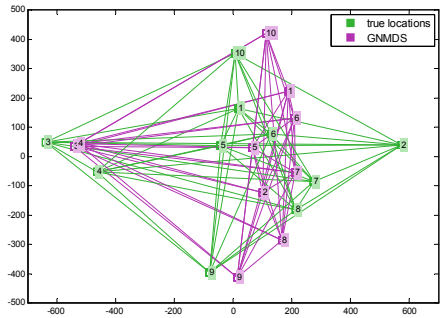


Figure 1. d-ranking toy problem: locations of ten European cities. Purple: true locations. Green: locations recovered by GNMDS. All the coefficients have been scaled before plotting.

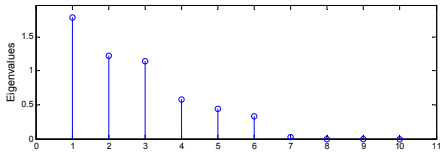


Figure 2. 10 sorted eigenvalues of the Gram matrix K learned by GNMDS.

slack variables ξ_{ijkl} in the inequality constraints, the learned embedding tends to push all the samples to the same point. The first problem can be solved by introducing better heuristics of the convex envelope of the rank. The second problem can be solved by the following slight modification of GNMDS:

$$\begin{aligned} &\text{Modified GNMDS:} \\ \min\ & \sum_{ijkl\in\mathcal{S}} \xi_{ijkl} + \lambda \mathrm{tr(K)}, \\ \text{s.t. } & (K_{kk} - 2K_{kl} + K_{ll}) - (K_{ii} - 2K_{ij} + K_{jj}) \\ & - \xi_{ijkl} \geq 1, \\ & \sum_{ab} K_{ab} = 0,\ \xi_{ijkl} \geq 0,\ K \succeq 0, \\ & \text{for all } (i,j,k,l) \in \mathcal{S}. \end{aligned} \tag{3}$$

The modified GNMDS just changes the slack variables from $+\xi_{ijkl}$ to $-\xi_{ijkl}$. This simple trick can ensure that all the differences between distances k,l and i,j are larger than 1, thus pulls the embedding samples apart. Figure 3 shows toy problem solved by the modified GNMDS. The recovered samples are closer to the true locations than those in Figure 1. There are 850 out of 990 pairs of ranks correctly preserved, i.e. 14.14% error rate, which is 6% lower than GNMDS. Figure

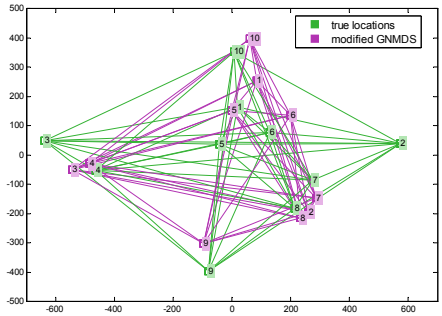


Figure 3. d-ranking toy problem: locations of ten European cities. Purple: true locations. Green: locations recovered by modified GNMDS. All the coefficients have been scaled before plotting.

4 shows the 10 eigenvalues of K learned by modified GNMDS. The first 2 eigenvalues account to 69.8% of the total variant, which is 20% higher than GNMDS.

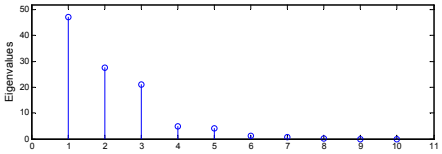


Figure 4. 10 sorted eigenvalues of the Gram matrix K learned by modified GNMDS.

4. Solving F2 by QP

As introduced in Section 2, instead of learning an embedding as in **F1**, a dissimilarity function $d(\mathbf{x}_i, \mathbf{x}_j) : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ is learned in **F2**, such that all the training ranks between $d(\cdot, \cdot)$ are preserved, and can generalize to new samples. This is indeed a dissimilarity learning problem.

Many previous metric learning methods (Hastie & Tibshirani, 1996; Goldberger et al., 2004; Kwok & Tsang, 2003) try to learn an alternative dissimilarity function by replacing the Euclidean metric with an properly learnt Mahalanobis metric, either globally or locally.

In this section we propose the *d-ranking Vector Machine* (*d-ranking-VM*) method. Unlike metric learning methods, d-ranking-VM is explicitly regularized. Thus we can have a full control over the complexity of $d(\mathbf{x}_i, \mathbf{x}_j)$. D-ranking-VM utilizes the technique of hyperkernel learning (Ong et al., 2005) which was originally proposed for learning a proper kernel.

D-ranking-VM is formulated as the following optimization problem:

$$\begin{aligned} &\text{d-ranking-VM (primal):} \\ \min\ & \frac{1}{N}\sum_{ijkl\in\mathcal{S}} \xi_{ijkl} + \lambda \|d\|^2_{\underline{\mathbb{H}}}, \\ \text{s.t. }\ & d(\mathbf{x}_k, \mathbf{x}_l) - d(\mathbf{x}_i, \mathbf{x}_j) - \xi_{ijkl} \geq 1, \\ & \xi_{ijkl} \geq 0, \\ & \text{for all } (i,j,k,l) \in \mathcal{S}. \end{aligned} \tag{4}$$

where $N = |\mathcal{S}|$. $\underline{\mathbb{H}}$ is a hyper-reproducing kernel Hilbert space (hyper-RKHS) from which the function $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ is drawn.

Like the representer theorem in RKHS (Kimeldorf & Wahba, 1971), there is also a representer theorem in hyper-RHKS (see (Ong et al., 2005) or (Kondor & Jebara, 2006) for the theorem and proofs):

$$d(\underline{\mathbf{x}}) = \sum_{p=1}^{M} c_p \underline{K}(\underline{\mathbf{x}}_p, \underline{\mathbf{x}}), \tag{5}$$

where $\underline{K}$ is a semidefinite hyperkernel, $\underline{\mathbf{x}}$ denotes a pair of samples $(\mathbf{x}_i, \mathbf{x}_j)$, and M is the number of distinct dissimilarity pairs provided by the training rank data $\mathcal{S}$. We denote the set of dissimilarity pairs as $\mathcal{D}$, and $M = |\mathcal{D}|$. Normally we have $M > N$ (the discussion of M and N is given in Section 5.1).

Substitute Eq.(5) into (4), we can change the primal problem to the following form:

$$\begin{aligned} \min\ & \frac{1}{N}\sum_{p\in\mathcal{S}} \xi_p + \lambda C^T \underline{K} C, \\ \text{s.t. }\ & \sum_{p=1}^{M} c_p \underline{K}(\underline{\mathbf{x}}_p; \mathbf{x}_k, \mathbf{x}_l) - \sum_{p=1}^{M} c_p \underline{K}(\underline{\mathbf{x}}_p; \mathbf{x}_i, \mathbf{x}_j) \\ & - \xi_p \geq 1, \\ & \xi_p \geq 0, \text{for all } p \in \mathcal{S}, \end{aligned} \tag{6}$$

where $C \in \mathbb{R}^M$ is a vector with the ith element being c_p, and $\underline{K} \in \mathbb{R}^{M\times M}$ is the hyper-kernel matrix.

The dual problem of (6) can be derived by using the Lagrangian technique. The solution to this optimization problem is given by the saddle point of the Lagrangian function:

$$\begin{aligned} L(C, \xi_p, \alpha_p, \zeta_p) = & \frac{1}{N}\sum_{p\in\mathcal{S}} \xi_p + \lambda C^T \underline{K} C - \sum_{p=1}^{N} \zeta_p \xi_p \\ & + \sum_{p=1}^{N} \alpha_p \left\{ \sum_{s=1}^{M} c_s \left[\underline{K}(\underline{\mathbf{x}}_s; \mathbf{x}_i, \mathbf{x}_j) - \underline{K}(\underline{\mathbf{x}}_s; \mathbf{x}_i, \mathbf{x}_j) \right] + 1 + \xi_p \right\}, \end{aligned} \tag{7}$$

where ζ_p and α_p are non-negative Lagrange multipliers. The primal problem is convex, thus there exist a strong duality between the primal and the dual. Utilizing the KKT optimality condition, we have:

$$\frac{\partial L}{\partial C} = 2\lambda \underline{K} C + \underline{K}(P-Q)A = 0, \tag{8}$$

and

$$\frac{\partial L}{\partial \xi_p} = \frac{1}{N} + \lambda_p - \zeta_p = 0, \tag{9}$$

where $A \in \mathbb{R}^N$ is a vector with the pth element being α_p. $P, Q \in \mathbb{R}^{M\times N}$ are two matrices with contain the rank information. Each column $p_{x\cdot}(x = 1 \dots N)$ of P and each column $q_{x\cdot}(x = 1 \dots N)$ of Q only contain one 1 and $M-1$ 0s. For example, if the rth training quadruples in $\mathcal{S}$ is (i, j, k, l), which means that $d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_k, \mathbf{x}_l)$, and if the pair (i, j) is the mth element in $\mathcal{D}$, while (k, l) is the $n$the element in $\mathcal{D}$, then $p_{rm} = 1$ and $q_{rn} = 1$.

From Eq.(8) and (9) we have:

$$C = \frac{(Q-P)A}{2\lambda}, \tag{10}$$

and

$$\alpha_p = \zeta_p - \frac{1}{N} \tag{11}$$

Substitute Eq.(10) and (11) into (6), we arrive at the following *dual* problem for d-ranking-VM:

$$\begin{aligned} &\text{d-ranking-VM (dual):} \\ &\max \sum_{p=1}^{N} \alpha_p - \frac{A^T(Q-P)^T \underline{K}(Q-P)A}{4\lambda}, \\ &\text{s.t. } \alpha_p \geq 0, \\ &\text{for all } (i,j,k,l) \in \mathcal{S}. \end{aligned} \tag{12}$$

This problem is a quadratic programming (QP) problem which shares a similar form as SVM. Thus the sequential optimization techniques of SVM can be readily employed for d-ranking-VM. To perform testing, we can use the learnt dissimilarity function in Eq.(5) and make pairwise comparisons.

An important problem for kernel learning methods is the selection of proper kernels. This problem also exists in hyperkernel learning methods. Here we propose some examples of hyperkernels, which are hyperextensions of Gaussian RBF kernels and polynomial kernels. The construction of these hyperkernels are based on the following proposition.

Proposition 4.1. *Let $k_a(\cdot,\cdot)$ and $k_b(\cdot,\cdot)$ be positive definite kernels, then $\forall \boldsymbol{x}_1, \boldsymbol{x}_1', \boldsymbol{x}_2, \boldsymbol{x}_2' \in \mathbb{X}$, and $\forall \alpha, \beta > 0$, $\big(k_a(\boldsymbol{x}_1, \boldsymbol{x}_2)\big)^\alpha \big(k_b(\boldsymbol{x}_1', \boldsymbol{x}_2')\big)^\beta$ or $\alpha k_a(\boldsymbol{x}_1, \boldsymbol{x}_2) + \beta k_b(\boldsymbol{x}_1', \boldsymbol{x}_2')$ can give a hyperkernel $\underline{k}$.*

Proof. See appendix. □

Example 4.2. ***(Gaussian symmetric product hyperkernel)*** *Let k_a and k_b be the same Gaussian RBF kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\big(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}\big)$, and let $\alpha = \beta = 1$, the Gaussian symmetric product hyperkernel is given by:*

$$\begin{aligned} &\underline{k}\big((\boldsymbol{x}_1, \boldsymbol{x}_1'), (\boldsymbol{x}_2, \boldsymbol{x}_2')\big) = k(\boldsymbol{x}_1, \boldsymbol{x}_1') k(\boldsymbol{x}_2, \boldsymbol{x}_2') \\ &= \exp\bigg(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_1'\|^2 + \|\boldsymbol{x}_2 - \boldsymbol{x}_2'\|^2}{2\sigma^2}\bigg) \end{aligned} \tag{13}$$

Example 4.3. ***(Gaussian symmetric sum hyperkernel)*** *Under the same conditions as Example 4.2, we can construct the Gaussian symmetric sum hyperkernel as:*

$$\begin{aligned} &\underline{k}\big((\boldsymbol{x}_1, \boldsymbol{x}_1'), (\boldsymbol{x}_2, \boldsymbol{x}_2')\big) = k(\boldsymbol{x}_1, \boldsymbol{x}_1') + k(\boldsymbol{x}_2, \boldsymbol{x}_2') \\ &= \exp\bigg(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_1'\|^2}{2\sigma^2}\bigg) + \exp\bigg(-\frac{\|\boldsymbol{x}_2 - \boldsymbol{x}_2'\|^2}{2\sigma^2}\bigg) \end{aligned} \tag{14}$$

Example 4.4. ***(polynomial symmetric product hyperkernel)*** *Let k_a and k_b be the same polynomial kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \big(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + q\big)^p$, and let $\alpha = \beta = 1$, we can construct the polynomial symmetric product hyperkernel as:*

$$\underline{k}\big((\boldsymbol{x}_1, \boldsymbol{x}_1'), (\boldsymbol{x}_2, \boldsymbol{x}_2')\big) = \big(\langle \boldsymbol{x}_1, \boldsymbol{x}_1' \rangle + q\big)^p \big(\langle \boldsymbol{x}_2, \boldsymbol{x}_2' \rangle + q\big)^p \tag{15}$$

Example 4.5. ***(polynomial symmetric sum hyperkernel)*** *Under the same conditions as Example 4.4, we can construct the polynomial symmetric sum hyperkernel as:*

$$\underline{k}\big((\boldsymbol{x}_1, \boldsymbol{x}_1'), (\boldsymbol{x}_2, \boldsymbol{x}_2')\big) = \big(\langle \boldsymbol{x}_1, \boldsymbol{x}_1' \rangle + q\big)^p + \big(\langle \boldsymbol{x}_2, \boldsymbol{x}_2' \rangle + q\big)^p \tag{16}$$

5. Experiments

The proposed d-ranking methods in Section 3 and Section 4 are evaluated by several experiments.

5.1. Obtaining Ranks of Pairs from Data

To test our methods, we need to obtain pairwise distance ranks. This can be done in many ways. Generally speaking, for a problem of n data samples, the total number of available distance pairs are $M = C_n^2 = \frac{n(n-1)}{2}$ (if we take $d(\mathbf{x}_i, \mathbf{x}_j)$ and $d(\mathbf{x}_j, \mathbf{x}_i)$ as the same

distance). The total number of pairwise distance ranks are $N = C_M^2 = \frac{M(M-1)}{2} = \frac{n^4}{8} - \frac{n^3}{4} - \frac{n^2}{8} + \frac{n}{4}$ (if we take $d(\mathbf{x}_i, \mathbf{x}_j) < d(\mathbf{x}_k, \mathbf{x}_l)$ and $d(\mathbf{x}_k, \mathbf{x}_l) > d(\mathbf{x}_i, \mathbf{x}_j)$ as the same rank pair). Table 1 gives some examples of the relation between n and N. When n grows, the number of rank constraints will increase dramatically. Even solving a problem of $n > 100$ will be impossible for some optimization solvers.

Table 1. Some examples of N v.s. n.

n	2	3	4	10	20	50	100	1000
N	0	3	15	990	17955	749700	12248775	1.2475e+11

Here we reduce N by considering order transitivities, i.e. if $A > B$ and $B > C$, then the rank pair $A > C$ can be ignored (automatically satisfied) in the optimization constraints. The method is very simple. Firstly we sort the M distances decreasingly. Then we take the adjacent two distances to form one distance rank pair. By doing this, N can be reduced to $\frac{n^2}{2} - \frac{n}{2} - 1$. This is the maximum number of N which carries full rank information of all the distances. Of course in some applications, the rank information is not be fully given, and $N < \frac{n^2}{2} - \frac{n}{2} - 1$.

We test our method on three data sets: 1) 2D locations of 109 largest cities in the continental US; 2) 100 images of handwritten digits "3" and "5" from the USPS database, each of size 16×16; 3) 126 face images of 4 people from the UMist database, each of size 112×92.

For GNMDS and modified GNMDS methods, all ranks of distance pairs are fed to a SDP solver, and the recovered 2D embeddings are plotted. We used SeDuMi (Strum, 1999) to get the results given in the following subsection. For d-ranking-VM, LibSVM (Chang & Lin, 2001) is employed as our QP solver. It is implemented by employing the sequential minimal optimization (SMO) technique. The learned dissimilarities are used as a "pseudo-distances", and are fed to the classical MDS. The recovered 2D embeddings are then plotted.

5.2. Results of US Cities

In this data set, $n = 109$ and $N = 5885$. Every location of the cities is given by a 2D coefficient. Figure 5 shows the true locations. Figure 6 shows the results given by GNMDS.

It can be observed that GNMDS cannot correctly recover the embedding based on distance ranks. Most of the embedded samples are pushed to a line. 50.3% of the distances ranks are preserved, which are the results of randomness. This gives an evidence to the analysis in Section 3. Figure 7 shows the result given by the modified GNMDS. The recovered embedding roughly reflects the geometry of the cities. 74.5% of the distances ranks have been preserved. Since the distance information is not provided, there is no hope to match the true locations exactly.

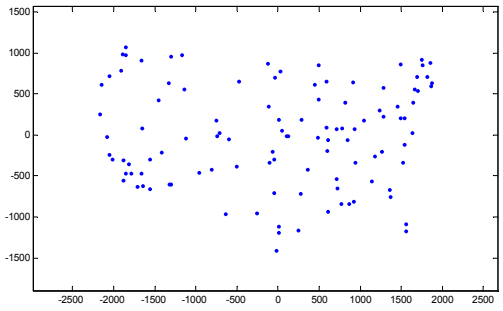


Figure 5. Locations of 109 largest cities in the continental United States.

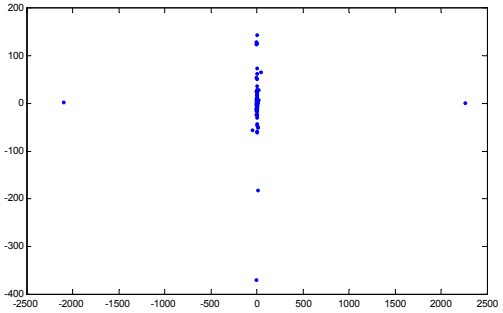


Figure 6. Recovered locations of 109 US cities given by GNMDS.

Figure 8 shows the results given by d-ranking-VM, where $\lambda = 10$, and the Gaussian symmetric product hyperkernel is used, with $\sigma = 15$. 97.9% of the distances ranks are preserved.

Table 2 shows the runtime of the above experiments.

Table 2. Runtime comparison for the three d-ranking methods.

Method	Runtime(minutes)
GNMDS	124
modified GNMDS	71
d-ranking-VM	4.5

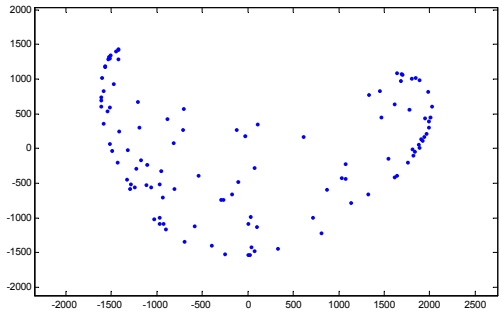


Figure 7. Recovered locations of 109 US cities given by the modified GNMDS.

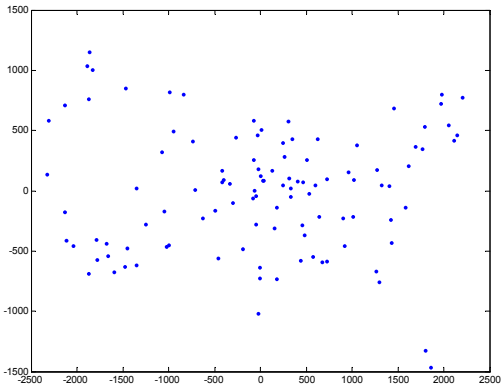


Figure 8. Recovered locations of 109 US cities given by the d-ranking-VM, using Gaussian symmetric product hyperkernel.

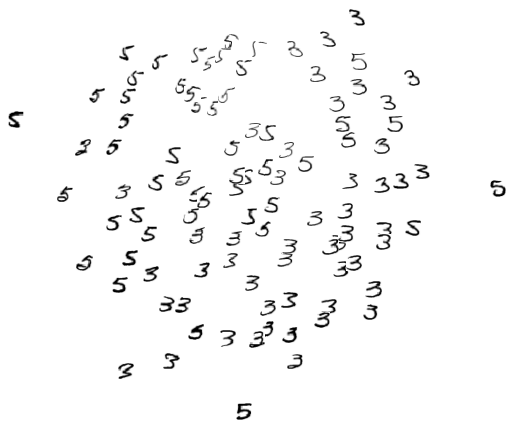


Figure 9. Recovered locations of USPS handwritten digits "3" and "5" given by d-ranking-VM.

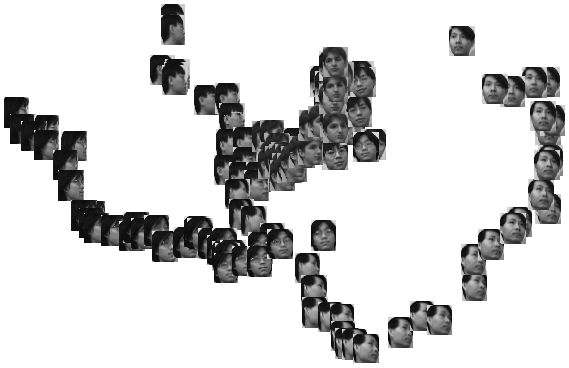


Figure 10. Recovered locations of UMist human faces given by d-ranking-VM.

5.3. Results of USPS Handwritten Digits

In this data set, $n = 100$ and $N = 4949$. The dimension every data sample is $16 \times 16 = 256$. Figure 9 shows the recovered 2D results given by RnakD-VM.

5.4. Results of UMist Human Faces

In this data set, $n = 126$ and $N = 7874$. The dimension every data sample is $112 \times 92 = 10304$. Figure 10 shows the recovered 2D results given by RnakD-VM.

6. Discussions and Conclusions

We have presented three d-ranking formulations, and give numerical solutions for two of them, namely solving d-ranking by SDP and solving d-ranking by QP. Each of them has its advantages and shortcomings. We list some pros and cons from different perspectives:

- Pros for d-ranking by SDP (GNMDS and the modified version): It can recover low dimensional embedding directly. Only ordering information is needed. There is no need to know the values of sample coefficients.
- Cons for d-ranking by SDP (GNMDS and the modified version): Solving SDP is hard, especially for large scale problems. Even sophistries SDP solver can only solve $N < 10^3$ problems with the number of constraints less than 10^5. It cannot be used to predict unseen samples.
- Pros for d-ranking by QP (d-ranking-VM): Solving QP in our case is much easier than SDP, since it can be converted to a similar form as SVM. Using sequential methods (SMO), can solve $N > 10^5$ problems. The learn dissimilarity function can be used to predict unseen samples.
- Cons for d-ranking by QP (d-ranking-VM): It can-

not recover low-dimensional embedding explicitly. One needs to use MDS or other embedding methods after learning the dissimilarities. Learning dissimilarity measure needs to know the coefficients of original samples. Like kernel methods, how to choose a good hyperkernel is crucial in solving a specific problem.

To our knowledge, this is the first work which brings out-of-sample prediction capability and large-scale scalability to d-ranking problems. Note that the technique of d-ranking-VM can also be employed in solving distances preserving problems. We will investigate the regularization properties and evaluate the performances of different hyperkernels in the following research. Finding a numerical solution for formulation **F3** will also be our future work.

7. Appendix: Proof of Proposition 4.1

We need to prove that $\underline{k}$ is a kernel on $\mathbb{X}^2$.

Denote $\bigotimes$ as the Kronecker product of two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$:

$$A \bigotimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}, \qquad (17)$$

and define $\bigoplus$ as:

$$A \bigoplus B = \begin{bmatrix} a_{11}\mathbb{1} + B & \cdots & a_{1n}\mathbb{1} + B \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbb{1} + B & \cdots & a_{mn}\mathbb{1} + B \end{bmatrix}, \qquad (18)$$

where $\mathbb{1}$ is a matrix of all ones.

Denote the kernel matrix of $k_a(\mathbf{x}_1, \mathbf{x}_2)$ as K_a, and that of $k_b(\mathbf{x}_1^{'}, \mathbf{x}_2^{'})$ as K_b. Denote the hyperkernel matrix of $\underline{k}$ as $\underline{K}$.

It is easy to verify that we can construct $\underline{K} = K_a \bigotimes K_b$. Since K_a and K_b are positive definite, their eigenvalues μ_a and μ_b are positive. Thus the eigenvalues of $\underline{K}$: $v_{ij} = \alpha\beta\mu_{ai}\mu_{bj}$ are also positive. A symmetric matrix $\underline{K}$ with positive eigenvalues is positive definite. Thus $\underline{k} = \big(k_a(\mathbf{x}_1, \mathbf{x}_2)\big)^{\alpha}\big(k_b(\mathbf{x}_1^{'}, \mathbf{x}_2^{'})\big)^{\beta}$ is a valid hyperkernel.

We can also verify that $\alpha K_a \bigoplus \beta K_b = \alpha K_a \bigotimes \mathbb{1} + \beta\mathbb{1} \bigotimes K_b$. Since K_a, K_b and $\mathbb{1}$ are all positive semidefinite and $\alpha, \beta > 0$, $\underline{K} = \alpha K_a \bigoplus \beta K_b$ is positive semidefinite. Thus $\underline{k} = \alpha k_a(\mathbf{x}_1, \mathbf{x}_2) + \beta k_b(\mathbf{x}_1^{'}, \mathbf{x}_2^{'})$ is a valid hyperkernel.

References

Agarwal, S. (2007). Generalized non-metric multidimensional scaling. *AISTATS 07*.

Borg, I., & Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. New York: Springer.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: a library for support vector machines. Software available at *http://www.csie.ntu.edu.tw/ ∼ cjlin/libsvm*.

Cohen, W. W., Schapire, R. E., & Singer, Y. (1998). Learning to order things. *NIPS*.

Goldberger, J., Roweis, S., Hinton, G., & Salakhutdinov, R. (2004). Neighbourhood Components Analysis. *NIPS*.

Hastie, T., & Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. *IEEE Trans. PAMI*, *18*, 607–616.

Herbrich, R., Graepel, T., & Obermayer, K. (1999). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, 115–132.

Joachims, T. (2002). Optimizing search engines using clickthrough data. *Proc. of SIGKDD '02*.

Kimeldorf, G., & Wahba, G. (1971). Some Results on Tchebycheffian Spline Functions. *Journal of Mathematical Analysis and Applications*, *33*, 82–75.

Kondor, R., & Jebara, T. (2006). Gaussian and Wishart Hyperkernels. *NIPS*.

Kwok, J. T., & Tsang, I. W. (2003). Learning with Idealized Kernels. *ICML*.

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L., & Jordan, M. I. (2004). Learning the Kernel Matrix with Semidenite Programming. *Journal of Machine Learning Research*, *5*, 27–72.

Micchelli, C. A., & Pontil, M. (2005). Learning the Kernel Function via Regularization. *Journal of Machine Learning Research*, *6*, 1099–1125.

Ong, C. S., Smola, A. J., & Williamson, R. C. (2005). Learning the Kernel with Hyperkernels. *Journal of Machine Learning Research*, *6*, 1043–1071.

Schultz, M., & Joachims, T. (2003). Learning a Distance Metric from Relative Comparisons. *NIPS*.

Strum, J. F. (1999). Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, *11*.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, *290*, 2319–2323.

Vandenberghe, L., & Boyd, S. (1996). Semidefinite Programming. *SIAM Review*, *38*, 49–95.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2002). Distance Metric Learning with Application to Clustering with Side-Information. *NIPS*.

A Distance Model for Rhythms

Jean-François Paiement PAIEMENT@IDIAP.CH
Yves Grandvalet YVES.GRANDVALET@UTC.FR
IDIAP Research Institute, Case Postale 592, CH-1920 Martigny, Switzerland

Samy Bengio BENGIO@GOOGLE.COM
Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA

Douglas Eck DOUGLAS.ECK@UMONTREAL.CA
Université de Montréal, Department of Computer Science, CP 6128, Succ. Centre-Ville, Montréal, Québec H3C 3J7, Canada

Abstract

Modeling long-term dependencies in time series has proved very difficult to achieve with traditional machine learning methods. This problem occurs when considering music data. In this paper, we introduce a model for rhythms based on the distributions of distances between subsequences. A specific implementation of the model when considering Hamming distances over a simple rhythm representation is described. The proposed model consistently outperforms a standard Hidden Markov Model in terms of conditional prediction accuracy on two different music databases.

1. Introduction

Reliable models for music would be useful in a broad range of applications, from contextual music generation to on-line music recommendation and retrieval. However, modeling music involves capturing long-term dependencies in time series, which has proved very difficult to achieve with traditional statistical methods. Note that the problem of long-term dependencies is not limited to music, nor to one particular probabilistic model (Bengio et al., 1994).

Music is characterized by strong hierarchical dependencies determined in large part by *meter*, the sense of strong and weak beats that arises from the interaction among hierarchical levels of sequences having nested periodic components. Such a hierarchy is implied in western music notation, where different levels are indicated by kinds of notes (whole notes, half notes, quarter notes, etc.) and where bars establish measures of an equal number of beats. Meter and rhythm provide a framework for developing musical melody. For example, a long melody is often composed by repeating with variation shorter sequences that fit into the metrical hierarchy (e.g. sequences of 4, 8 or 16 measures). It is well know in music theory that *distance patterns* are more important than the actual choice of notes in order to create coherent music (Handel, 1993). In this work, distance patterns refer to distances between subsequences of equal length in particular positions. For instance, measure 1 may be always similar to measure 5 in a particular musical genre. In fact, even random music can sound structured and melodic if it is built by repeating random subsequences with slight variation.

Many algorithms have been proposed for audio beat tracking (Dixon, 2007; Scheirer, 1998). Probabilistic models have also been proposed for tempo tracking and inference of rhythmic structure in musical audio (Whiteley et al., 2007; Cemgil & Kappen, 2002). The goal of these models is to align rhythm events with the metrical structure. However, simple Markovian assumptions are used to model the transitions between rhythms themselves. Hence, these models do not take into account long-term dependencies. A few generative models have already been proposed for music in general (Pachet, 2003; Dubnov et al., 2003). While these models generate impressive musical results, we are not aware of quantitative comparisons between models of music with machine learning standards, as it is done in Section 3 in terms of out-of-sample prediction accuracy. In this paper, we focus on modeling rhyth-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

mic sequences, ignoring for the moment other aspects of music such as pitch, timbre and dynamics. However, by capturing aspects of global temporal structure in music, this model should be valuable for full melodic prediction and generation: combined with an audio transcription algorithm, it should help improve the poor performance of state-of-the-art transcription systems; it could as well be included in genre classifiers or automatic composition systems (Eck & Schmidhuber, 2002); used to generate rhythms, the model could act as a drum machine or automatic accompaniment system which learns by example.

Our main contribution is to propose a generative model for distance patterns, specifically designed for capturing long-term dependencies in rhythms. In Section 2, we describe the model, detail its implementation and present an algorithm using this model for rhythm prediction. The algorithm solves a constrained optimization problem, where the distance model is used to filter out rhythms that do not comply with the inferred structure. The proposed model is evaluated in terms of conditional prediction error on two distinct databases in Section 3 and a discussion follows.

2. Distance Model

In this Section, we present a generative model for distance patterns and its application to rhythm sequences. Such a model is appropriate for most music data, where distances between subsequences of data exhibit strong regularities.

2.1. Motivation

Let $\mathbf{x}^l = (x^l_1, \ldots, x^l_m) \in \mathbb{R}^m$ be the l-th rhythm sequence in a dataset $\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ where all the sequences contain m elements. Suppose that we construct a *partition* of this sequence by dividing it into ρ parts defined by $y^l_i = (x^l_{1+(i-1)m/\rho}, \ldots, x^l_{im/\rho})$ with $i \in \{1, \ldots, \rho\}$. We are interested in modeling the distances between these subsequences, given a suitable metric $d(y_i, y_j) : \mathbb{R}^{m/\rho} \times \mathbb{R}^{m/\rho} \to \mathbb{R}$. As was pointed out in Section 1, the distribution of $d(y_i, y_j)$ for each specific choice of i and j may be more important when modeling rhythms (and music in general) than the actual choice of subsequences y_i.

Hidden Markov Models (HMM) (Rabiner, 1989) are commonly used to model temporal data. In principle, an HMM is able to capture complex regularities in patterns between subsequences of data, provided its number of hidden states is large enough. However, when dealing with music, such a model would lead to a learning process requiring a prohibitive amount of data: in order to learn long range interactions, the training set should be representative of the joint distribution of subsequences. To overcome this problem, we summarize the joint distribution of subsequences by the distribution of distances between these subsequences. This summary is clearly not a sufficient statistics for the distribution of subsequences, but its distribution can be learned from a limited number of examples. The resulting model, which generates distances, is then used to recover subsequences.

2.2. Decomposition of Distances

Let $D(\mathbf{x}^l) = (d^l_{i,j})_{\rho\times\rho}$ be the distance matrix associated with each sequence $\mathbf{x}^l$, where $d^l_{i,j} = d(y^l_i, y^l_j)$. Since $D(\mathbf{x}^l)$ is symmetric and contains only zeros on the diagonal, it is completely characterized by the upper triangular matrix of distances *without* the diagonal. Hence,

$$p(D(\mathbf{x}^l)) = \prod_{i=1}^{\rho-1} \prod_{j=i+1}^{\rho} p(d^l_{i,j} | S_{l,i,j}) \tag{1}$$

where

$$S_{l,i,j} = \{d^l_{r,s} | \quad (1 < s < j \text{ and } 1 \le r < s) \\ \text{or} \quad (s = j \text{ and } 1 \le r < i)\} \ . \tag{2}$$

In words, we order the elements column-wise and do a standard factorization, where each random variable depends on the previous elements in the ordering. Hence, we do not assume any conditional independence between the distances.

Since $d(y_i, y_j)$ is a metric, we have that $d(y_i, y_j) \le d(y_i, y_k) + d(y_k, y_j)$ for all $i, j, k \in \{1, \ldots, \rho\}$. This inequality is usually referred to as the *triangle inequality*. Defining

$$\begin{aligned} \alpha^l_{i,j} &= \min_{k\in\{1,\ldots,(i-1)\}} (d^l_{k,j} + d^l_{i,k}) \text{ and} \\ \beta^l_{i,j} &= \max_{k\in\{1,\ldots,(i-1)\}} (|d^l_{k,j} - d^l_{i,k}|) \ , \end{aligned} \tag{3}$$

we know that given previously observed (or sampled) distances, constraints imposed by the triangle inequality on $d^l_{i,j}$ are simply

$$\beta^l_{i,j} \le d^l_{i,j} \le \alpha^l_{i,j} \ . \tag{4}$$

One may observe that the boundaries given in Eq. (3) contain a subset of the distances that are on the conditioning side of each factor in Eq. (1) for each indexes i and j. Thus, constraints imposed by the triangle inequality can be taken into account when modeling each factor of $p(D(\mathbf{x}^l))$: each $d^l_{i,j}$ must lie in the interval imposed by previously observed/sampled distances given

in Eq. (4). Figure 1 shows an example where $\rho = 4$. Using Eq. (1), the distribution of $d^l_{2,4}$ would be conditioned on $d^l_{1,2}$, $d^l_{1,3}$, $d^l_{2,3}$, and $d^l_{1,4}$, and Eq. (4) reads $|d^l_{1,2} - d^l_{1,4}| \leq d^l_{2,4} \leq d^l_{1,2} + d^l_{1,4}$. Then, if subsequences y^l_1 and y^l_2 are close and y^l_1 and y^l_4 are also close, we know that y^l_2 and y^l_4 cannot be far. Conversely, if subsequences y^l_1 and y^l_2 are far and y^l_1 and y^l_4 are close, we know that y^l_2 and y^l_4 cannot be close.

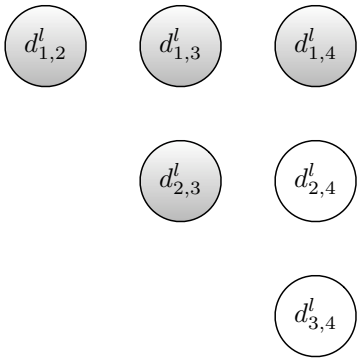


Figure 1. Each circle represents the random variable associated with the corresponding factor in Eq. (1), when $\rho = 4$. For instance, the conditional distribution for $d^l_{2,4}$ possibly depends on the variables associated to the grey circles.

2.3. Modeling Relative Distances Between Rhythms

We want to model rhythms in a music dataset $\mathcal{X}$ consisting of melodies of the same musical genre. We first quantize the database by segmenting each song in m time steps and associate each note to the nearest time step, such that all melodies have the same length m[1]. It is then possible to represent rhythms by sequences containing potentially three different symbols: 1) Note onset, 2) Note continuation, and 3) Silence. When using quantization, there is a one to one mapping between this representation and the set of all possible rhythms. Using this representation, symbol 2 can never follow symbol 3. Let $A = \{1, 2, 3\}$; in the remaining of this paper, we assume that $\mathbf{x}^l \in A^m$ for all $\mathbf{x}^l \in \mathcal{X}$.

When using this representation, $d^l_{i,j}$ can simply be chosen to be the Hamming distance (i.e. counting the number of positions on which corresponding symbols are different.) One could think of using more general edit distance such as the Levenshtein distance. However, this approach would not make sense psychoacoustically: doing an insertion or a deletion in a rhythm produces a translation that alters dramatically the nature of the sequence. Putting it another way, rhythm perception heavily depends on the *position* on which rhythmic events occur. In the remainder of this paper, $d^l_{i,j}$ is the Hamming distance between subsequences y_i and y_j.

We now have to encode our belief that melodies of the same musical genre have a common distance structure. For instance, drum beats in rock music can be very repetitive, except in the endings of every four measures, without regard to the actual beats being played. This should be accounted for in the distributions of the corresponding $d^l_{i,j}$. With Hamming distances, the conditional distributions of $d^l_{i,j}$ in Eq. (1) should be modeled by discrete distributions, whose range of possible values *must* obey Eq. (4). Hence, we assume that the random variables $(d^l_{i,j} - \beta^l_{i,j})/(\alpha^l_{i,j} - \beta^l_{i,j})$ should be identically distributed for $l = 1, \ldots, n$. As an example, suppose that measures 1 and 4 always tend to be far away, that measures 1 and 3 are close, and that measures 3 and 4 are close; Triangle inequality states that 1 and 4 should be close in this case, but the desired model would still favor a solution with the greatest distance possible *within* the constrains imposed by triangle inequalities.

All these requirements are fulfilled if we model $d_{i,j} - \beta_{i,j}$ by a binomial distribution of parameters $(\alpha_{i,j} - \beta_{i,j}, p_{i,j})$, where $p_{i,j}$ is the probability that two symbols of subsequences y_i and y_j differ. With this choice, the conditional probability of getting $d_{i,j} = \beta_{i,j} + \delta$ would be

$$B(\delta, \alpha_{i,j}, \beta_{i,j}, p_{i,j}) = \binom{\alpha_{i,j} - \beta_{i,j}}{\delta} (p_{i,j})^{\delta} (1 - p_{i,j})^{(\alpha_{i,j} - \beta_{i,j} - \delta)} \quad , \tag{5}$$

with $0 \leq p_{i,j} \leq 1$. If $p_{i,j}$ is close to zero/one, the relative distance between subsequences y_i and y_j is small/large. However, the binomial distribution is not flexible enough since there is no indication that the distribution of $d_{i,j} - \beta_{i,j}$ is unimodal. We thus model each $d_{i,j} - \beta_{i,j}$ with a binomial *mixture* distribution in order to allow multiple modes. We thus use

$$p(d_{i,j} = \beta_{i,j} + \delta | S_{i,j}) = \sum_{k=1}^{c} w^{(k)}_{i,j} B(\delta, \alpha_{i,j}, \beta_{i,j}, p^{(k)}_{i,j}) \tag{6}$$

with $w^{(k)}_{i,j} \geq 0$, $\sum_{k=1}^{c} w^{(k)}_{i,j} = 1$ for every indexes i and j, and $S_{i,j}$ defined similarly as in Eq. (2). Parameters

$$\theta_{i,j} = \{w^{(1)}_{i,j}, \ldots, w^{(c-1)}_{i,j}\} \cup \{p^{(1)}_{i,j}, \ldots, p^{(c)}_{i,j}\}$$

[1]This hypothesis is not fundamental in the proposed model and could easily be avoided if one would have to deal with more general datasets.

can be learned with the EM algorithm (Dempster et al., 1977) on rhythm data for a specific music style.

In words, we model the *difference* between the observed distance $d^l_{i,j}$ between two subsequences and the minimum possible value $\beta_{i,j}$ for such a difference by a binomial mixture.

The parameters $\theta_{i,j}$ can be initialized to arbitrary values before applying the EM algorithm. However, as the likelihood of mixture models is not a convex function, one may get better models and speed up the learning process by choosing sensible values for the initial parameters. In the experiments reported in Section 3, the k-means algorithm for clustering (Duda et al., 2000) was used. More precisely, k-means was used to partition the values $(d^l_{i,j} - \beta^l_{i,j})/(\alpha^l_{i,j} - \beta^l_{i,j})$ into c clusters corresponding to each component of the mixture in Eq. (6). Let $\{\mu^{(1)}_{i,j}, \ldots, \mu^{(c)}_{i,j}\}$ be the centroids and $\{n^{(1)}_{i,j}, \ldots, n^{(c)}_{i,j}\}$ the number of elements in each of these clusters. We initialize the parameters $\theta_{i,j}$ with

$$w^{(k)}_{i,j} = \frac{n^{(k)}_{i,j}}{n} \quad \text{and} \quad p^{(k)}_{i,j} = \mu^{(k)}_{i,j}.$$

We then follow a standard approach (Bilmes, 1997) to apply the EM algorithm to the binomial mixture in Eq. (6). Let $z^l_{i,j} \in \{1, \ldots, c\}$ be a hidden variable telling which component density generated $d^l_{i,j}$. For every iteration of the EM algorithm, we first compute

$$p(z^l_{i,j} = k | d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, \hat{\theta}_{i,j}) = \frac{\psi_{k,i,j,l}}{\sum_{t=1}^{c} \psi_{t,i,j,l}}$$

where $\hat{\theta}_{i,j}$ are the parameters estimated in the previous iteration, or the parameters guessed with k-means on the first iteration of EM, and

$$\psi_{k,i,j,l} = \hat{w}^{(k)}_{i,j} B(d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, p^{(k)}) \ .$$

Then, the parameters can be updated with

$$p^{(k)}_{i,j} = \frac{\sum_{l=1}^{n} (d^l_{i,j} - \beta^l_{i,j}) p(z^l_{i,j} = k | d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, \hat{\theta}_{i,j})}{\sum_{l=1}^{n} (\alpha^l_{i,j} - \beta^l_{i,j}) p(z^l_{i,j} = k | d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, \hat{\theta}_{i,j})}$$

and

$$w^{(k)}_{i,j} = \frac{1}{n} \sum_{l=1}^{n} p(z^l_{i,j} = k | d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, \hat{\theta}_{i,j}).$$

This process is repeated until convergence.

Note that using mixture models for discrete data is known to lead to *identifiability* problems. Identifiability refers here to the uniqueness of the representation (up to an irrelevant permutation of parameters) of any distribution that can be modeled by a mixture. Estimation procedures may not be well-defined and asymptotic theory may not hold if a model is not identifiable. However, the model defined in Eq. (6) is identifiable if $\alpha_{i,j} - \beta_{i,j} > 2c - 1$ (Titterington et al., 1985, p.40). While this is the case for most $d_{i,j}$, we observed that this condition is sometimes violated. Whatever happens, there is no impact on the estimation because we only care about what happens at the distribution level: there may be several parameters leading to the same distribution, some components may vanish in the fitting process, but this is easily remedied, and EM behaves well.

As stated in Section 1, musical patterns form hierarchical structures closely related to meter (Handel, 1993). Thus, the distribution of $p(D(\mathbf{x}^l))$ can be computed for many numbers of partitions within each rhythmic sequence. Let $\mathcal{P} = \{\rho_1, \ldots \rho_h\}$ be a set of numbers of partitions to be considered by our model, where h is the number of such numbers of partitions. The choice of $\mathcal{P}$ depends on the domain of application. Following meter, $\mathcal{P}$ may have dyadic[2] tree-like structure when modeling music (e.g. $\mathcal{P} = \{2, 4, 8, 16\}$). Let $D_{\rho_r}(\mathbf{x}^l)$ be the distance matrix associated with sequence $\mathbf{x}^l$ divided into ρ_r parts. Estimating the joint probability $\prod_{r=1}^{h} p(D_{\rho_r}(\mathbf{x}^l))$ with the EM algorithm as described in this section leads to a model of the distance structures in music datasets. Suppose we consider 16 bars songs with four beats per bar. Using $\mathcal{P} = \{8, 16\}$ would mean that we consider pairs of distances between every group of two measures ($\rho = 8$), and every single measures ($\rho = 16$).

One may argue that our proposed model for long-term dependencies is rather unorthodox. However, simpler models like Poisson or Bernoulli process (we are working in discrete time) defined over the whole sequence would not be flexible enough to represent the particular long-term structures in music.

2.4. Conditional Prediction

For most music applications, it would be particularly helpful to know which sequence $\hat{x}_s, \ldots, \hat{x}_m$ maximizes $p(\hat{x}_s, \ldots, \hat{x}_m | x_1, \ldots, x_{s-1})$. Knowing which musical events are the most likely given the past $s-1$ observations would be useful both for prediction and generation. Note that in the remaining of the paper, we refer to prediction of musical events given past observations only for notational simplicity. The distance model presented in this paper could be used to predict

[2]Even when considering non-dyadic measures (e.g. a three-beat waltz), the *very large* majority of the hierarchical levels in metric structures follow dyadic patterns (Handel, 1993) in most tonal music.

any part of a music sequence given any other part with only minor modifications.

While the described modeling approach captures long range interactions in the music signal, it has two shortcomings. First, it does not model local dependencies: it does not predict how the distances in the smallest subsequences (i.e. with length smaller than $m/\max(\mathcal{P})$) are distributed on the events contained in these subsequences. Second, as the mapping from sequences to distances is many to one, there exists several admissible sequences $\mathbf{x}^l$ for a given set of distances. These limitations are addressed by using another sequence learner designed to capture short-term dependencies between musical events. Here, we use a standard Hidden Markov Model (HMM) (Rabiner, 1989) displayed in Figure 2, following standard graphical model formalism. Each node is associated to a random variable and arrows denote conditional dependencies. Learning the parameters of the HMM can be done as usual with the EM algorithm.

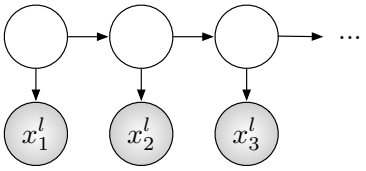


Figure 2. Hidden Markov Model. Each node is associated to a random variable and arrows denote conditional dependencies. During training of the model, white nodes are hidden while grey nodes are observed.

The two models are trained separately using their respective version of the EM algorithm. For predicting the continuation of new sequences, they are combined by choosing the sequence that is most likely according to the local HMM model, provided it is also plausible regarding the model of long-term dependencies. Let $p_{\text{HMM}}(\mathbf{x}^l)$ be the probability of observing sequence $\mathbf{x}^l$ estimated by the HMM after training. The final predicted sequence is the solution of the following optimization problem:

$$\begin{cases} \max\limits_{\tilde{x}_s,\ldots,\tilde{x}_m} & p_{\text{HMM}}(\tilde{x}_s,\ldots,\tilde{x}_m|x_1,\ldots,x_{s-1}) \\ \text{subject to} & \prod\limits_{r=1}^{h} p(D_{\rho_r}(\mathbf{x}^l)) \geq P_0 \ , \end{cases} \tag{7}$$

where P_0 is a threshold. In practice, one solves a Lagrangian formulation of problem (7), where we use log-probabilities for obvious computational reasons:

$$\begin{aligned} \max_{\tilde{x}_s,\ldots,\tilde{x}_m} [&\log p_{\text{HMM}}(\tilde{x}_s,\ldots,\tilde{x}_m|x_1,\ldots,x_{s-1}) \\ &+\lambda \textstyle\sum_{r=1}^{h} \log p(D_{\rho_r}(\mathbf{x}^l))] \ , \end{aligned} \tag{8}$$

where tuning λ has the same effect as choosing a threshold P_0 in Eq. (7) and can be done by cross-validation.

Multidimensional Scaling (MDS) is an algorithm that tries to embed points (here "local" subsequences) into a potentially lower dimensional space while trying to be faithful to the pairwise affinities given by a "global" distance matrix. Here, we propose to consider the prediction problem as finding sequences that maximize the likelihood of a "local" model of subsequences under the constraints imposed by a "global" generative model of distances between subsequences. In other words, solving problem (7) is similar to finding points between which distances are as close as possible to a given set of distances (i.e. minimizing a stress function in MDS). Naively trying all possible subsequences to maximize (8) leads to $O(|A|^{(m-s+1)})$ computations. Instead, we propose to search the space of sequences using a variant of the Greedy Max Cut (GMC) method (Rohde, 2002) that has proven to be optimal in terms of running time and performance for binary MDS optimization.

The subsequence $\hat{x}_s,\ldots,\hat{x}_m$ can be simply initialized with

$$(\hat{x}_s,\ldots,\hat{x}_m) = \max_{\tilde{x}_s,\ldots,\tilde{x}_m} p_{\text{HMM}}(\tilde{x}_s,\ldots,\tilde{x}_m|x_1,\ldots,x_{s-1}) \tag{9}$$

using the local HMM model. Then, Algorithm 1 carries on complete optimization.

Algorithm 1 Simple optimization algorithm to maximize $p(\hat{x}_i,\ldots,\hat{x}_m|x_1,\ldots,x_{i-1})$

```
Initialize x̂_s,...,x̂_m using Eq. (9)
Initialize end = false
while end = false do
  Set end = true
  for j = s to m do
    Set x̂_j = arg max_{a∈A} [log p_HMM(x*|x_1,...,x_{s-1}) +
                λ Σ_{r=1}^{h} log p(D_{ρ_r}(x_1,...,x_{s-1},x*))]
    where x* = (x̂_s,...,x̂_{j-1}, a, x̂_{j+1},...,x̂_m)
    if x̂_j has been modified in the last step then
      Set end = false
    end if
  end for
end while
Output x̂_s,...,x̂_m.
```

For each position, we try every admissible symbol of the alphabet and test if a change increases the probability of the sequence. We stop when no further change can increase the value of the utility function. Obviously, many other methods could have been used to search the space of possible sequences $\hat{x}_s, \ldots, \hat{x}_m$, such as simulated annealing (Kirkpatrick et al., 1983). We chose Algorithm 1 for its simplicity and the fact that it yields excellent results, as reported in the following section.

3. Experiments

Two rhythm databases from different musical genres were used to evaluate the proposed model. Firstly, 47 jazz standards melodies (Sher, 1988) were interpreted and recorded by the first author in MIDI format. Appropriate rhythmic representations as described in Section 2.3 have been extracted from these files. The complexity of the rhythm sequences found in this corpus is representative of the complexity of common jazz and pop music. We used the last 16 bars of each song to train the models, with four beats per bar. Two rhythmic observations were made for each beat, yielding observed sequences of length 128. We also used a subset of the Nottingham database [3] consisting of 53 traditional British folk dance tunes called "hornpipes". In this case, we used the first 16 bars of each song to train the models, with four beats per bar. Three rhythmic observations were made for each beat, yielding observed sequences of length 192. The sequences from this second database contain no silence (i.e. rests), leading to sequences with binary states.

The goal of the proposed model is to predict or generate rhythms *given* previously observed rhythm patterns. As pointed out in Section 1, such a model could be particularly useful for music information retrieval, transcription, or music generation applications. Let $\varepsilon_i^t = 1$ if $\hat{x}_i^t = x_i^t$, and 0 otherwise, with $\mathbf{x}^t = (x_1^t, \ldots, x_m^t)$ a test sequence, and $\hat{x}_i^t$ the output of the evaluated prediction model on the i-th position when given $(x_1^t, \ldots, x_s^t)$ with $s < i$. Assume that the dataset is divided into K folds $T_1, \ldots, T_K$ (each containing different sequences), and that the k-th fold T_k contains n_k test sequences. When using cross-validation, the accuracy *Acc* of an evaluated model is given by

$$Acc = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{t \in T_k} \frac{1}{m-s} \sum_{i=s+1}^{m} \varepsilon_i^t \ . \qquad (10)$$

Note that, while the prediction accuracy is simple to estimate and to interpret, other performance criteria, such as ratings provided by a panel of experts, should be more appropriate to evaluate the relevance of music models. We plan to define such an evaluation protocol in future work. We used 5-fold double cross-validation to estimate the accuracies. Double cross-validation is a recursive application of cross-validation that enables to jointly optimize the hyper-parameters of the model and evaluate its generalization performance. Standard cross-validation is applied to each subset of $K-1$ folds with each hyper-parameter setting and tested with the best estimated setting on the remaining hold-out fold. The reported accuracies are the averages of the results of each of the K applications of simple cross-validation during this process.

For the baseline HMM model, double cross-validation optimizes the number of possible states for the hidden variables. 2 to 20 possible states were tried in the reported experiments. In the case of the model with distance constraints, referred to as the global model, the hyper-parameters that were optimized are the number of possible states for hidden variables in the local HMM model (i.e. 2 to 20), the Lagrange multiplier λ, the number of components c (common to all distances) for each binomial mixture, and the choice of $\mathcal{P}$, i.e. which partitions of the sequences to consider. Values of λ ranging between 0.1 and 4 and values of c ranging between 2 and 5 were tried during double cross-validation. Since music data commonly shows strong dyadic structure following meter, many subsets of $\mathcal{P} = \{2, 4, 8, 16\}$ were allowed during double cross-validation.

Note that the baseline HMM model is a poor benchmark on this task, since the predicted sequence, when prediction consists in choosing the most probable subsequence given previous observations, only depends on the state of the hidden variable in position s, where s is the index of the last observation. This observation implies that the number of possible states for the hidden variables of the HMM upper-bounds the number of different sequences that the HMM can predict. However, this behavior of the HMM does not harm the validity of the reported experiments. The main goal of this quantitative study is to measure to what extent distance patterns are present in music data and how well these dependencies can be captured by the proposed model. What we really want to measure is how much gain we observe in terms of out-of-sample prediction accuracy when using an arbitrary model if we impose additional constraints based on distance patterns. That being said, it would be interesting to measure the effect of appending distance constraints to more complex music prediction models (Pachet, 2003; Dubnov

[3] `http://www.cs.nott.ac.uk/~ef/music/database.htm`.

Table 1. Accuracy (the higher the better) for best models on the jazz standards database.

Observed	Predicted	HMM	Global
32	96	34.5%	54.6%
64	64	34.5%	55.6%
96	32	41.6%	47.2%

Table 2. Accuracy (the higher the better) for best models on the hornpipes database.

Observed	Predicted	HMM	Global
48	144	75.1%	83.0%
96	96	75.6%	82.1%
144	48	76.6%	80.1%

et al., 2003) in future work.

Results in Table 1 for the jazz standards database show that considering distance patterns significantly improves the HMM model. One can observe that the baseline HMM model performs much better when trying to predict the last 32 symbols. This is due to the fact that this database contains song endings. Such endings contain many silences and, in terms of accuracy, a useless model predicting silence at any position performs already well. On the other hand, the endings are generally different from the rest of the rhythm structures, thus harming the performance of the global model when just trying to predict the last 32 symbols.

Results in Table 2 for the hornpipes database again show that the prediction accuracy of the global model is consistently better than the prediction accuracy of the HMM, but the difference is less marked. This is mainly due to the fact that this dataset only contains two symbols, associated to note onset and note continuation. Moreover, the frequency of these symbols is quite unbalanced, making the HMM model much more accurate when almost always predicting the most common symbol.

In Table 3, the set of partitions $\mathcal{P}$ is not optimized by double cross-validation. Results are shown for different fixed sets of partitions. The best results are reached with "deeper" dyadic structure. This is a good indication that the basic hypothesis underlying the proposed model is well-suited to music data, namely that dyadic distance patterns exhibit strong regularities in music data. We did not compute accuracies for $\rho > 16$ because it makes no sense to estimate distribution of distances between too short subsequences.

Table 3. Accuracy over the last 64 positions for many sets of partitions $\mathcal{P}$ on the jazz database, given the first 64 observations. The higher the better.

$\mathcal{P}$	Global
$\{2\}$	49.3%
$\{2, 4\}$	49.3%
$\{2, 4, 8\}$	51.4%
$\{2, 4, 8, 16\}$	55.6%

4. Conclusion

The main contribution of this paper is the design and evaluation of a generative model for distance patterns in temporal data. The model is specifically well-suited to music data, which exhibits strong regularities in dyadic distance patterns between subsequences. Reported conditional prediction accuracies show that such regularities are present in music data and can be effectively captured by the proposed model. Moreover, learning distributions of distances between subsequences really helps for accurate rhythm prediction. Rhythm prediction can be seen as the first step towards full melodic prediction and generation. A promising approach would be to apply the proposed model to melody prediction. It could also be readily used to increase the performance of transcription algorithms, genre classifiers, or even automatic composition systems.

The choice of the HMM to initialize the model is not optimal. However, this has no impact on the validity of the reported results, since our goal was to show the importance of distance patterns between subsequences in rhythm data. In order to sample to models to generate subjectively good results (Pachet, 2003; Dubnov et al., 2003), one could use other benchmark and initialization techniques, such as repetition of common patterns.

Finally, besides being fundamental in music, modeling distance between subsequences should also be useful in other application domains, such as in natural language processing. Being able to characterize and constrain the relative distances between various parts of a sequence of bags-of-concepts could be an efficient means to improve performance of automatic systems such as machine translation (Och & Ney, 2004). On a more general level, learning constraints related to distances between subsequences can boost the performance of "short memory" models such as the HMM.

Acknowledgments

This work was supported in part by the IST Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, funded in part by the Swiss Federal Office for Education and Science (OFES) and the Swiss NSF through the NCCR on IM2.

References

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*, 157–166.

Bilmes, J. (1997). A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models.

Cemgil, A. T., & Kappen, H. J. (2002). Rhythm quantization and tempo tracking by sequential Monte Carlo. *Advances in Neural Information Processing Systems 14* (pp. 1361–1368).

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, *39*, 1–38.

Dixon, S. (2007). Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, *36*, 39–50.

Dubnov, S., Assayag, G., Lartillot, O., & Bejerano, G. (2003). Using machine-learning methods for musical style modeling. *IEEE Computer*, *10*.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification, second edition*. Wiley Interscience.

Eck, D., & Schmidhuber, J. (2002). Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. *Neural Networks for Signal Processing XII, Proc. 2002 IEEE Workshop* (pp. 747–756). New York: IEEE.

Handel, S. (1993). *Listening: An introduction to the perception of auditory events*. Cambridge, Mass.: MIT Press.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, *220, 4598*, 671–680.

Och, F. J., & Ney, H. (2004). The alignment template approach to statistical machine translation. *Computational Linguistics*, *30*, 417–449.

Pachet, F. (2003). The continuator: Musical interaction with style. *Journal of New Music Research*, *32*, 333–341.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–285.

Rohde, D. L. T. (2002). Methods for binary multidimensional scaling. *Neural Comput.*, *14*, 1195–1232.

Scheirer, E. (1998). Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, *103*, 588–601.

Sher, C. (Ed.). (1988). *The New Real Book*, vol. 1-3. Sher Music Co.

Titterington, D. M., Smith, A. F. M., & Makov, U. E. (1985). *Statistical analysis of finite mixture distributions*. Wiley.

Whiteley, N., Cemgil, A. T., & Godsill, S. J. (2007). Sequential inference of rhythmic structure in musical audio. *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP 07)* (pp. 1321–1324).

On the Chance Accuracies of Large Collections of Classifiers

Mark Palatucci MARKMP@CMU.EDU
Andrew Carlson ACARLSON@CS.CMU.EDU
Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

We provide a theoretical analysis of the chance accuracies of large collections of classifiers. We show that on problems with small numbers of examples, some classifier can perform well by random chance, and we derive a theorem to explicitly calculate this accuracy. We use this theorem to provide a principled feature selection criterion for sparse, high-dimensional problems. We evaluate this method on microarray and fMRI datasets and show that it performs very close to the optimal accuracy obtained from an oracle. We also show that on the fMRI dataset this technique chooses relevant features successfully while another state-of-the-art method, the False Discovery Rate (FDR), completely fails at standard significance levels.

1. Introduction

There are many real world problems in which a large number of *experts* predict the outcome of a small number of events. For example, we may ask one hundred football fans to predict the outcome of twenty games, or we may ask fifty political pundits to predict the outcome of ten elections.

With only a small number of events to predict, there may be a reasonable chance that some expert may predict all the outcomes perfectly, even if the outcomes are chosen at random.

For example, suppose we ask a person to predict the outcome of five coin flips where the probability of obtaining heads is 0.5. Since the flips are independent, this person has a $(0.5)^5 = \frac{1}{32}$ chance of guessing the outcome of all flips correctly. Now, if we ask ten people to predict the outcome of the five flips, there is a much higher chance that *someone* will predict all outcomes perfectly. With thirty-two people, someone would (in expectation) guess correctly each time.

Suppose we repeated this experiment again but asked our participants to predict the outcome of thirty coin flips. In this case, the chance of obtaining a perfect prediction would be nearly 1 in 1 billion. Given any number of participants less than 1 billion, we would not expect any participant to perfectly predict all the outcomes. But some participant will predict a series of outcomes that is most similar to the true flips.

How accurate should we expect this participant's predictions to be?

We consider this question and its relevance to machine learning. In our setting, we consider *experts* that are not people, but rather classification algorithms that predict labels for a set of examples.

When a large number of classifiers predict labels for a small number of examples, some classifiers will predict the labels well purely by random chance. This may lead us to believe that a subset of the classifiers are actually good predictors, when in fact they may be just guessing randomly.

This effect is commonly seen in discriminative feature selection, where a feature is selected based on the accuracy of a classifier trained on that single feature and tested on a held-out set of validation examples. In modern high-dimensional machine learning applications such as fMRI or microarray analysis, there are typically thousands of features with less than one hundred examples. Classification tasks in such settings often have *sparse* solutions, meaning that only a small subset of the features are useful for predicting the correct class.

To determine which features are relevant, it would be useful to know how well *some* classifier could perform if all classifiers just chose labels at random. We would like to know how this accuracy changes with both the number of features and number of examples. This pa-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

per poses and answers the following question:

Given M classifiers that each produce labels randomly for N examples, what is the highest accuracy that we would expect some classifier to achieve?

1.1. Related Work

Our work is closely related to the multiple-testing problem in the statistics community. In statistics, hypothesis tests are the standard way to test if some assertion is true with high probability. While a single test has a low probability of making an error, when multiple hypothesis tests are performed simultaneously, the probability of at least one of the tests making an error can be much higher. It is common to *correct* the tests by making them more conservative to compensate.

Two of the most popular methods for correcting multiple tests are the Bonferroni Correction and the False Discovery Rate (Benjamini & Hochberg, 1995). We can apply these methods to the problem of feature selection, but in practice they are often too conservative at standard significance levels (e.g. 5%). See Wong et al. (2002) and Frank and Witten (1998). With many high-dimensional classification problems they may simply state that no feature is significant. This is not particularly helpful when building a classifier.

We could lower the significance level so more features are considered relevant, but it is unclear what significance level to choose. Since different learning algorithms have different tolerances to noisy, irrelevant features, there is no single significance level that is appropriate for all learning algorithms.

This fact, along with the large number of available tests and correction methods, makes hypothesis testing a difficult task for non-experts.

In our work, we approach the problem of significance from a different angle. Using *order statistics*, we explicitly model small chance events in a group setting. These techniques are relatively unknown in the machine learning literature although the *multiple comparison procedures* described in Jensen (2000) are similar in spirit.

We feel an order statistic approach is much more intuitive than hypothesis testing, and is well suited to problems in machine learning.

One such problem is discriminative feature selection. This feature selection technique is often called a *wrapper* method in contrast to more recent *embedded* methods like the $\mathcal{L}_1$ regularized Lasso (Tibshirani, 1996). While a full comparison of wrapper and embedded methods is beyond the scope of this paper, we believe that wrapped methods will continue to play a role in machine learning due to their simplicity and tractability. An excellent overview of the feature selection literature is available in Guyon (2003).

The work most similar to ours is by Li and Grosse (2003), which uses extreme value distribution theory to choose a significance threshold for selecting relevant features. While the general theme is similar, we do not use asymptotic results of extreme value theory, nor do we use simulation to compute moments of order statistics. By contrast, we focus on classification problems and show exact solutions that do not require any simulation.

2. Preliminaries

2.1. Order Statistics

We use *order statistics* extensively in this paper, thus we begin with a small introduction to define some basic concepts and notation. Consider M samples (i.i.d.) drawn from some distribution: $X_1, \ldots, X_M \sim F_X(x)$. If we order these samples from smallest to largest we obtain:

$$X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(M)}$$

and we use the notation $X_{(r)}$ to denote the r^{th} smallest sample which we call the r^{th} order statistic. $X_{(1)}$ and $X_{(M)}$ have special meaning which we call the *extreme values*:

$$\begin{aligned} X_{(1)} &= \min(X_1, X_2, \ldots, X_M) \\ X_{(M)} &= \max(X_1, X_2, \ldots, X_M) \end{aligned}$$

Each order statistic $X_{(r)}$ is also a random variable and can be described by a cumulative distribution function $F_{(r)}(x)$ or a density function $f_{(r)}(x)$. We will refer to an order statistic's *parent distribution*, which is the original distribution from which the M unordered samples were drawn. In our example this is $F_X(x)$.

We will use the notation $\mu_{r:M}$ to denote the mean of the r^{th} order statistic for M samples drawn from the parent distribution.

3. Expected Chance Accuracies

Using order statistics we can now answer the question we posed earlier:

Given M classifiers that each produce labels randomly for N examples, what is the highest accuracy that we would expect some classifier to achieve?

To answer this question, first consider a classifier that labels some collection of examples at random. If the classifier labels an example incorrectly with probability p_{err}, we can model the number of errors the classifier makes as a binomial random variable. Formally, let X be defined as the number of errors the classifier makes on some true labeling of N examples. Then:

$$X \sim \text{Binomial}(N, p_{err})$$

and the mean and variance of X are:

$$\begin{aligned}\mathbb{E}[X] &= N \cdot p_{err} \\ \mathbb{V}[X] &= N \cdot p_{err} \cdot (1 - p_{err})\end{aligned}$$

Now, suppose instead we have M independent classifiers where each produces a set of N labels at random. Once again, the probability that each classifier makes a mistake on a single example is p_{err}. Let X_i be the number of errors made by the ith classifier. We then have:

$$X_1, X_2, \ldots, X_M \sim \text{Binomial}(N, p_{err})$$

One of these classifiers will have the minimal number of errors. Using our order statistic notation we have:

$$X_{(1)} = \min(X_1, X_2, \ldots, X_M)$$

and the expected minimum number of errors is:

$$\mu_{1:M} = \mathbb{E}[X_{(1)}]$$

If we knew the density function of $X_{(1)}$ for M samples from a Binomial(N, p_{err}) we could compute the mean $\mu_{1:M}$ directly:

$$\mu_{1:M} = \sum_{x=0}^{\infty} x f_{(1)}(x)$$

If the parent distribution were a continuous variable, obtaining $f_{(1)}$ would not be difficult and many references show simple methods to compute the density for any order statistic of a continuous distribution (Casella & Berger, 2002). Since our parent distribution is the discrete binomial, computing $f_{(1)}$ and more importantly $\mu_{1:M}$ is more difficult.

We could resort to simulation to find the mean, but this can be quite time consuming for large collections of variables. We will show later, however, that an exact solution does exist.

3.1. The Multiplicity Gap

For any problem with M classifiers and N examples there is a risk that some classifier will perform well by random chance. *What is a good measure of this risk?*

As we showed earlier, $\mathbb{E}[X_{(1)}]$ is the minimum number of errors that we should expect *some* classifier to make. We also know that $\mathbb{E}[X]$ is the expected number of errors an *individual* classifier will make.

Thus, one natural measure of this risk is the difference between these two values. We define the *multiplicity gap* $\mathcal{G}_{M,N}$ for M classifiers and N examples as:

$$\mathcal{G}_{M,N} = \mathbb{E}[X] - \mathbb{E}[X_{(1)}]$$

Reducing the number of examples N or increasing the number of classifiers M *increases* the risk.

4. Derivation

Theorem 4.1. *Highest Chance Accuracy*
Consider a classification problem with M classifiers and N examples. If the probability that a classifier makes a mistake on a single example is p_{err}, the highest expected accuracy $\mathcal{A}_H$ of any classifier is given by:

$$\mathbb{E}[\mathcal{A}_H] = 1 - \frac{1}{N} \sum_{i=0}^{N-1} I_{p_{err}}(i+1, N-i)^M \quad (1)$$

where $I_p(a,b)$ is the incomplete beta function[1]*:*

$$I_p(a,b) = \frac{1}{\beta(a,b)} \int_0^p t^{a-1}(1-t)^{b-1} dt$$

Proof. Let X_i, $(1 \le i \le M)$ be the total number of errors classifier i makes on some true labeling of N examples. If the probability that a classifier makes a mistake on a single example is p_{err}, then:

$$X_1, X_2, \ldots, X_M \sim \text{Binomial}(N, p_{err})$$

Therefore, the expected minimum number of errors is:

$$\mu_{1:M} = \mathbb{E}[X_{(1)}]$$

To compute the value of $\mu_{1:M}$ we utilize a useful result from Feller (1957) that relates the mean of a discrete random variable to its distribution function:

$$\mu_X = \sum_{i=0}^{\infty} [1 - F_X(i)]$$

therefore

$$\mu_{1:M} = \sum_{i=0}^{\infty} [1 - F_{(1)}(i)]$$

[1] Some texts refer to this form as the regularized incomplete beta function.

A result from David and Nagaraja (2003) shows that is equivalent to:

$$\mu_{1:M} = \sum_{i=0}^{\infty} [1 - F_X(i)]^M \tag{2}$$

Now, for a Binomial(N, p_{err}), $F_X(i) = 1$ when $i \geq N$. Therefore, the upper limit of the sum becomes $N-1$:

$$\mu_{1:M} = \sum_{i=0}^{N-1} [1 - F_X(i)]^M$$

Note that the incomplete beta function $I_p(a,b)$ has an expansion that looks similar to the distribution function of a binomial:

$$I_p(a,b) = \sum_{j=a}^{a+b-1} \frac{(a+b-1)!}{j!(a+b-1-j)!} p^j (1-p)^{a+b-1-j}$$

Using this expansion and a few algebraic manipulations we can express the tail of the distribution function in terms of the incomplete beta function[2]:

$$\begin{aligned} 1 - F_X(i) &= \mathbb{P}(X \geq i+1) \\ &= \sum_{j=i+1}^{N} \frac{N!}{j!(N-j)!} (p_{err})^j (1-p_{err})^{N-j} \\ &= I_{p_{err}}(i+1, N-i) \end{aligned}$$

Substituting this form into (2) we have:

$$\mu_{1:M} = \sum_{i=0}^{N-1} I_{p_{err}}(i+1, N-i)^M$$

To put our answer in terms of accuracy rather than errors we rearrange:

$$\begin{aligned} \frac{1}{N}(N - \mu_{1:M}) &= 1 - \frac{1}{N}\mu_{1:M} \\ &= 1 - \frac{1}{N} \sum_{i=0}^{N-1} I_{p_{err}}(i+1, N-i)^M \end{aligned}$$

Note that this theorem depends on the number of classes only through p_{err}. It does not require any modification to adapt to many classes. □

[2]We feel it is numerically advantageous to use the incomplete beta function rather than computing the binomial CDF directly. Many numerical computing environments have fast implementations of the incomplete beta function $I_p(a,b)$. For example, the `betainc(p,a,b)` command in MATLAB can implement Equation 1 in one line of code.

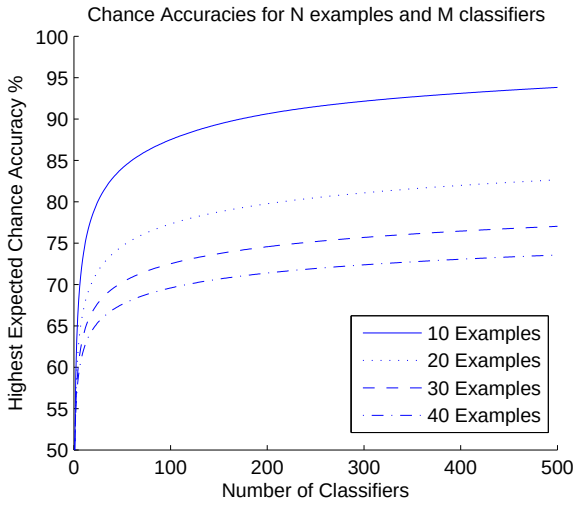


Figure 1. The highest expected chance accuracy as a function of the number of examples and classifiers. Each line represents a different number of examples. The x-axis is the number of classifiers and the y-axis is the accuracy.

Example 4.1 Predicting NFL games
Consider an office football pool with 200 participants betting on the outcome of 20 games. If each participant selects the outcome of a game according to a fair coin flip, how well would we expect the "winner" to perform?

To answer this question, we apply Equation 1 where $M = 200$, $N = 20$, and $p_{err} = 0.5$. In this case, the highest expected accuracy of some participant is 80%.

Although the chance probability of obtaining a *perfect* labeling is extremely small, in this case only $1/2^{20} = 1/1,048,576$, the chance of obtaining a *very good* labeling is much higher. Exactly 1,048,576 participants would be needed for us to expect one to obtain a perfect labeling. Yet, with only 200 participants, the expected accuracy of the top performer is 80%.

This effect can be seen by plotting Equation 1 for a two class problem where $p_{err} = 0.5$ (see Figure 1). The graph shows the highest expected chance accuracy (y-axis) for a given number of classifiers (x-axis). Each line represents a different number of examples N. As we increase the number of examples, the *multiplicity gap* closes, and highest expected chance accuracy for some classifier approaches the expected chance accuracy for a single classifier.

With small numbers of examples and large numbers of classifiers, the chance of obtaining a very good labeling may be very high, even if the chance of obtaining a perfect labeling is very low.

5. Case Study: Discriminative Feature Selection in Sparse, High-Dimensional Problems

A simple and popular method for finding relevant features in a classification task is discriminative feature selection. This method evaluates how well individual features discriminate between different classes and selects features with high predictive accuracy.

For example, if we have M features in a classification task, we train M distinct classifiers, where each classifier is trained using a single feature. After training, we evaluate all the classifiers on a set of *validation examples* and select the top performing features according to some criterion. A final classifier is then trained using only these top performing features, and then evaluated on some set of test examples.

This method is popular because it is simple to implement and often performs well in practice. The main difficulty is: *What are appropriate criteria for selecting significant features?*

One approach is to run a *cross-validation* loop, testing different significance thresholds to find one that has high empirical performance. This loop is computationally expensive and also requires additional validation examples. To avoid these difficulties in practice, it is common to choose some arbitrary threshold, and hope that performance is sufficient for the classification task.

Besides being pedantically unsatisfying, choosing an arbitrary threshold in a high-dimensional problem with a small number of examples is very risky. For example, a simple threshold might choose all features that perform better than 80% accuracy. As we showed earlier, many features may exceed this seemingly high threshold purely by random chance.

In high-dimensional problems with small numbers of examples, the accuracy required for statistical significance is often much higher than intuition might suggest.

A more principled approach for determining significance is to use a hypothesis test. With a hypothesis test, one tries to disprove a certain assertion. For example, one might assume that a classifier performs with a true accuracy of 50%. This assumption is called the *null hypothesis*. The goal then is to reject the null hypothesis if the evidence (e.g. the discriminative accuracy) is sufficiently strong.

Hypothesis testing has a vast literature in the statistics community. A good introduction can be found in Wasserman (2005). The Wald, "t", binomial, permutation, and χ^2 tests are just a few of the possible testing methods available. It is difficult, however, for a non-expert to know when to apply a particular test. To complicate matters, adjustments must be made when multiple tests are considered simultaneously. This is known in the statistics community as the *multiple testing problem*. Several methods such as the Bonferroni correction, family-wise error rate, and the false discovery rate (FDR) are used to compensate for multiple tests (Benjamini & Hochberg, 1995).

For the problem of discriminative feature selection, the use of a binomial test along with a false discovery rate adjustment is an appropriate choice. As we mentioned earlier, however, hypothesis tests require the choice of a significance level α. As is common in the scientific literature, the level $\alpha = 0.05$ is typically considered statistically significant.

For the purpose of feature selection, however, an appropriate choice of α is highly dependent on the classification algorithm used. Some classifiers are more tolerant to irrelevant features than others. Thus, there is no single α value appropriate for all classifiers. We could use a cross-validation loop to search for an appropriate α, but then we could have avoided the hypothesis test altogether and searched empirically for an appropriate threshold.

5.1. The Multiplicity Gap Midpoint (MGM) Method

Earlier in Equation 1 we derived the highest expected chance accuracy of some classifier assuming all classifiers choose their labels according to random chance. In some sense, this accuracy is a *natural significance threshold*, since we would not expect any classifer to perform better than this threshold by random chance.

While this may seem like an intuitive threshold for feature selection, in practice the threshold is overly conservative for several reasons. First, this threshold assumes all features are independent. This rarely holds in practice, and in many high-dimensional datasets it is very common to see strong correlations between features.

Further, the threshold assumes that all features are irrelevant and produce labels at random. In practice, some subset of the features will actually be significant, thereby lowering the effective number of random features. There is also no guarantee that errors for a feature can be modeled as a binomial random variable.

These violations of independence and irrelevance effectively lower the highest expected chance accuracy

(and increase the expected minimum number of errors). While this threshold may be overly conservative, it effectively serves as an upper bound on the highest expected chance accuracy.

At the other extreme, we might consider any feature significant that performs better than the expected chance accuracy of a *single* feature. As we showed before, this will clearly allow many irrelevant features to be considered significant. Note that these two extremes are the endpoints of the multiplicity gap that we defined earlier. If we model the number of errors made by a classifier as a binomial random variable, and we have M classifiers and N examples then the multiplicity gap $\mathcal{G}_{M,N} = \mathbb{E}[X] - \mathbb{E}[X_{(1)}]$

We conjecture that the optimal threshold should fall within the multiplicity gap.

In practice, we can choose any threshold between these two extremes. If we believe that our classifier is sensitive to irrelevant features, we should choose a threshold closer to $\mathbb{E}[X_{(1)}]$. Similarly, if our classifier is robust to irrelevant features, we should choose a threshold closer to $\mathbb{E}[X]$.

Without any knowledge of the particular classifier it is impossible to know what the optimal threshold should be. Therefore, as a simple heuristic we propose the *multiplicity gap midpoint* method, which chooses the midpoint of the extremes of the multiplicity gap. This yields a threshold τ_{MGM} on the maximum number of errors a classifier could make and still be considered significant:

$$\tau_{MGM} = \frac{\left(\mathbb{E}[X] + \mathbb{E}[X_{(1)}]\right)}{2}$$

where $\mathbb{E}[X_{(1)}]$ is computed as in Equation 1:

$$\mathbb{E}[X_{(1)}] \quad = \quad \mu_{1:M} = \sum_{i=0}^{N-1} I_{p_{err}}(i+1, N-i)^M$$

and $\mathbb{E}[X]$ is the number of examples N multiplied by the probability p_{err} that a classifier makes an error on a particular example: $\mathbb{E}[X] = N \cdot p_{err}$

To use this threshold, we perform a discriminative feature selection and select all features that make less than τ_{MGM} errors on a validation set with N examples.

5.2. Experimental Methodology

We perform discriminative feature selection experiments on two high-dimensional classification tasks that have few relevant features and limited training data:

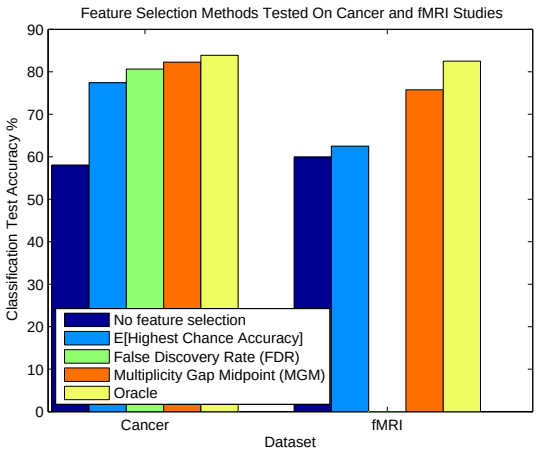


Figure 2. Accuracies for different feature selection methods for two classification tasks: Cancer (left) fMRI (right). The False Discovery Rate (FDR) method selected no features in the fMRI task.

Task 1: Cognitive state classification using functional magnetic resonance imaging (fMRI) In this task, we are given a time series of neural activity from thirteen human subjects. Each feature is the neuro-activation of a particular region of the brain at a given time. The goal is to distinguish between two cognitive states: reading a sentence, and viewing a picture (Mitchell et al., 2004). Each subject has ≈80,000 features and 40 examples.

Task 2: Colon cancer patient classification using microarray gene expression levels (Cancer) In this task, the goal is to predict whether a patient is diagnosed with colon cancer. The data are microarray gene expression levels from tissue samples (Alon et al., 1999). There are 2,000 features and 62 examples.

Testing Method In each experiment, we use a Gaussian Naive Bayes classifier and perform a leave-one-out-cross-validation. On each round, we leave out one example, and split the remaining examples into equal training and validation sets. We train using the first set, and measure classification accuracy on the validation set. We select the best performing features according to a specific criterion. After selecting features, we retrain by combining the validation and training sets. We then test the left out example. We repeat the process for each example.

We tested five different feature selection criteria:

1. **No feature selection** Uses all features.
2. **Highest Expected Chance Accuracy** Selects features that make fewer than $\mathbb{E}[X_{(1)}]$ mistakes.

3. **Binomial Hypothesis Test with False Discovery Rate correction** We select a feature if we reject the hypothesis that a classifier's true accuracy, trained on that feature, is 50%.[3] We use an $\alpha = 5\%$ level in the tests.
4. **Multiplicity Gap Midpoint (MGM) method** The method proposed in Section 5.1.
5. **Oracle Threshold** This is the threshold that would have led to the optimal testing accuracy.

5.3. Results and Discussion

In Figure 2, we see the classification results of five discriminative feature selection methods for both the colon cancer and fMRI datasets (for the fMRI dataset, we averaged the results of the 13 subjects together).

In both datasets, the threshold $\mathbb{E}[X_{(1)}]$ yields an improvement over no feature selection. But the assumptions made in calculating that threshold, namely that all features are independent and irrelevant, result in a very conservative threshold which admits few features.

The multiplicity gap midpoint (MGM) method relaxes these assumptions and performs significantly better. This method comes closest to the accuracy that could have been achieved had an oracle told us the optimal threshold to use[4].

As a state-of-the-art baseline, we tried a binomial hypothesis test with a false discovery rate correction. As is common in the statistical and scientific literature, we chose a significance level $\alpha = 0.05$. This method completely failed to select any features for the fMRI task, indicating that it is overly conservative for very high-dimensional problems. The method performed fairly well on the colon cancer dataset, but did so after selecting fewer than ten features.

It is worth noting that we could tune the α value of the false discovery rate test to admit more features and help performance. But the goal of the midpoint heuristic is to avoid this tuning (in fact, if we were to do tuning, it would make more sense to just tune the threshold for selecting features directly). Thus we feel the midpoint method provides a more appropriate *default threshold* than a specific value of α would in a classical test.

We chose the Gaussian Naive Bayes classifier because it is extremely fast to train and test making it very appropriate for use in a *wrapped feature selector*. This classifier is also robust to noise but is not entirely immune to overfitting. We found that adding additional features increased performance up to a point, but eventually noisy features overwhelmed the classifier, and performance degraded.

Figure 3 shows this effect for three fMRI subjects and the colon cancer dataset. The curves shows test accuracies at various feature selection thresholds. In each plot, the x-axis is the number of errors allowed, and the y-axis is the test accuracy of the resulting classifier. We mark the extremes of the multiplicity gap $\mathbb{E}[X_{(1)}]$ and $E[X]$ on each plot. On all thirteen subjects as well as the colon cancer dataset, the optimal (oracle chosen) threshold falls within this gap.

5.4. Future Work

The goal of this paper was to show how order statistics can be a useful tool for problems in machine learning. While our initial work focused on accuracy, we feel similar techniques can be applied to other measures such as information gain, entropy, and AUC.

Also, in our initial analysis we compute a significance threshold assuming that all features are independent. One natural extension of this work is to develop a method that adjusts for correlations between features.

6. Conclusion

We provided a theoretical analysis of the chance accuracy of large collections of classifiers. We showed that on problems with small numbers of examples and large numbers of features, we should expect some classifier to be highly accurate by random chance. We derived a theorem to directly calculate this accuracy.

We used this theorem to provide a principled feature selection criterion for sparse, high-dimensional problems. This criterion is theoretically well-motivated, simple to implement, and computationally inexpensive.

We demonstrated this method on microarray and fMRI datasets and showed that this method performs very close to the optimal oracle accuracy. We also showed that on the fMRI dataset this technique chooses relevant features while another state-of-the-art method, the False Discovery Rate (FDR), completely fails at standard significance levels.

Acknowledgments

We would like to offer our tremendous thanks to Haikady Nagaraja, Larry Wasserman, Tom Mitchell,

[3] This is appropriate since both datasets have nearly equal class priors.

[4] The oracle is determined by calculating the highest accuracy on a test set for every possible "number of errors" threshold on the validation set.

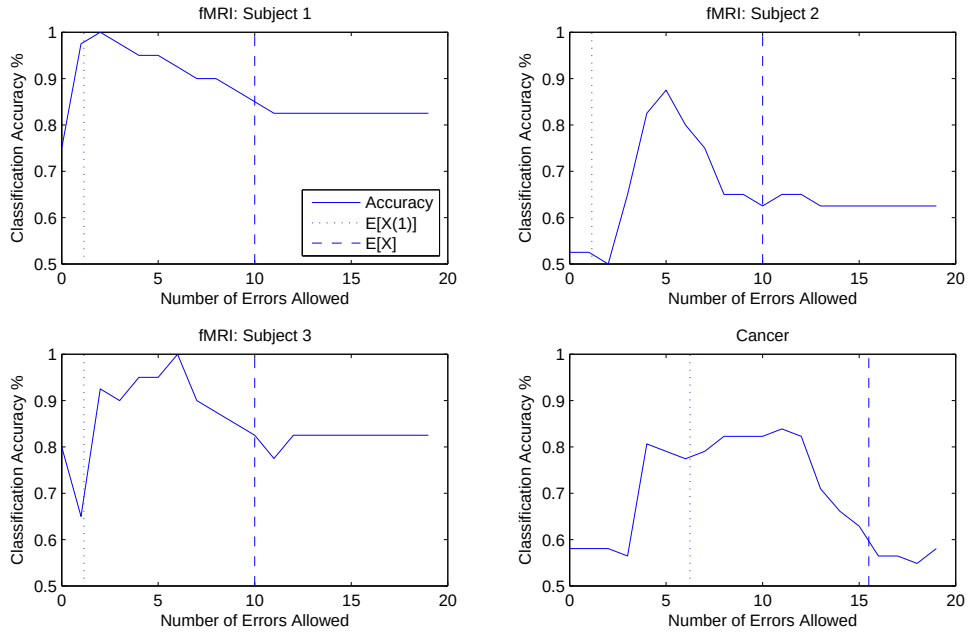


Figure 3. Test accuracies at various feature selection thresholds. In each plot, the x-axis is the number of errors allowed, and the y-axis is the test accuracy of the resulting classifier. We mark the extremes of the multiplicity gap $\mathbb{E}[X_{(1)}]$ and $E[X]$ with vertical lines on each plot.

and Geoff Gordon for their useful comments. We would also like to acknowledge the National Science Foundation, W.M Keck Foundation, and Yahoo! for their generous financial support.

References

Alon, U., et al. (1999). Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences of the United States of America*, *96*, 6745–6750.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B (Methodological)*, *57*, 289300.

Casella, G., & Berger, R. L. (2002). *Statistical inference*. Pacific Grove, CA: Duxbury.

David, H., & Nagaraja, H. (2003). *Order statistics*. Hoboken, NJ: Wiley.

Feller, W. (1957). *An introduction to probability theory and its applications*. New York, NY: Wiley.

Frank, E., & Witten, I. H. (1998). Using a permutation test for attribute selection in decision trees. *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 152–160). Morgan Kaufmann Publishers Inc.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Jensen, D., & Cohen, P. (2000). Multiple comparisons in induction algorithms. *Machine Learning*, *38*, 309–338.

Li, W., & Grosse, I. (2003). Gene selection criterion for discriminant microarray data analysis based on extreme value distributions. *RECOMB '03: Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology* (pp. 217–223). New York, NY, USA.

Mitchell, T. M., Hutchinson, R., Niculescu, R. S., Pereira, F., Wang, X., Just, M., & Newman, S. (2004). Learning to decode cognitive states from brain images. *Machine Learning*, *57*, 145–175.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, *58*, 267–288.

Wasserman, L. (2005). *All of statistics*. New York, NY: Springer.

Wong, W.-K., Moore, A., Cooper, G., & Wagner, M. (2002). Rule-based anomaly pattern detection for detecting disease outbreaks. *Eighteenth national conference on Artificial intelligence* (pp. 217–223). Edmonton, Alberta, Canada: American Association for Artificial Intelligence.

An Analysis of Linear Models, Linear Value-Function Approximation, and Feature Selection for Reinforcement Learning

Ronald Parr[⋆] PARR@CS.DUKE.EDU
Lihong Li[†] LIHONG@CS.RUTGERS.EDU
Gavin Taylor[⋆] GVTAYLOR@CS.DUKE.EDU
Christopher Painter-Wakefield[⋆] PAINT007@CS.DUKE.EDU
Michael L. Littman[†] MLITTMAN@CS.RUTGERS.EDU

[⋆]Department of Computer Science, Duke University, Durham, NC 27708 USA

[†]Department of Computer Science, Rutgers University, Piscataway, NJ 08854 USA

Abstract

We show that linear value-function approximation is equivalent to a form of linear model approximation. We then derive a relationship between the model-approximation error and the Bellman error, and show how this relationship can guide feature selection for model improvement and/or value-function improvement. We also show how these results give insight into the behavior of existing feature-selection algorithms.

1. Introduction

Broadly speaking, there are two types of reinforcement-learning (RL) algorithms: model-free and model-based algorithms. Model-free approaches typically use samples to learn a value function, from which a policy is implicitly derived. In contrast, model-based approaches build a model of system behavior from samples, and the model is used to compute a value function or policy. Both approaches have advantages and disadvantages, and function approximation can be applied to either, to represent a value function or a model. Examples of function approximators include decision trees, neural networks, and linear functions.

The first contribution of this paper shows that, when linear value-function approximation is used for policy evaluation as in nominally model-free approaches such as linear TD learning (Sutton, 1988) or LSTD (Bradtke & Barto, 1996), the value function is *precisely* the same as the value function that results from an exact solution to a corresponding approximate, linear model, where the value function and linear model are defined over the same set of features.

This insight results in a novel view of the Bellman error and a deeper understanding of the problem of feature selection when linear function approximation is used. Specifically, we show that the Bellman error can be decomposed into two types of errors in the learned linear model: the reward error and the feature error. This decomposition gives insight into the behavior of existing approximation techniques, suggests new views of feature selection, and explains the behavior of existing feature-selection algorithms.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Formal Framework and Notation

We are interested in both controlled and uncontrolled Markov processes with a set S of states s and, when applicable, a set A of actions a. Our main results and experiments consider the uncontrolled or policy-evaluation case, but many of the ideas can be applied to the controlled case, as is discussed in more detail in Section 3.2.

We refer to the uncontrolled case as a Markov reward process (MRP): $M = (S, P, R, \gamma)$, and the controlled case as a Markov decision process (MDP): $M = (S, A, P, R, \gamma)$. Given a state s_i, the probability of a transition to a state s_j given action a is given by P^a_{ij} and results in an expected reward of R^a_i. In the uncontrolled case, we use P_{ij} and R_i to stand for the transitions and rewards.

We are concerned with finding value functions V that map each state s_i to the expected total γ-discounted reward for the process. In particular, we would like to find the solution to the Bellman equation

$$V[s_i] = \max_a (R^a_i + \gamma \sum_j P^a_{ij} V[s_j])$$

in the controlled case (the "max" and a's are eliminated from the equation in the uncontrolled case).

For any matrix, A, we use A^T to indicate the transpose of A and $span(A)$ to indicate the column space of A.

2.1. Linear Value Functions

In cases where the value function cannot be represented exactly, it is common to use some form of parametric value-function approximation, such as a linear combination of features or basis functions:

$$\hat{V} = \sum_{i=1}^{k} w_i \phi_i,$$

where $\Phi = \{\phi_1, \ldots, \phi_k\}$ is a set of linearly independent[1] basis functions of the state, with $\phi_i(s)$ defined as the value of feature i in state s. The vector $\mathbf{w} = \{w_1, \ldots, w_k\}$ is a set of scalar weights. We can think of Φ as a design matrix with $\Phi[i,j] = \phi_j(s_i)$, that is, the basis functions span the columns of Φ and the states span the rows. Expressing the weights $\mathbf{w}$ as a column vector, we have $\hat{V} = \Phi\mathbf{w}$.

Methods for finding a reasonable $\mathbf{w}$ given Φ and a set of samples include linear TD (Sutton, 1988), LSTD (Bradtke & Barto, 1996) and LSPE (Yu & Bertsekas, 2006). If the model can be expressed as a factored MDP, then the weights can be found directly (Koller & Parr, 1999). We refer to this family of methods as *linear fixed-point* methods because they all solve for the fixed point

$$\hat{V} = \Phi\mathbf{w}_\Phi = \Pi_\sigma(R + \gamma P \Phi \mathbf{w}_\Phi), \tag{1}$$

where Π_σ is an operator that is the σ-weighted L_2 projection into $span(\Phi)$, where σ is a state weighting distribution, typically the stationary distribution of P. If $\Sigma = diag(\sigma)$, $\Pi_\sigma = \Phi(\Phi^T\Sigma\Phi)^{-1}\Phi^T\Sigma$. In some cases, an unweighted projection (uniform σ) or some other σ is used. Most of our results do not depend upon the projection weights, so we shall assume uniform σ unless otherwise stated. Solving for $\mathbf{w}_\Phi$ yields:

$$\begin{aligned} \mathbf{w}_\Phi &= (I - \gamma(\Phi^T\Phi)^{-1}\Phi^T P\Phi)^{-1}(\Phi^T\Phi)^{-1}\Phi^T R \quad (2)\\ &= (\Phi^T\Phi - \gamma\Phi^T P\Phi)^{-1}\Phi^T R. \quad (3)\end{aligned}$$

In this paper, we assume that P is known and that Φ can be constructed exactly, while in all but the factored model case, these would be sampled. This assumption allows us to characterize the representational power of the features as a separate issue from the variance introduced by sampling.

2.2. Linear Models

As in the case of linear value functions, we assume the existence of a set of linearly independent features $\phi_1 \ldots \phi_k$ for representing transition and reward models, with $\phi_i(s)$ defined as the value of feature i in state s. While value-function approximation typically uses features to predict values, we will consider the use of these features to predict *next* features. For feature vector $\Phi(s) = [\phi_1(s) \ldots \phi_k(s)]^T$, we define $\Phi(s'|s)$ as the random vector of next features:

$$\Phi(s'|s) \overset{s' \sim P(s'|s)}{=} [\phi_1(s'), \ldots, \phi_k(s')]^T,$$

our objective will be to produce a $k \times k$ matrix P_Φ that predicts expected next feature vectors,

$$P_\Phi{}^T \Phi(s) \approx E_{s' \sim P(s'|s)}\{\Phi(s'|s)\},$$

and minimizes the expected feature-prediction error:

$$P_\Phi = \arg\min_{P_k} \sum_s \| P_k^T \Phi(s) - E\{\Phi(s'|s)\} \|_2^2. \tag{4}$$

(For brevity, we shall henceforth leave $s' \sim P(s'|s)$ implicit). One way to solve the minimization problem in Eq. (4) is to compute the expected next feature explicitly as the $n \times k$ matrix $P\Phi$ and then find the least-squares solution to the over-constrained system $\Phi P_\Phi \approx P\Phi$, since the i^{th} row of ΦP_Φ is P_Φ's prediction of the next feature values for state i and the i^{th} row of $P\Phi$ is the expected value of these features. The least-squares solution is

$$P_\Phi = (\Phi^T\Phi)^{-1}\Phi^T P\Phi, \tag{5}$$

with approximate next feature values $\widehat{P\Phi} = \Phi P_\Phi$. To predict the reward model using the same features, we could perform a standard least-squares projection into $span(\Phi)$ to compute an approximate reward predictor:

$$r_\Phi = (\Phi^T\Phi)^{-1}\Phi^T R, \tag{6}$$

with corresponding approximate reward: $\hat{R} = \Phi r_\Phi$. As in the value-function approximation case, it is possible to do a weighted L_2 projection, a straightforward generalization that we omit for conciseness of presentation.

Classically, an advantage of learning a model and deriving values from the model (indirect learning) over using samples to estimate the values (direct learning) is that such a method can be very data efficient. On the other hand, learning an accurate model can require a great deal of experience. Surprisingly, we find that the two approaches are the same, at least in the linear approximation setting.

3. Linear Fixed-Point Solution = Linear-Model Solution

The notion that linear fixed-point methods are implicitly computing some sort of model has been recognized in varying degrees for several years. For example, Boyan (1999)

[1] Permitting linearly dependent basis functions would not change our results, but would complicate exposition since the resulting weight vectors would no longer be unique. In practice, one may use SVD to enforce the selection of a unique solution when features are linearly dependent.

considered the intermediate calculations performed by LSTD in some special cases, and interpreted parts of the LSTD algorithm as computing a compressed model. In this section, we show that the linear fixed-point solution for features Φ is *exactly* the solution to the linear model described by P_Φ and r_Φ. We first prove it for the uncontrolled case, and then generalize our result to the controlled case. Our results concern unweighted projections, but generalize readily to weighted projections.

3.1. The Uncontrolled Case

Recall that the approximate model transforms feature vectors to feature vectors, so any k-vector is a state in the approximate model. If $\mathbf{x}$ is such a state, then, in the approximate model, ${r_\Phi}^T\mathbf{x}$ is the reward for this state and ${P_\Phi}^T\mathbf{x}$ is the next state vector. The Bellman equation for state $\mathbf{x}$ is:

$$V[\mathbf{x}] = {r_\Phi}^T\mathbf{x} + \gamma V[{P_\Phi}^T\mathbf{x}] = \sum_{i=0}^{\infty} \gamma^i {r_\Phi}^T ({P_\Phi}^i)^T\mathbf{x}.$$

Expressed with respect to the original state space, the value function becomes

$$V = \Phi \sum_{i=0}^{\infty} \gamma^i {P_\Phi}^i r_\Phi,$$

which is a linear combination of the columns of Φ. Since $V = \Phi\mathbf{w}$ for some $\mathbf{w}$, the fixed-point equation becomes:

$$\begin{aligned} V &= \hat{R} + \gamma\widehat{P\Phi\mathbf{w}} && (7)\\ \Phi\mathbf{w} &= \Phi r_\Phi + \gamma\Phi P_\Phi\mathbf{w} && (8)\\ \mathbf{w} &= (I - \gamma P_\Phi)^{-1} r_\Phi. && (9) \end{aligned}$$

We call the solution to the system above the *linear model solution*. A solution will exist when P_Φ has a spectral radius less than $1/\gamma$. This condition is not guaranteed because P_Φ is not necessarily a stochastic matrix; it is simply a matrix that predicts expected next feature values. The cases where the spectral radius of P_Φ exceeds $1/\gamma$ correspond to the cases where the value function defined by P_Φ and r_Φ assigns unbounded value to some states.

Theorem 3.1 *For any MRP M and set of features Φ, the linear-model solution and the linear fixed-point solution are identical.*

Proof We begin with the expression for the linear-model solution from Eq. (9) and then proceed by substituting the definitions of P_Φ and r_Φ from Eq. (5) and Eq. (6), yielding:

$$\begin{aligned} \mathbf{w} &= (I - \gamma P_\Phi)^{-1} r_\Phi \\ &= (I - \gamma(\Phi^T\Phi)^{-1}\Phi^T P\Phi)^{-1}(\Phi^T\Phi)^{-1}\Phi^T R \\ &= \mathbf{w}_\Phi. \ \blacksquare \end{aligned}$$

This result demonstrates that for a given set of features Φ, there is no difference between using the exact model to find an *approximate* linear fixed-point value function in terms of Φ and first constructing an approximate linear model in terms of Φ and then solving for the *exact* value function of the approximate model using Eq. (9). Although the model-based view produces exactly the same value function as the value-function-based view, the model-based view can give a new perspective on error analysis and feature selection, as shown in later sections.

3.2. The Controlled Case: LSPI

For the controlled case, we denote a *policy* as $\pi : S \mapsto A$. Since rewards and transitions are action dependent, the value function is defined over state–action pairs and is called a Q-function: For a fixed policy π, Q^π is the unique fixed-point solution to the Bellman equation

$$Q^\pi[s_i, a] = R_i^a + \gamma \sum_j P_{ij}^a Q^\pi[s_j, \pi(s_j)].$$

As in Section 2.1, Q^π can be approximated by functions in $span(\Phi)$: $\hat{Q} = \sum_{i=1}^k w_i\phi_i$, but now the basis functions ϕ_i are defined over state–action pairs rather than states.

In the controlled case, the policy π can be refined iteratively, as in the Least-Squares Policy Iteration (LSPI) algorithm (Lagoudakis & Parr, 2003). Starting with an arbitrary policy π_1, LSPI performs two steps iteratively until certain termination conditions are satisfied. In iteration i, it first computes an approximate linear value function $\hat{Q}_i$ for the current policy π_i (the *policy-evaluation* step), and then computes a new policy π_{i+1} that is greedy with respect to $\hat{Q}_i$ (the *policy-improvement* step).

In the policy-evaluation step, an algorithm LSTDQ, which is the Q-version of the LSTD algorithm, is used to compute $\hat{Q}_i$. Since a Markov decision process controlled by a fixed policy is equivalent to an induced Markov reward process whose state space is $S \times A$, LSTDQ can be viewed as LSTD running over this induced MRP. Due to Theorem 3.1, LSTDQ effectively builds a least-squares linear model approximation and then finds the *exact* solution to this model. Therefore, the intermediate value functions $\hat{Q}_i$ found by LSPI are the exact value functions of the respective approximate linear models with the smallest (weighted) L_2 error.

4. Analysis of Error: Uncontrolled Case

Value-function-based methods often analyze the error of a value function $\hat{V}$ in terms of the one-step lookahead error, or *Bellman error*:

$$BE(\hat{V}) = R + \gamma P\hat{V} - \hat{V}.$$

In the context of linear value functions and linear models, we shall define the Bellman error for a set Φ of features as the error in the linear fixed-point value function for Φ:

$$BE(\Phi) = BE(\Phi \mathbf{w}_\Phi) = R + \gamma P \Phi \mathbf{w}_\Phi - \Phi \mathbf{w}_\Phi.$$

To understand the relationship between the error in the linear model and the Bellman error, we define two components of the model error, the *reward error*:

$$\Delta_R = R - \hat{R},$$

and the *per-feature error*:

$$\Delta_\Phi = P\Phi - \widehat{P\Phi}.$$

The per-feature error is the error in the prediction of the expected next feature values, so both terms can be thought of as the residual error of the linear model. The next theorem relates the Bellman error to these model errors.

Theorem 4.1 *For any MRP M and features Φ,*

$$BE(\Phi) = \Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi. \tag{10}$$

Proof Using the definitions of $BE(\Phi)$, Δ_R, and Δ_Φ:

$$\begin{aligned} BE(\Phi) &= R + \gamma P \Phi \mathbf{w}_\Phi - \Phi \mathbf{w}_\Phi \\ &= (\Delta_R + \hat{R}) + \gamma(\Delta_\Phi + \widehat{P\Phi})\mathbf{w}_\Phi - \Phi \mathbf{w}_\Phi \\ &= (\Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi) + \hat{R} + (\gamma \Phi P_\Phi - \Phi)\mathbf{w}_\Phi \\ &= (\Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi) + \hat{R} - \Phi(I - \gamma P_\Phi)\mathbf{w}_\Phi \\ &= (\Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi) + \hat{R} - \Phi r_\Phi \\ &= \Delta_R + \gamma \Delta_\Phi \mathbf{w}_\Phi. \end{aligned}$$

The final step follows from the definition of $\hat{R}$, and the penultimate step follows from Eq. (9) and Theorem 3.1. ∎

This decomposition of the Bellman error lets us think of the Bellman error as composed of two separate sources of error: reward error, and per-feature error. In the next section, we show that this view can give insight into the problem of feature selection, but we also caution that there can be interactions between Δ_R and Δ_Φ. For example, consider the basis composed of the single basis function $\Phi^* = [V^*]$. Clearly, $BE(\Phi^*) = 0$, but for any non-trivial problem and approximate model, Δ_R and Δ_Φ will be nonzero and will cancel each other out in Eq. (10).

A similar result may be possible for the controlled case, but there are some subtleties. For example, there is not a clean notion of a fixed point for the outer loop of the LSPI algorithm since the algorithm is not guaranteed to converge to a single policy or $\mathbf{w}$.

5. Feature Selection

We present several insights on the problem of feature selection that follow from the results presented above.

5.1. General Observations about Δ_R and Δ_Φ

The condition $\Delta_R = \Delta_\Phi = 0$ is sufficient (but not necessary) to achieve zero Bellman error and a perfect value function. Specifically, it requires that the features of the approximate model capture the structure of the reward function, and that the features of the approximate model are sufficient to predict expected next features. In the case where Φ is a set of indicator functions over disjoint partitions of S, these conditions are similar to those specified for model minimization (Dean & Givan, 1997) in MDPs.

Features that are insufficient to represent the immediate reward are likely to be problematic since any error in the prediction of the immediate reward based upon the features (Δ_R) can appear directly in the Bellman error through the first summand of Eq. (10). This finding is consistent with the observation of Petrik (2007) of the problems that arise when the reward is orthogonal to the features.

For $\Delta_\Phi = 0$, the Bellman error is determined entirely by Δ_R, with no dependence on γ. This observation has some interesting implications for feature selection and the analysis of the resulting approximate value function, topics we address further in Section 5.3.

5.2. Incremental Feature Generation

This section presents two existing methods for incrementally building a basis, the *Krylov basis*, and *Bellman Error Basis Functions* (BEBFs). We also propose a new method based upon the model error, Model Error Basis Functions (MEBFs), then show that all three methods are equivalent given the same initial conditions.

5.2.1. The Krylov Basis

The Krylov basis is defined in terms of powers of the transition matrix multiplied by R. We refer to the Krylov basis with k basis functions, starting from $\mathbf{X}$, as $Krylov_k(\mathbf{X})$, with $Krylov_k(\mathbf{X}) = \{P^{i-1}\mathbf{X} : 1 \leq i \leq k\}$. For an MRP, typically $\mathbf{X} = R$. The Krylov basis, and Krylov methods in general, are standard techniques for the iterative solution to systems of linear equations. Its relevance to feature selection for RL was demonstrated by Petrik (2007).

5.2.2. BEBFs

Many researchers have proposed using features based upon the residual error in the current feature set (Wu & Givan, 2004; Sanner & Boutilier, 2005; Keller et al., 2006). Parr et al. (2007) describe this family of techniques as Bellman Error Basis Functions (BEBFs), and analyze some of the properties of this approach. More formally, if $\Phi_k \mathbf{w}_{\Phi_k}$ is the current value function, BEBF adds $\phi_{k+1} = BE(\Phi_k)$ as the next basis function. We refer to the basis resulting from $k-1$ iterations of BEBF, starting from $\mathbf{X}$, as $BEBF_k(\mathbf{X})$.

Theorem 5.1 *(Petrik[2]) For any $k \geq 1$,*

$$span(Krylov_k(R)) = span(BEBF_k(R)).$$

Proof The proof is by induction on k. For the basis:

$$Krylov_1(R) = BEBF_1(R) = R.$$

For the inductive step, we assume equality up to k, so for both methods the value function can be expressed as:

$$\Phi_k \mathbf{w}_{\Phi_k} = \sum_{i=1}^{k} w_i P^{i-1} R.$$

Now, observe that:

$$BE(\Phi_k) = R + \gamma P(\sum_{i=1}^{k} w_i P^{i-1} R) - \sum_{i=1}^{k} w_i P^{i-1} R.$$

The only part of the above that is not already in the basis is the contribution from $P^{k+1}R$, which is precisely what is added in $Krylov_{k+1}(R)$. ∎

5.2.3. MEBFs

A natural generalization of BEBFs to the model-based view would be to add features that capture the residual error in the model. Starting from Φ_k, this technique (MEBF) adds Δ_R and Δ_Φ (or the linearly independent components thereof) to the basis at each iteration to create Φ_{k+1}. In contrast to BEBFs, this method can add a large number of basis functions at each iteration since Δ_Φ has as many columns as Φ. One might imagine that this process could result in an exponential growth in the number of basis functions. In fact, however, the number of new basis functions added at each iteration will not grow since each new set of basis functions that is added will drive the error in the previous basis functions to 0.

We refer to the basis resulting from $k-1$ iterations of MEBF, starting from $\mathbf{X}$, as $MEBF_k(\mathbf{X})$. For an initial basis of Φ, the MEBF basis expansion is guaranteed to contain the BEBF basis expansion.

Theorem 5.2 $span(BEBF_2(\Phi)) \subseteq span(MEBF_2(\Phi))$.

Proof Follows immediately from Eq. (10). ∎

Theorem 5.3 *For $k \geq 1$:*

$$span(Krylov_k(R)) = span(MEBF_k(R)).$$

Proof The proof is by induction on k. For the basis:

$$Krylov_1(R) = MEBF_1(R) = R.$$

For the inductive step, we assume equality up to k and consider the behavior of MEBF. For $k \geq 1$, $\Delta_R = 0$, since R is the first basis function added. The basis Φ_k is equivalent to a collection of basis functions of the form $\phi_i = P^{i-1}R$ for $1 \leq i \leq k$. As a result, $P\phi_i$ is already in the basis for all $1 \leq i < k$. Thus, the only nonzero column of Δ_Φ will correspond to feature ϕ_k and will be $P^k R - P_\Phi P^{k-1} R$. Since $P_\Phi P^{k-1} R$ is necessarily in $span(\Phi_k)$, the only new contribution to the basis made by MEBF will be from $P^k R$, which is precisely what is added by $Krylov_{k+1}(R)$. ∎

These results show that, starting from R, all three methods will produce the same basis. An advantage of BEBF is that it will produce orthogonal basis vectors. An advantage of MEBF is that it can add multiple new basis vectors at each iteration if it is initialized with a set of basis functions.

5.3. Invariant Subspaces of P

The form of the Bellman error in Eq. (10) suggests that features for which $\Delta_\Phi = 0$ are particularly interesting. If a dictionary of such features were readily available, then the feature-selection problem would reduce to the problem of predicting the immediate reward using this dictionary.

The condition $\Delta_\Phi = 0$ means that, collectively, the features are a basis for a perfect linear predictor of their *own* next, expected values. More formally, features Φ are *subspace invariant* with respect to P if $P\Phi$ is in $span(\Phi)$, which means that there exists a Λ such that $P\Phi = \Phi\Lambda$.

At first, it may seem like subspace invariance is an extraordinary requirement that could hold only for a complete basis for P. It turns out, however, that there are many ways to describe invariant subspaces of P. Any set of eigenvectors of P forms an invariant subspace. For eigenvectors $\mathcal{X}_1 \ldots \mathcal{X}_k$ with eigenvalues $\lambda_1 \ldots \lambda_k$, $\Lambda = \text{diag}(\lambda_1 \ldots \lambda_k)$. The set of generalized eigenvectors corresponding to a particular eigenvalue λ of P is subspace invariant with respect to P. For an eigenvalue λ with multiplicity i, there will be i generalized eigenvectors, $\mathcal{X}_1 \ldots \mathcal{X}_i$ satisfying $(P - \lambda I)\mathcal{X}_j = \mathcal{X}_{j-1}$ for $1 \leq j \leq i$ and $(P - \lambda I)^j \mathcal{X}_j = 0$ for $0 \leq j \leq i$. More generally, if Φ^1 and Φ^2 are subspace invariant with respect to P, then so is their union. Finally, the Schur decomposition of a matrix P provides a set of nested invariant subspaces of P.

In fairness, we point out that these methods all require knowledge of P and superlinear computation time in the dimension of P. We defer discussion of the practicality of implementing these methods to Section 6 and Section 7.

Theorem 5.4 *For any MRP M and subspace invariant feature set Φ, $\Delta_\Phi = 0$.*

Proof First, we observe that P_Φ has a particularly simple

[2] M. Petrik, personal communication, 2007.

form as a consequence of subspace invariance:

$$P_\Phi = (\Phi^T\Phi)^{-1}\Phi^T P\Phi = (\Phi^T\Phi)^{-1}\Phi^T\Phi\Lambda = \Lambda.$$

Substituting into the definition of Δ_Φ:

$$\Delta_\Phi = P\Phi - \widehat{P\Phi} = P\Phi - \Phi P_\Phi = \Phi\Lambda - \Phi\Lambda = 0. \ \blacksquare$$

Subspace invariant features have additional intriguing properties. The resulting value function always exists and can be interpreted as the result of using the true transition model with the approximate reward function $\hat{R}$.

Theorem 5.5 *For any MRP M and subspace invariant feature set Φ, $\mathbf{w}_\Phi$ always exists and*

$$\Phi\mathbf{w}_\Phi = (I - \gamma P)^{-1}\hat{R}.$$

Proof Starting with the form of $\mathbf{w}_\Phi$ from Eq. (7) and the fact that $\Delta_\Phi = 0$:

$$\begin{aligned} \Phi\mathbf{w}_\Phi &= \hat{R} + \gamma\widehat{P\Phi}\mathbf{w}_\Phi \\ &= \hat{R} + \gamma P\Phi\mathbf{w}_\Phi \\ \Phi\mathbf{w}_\Phi - \gamma P\Phi\mathbf{w}_\Phi &= \hat{R} \\ \Phi\mathbf{w}_\Phi &= (I - \gamma P)^{-1}\hat{R}. \end{aligned}$$

To confirm that such a $\mathbf{w}_\Phi$ actually exists, we note that $\hat{R} \in span(\Phi)$ by construction, and that $(I - \gamma P)^{-1}$ must exist for the actual P and $0 \leq \gamma < 1$, allowing us to rewrite:

$$\Phi\mathbf{w}_\Phi = \sum_{i=0}^{\infty} \gamma^i P^i \hat{R},$$

which remains in $span(\Phi)$ because of Φ's subspace invariance with respect to P. $\blacksquare$

Our analysis has some similarities with that of Petrik (2007). Petrik considered the eigenvalue decomposition of P as a basis and considered the error in the projection of V^* into this basis. Petrik also suggested the use of the Jordan form, which would provide generalized eigenvectors for matrices that are not diagonalizable. Our analysis focuses on the Bellman error of the linear fixed-point solution. Insights from the model-based view of linear approximation architectures allow us to decompose the error into distinct components corresponding to the reward and transition models, making the role of invariant subspaces particularly salient.

6. Experimental Results

We present policy-evaluation results on three different problems. Our objective is to demonstrate how our theoretical results can inform the feature-selection process and explain the behavior of known feature-selection algorithms. We consider 4 algorithms:

PVF: This is the proto-value function (PVF) framework described by Mahadevan and Maggioni (2007). PVFs use eigenvalues of the Laplacian derived from an empirically constructed adjacency matrix (from random walk trajectories), enumerated in increasing order of eigenvalue. We reproduced their method as closely as possible, including adding links to the adjacency matrix for all policies, not just the policy under evaluation. Curiously, removing the off-policy links seemed to produce worse performance. We avoided using samples to eliminate the confounding (for our purposes) issue of variance between experiments. We used the combinatorial Laplacian for the 50-state and blackjack problems, but used the normalized Laplacian in the two-room problem to match Mahadevan and Maggioni (2007).

PVF-MP: This algorithm selects basis functions from the set of PVFs, but selects them incrementally based upon the Bellman error. Specifically, basis function $k+1$ is the PVF that has highest dot product with the Bellman error resulting from the previous k basis functions. It can be interpreted as a form of matching pursuits (Mallat & Zhang, 1993) on the Bellman error with a dictionary of PVFs.

Eig-MP: This algorithm is similar to PVF-MP, but selects from a dictionary of the eigenvectors of P. Both Eig-MP and PVF-MP are similar in spirit to Petrik's WL algorithm.

BEBF: This is the BEBF algorithm starting with $\Phi_0 = R$, as described in Section 5.2.

Our experiments performed unweighted L_2 projection and report unweighted L_2 norm error. We also considered L_∞ error and L_2 projections weighted by stationary distributions, but the results were not qualitatively different. We report the Bellman error, the reward error, and the *feature error*, which is the contribution of the per-feature errors to the Bellman error: $\gamma\Delta_\Phi\mathbf{w}_\Phi$. These metrics are presented as a function of the number of basis functions.

6.1. 50-state Chain

We applied all 4 algorithms to the 50-state chain problem from Lagoudakis and Parr (2003), with the results shown in Figure 1(a–c). As demanded by theory, Eig-MP has 0 feature error, which means that the entirety of the Bellman error is expressed in Δ_R. BEBFs represent the other extreme since $\Delta_R = 0$ after the first basis function is added and the entirety of the Bellman error is expressed through Δ_Φ. For this problem, PVFs appear to be approximately subspace invariant, resulting in low Δ_Φ. However, both Eig-MP and the PVF methods do poorly because the reward is not easily expressed as linear combination of a small number of PVFs. PVF-MP does better than plain PVFs because it is actively trying to reduce the error, while plain PVFs choose basis functions in an order that ignores the reward.

6.2. Two-room Problem

We tried all four algorithms on an optimal policy for the two-room navigation problem from Mahadevan and Maggioni (2007). The transition matrix for this problem is not diagonalizable and typical methods for extracting generalized eigenvectors proved unreliable, so we do not show results for the Eig-MP method. Figure 1(d–f) shows the breakdown of error for the remaining algorithms. In this case, the Laplacian approach produces features that behave less like an invariant subspace, resulting in high Δ_R and Δ_Φ. However, there is some cancellation between them.

6.3. Blackjack

We tested a version of the bottomless-deck blackjack problem from Sutton and Barto (1998), evaluating the policy they propose. For the model described in the book, all methods except BEBF performed extremely poorly. To make the problem more amenable to eigenvector-based methods, we implemented an ergodic version that resets to an initial distribution over hands with a value of 12 or larger and used a discount of 0.999. The breakdown of error for the different algorithms is shown in Figure 1(g–i), where we again omit Eig-MP. As expected, BEBFs exhibit $\Delta_R = 0$, and drive the Bellman error down fairly rapidly. PVFs exhibit some interesting behaviors: When the PVFs are enumerated in order of increasing eigenvalue, they form an invariant subspace. As a result, the feature error for PVFs hugs the abscissa in Figure 1(i). However, this ordering completely fails to match R until the very last eigenvectors are added, resulting in very poor performance overall. In contrast, PVF-MP adds basis eigenvectors in an order that does not result in subspace invariant features sets, but that does match R earlier, resulting in a more consistent reduction of error.

7. Discussion and Future Work

A significant finding in our work is the close relationship between value-function approximation and model-based learning. Our experimental results illustrate the relationship between the power of the features to represent an approximate model and the Bellman error.

While features that represent feature transitions accurately have highly desirable properties, both components of the model, the reward function and the transition function, should be respected by the features. Both a strength and weakness of the BEBF/MEBF/Krylov methods is their connection to specific policies and reward structures. Our results are consonant with those of Petrik (2007), which showed good performance for the Krylov basis and some surprisingly weak performance for eigenvector-based methods despite their appealing properties.

To focus on the expressive power of the features, our results in this paper do not directly consider sampled data, the regime in which linear fixed-point methods are most often employed. Some initial results on the effects of noise in feature generation for BEBF/MEBF/Krylov methods can be found in Parr et al. (2007), however further analysis would still be helpful. For eigenvector-based methods, there are some questions about the cost of estimating eigenvectors of P, or an approximation to P via the Laplacian. Computing eigenvectors can be computationally intensive and, for general P, prone to numerical instabilities.

An important direction for future work is seeking a deeper understanding of the interaction between feature-selection and policy-improvement algorithms such as LSPI.

8. Conclusion

This paper demonstrated a fundamental equivalence between linear value-function approximation and linear model approximation for RL. This equivalence led to a novel view of the Bellman error, which then gave insight into the problem of feature selection. These insights were used to explain the behavior of existing feature-selection algorithms on some sample problems. While this research has not, yet, led to a novel algorithmic approach, we believe that it helps address fundamental questions of representation and feature selection encountered by anyone wishing to solve real RL problems.

Acknowledgment

We thank Carlo Tomasi and Xiaobai Sun for helpful discussions, Sridhar Mahadevan and Jeff Johns for pointing out some discrepancies between our interpretation of the two-room problem in an earlier version of this paper and the version in Mahadevan and Maggioni (2007), and Marek Petrik for Theorem 5.1. This work was supported in part by DARPA CSSG HR0011-06-1-0027, and by NSF IIS-0713435. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

References

Boyan, J. A. (1999). Least-squares temporal difference learning. *ICML-99*.

Bradtke, S., & Barto, A. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, *2*.

Dean, T., & Givan, R. (1997). Model minimization in Markov decision processes. *AAAI-97*.

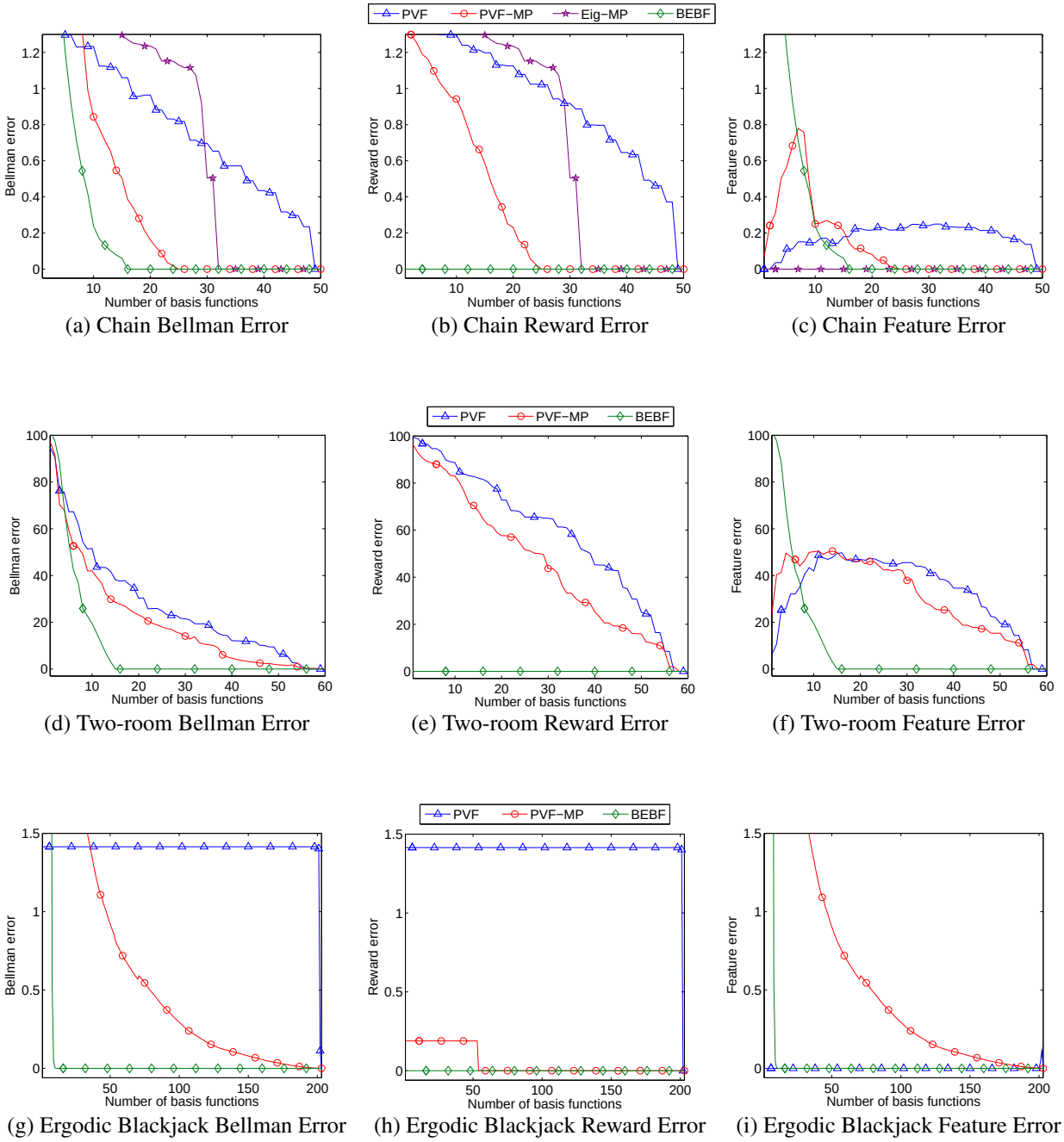
Keller, P., Mannor, S., & Precup, D. (2006). Automatic basis function construction for approximate dynamic programming and reinforcement learning. *ICML 2006*.

Koller, D., & Parr, R. (1999). Computing factored value functions for policies in structured MDPs. *IJCAI-99*.

Lagoudakis, M., & Parr, R. (2003). Least squares policy iteration. *JMLR*, *4*.

Mahadevan, S., & Maggioni, M. (2007). Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *JMLR*, *8*.

Mallat, S. G., & Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Trans. on Signal Processing*, *41*.



(a) Chain Bellman Error (b) Chain Reward Error (c) Chain Feature Error

(d) Two-room Bellman Error (e) Two-room Reward Error (f) Two-room Feature Error

(g) Ergodic Blackjack Bellman Error (h) Ergodic Blackjack Reward Error (i) Ergodic Blackjack Feature Error

Figure 1. Decomposition of the Bellman error for three different problems. First row: 50-state chain; Second row: Two-room problem; Third row: Ergodic Blackjack. First column: Bellman error; Second Column: reward error; Third Column: feature error

Parr, R., Painter-Wakefield, C., Li, L., & Littman, M. (2007). Analyzing feature generation for value-function approximation. *ICML-07*.

Petrik, M. (2007). An analysis of Laplacian methods for value function approximation in MDPs. *IJCAI-07*.

Sanner, S., & Boutilier, C. (2005). Approximate linear programming for first-order MDPs. *UAI-05*.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. The MIT Press.

Wu, J.-H., & Givan, R. (2004). *Feature-discovering approximate value iteration methods* (Technical Report TR-ECE-04-06). Purdue University.

Yu, H., & Bertsekas, D. (2006). *Convergence results for some temporal difference methods based on least squares* (Technical Report LIDS-2697). MIT.

Learning to Learn Implicit Queries from Gaze Patterns

Kai Puolamäki KAI.PUOLAMAKI@TKK.FI
Antti Ajanki ANTTI.AJANKI@TKK.FI
Samuel Kaski SAMUEL.KASKI@TKK.FI
Helsinki Institute for Information Technology, Department of Information and Computer Science, Helsinki University of Technology, P.O. Box 5400, FI-02015 TKK, Finland

Abstract

In the absence of explicit queries, an alternative is to try to infer users' interests from implicit feedback signals, such as clickstreams or eye tracking. The interests, formulated as an implicit query, can then be used in further searches. We formulate this task as a probabilistic model, which can be interpreted as a kind of transfer or meta-learning. The probabilistic model is demonstrated to outperform an earlier kernel-based method in a small-scale information retrieval task.

1. Introduction

The classic problem in information retrieval (IR) is to rank a set of documents according to the user's current interest, with the documents most relevant for the user ranked among the first. The same theme recurs currently in other applications of machine learning such as recommender systems. Current IR systems rely mostly on explicit, typed queries to perform the ranking.

The main problem in this traditional IR scenario is that it is difficult even for experienced users to formulate good textual queries (Turpin & Scholer, 2006), and therefore user's interest needs to be inferred partly from other sources. A straightforward way is to collect explicit feedback, that is, the user labels some of the documents relevant or irrelevant for her interests. Giving explicit feedback is however laborious. It would be ideal if the IR system would be able to unobtrusively collect and use *implicit feedback* to infer the interest of the user while she works and use this information to improve the quality of the search results. We call this task *proactive information retrieval*.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Several forms of implicit feedback, such as clickstream data, time spent during reading, and amount of scrolling and exit behaviour, have been used with some success (Kelly & Teevan, 2003; Claypool et al., 2001; Fox et al., 2005; Joachims et al., 2005; Joachims & Radlinski, 2007). While these sources of feedback are often readily available, they offer only limited information of users' interests.

Gaze patterns are a promising source of information about the attention of the user, and hence of implicit feedback. They have been used for information retrieval in two papers (Puolamäki et al., 2005; Hardoon et al., 2007). In the latter, eye tracking-based feedback improved information retrieval performance in an experiment where no explicit queries were available, and everything was inferred from the eye movements and the texts. The setup was slightly different from standard IR. The users saw sets of ten simplified (Wikipedia) documents, about half of which were relevant to a topic given to them beforehand, while the remaining documents were of other randomly selected topics. Based on the gaze pattern, an implicit query was constructed and used to rank unseen documents. The results were significantly better than random rankings.

We extend these results in two ways. First, we will use the eye tracking-based feedback in a more realistic IR scenario. Instead of randomly sampled documents, the user is shown a ranked list of top-5 results, and the task of the system is to improve the ranking of the yet unseen documents. Second, we will improve on the methodology. In (Hardoon et al., 2007) we introduced a two-stage prediction algorithm ("SVM model" in the following), where the latter stage was an SVM which classified new documents into relevant and irrelevant, given a parameter vector that consists of a weight for each word. The parameter vector was inferred with a regressor which had been trained to predict the weight of a word based on the eye movement pattern on the

word. The problem with this method is that learning of the regressor requires a ground truth which is tricky. In the earlier paper we used the parameter vector of an SVM taught to classify the relevant and irrelevant documents in the off-line learning stage, based on their textual content. This is intuitively a sensible strategy, and the experimental results validated it, but the choice is unlikely to be optimal.

In this paper, we introduce a probabilistic model for inferring relevance of the documents. Incorporating both of the two stages, inference of the relevance of a new document and inference of the implicit query, into a single generative model solves rigorously the problem of getting the ground truth and the learning procedure will be optimal for the task, given our modeling assumptions. The method indeed outperforms the earlier one (Hardoon et al., 2007).

Learning of the probabilistic model is related to transfer learning and meta-learning. The implicit query is expressed in the model as latent variables shared within each search task. The central task is to learn an implicit query for a topic unseen in the original training phase, which translates to transfer learning. The eye movements from which the implicit query is inferred can be considered as meta-data for the documents.

We test the method in an experimental scenario designed to closely resemble a real information retrieval setup. The user makes a query within a restricted Wikipedia corpus located in our customized Wikipedia server. A search engine then ranks the top-10 documents for this query, and the first five are shown sequentially to the user using a web browser. The browser has been modified to record and transmit the eye movement measurements to the Wikipedia server. We rank the remaining 5 documents by our eye movement-based model and aggregate the new and the original ranking to produce an ordering for the remaining 5 documents. Average precision in the re-ranked 5 documents was used as the goodness criterion.

Our method, once trained, consists only of a linear discriminator applied to term-specific gaze and term features. Therefore, the method can be applied efficiently in linear time whenever the eye tracking data is available.

2. The Information Retrieval Task

The usual approach in IR is to rank the documents based on their match to a textual query (Baeza-Yates & Ribeiro-Neto, 1999). In our setup the user types in a textual query that reflects her interest but is typically an incomplete description. Then the search engine shows her the top-ranked documents. The eye movements of the user are measured while she reads the documents presented sequentially in the ranked order. Our objective is to use the gaze patterns to improve the ranking of the yet unseen documents. These documents are consequently shown to the user in an order modified using the implicit query inferred from the gaze patterns. In doing so the relevant documents are hopefully shown to the user earlier, that is, the average precision of the search result is improved.

2.1. Okapi BM25 Ranking Function

A widely used ranking function is given by Okapi BM25 (Robertson & Walker, 1994; Robertson & Zaragoza, 2007), which is also used as a baseline method throughout this paper. Okapi BM25 ranks the documents given a textual query q that is a set of terms. Our approach is independent of the actual ranking function, however; indeed, in this work, we could replace Okapi BM25 with any information retrieval system that outputs a ranking of documents for a given query.

Consider a document collection C where each document $d = \{tf_t\}_{t \in V}$ in the collection is a vector of term frequencies, where tf_t is the frequency of term t in the document and V is the vocabulary. For ad hoc retrieval the BM25 weighting function can be expressed as

$$w_t(d, C) = \frac{(1 + k_1) tf_t}{k_1 \left((1 - b) + b \frac{dl}{avdl}\right) + tf_t} \log \frac{|C| - df_t + \frac{1}{2}}{df_t + \frac{1}{2}}, \quad (1)$$

where df_t is the document frequency of term t, dl is the document length and $avdl$ is the average document length across the collection. The k_1 and b are free parameters which we for the purposes of this paper fix to $k_1 = 1.2$ and $b = 0.75$, as suggested by Robertson and Walker (1999). The documents are ranked according to the sum of the weights of the terms in query q:

$$W(d, q, C) = \sum_{t \in q} w_t(d, C). \quad (2)$$

2.2. Metasearch

We have at our disposal a separate and independent ranking system, described in detail in Section 3, that ranks the yet unseen documents based on the gaze patterns of the users. In effect, we have two rankings: the ranking derived from the textual query and given by the Okapi BM25 ranking function described in Section 2.1, and the ranking derived from the gaze patterns.

The problem of combining multiple search engine rankings into one is known as *metasearch* and it has been studied extensively during the past years (see, for example, Cohen et al., 1998; Aslam & Montague, 2001). Because in this work we aim for simplicity and robustness, we use a straightforward linear combination of rankings. Another reason for this choice is that we want our approach to work also with a "black box" search engine which only gives us a ranking of the documents, without any probability of relevance associated with the documents.

In more detail, let $r_{BM25}(d)$ and $r_{EYE}(d)$ be the ranks of the document d given by the Okapi BM25 ranking function and the eye movement model, respectively. We re-rank the yet unseen documents using $score(d)$ defined by

$$score(d) = \gamma r_{BM25}(d) + (1-\gamma) r_{EYE}(d), \quad (3)$$

with the document having the smallest $score(d)$ ranked first. Here γ is a constant between zero and one.

3. Learning to Learn: A Probabilistic Model

In this section we introduce a probabilistic model that can be used to infer the ranking of yet unseen documents for a new and unknown query, given how the user has viewed a set of documents. In practice, the viewed documents are the highest-ranked documents for a given unknown query, and the inferred ranking is used to modify the order in which the further documents are presented.

3.1. Probabilistic Model

The available data is a collection of documents, encoded as TFIDF vectors $\mathbf{d}$, where the component corresponding to term t is

$$\mathbf{d}_t = tf_t \log \frac{|C|}{df_t}.$$

For the viewed documents we additionally have eye movement features. The feature vector $\mathbf{e}_{it}$ contains feature values for term t in document i. There are two types of features: eye movement features that are computed from the eye movement pattern over the term, and textual features that depend only on the term and its location in the document. The features are described in Section 4.3.

In the model, the relevancy r of a document is assumed to depend on the TFIDF vector $\mathbf{d}$ of the document and a search task specific *query vector* $\mathbf{w}$. Our main assumption is that the importance of a certain term for a search query depends only on the way the term is viewed, not on the meaning of the term. This allows us to learn global parameters α and $\boldsymbol{\beta}$, which are common for all search tasks, for the mapping from the features $\mathbf{e}$ to the term's weights w_t. The model is illustrated in Figure 1.

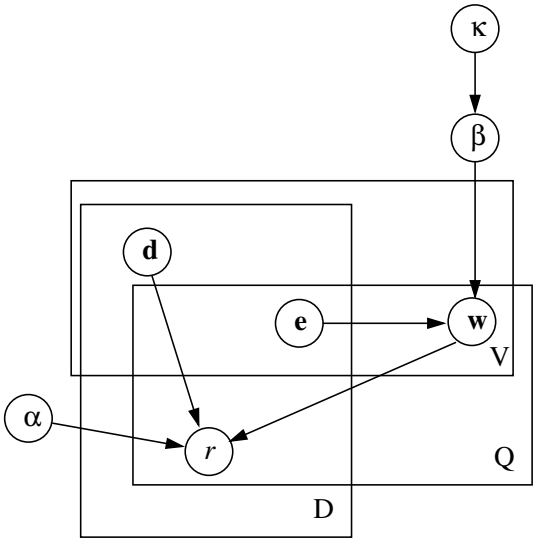


Figure 1. Graphical representation for the generative process for learning to learn. The plates are repeated the number of times shown in their bottom right corner; V is the number of terms in the vocabulary, Q of search tasks, and D of documents in the collection. The $\mathbf{d}$ is the TFIDF representation of the document and r is the 0–1 relevance of a document in a given search task. The $\mathbf{e}$ are the term-specific eye movement and text features and $\mathbf{w}$ is the query inferred from the eye movement features. The α and $\boldsymbol{\beta}$ are parameters shared by all search tasks, and κ is a prior parameter.

The query vector $\mathbf{w}$ of a search task is a vector in $\mathbb{R}^{|V|}$, where $|V|$ is the number of terms in the vocabulary. The entries in the query vector can be interpreted as the relative importance of the terms for the query. We assume that the entries in the query vector are normally distributed, with the mean depending on the eye movement features during the search task. We further assume that the mapping from the eye movements to the query weights is universal in the sense that the parameters of the mapping depend neither on the search task nor on the specific term.

The query weight w_{qt} for term t in the search task q depends on the viewed documents (in our experiments top-k with $k = 5$) in the search task, denoted by D_q, and on all term features in the search task, denoted collectively by $\mathbf{E}^q$. We assume that the dependency is

linear,

$$p(w_{qt}|\mathbf{E}^q, \boldsymbol{\beta}, \boldsymbol{\beta}') = N\left(\frac{1}{|D_q|}\sum_{i\in D_q}\left(\boldsymbol{\beta}^T\mathbf{e}_{it}I_{it} + \boldsymbol{\beta}'^T\mathbf{e}_{it}(1-I_{it})\right), \sigma^2\right), \quad (4)$$

where the indicator variable $I_{it} = 1$ if term t is viewed in document i, and 0 otherwise. If a term appears in the document but is not viewed only the textual features have non-zero values. If a word t does not appear in document i, the term's features are set to zero: $\mathbf{e}_{it} = \mathbf{0}$. The two regression coefficient vectors $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$, for viewed and unviewed terms, respectively, are common for all tasks. For notational simplicity we denote both of these parameters by $\boldsymbol{\beta}$ in the following.

For the probability of relevance r of a document in a search task q we assume the functional form of logistic regression. The probability is assumed to be a sigmoidal function of the dot product of the document TFIDF vector $\mathbf{d}$ and the query vector $\mathbf{w}_q$,

$$p(r|\mathbf{d}, \mathbf{w}_q, \alpha) = \frac{1}{1 + e^{-(\alpha + \mathbf{d}^T\mathbf{w}_q)}}. \quad (5)$$

The parameter α is common for all tasks.

We assume availability of a collection of background tasks for learning. Each background task is a search session where we know the relevances of the displayed documents, and have observed users' eye movements. The background tasks are used to learn the shared parameters.

The hyperparameters of the model, α and $\boldsymbol{\beta}$ on which the transfer learning is based, are estimated by maximizing the posterior from which all other parameters have been marginalized out. The logarithm of the marginalized posterior to be maximized is

$$\begin{aligned} L &= \sum_{i\in D_{BG}} \log p(r_i|\mathbf{d}_i, \alpha, \boldsymbol{\beta}) + \log p(\boldsymbol{\beta}|\kappa) \\ &= \sum_{i\in D_{BG}} \log \int p(r_i|\mathbf{d}_i, \mathbf{w}_{q(i)}, \alpha) p(\mathbf{w}_{q(i)}|\mathbf{E}^{q(i)}, \boldsymbol{\beta}) d\mathbf{w}_{q(i)} \\ &\quad + \log p(\boldsymbol{\beta}|\kappa) \\ &\approx \sum_{i\in D_{BG}} \log p(r_i|\mathbf{d}_i, \hat{\mathbf{w}}(q(i), \boldsymbol{\beta}), \alpha) + \log p(\boldsymbol{\beta}|\kappa), \end{aligned}$$

where $q(i)$ is the index of the background task during which document i was shown, D_{BG} is the set of all top-k documents in background tasks, and $\log p(\boldsymbol{\beta}|\kappa) = -\kappa/2\boldsymbol{\beta}^T\boldsymbol{\beta}$ is a Gaussian prior for $\boldsymbol{\beta}$. We assume a uniform prior for α. In the last step, the integral is approximated for computational reasons by a point estimate evaluated at the mode $\hat{\mathbf{w}}$ of $p(\mathbf{w}_{q(i)}|\mathbf{E}^{q(i)}, \boldsymbol{\beta})$; the entries of the mode are

$$\begin{aligned} \hat{w}_t(q, \boldsymbol{\beta}) &= \arg\max_{w_{qt}} p(w_{qt}|\mathbf{E}^q, \boldsymbol{\beta}) \\ &= \frac{1}{|D_q|}\sum_{i\in D_q}\left(\boldsymbol{\beta}^T\mathbf{e}_{it}I_{it} + \boldsymbol{\beta}'^T\mathbf{e}_{it}(1-I_{it})\right). \quad (6) \end{aligned}$$

Here $|D_q| = k$ is the number of viewed documents (top-k documents) in the search task q.

The learned values of the hyperparameters α and $\boldsymbol{\beta}$ are used to transfer knowledge from the old tasks to a new one. For the new task, we observe only eye movements on a small number of documents; we do not know the relevances of the documents as in the learning phase. The best prediction of relevance would result by integrating over the potential query vectors, but for computational reasons we again estimate the integral by the mode of $p(\mathbf{w}_{q(i)}|\mathbf{E}^{q(i)}, \boldsymbol{\beta})$. The mode can be interpreted as the estimated query vector $\mathbf{w}_{new}$, estimated using the equation (6), where the sum now is over the documents in the new task.

The ultimate goal is to find documents which are relevant to the new query. We rank the test set of unseen documents according to the probabilities (5) computed using $\mathbf{w}_{new}$.

3.2. SVM Model

The probabilistic model was motivated by the model (denoted by "SVM model") of Hardoon et al. (2007); we compared our model with the linear variant of the SVM model that performed almost as well as the best one in the original paper.

The main difference between our probabilistic model and the SVM model is that we have a full generative framework for all observations. A useful consequence of this is that we have a principled way for generating the search task specific implicit query $\mathbf{w}$ from the term-specific eye movements patterns. In the SVM model this step was somewhat ad hoc; the ground truth was obtained by classifying relevant vs. irrelevant documents in the training data. The regressor then tried to predict the SVM discriminator weights from eye tracking data. There is no guarantee that the discriminator weights used by the SVM are optimal, or even always good, targets taking into account uncertainties stemming from the noisy eye movements.

Note that in this paper we used fairly simple computational approximations for generating the task-specific implicit queries $\mathbf{w}$. The fact that the results are still good gives the model further support; if necessary,

more accurate approximations can be developed later.

3.3. Connection to Transfer Learning and Meta-Learning

In our problem we need to learn to learn an implicit query from the text of the document and gaze pattern. This needs to be done for search topics unseen in the training phase. Each search topic has a hidden representation that we have to learn, namely the query vector $\mathbf{w}$. We have additionally introduced hyperparameters, namely the α and $\boldsymbol{\beta}$, that are shared across all search tasks. The shared parameters contain information that is needed to learn the search task-specific query vector.

Our modeling assumption is that there exists information in the gaze pattern that is independent of the actual semantic content of the words and of the specific query. We encode independence of the specific query by introducing the parameters α and $\boldsymbol{\beta}$ that are shared across the search tasks. Independence of the semantic content is achieved by constructing the model so that the query vector $\mathbf{w}$ depends only on text and eye movement features associated with a specific term, but not on the semantic content of the terms. In other words, the model is invariant with respect to any permutation of term labels.

The learning process, described in Section 3.1, can be interpreted to have two phases: in the first phase the parameters α and $\boldsymbol{\beta}$ that are shared across all search tasks are learned using several background search tasks. In the second "on-line" phase a search task-specific query vector $\mathbf{w}$ (for a previously unseen search task) is estimated using the shared parameters.

Our problem is related to transfer learning and meta-learning (Thrun, 1996; Baxter, 2004; Caruana, 1997; Ando & Zhang, 2005; Thrun, 1998; Pratt & Thrun, 1997; Vilalta & Drissi, 2002; Giraud-Carrier et al., 2004). Transfer learning and meta learning utilize data from other "similar" learning tasks and from multiple applications of the learning system. For example, learning to recognize objects in cartoons might help to recognize objects in photographs (Elidan et al., 2006); or in our case, learning to infer query vectors in search tasks helps in learning a query vector in a yet unseen search task. We can think that in our case the inductive bias extracted is parametrized by α and $\boldsymbol{\beta}$ and fixed when these parameters are learned. When we observe a new search task we can then use the information coded in α and $\boldsymbol{\beta}$ to learn the task-specific query vector $\mathbf{w}$ that can finally be used to predict the relevance of a given document.

4. Experiments

We conducted small scale eye tracking experiments to validate the proposed model. The experiments were designed to simulate the common case where some keywords are available but they are not a sufficient description of the interests.

4.1. Search Tasks

We constructed 13 search tasks for text documents (see Table 1). The search tasks were chosen prior to doing any experiments. The criteria for selecting the search tasks were that there should be several relevant documents for each task and, furthermore, that the original query should also suggest irrelevant documents. That is, there should be irrelevant documents which are ranked quite high. This is why the search terms were purposefully ambiguous.

For example, in search task number 3 the task was to find information about "ancient Rome." The user would be instructed to find documents that would tell about ancient Rome. The textual search query, forced by us, was "Rome." As a result, the search results included articles also, for example, of modern Rome. Our purpose was to see whether the gaze pattern could be used to infer a new query. Intuitively, the query vector $\mathbf{w}$ inferred from the eye movements could include with positive weight terms related to ancient Rome, such as "ancient", "Caesar" and "Carthage"; and possibly with negative weight terms related to the modern times, such as "airport" or "president".

The document corpus consisted of articles downloaded from Wikipedia. Only the lead section of the documents before the first section header was shown to the user and used in the experiments. For each search task, we selected 10 documents having the highest BM25 score.

4.2. Experimental Procedure

There were three participants in the experiments, one female and two males. The test subjects were voluntary under and post-graduate researchers (the authors were not included).

The participants were asked to search for documents of a given topic using a web search engine. They were advised to act as if they were collecting the relevant documents of the topic for later reading, that is, they were supposed to stop reading the document once they had determined whether the document was relevant. Each task was started by clicking a search button which submitted the pre-entered search term to a custom server.

Table 1. The search tasks and the given search terms.

Task number	Desired topic	Search term
1	American football	football
2	Alternative medicine	medicine
3	Ancient Rome	Rome
4	Adhesive tape	tape
5	Environmental conservation	conservation
6	Seal (marine mammal)	seal
7	Extra-solar planets	planets
8	Visual nervous system	vision
9	Internet forums	forum
10	Marketing strategies	strategies
11	National libraries	libraries
12	British Royal Navy	navy
13	Space shuttles	shuttle

Our search engine did not return a list of the most relevant documents as search engines usually do. Instead, it returned the most relevant document directly. In the bottom of each document there were two links which were used for marking the document as either relevant or not relevant. Clicking one of these links retrieved the next document. All other links were removed from the documents. The search engine returned the documents in the order determined by the BM25 algorithm. Each search session included 10 documents. After finishing one session the test subject was automatically given a topic and search terms for the next task.

The relevance judgements given by the users were used as ground truth during the training phase of the model. In testing phase they were used to validate the results.

During the search tasks the users' eye movements were recorded with a Tobii 1750 eye tracker. Tobii tracks gaze location by measuring the reflection pattern on the cornea of eye. It does not require wearing a helmet or a headrest. The users were sitting 60 cm away from a 17 inch computer screen. The system was calibrated once in the beginning of the experiment.

4.3. Term Features

The eye tracker and the browser recorded the sequence of fixations; it was then transmitted to the Wikipedia document server when the user clicked any link. A part of the gaze trajectory was considered a fixation if the gaze stayed inside a 30 pixel square (about 0.6 visual angle) for more than 100 ms. Fixations that appeared outside the bounding boxes of vocabulary words were ignored. The vocabulary consisted of stemmed words, with stop words removed. The size of the vocabulary was 3030 words.

For each term t, we extracted 19 eye movement features and 3 text features, denoted collectively by $\mathbf{e}_t$. We used the same eye movement features, such as number of fixations, absolute, relative and mean fixation durations, used by Hardoon et al. (2007) as well. The 3 text features were independent of the actual search task; they are the number of characters in the word, the relative position of the word in the document, and the inverse document frequency of the word.

4.4. Combination of Textual and Eye Movement Based Searches

We show that a simple combination of our eye movement-based ranking and a ranking by a state-of-the-art textual IR algorithm has higher precision than the textual search alone. For the textual search we use BM25, a well-known bag-of-words ranking function.

We measured the eye movements while the users were reading top-5 documents as returned by BM25. The documents below the rank 5 were used for testing. We ranked the test documents with our method thus getting a second ranking in addition to the original BM25 ranking. We combined the two rankings by reordering the documents according to the weighted average (3).

To compare the original BM25 ranking and the combined ranking, we compare their average precision, a common measure for evaluating search results. It is computed as the average of the precisions at positive rankings: $\frac{1}{R}\sum_{i=1}^{R}\frac{i}{r_i}$, where R is the total number of relevant documents, and the r_i are the rankings of the positive documents such that $r_i < r_{i+1}$. The best possible average precision is one, which corresponds to all relevant documents being ranked in the first positions.

We learn the query vector for each search task by leaving the data for that task out and using the remaining tasks as background tasks in the first phase of the training, as was discussed in Section 3.1. In the "online" phase we use the top-5 documents from the left-out task to infer the query vector.

We combine BM25 and the probabilistic model by using equation (3). For that purpose we need to decide a value for the weighting factor γ. We compare the mean average precision of the plain BM25 and the combination of BM25 and the proposed probabilistic model for documents that are ranked 6–10 in each background task for several discrete values of γ, and for each task select the γ value that has the best mean improvement in average precision. We use the average of these task-specific best values of γ in order to obtain a common value of $\gamma = 0.2$, which is then the value used in all of the experiments. The value $\kappa = 1$ for the prior pa-

rameter is selected in an analogous fashion at the same time.

The results are shown in Table 2. On the test set, in 9 search tasks out of 13 the mean average precision of the combined ranking across test subjects outperformed the baseline BM25 ranking. The difference in average precision is statistically significant ($p = 0.047$, one-tailed Wilcoxon Signed Rank Test).

The worst performance is shown by task 6 in which the only relevant document in ranks 6–10 (according to the users' relevance judgement) was moved from being 6th to 10th, thus reducing the average precision for documents in ranks 6–10 from unity of the BM25 baseline down to 0.29. In all the other tasks the performance either improved clearly, or for some degraded slightly.

In order to examine the relative contributions of the eye movement and textual features we compared two probabilistic models, one using just the text features (of all words, irrespective of whether they were looked at or not) and one using all the features. The mean average precision of the latter was 6.3 percentage units higher but for this amount of data the difference was not statistically significant ($p = 0.22$).

4.5. Comparison to the SVM Model

We compared the performance of the combination of BM25 and the probabilistic model to an analogous combination of BM25 and the SVM (of Section 3.2). The SVM model is trained using the eye movements on the training documents, as described in (Hardoon et al., 2007), and the resulting ranking is combined to the BM25 ranking identically as was done above for the probabilistic model.

The mean average precision of the combination of BM25 and the SVM is worse than the average precision of BM25 for all values of the weighting parameter γ. The bad performance of the SVM model here is probably due to the fact that the ground truth for the query vectors, estimated with SVM from the very small data sets, is likely to be very noisy. It is also possible that the probabilistic model is otherwise better suited for the relatively small training set sizes.

5. Discussion

We introduced a generative model of how the relevance of documents is related to the viewing patterns of people during a search task. The model is trained in two phases. In the first phase, the hyperparameters that are independent of the search topic are learned. In the second "on-line" phase, the learned parameters are used to infer an implicit query from the gaze patterns during a new search session.

The system is realistically applicable; we have indeed implemented it using a standard web browser that has been modified to record and transmit the information about the gaze patterns to the web server. The document corpus used in the experiments consisted of abstracts of Wikipedia articles.

Our results imply that the performance of the method correlates with the search task; in particular, there are some tasks for which the methods seems to perform quite badly although on average it clearly improves the results. It needs to be investigated more carefully later, which kinds of search tasks the eye movements are helpful in, and whether different types of models are useful for different kinds of tasks.

Acknowledgements

KP belongs to the Finnish Centre of Excellence in Algorithmic Data Analysis, and AA and SK to Finnish Centre of Excellence in Adaptive Informatics Research. We would like to thank David Hardoon and John Shawe-Taylor for valuable input, Wray Buntine for the Wikipedia data and Janne Kataja for the Gazillion browser. This work was supported in part by the PASCAL2 Network of Excellence of the European Community.

References

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*, 1817–1853.

Aslam, J. A., & Montague, M. (2001). Models for metasearch. *SIGIR '01: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 276–284). New York, NY: ACM.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Reading, MA: Addison-Wesley.

Baxter, J. (2004). A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, *28*, 7–39.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Claypool, M., Le, P., Wased, M., & Brown, D. (2001). Implicit interest indicators. *IUI '01: Proceedings*

Table 2. Mean improvement in the average precision between plain BM25 and the combination of BM25 and the proposed probabilistic model (top row, $\Delta AVGPREC_{Prob}$). The mean average precision of the plain BM25 which shows the baseline performance on purely textual searches is shown in the bottom row, titled $AVGPREC_{BM25}$. The probabilistic model outperforms the BM25 baseline model ($p = 0.047$, one-tailed Wilcoxon Signed Rank Test), while the performance of the SVM model is comparable to the baseline. Larger values are better.

	Search task												
	1	2	3	4	5	6	7	8	9	10	11	12	13
$\Delta AVGPREC_{Prob}$	0.17	0.28	0.53	0.18	0.22	-0.71	-0.03	0.16	0.04	0.14	-0.05	-0.08	0.34
$AVGPREC_{BM25}$	0.50	0.61	0.13	0.50	0.50	1.00	0.91	0.50	0.78	0.70	0.71	0.50	0.42

of the International Conference on Intelligent User Interfaces (pp. 33–40). New York, NY: ACM Press.

Cohen, W. W., Schapire, R. E., & Singer, Y. (1998). Learning to order things. *NIPS '97: Proceedings of the Conference on Advances in Neural Information Processing Systems 10* (pp. 451–457). Cambridge, MA: MIT Press.

Elidan, G., Heitz, G., & Koller, D. (2006). Learning object shape: From drawings to images. *CVPR'06: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2*, 2064–2071.

Fox, S., Karnawat, K., Mydland, M., Dumais, S., & White, T. (2005). Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems*, *23*, 147–168.

Giraud-Carrier, C., Vilalta, R., & Brazdil, P. (2004). Special issue on meta-learning. *Machine Learning*, *54*, 187–312.

Hardoon, D. R., Ajanki, A., Puolamäki, K., Shawe-Taylor, J., & Kaski, S. (2007). Information retrieval by inferring implicit queries from eye movements. *AISTATS '07: The International Conference on Artificial Intelligence and Statistics*. Electronic proceedings at www.stat.umn.edu/∼aistat/proceedings/start.htm.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., & Gay, G. (2005). Accurately interpreting clickthrough data as implicit feedback. *SIGIR '05: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 154–161). New York, NY: ACM.

Joachims, T., & Radlinski, F. (2007). Search engines that learn from implicit feedback. *IEEE Computer*, *40*, 34–40.

Kelly, D., & Teevan, J. (2003). Implicit feedback for inferring user preference: a bibliography. *ACM SIGIR Forum*, *37*, 18–28.

Pratt, L., & Thrun, S. (1997). Second special issue on inductive transfer. *Machine Learning*, *28*, 5–130.

Puolamäki, K., Salojärvi, J., Savia, E., Simola, J., & Kaski, S. (2005). Combining eye movements and collaborative filtering for proactive information retrieval. *SIGIR '05: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 146–153). New York, NY: ACM.

Robertson, S., & Zaragoza, H. (2007). The probabilistic relevance model: BM25 and beyond. Tutorial at the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07), http://www.sigir2007.org/tutorial2d.html.

Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. *SIGIR '94: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 232–241). New York, NY: ACM.

Robertson, S. E., & Walker, S. (1999). Okapi/Keenbow at TREC-8. *Proceedings of the Eighth Text Retrieval Conference TREC-8* (pp. 151–162). Washington, DC: GPO.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? *In Advances in Neural Information Processing Systems 8* (pp. 640–646). Cambridge, MA: MIT Press.

Thrun, S. (1998). *Learning to learn*, chapter Lifelong learning algorithms, 181–209. Norwell, MA: Kluwer Academic Publishers.

Turpin, A., & Scholer, F. (2006). User performance versus precision measures for simple search tasks. *SIGIR '06: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 11–18). New York, NY: ACM.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*, 77–95.

Multi-Task Compressive Sensing with Dirichlet Process Priors

Yuting Qi YUTING@EE.DUKE.EDU
Dehong Liu LIUDH@EE.DUKE.EDU
Department of Electrical and Computer Engineering, Duke University, Durham, NC, 27708 USA

David Dunson DUNSON@STAT.DUKE.EDU
Department of Statistical Science, Duke University, Durham, NC, 27708 USA

Lawrence Carin LCARIN@EE.DUKE.EDU
Department of Electrical and Computer Engineering, Duke University, Durham, NC, 27708 USA

Abstract

Compressive sensing (CS) is an emerging £eld that, under appropriate conditions, can signi£cantly reduce the number of measurements required for a given signal. In many applications, one is interested in multiple signals that may be measured in multiple CS-type measurements, where here each signal corresponds to a sensing "task". In this paper we propose a novel multi-task compressive sensing framework based on a Bayesian formalism, where a Dirichlet process (DP) prior is employed, yielding a principled means of simultaneously inferring the appropriate sharing mechanisms as well as CS inversion for each task. A variational Bayesian (VB) inference algorithm is employed to estimate the full posterior on the model parameters.

1. Introduction

Over the last two decades researchers have considered sparse signal representations in terms of orthonormal basis functions (*e.g.*, the wavelet transform). For example, consider an m-dimensional real-valued signal $\boldsymbol{u}$ and assume an $m \quad m$ orthonormal basis matrix $\boldsymbol{\Psi}$; we may then express $\boldsymbol{u} = \boldsymbol{\Psi}\boldsymbol{\theta}$, where $\boldsymbol{\theta}$ is an m-dimensional column vector of weighting coef£cients. For most natural signals there exists an orthonormal basis $\boldsymbol{\Psi}$ such that $\boldsymbol{\theta}$ is sparse. Consider now an approximation to $\boldsymbol{u}$, $\hat{\boldsymbol{u}} = \boldsymbol{\Psi}\hat{\boldsymbol{\theta}}$, where $\hat{\boldsymbol{\theta}}$ approximates $\boldsymbol{\theta}$ by retaining the largest N coef£cients and setting the remaining $m \quad N$ coef£cients to zero; due to the aforementioned sparseness properties, $||\boldsymbol{u} \quad \hat{\boldsymbol{u}}||^2$ is typically very small even for $N \ll m$. Conventional techniques require one to measure the m-dimensional signal $\boldsymbol{u}$ but £nally discard $m \quad N$ coef£cients (Charilaos, 1999). This sample-then-compress framework is often wasteful since the signal acquisition is potentially expensive, and only a small amount of data N is eventually required for the accurate approximation $\hat{\boldsymbol{u}}$. One may therefore consider the following fundamental question: Is it possible to directly measure the informative part of the signal? Recent research in the £eld of compressive sensing shows that this is indeed possible (Candes, 2006)(Donoho, 2006).

Exploiting the same sparseness properties of $\boldsymbol{u}$ employed in transform coding ($\boldsymbol{u} = \boldsymbol{\Psi}\boldsymbol{\theta}$ with $\boldsymbol{\theta}$ sparse), in compressive sensing one measures $\boldsymbol{v} = \boldsymbol{\Phi}\boldsymbol{\theta}$, where $\boldsymbol{v}$ is an n-dimensional vector with $n < m$, and $\boldsymbol{\Phi}$ is the $n \quad m$ sensing matrix. There are several ways in which $\boldsymbol{\Phi}$ may be constituted, with the reader referred to (Donoho, 2006) for details. In most cases $\boldsymbol{\Phi}$ is represented as $\boldsymbol{\Phi} = \mathbf{T}\boldsymbol{\Psi}$, where $\mathbf{T}$ is an $n \quad m$ matrix with components constituted randomly (Tsaig & Donoho, 2006); hence, the CS measurements correspond to projections of $\boldsymbol{u}$ with the rows of $\mathbf{T}$: $\boldsymbol{v} = \mathbf{T}\boldsymbol{u} = \mathbf{T}\boldsymbol{\Psi}\boldsymbol{\theta} = \boldsymbol{\Phi}\boldsymbol{\theta}$, which is an underdetermined problem. Assuming the signal $\boldsymbol{u}$ is N-sparse in $\boldsymbol{\Psi}$, implying that the coef£cients $\boldsymbol{\theta}$ only have N nonzero values (Candes, 2006) (Donoho, 2006), Candès, Romberg and Tao in (Candes et al., 2006) show that, with overwhelming probability, $\boldsymbol{\theta}$ (and hence $\boldsymbol{u}$) is recovered via

$$\min ||\boldsymbol{\theta}||_{l_1}, \quad \text{s.t.}, \quad \boldsymbol{v} = \boldsymbol{\Phi}\boldsymbol{\theta}, \tag{1}$$

if the number of CS measurements $n > C \ N \ \log m$ (C is a small constant); if N is small (*i.e.*, if u is highly compressible in the basis $\boldsymbol{\Psi}$) then $n \ll m$. In practice the signal $\boldsymbol{u}$ is not exactly sparse, but a large number of coef£cients in the basis $\boldsymbol{\Psi}$ may be discarded with minimal error in reconstructing $\boldsymbol{u}$; in this practical case the CS framework has also been shown to operate effectively.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

The problem in (1) may be solved by linear programming (S. Chen & Saunders, 1999) and greedy algorithms (Tropp & Gilbert, 2005) (Donoho et al., 2006). A Bayesian compressive sensing (BCS) methodology is proposed in (Ji et al., 2007b), by posing the CS inversion problem as a linear-regression problem with a sparseness prior on the regression weights $\boldsymbol{\theta}$. One advantage of BCS is that this framework may be extended to multi-task compressive sensing (Ji et al., 2007a), in which each CS measurement $\boldsymbol{v}_i = \boldsymbol{\Phi}_i \boldsymbol{\theta}_i$ represents a sensing "task" and the objective is to jointly invert for all $\{\boldsymbol{\theta}_i\}_{i=1,M}$, through an appropriate sharing of information between the M data collections. In multi-task CS, one may potentially reduce the number of measurements required for each task by exploiting the statistical relationships among the tasks, for example, "Distributed Compressed Sensing" (DCS) (Baron et al., 2005), an empirical Bayesian strategy "Simultaneous Sparse Approximation" in (Wipf & Rao, 2007), and a hierarchical Bayesian model for multi-task CS (Ji et al., 2007a). However, these multi-task algorithms assume all tasks are appropriate for sharing, which may not be true in many practical applications. In this paper we introduce a Dirichlet process (DP) prior (West et al., 1994) to the hierarchical BCS model, which can simultaneously perform the inversion of the underlying signals *and* infer the appropriate sharing/clustering structure across the M tasks.

As detailed below, an important property of DP for the work presented here is that it provides a tool for semi-parametric clustering (*i.e.*, the number of clusters need not be set in advance). The DP-based hierarchical model is employed to realize the desired property of simultaneously clustering and CS inversion of the M measurements $\{\boldsymbol{v}_i\}_{i=1,M}$. A variational Bayes (Blei & Jordan, 2004) inference algorithm is considered, yielding a full posterior over the model parameters $\boldsymbol{\theta}_i$.

2. Multi-Task CS Modeling with DP Priors

2.1. Multi-Task CS Formulation for Global Sharing

Let $\boldsymbol{v}_i$ represent the CS measurements associated with task i, and assume a total of M tasks. The i-th CS measurement may be represented as

$$\boldsymbol{v}_i = \boldsymbol{\Phi}_i \boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i, \tag{2}$$

where the CS measurements $\boldsymbol{v}_i$ are characterized by an n_i-dimensional real vector, the sensing matrix $\boldsymbol{\Phi}_i$ corresponding to task i is of size $n_i \quad m$, and $\boldsymbol{\theta}_i$ is the set of (sparse) transform coef£cients associated with task i. The j^{th} coef£cient of $\boldsymbol{\theta}_i$ is denoted $_{i,j}$. The residual error vector $\boldsymbol{\epsilon}_i \in \mathbb{R}^{n_i}$ is modeled as n_i *i.i.d.* draws from a zero-mean Gaussian distribution with an unknown precision $_0$ (variance $1/\ _0$); the residual corresponds to the error imposed by setting the small transform coef£cients exactly to zero when performing the CS inversion.

We impose a hierarchical sparseness prior on the parameters $\boldsymbol{\theta}_i$, the lower level of which is

$$p(\boldsymbol{\theta}_i|\boldsymbol{\alpha}_i) = \prod_{j=1}^{m} \mathcal{N}(\ _{i,j}|0,\ \ ^{-1}_{i,j}), \tag{3}$$

where $_{i,j}$ is the j^{th} component of the vector $\boldsymbol{\alpha}_i$. To impose sparseness, on a layer above a Gamma hyperprior is employed independently on the precisions $_{i,j}$. The likelihood function for the parameters $\boldsymbol{\theta}_i$ and $_0$, given the CS measurements $\boldsymbol{v}_i$, may be expressed as

$$p(\boldsymbol{v}_i|\boldsymbol{\theta}_i,\ \ _0) = (\frac{2}{_0})^{-\frac{n_i}{2}} \exp(\ \ \frac{_0}{2}\|\boldsymbol{v}_i \quad \boldsymbol{\Phi}_i\boldsymbol{\theta}_i\|_2^2). \tag{4}$$

Concerning the aforementioned hyperprior, for the multi-task CS model proposed in (Ji et al., 2007a), the parameters $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}$, for $i = 1, \quad , M$, and $\boldsymbol{\alpha} \quad \prod_{j=1}^{m} Ga(c,d)$. In this framework the CS data from all M tasks are used to jointly infer the hyper-parameters $\boldsymbol{\alpha}$ (global processing). However, the assumption in such a setting is that it is appropriate to employ all of the M tasks jointly to infer the hyper-parameters. One may envision problems for which the M tasks may be clustered into several sets of tasks (with the union of these sets constituting the M tasks), and data sharing may only be appropriate within each cluster. Through use of the Dirichlet process (DP) (Escobar & West, 1995) employed as the prior over $\boldsymbol{\alpha}_i$, we simultaneously cluster the multi-task CS data, and within each cluster the CS inversion is performed jointly. Consequently, we no longer need assume that all CS data from the M tasks are appropriate for sharing.

2.2. Dirichlet Process for Clustered Sharing

The Dirichlet process, denoted as $DP(\lambda, G_0)$, is a measure on measures, and is parameterized by a positive scaling parameter λ and the base distribution G_0. Assume we have $\{\boldsymbol{\alpha}_i\}_{i=1,M}$ and each $\boldsymbol{\alpha}_i$ is drawn identically from G, and G itself is a random measure drawn from a Dirichlet process,

$$\begin{aligned} \boldsymbol{\alpha}_i|G \quad &^{iid} \quad G, \quad i = 1, \quad , M, \\ G \quad &\quad DP(\lambda, G_0), \end{aligned} \tag{5}$$

where G_0 is a non-atomic base measure.

Sethuraman (Sethuraman, 1994) provides an explicit characterization of G in terms of a stick-breaking construction. Consider two in£nite collections of independent random variables $_k$ and $\boldsymbol{\alpha}_k^*$, $k = 1, 2, \quad , \infty$, where the $_k$ are drawn *i.i.d.* from a Beta distribution, denoted $Beta(1, \lambda)$, and the $\boldsymbol{\alpha}_k^*$ are drawn *i.i.d.* from the base distribution G_0.

The stick-breaking representation of G is then de£ned as

$$G = \sum_{k=1}^{\infty} w_k \delta_{\boldsymbol{\alpha}_k^*}, \quad \text{with} \tag{6}$$

$$w_k = {}_k \prod_{i=1}^{k-1} (1 \quad {}_i), \tag{7}$$

where ${}_k|\lambda \overset{iid}{} Beta(1,\lambda)$ and $\boldsymbol{\alpha}_k^*|G_0 \overset{iid}{} G_0$. This representation makes explicit that the random measure G is discrete with probability one and the support of G consists of an in£nite set of atoms located at $\boldsymbol{\alpha}_k^*$, drawn independently from G_0. The mixing weights w_k for atom $\boldsymbol{\alpha}_k^*$ are given by successively breaking a unit length "stick" into an in£nite number of pieces, with $0 \le w_k \le 1$ and $\sum_{k=1}^{\infty} w_k = 1$.

2.3. Multi-Task CS with DP Priors

We employ a DP prior with stick-breaking representation for $\boldsymbol{\alpha}_i$ in the model in (3), which assumes that $\boldsymbol{\alpha}_i|G \quad G$ and $G = \sum_{k=1}^{\infty} w_k \delta_{\boldsymbol{\alpha}_k^*}$. The base distribution G_0 corresponds to the sparseness promoting representation discussed in Sec 2.1. To facilitate posterior computation we introduce an indicator variable z_i with $z_i = k$ indicating $\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_k^*$. Therefore the DP multi-task CS model is expressed as

$$\begin{aligned}
\boldsymbol{v}_i|\boldsymbol{\theta}_i, \ {}_0 \quad & \mathcal{N}(\boldsymbol{\Phi}_i\boldsymbol{\theta}_i, \ {}_0^{-1} I), \\
{}_{i,j}|z_i, \{\boldsymbol{\alpha}_k^*\}_{k=1,K} \quad & \mathcal{N}(0, \ {}_{z_i,j}^{*\,-1}), \\
z_i|\{w_k\}_{k=1,K} \quad & \overset{iid}{} \ Multinomial(\{w_k\}_{k=1,K}), \\
w_k \quad = \quad & {}_k \prod_{l=1}^{k-1} (1 \quad {}_l), \\
{}_k \quad & \overset{iid}{} \ Beta(1,\lambda), \\
\lambda|e,f \quad & Ga(e,f), \\
\boldsymbol{\alpha}_k^*|c,d \quad & \overset{iid}{} \ \prod_{j=1}^{m} Ga(c,d), \\
{}_0 \quad & Ga(a,b),
\end{aligned} \tag{8}$$

where $i = 1, \quad , M$, $j = 1, \quad , m$, $k = 1, \quad , K$, $1 \le K \le \infty$, and ${}_{i,j}$ is the j-th element of $\boldsymbol{\alpha}_i$. For convenience, we denote the model in (8) as DP-MT CS. In practice K is chosen as a relatively large integer (*e.g.*, $K = M$ if M is relatively large) which yields a negligible difference compared to the true DP (Ishwaran & James, 2001), while making the computation practical.

The choice of G_0 here is consistent with the sparseness-promoting hierarchical prior discussed in Section II-A. Consider task i and assume $\boldsymbol{\alpha}_i$ takes value $\boldsymbol{\alpha}_k^*$; the prior distribution over $\boldsymbol{\theta}_i$ is then

$$p(\boldsymbol{\theta}_i|c,d) = \prod_{j=1}^{m} \int \mathcal{N}(\ {}_{i,j}|0, \ {}_{k,j}^{*\,-1}) Ga(\ {}_{k,j}^{*}|c,d) d \ {}_{k,j}^{*}. \tag{9}$$

Equation (9) is a type of automatic relevance determination (ARD) prior which enforces the sparsity over $\boldsymbol{\theta}_i$ (Tipping, 2001). We usually set c and d very close to zero (*e.g.*, 10^{-4}) to make a broad prior over $\boldsymbol{\alpha}_k^*$, which allows the posteriors on many of the elements of $\boldsymbol{\alpha}_k^*$ to concentrate at very large values, consequently the posteriors on the associated elements of $\boldsymbol{\theta}_i$ concentrate at zero, and therefore the sparseness of $\boldsymbol{\theta}_i$ is achieved (MacKay, 1994) (Neal, 1996). Since these posteriors have "heavy tails" compared to a Gaussian distribution, they allow for more robust shrinkage and borrowing of information. Similarly, hyper-parameters a, b, e, and f are all set to a small value to have a non-informative prior over ${}_0$ and λ respectively.

3. Variational Bayesian Inference

One may perform inference via MCMC (Gilks et al., 1996), however this requires vast computational resources and MCMC convergence is often dif£cult to diagnose (Gilks et al., 1996). Variational Bayes inference is therefore introduced as a relatively ef£cient method for approximating the posterior. From Bayes' rule, we have

$$p(\mathbf{H}|\mathbf{V},\Upsilon) = \frac{p(\mathbf{V}|\mathbf{H})p(\mathbf{H}|\Upsilon)}{\int p(\mathbf{V}|\mathbf{H})p(\mathbf{H}|\Upsilon)d\mathbf{H}}, \tag{10}$$

where $\mathbf{V} = \{\boldsymbol{v}_i\}_{i=1,M}$ are CS measurements from M CS tasks, $\mathbf{H} = \{ \ {}_0, \lambda, \boldsymbol{\pi}, \{z_i\}_{i=1,M}, \{\boldsymbol{\theta}_i\}_{i=1,M}, \{\boldsymbol{\alpha}_k^*\}_{k=1,K}\}$ are hidden variables (with $\boldsymbol{\pi} = \{ \ {}_k\}_{k=1,K}$) and $\Upsilon = \{a,b,c,d,e,f\}$ are known hyper-parameters. The integration in the denominator of (10), called the *marginal likelihood*, or *"evidence"* (Beal, 2003), is generally intractable to compute analytically. Instead of directly estimating $p(\mathbf{H}|\mathbf{V},\Upsilon)$, variational methods seek a distribution $q(\mathbf{H})$ to approximate the true posterior distribution $p(\mathbf{H}|\mathbf{V},\Upsilon)$. Consider the log marginal likelihood

$$\log p(\mathbf{V}|\Upsilon) = \mathcal{F}(q(\mathbf{H})) + \mathcal{D}_{KL}(q(\mathbf{H})||p(\mathbf{H}|\mathbf{V},\Upsilon)), \tag{11}$$

where

$$\mathcal{F}(q(\mathbf{H})) = \int q(\mathbf{H}) \log \frac{p(\mathbf{V}|\mathbf{H}|,\Upsilon)p(\mathbf{H},\Upsilon)}{q(\mathbf{H})} d\mathbf{H}, \tag{12}$$

and $\mathcal{D}_{KL}(q(\mathbf{H})||p(\mathbf{H}|\mathbf{V},\Upsilon))$ is the KL divergence between $q(\mathbf{H})$ and $p(\mathbf{H}|\mathbf{V},\Upsilon)$. The approximation of $p(\mathbf{H}|\mathbf{V},\Upsilon)$ using $q(\mathbf{H})$ can be achieved by maximizing $\mathcal{F}(q(\mathbf{H}))$, which forms a strict lower bound on $\log p(\mathbf{V}|\Upsilon)$. In this way estimation of $q(\mathbf{H})$ may be made computationally tractable. In particular, for computational convenience,

$q(\mathbf{H})$ is expressed in a factorized form, with the same functional form as the priors $p(\mathbf{H}|\Upsilon)$. For the model in (8), we assume

$$q(\mathbf{H}) = q(\ \ _0)q(\lambda)q(\boldsymbol{\pi})\prod_{i=1}^{M} q(z_i)\prod_{i=1}^{M} q(\boldsymbol{\theta}_i)\prod_{k=1}^{K} q(\boldsymbol{\alpha}_k^*), \tag{13}$$

where $q(\ \ _0)$ $Ga(\tilde{a},\tilde{b})$, $q(\lambda)$ $Ga(\tilde{e},\tilde{f})$, $q(\boldsymbol{\pi})$ $\prod_{k=1}^{K-1} Beta(\ _{1k},\ _{2k})$, $q(z_i)$ $Multinomial(\boldsymbol{w})$, $q(\boldsymbol{\theta}_i)$ $\mathcal{N}(\ _i,\boldsymbol{\Gamma}_i)$, $q(\boldsymbol{\alpha}_k^*)$ $\prod_{j=1}^{m} Ga(\ ^*_{k,j}|\tilde{c}_{k,j},\tilde{d}_{k,j})$, with $\boldsymbol{w} = \{w_k\}_{k=1,K}$.

By substituting (13) and (8) into (12), the lower bound $\mathcal{F}(q)$ is readily obtained. The optimization of the lower bound $\mathcal{F}(q)$ is realized by taking functional derivatives with respect to each of the $q(\ )$ distributions while £xing the other q distributions, and setting $\partial\mathcal{F}(q)/\partial q(\ ) = 0$ to £nd the distribution $q(\ )$ that increases $\mathcal{F}$ (Beal, 2003). The update equations for the variational posteriors are summarized in the Appendix. The convergence of the algorithm is monitored by the increase of the lower bound $\mathcal{F}$. One practical issue of the variational Bayesian inference is that the VB algorithm converges to a local maximum of the lower bound of the marginal log-likelihood since the true posterior usually is multi-modal. Therefore the average of multiple runs of the algorithm from different starting points may avoid this issue and yield better performance.

4. Experimental Results

4.1. Synthetic data

In the £rst set of examples we consider synthesized data to examine the sharing mechanisms associated with the DP-MT CS inversion. In the £rst example we generate data with 10 underlying clusters. Figure 1 shows ten "templates", each corresponding to a 256-length signal, with 30 non-zero components (the values of those non-zero components are randomly drawn from $\mathcal{N}(0,1)$). The non-zero locations are chosen randomly for each template such that the correlation between these sparse templates is zero. For each template, £ve sparse signals (each with 256 samples) are generated by randomly selecting three non-zero elements from the associated template and setting the coef£cients to zero, and three zero-amplitude points in the template are randomly now set to be non-zero (each of these three non-zero values again drawn from $\mathcal{N}(0,1)$). In this manner the sparseness properties of the £ve signals generated from a given template are highly related, and the ten clusters of sparse signals have distinct sparseness properties. For each sparse signal a set of CS random projections are performed, with the components of each projection vector drawn randomly from $\mathcal{N}(0,1)$(Donoho, 2006). The reconstruction error is de£ned as $||\hat{\boldsymbol{u}} \quad \boldsymbol{u}||_2/||\boldsymbol{u}||_2$, where $\hat{\boldsymbol{u}}$ is the recovered signal and $\boldsymbol{u}$ is the original one.

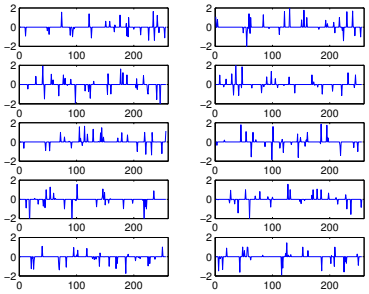


Figure 1. Ten template signals for 10-cluster case.

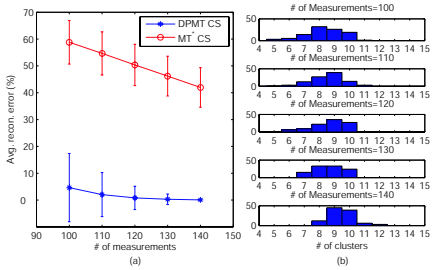


Figure 2. Multi-task CS inversion error (%) for DP-MT and MT CS for the ten-cluster case. (a) Reconstruction errors, (b) histograms of the number of clusters yielded by DP-MT.

Figure 2 shows the reconstruction errors of the CS inversion by DP-MT CS as well as the global-sharing MT CS discussed in Sec 2.1 (denoted as MT* CS for simplicity), as a function of the number of CS measurements. Both CS algorithms are based on the VB DP-MT algorithm described in Sec 3, however for MT*, we set $\kappa_{i,1} = 1$, and $\kappa_{i,k} = 0$ for $k > 1$ for all tasks and £x the values of $\kappa_{i,k}$ in each iteration without update. The experiment was run 100 times (with 100 different random generations of random projection as well as initial membership), and the error bars in Figure 2 represent the standard deviation about the mean. From Figure 2 the advantage of the DP-based formulation is evident. In Figure 2 we also present histograms for the number of different clusters inferred by the DP-MT CS. It is clear from Figure 2 that the algorithm tends to infer about 10 clusters, but there is some variation, with the variation in the number of clusters increasing with decreasing number of CS measurements.

To further examine the impact of the number of underlying clusters, we now consider examples for which the data are generated for 5, 3, 2 and 1 underlying. For each of templates, £ve sparse signals are generated randomly, in the manner discussed above for the ten-cluster case. In Figures 3-6 are shown results in the form considered in Figure 2, for the case of 5, 3, 2 and 1 underlying clusters for data generation. One notes the following phenomenon: As the number of underlying clusters diminishes, the difference between DP-MT and MT* CS algorithms diminishes, with almost identical performance witnessed for the case

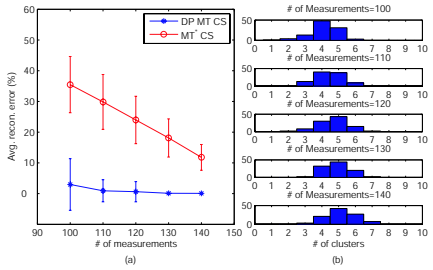


Figure 3. Multi-task CS inversion error (%) for DP-MT and MT CS for the £ve-cluster case. (a) Reconstruction errors, (b) histograms of the number of clusters yielded by DP-MT.

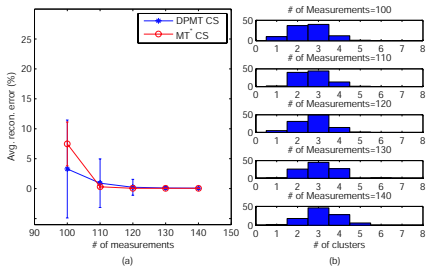


Figure 4. Multi-task CS inversion error (%) for DP-MT and MT CS for the three-cluster case. (a) Reconstruction errors, (b) histograms of the number of clusters yielded by DP-MT.

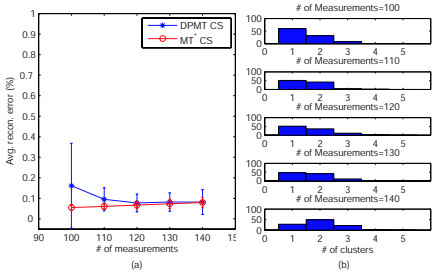


Figure 5. Multi-task CS inversion error (%) for DP-MT and MT CS for the two-cluster case. (a) Reconstruction errors, (b) histograms of the number of clusters yielded by DP-MT.

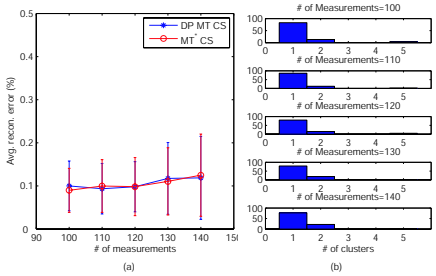


Figure 6. Multi-task CS inversion error (%) for DP-MT and MT CS for the one-cluster case. (a) Reconstruction errors, (b) histograms of the number of clusters yielded by DP-MT.

of three and two clusters; this phenomenon is particularly evident as the number of CS measurements increases. As an aside, we also note that the DP-based inference of the number of underlying clusters adapts well to the underlying data generation.

We now provide an explanation for the relationships between the DP-MT and MT* CS algorithms. For two sparse signals like those in Figure 1, they have distinct non-zero coef£cients and therefore one would typically infer that they have dissimilar sparseness properties. However, they share many zero-amplitude coef£cients. If we consider M sparse signals, and if *all* of the M signals share the same large set of zero-amplitude coef£cients, then they are appropriate for sharing even if the associated (small number of) non-zero coef£cients are entirely distinct. For the 10-cluster case, because of the large number of clusters, the templates do not cumulatively share the same set of zero-amplitude coef£cients; in this case global sharing for CS inversion is inappropriate, and the same is true for the 5-cluster case. However, for the 3 and 2 cluster cases, the templates share a signi£cant number of zero-amplitude coef£cients, and therefore global sharing is appropriate. This underscores that global sharing across M tasks is appropriate when there is substantial sharing of zero-amplitude coef£cients, even when all of the non-zero-amplitude coef£cients are distinct. However, one typically does not know *a priori* if global sharing is appropriate (as it was *not* in Figures 2 and 3), and therefore the DP-based formulation offers generally high-quality results when global sharing is appropriate *and* when it is not.

We consider the sharing mechanisms manifested for two examples from the three-cluster case considered in Figure 4. The truncation level K can be set either to a large number or be estimated in principle by increase the number of sticks included until the log-marginal likelihood (the lower bound) in the VB algorithm starts to decrease. In this example we choose the number of sticks in the DP formulation to $K = 8$ which corresponds to the upper bound of the log-marginal likelihood, and we show the stick (cluster) with which each of the 15 tasks were grouped at the end of the inference process. These examples were selected because they both yielded roughly the same average CS inversion accuracy across the 15 CS inversions (0.40% and 0.38% error), but these two runs yield distinct clusterings. This example emphasizes that because the underlying signals are very sparse and they have signi£cant overlap in the set of zero-amplitude coef£cients, the particular clustering manifested by the DP formulation is not particularly important for the £nal CS-inversion quality.

4.2. Real images

In the following examples, applied to imagery, we perform comparisons between DP-MT, MT*, and also a single-task Bayesian CS (ST), in which the CS inversion is performed independently on each of the tasks. ST CS is realized with

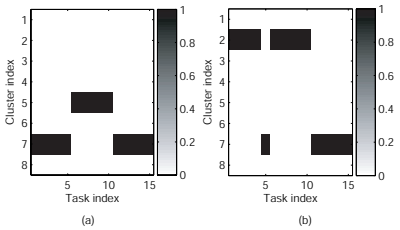


Figure 7. 2 example runs of the DP-MT CS clustering for the 3-cluster case (100 CS measurements). The grey scale denotes the probability that a given task is associated with a particular cluster. (a) Reconstruction error was 0.40%, (b) reconstruction error of 0.38%.

the same algorithm as DP-MT and MT*, but set $\kappa_{i,1} = 1$, and $\kappa_{i,k} = 0$ for $k > 1$, and consider only one CS task at a time ($M = 1$)..

We conduct two examples on CS reconstruction of typical imagery from "natural" scenes. All the images in these examples are of size $256 \quad 256$ and are highly compressible in a wavelet basis. We choose the "Daubechies 8" wavelet as our orthonormal basis, and the sensing matrix $\boldsymbol{\Phi}$ is constructed in the same manner as in Sec 4.1. In this experiment we adopt a hybrid CS scheme, in which using CS we measure only £ne-scale wavelet coef£cients, while retaining all coarse-scale coef£cients (no compression in the coarse scale) (Tsaig & Donoho, 2006). We also assume all the wavelet coef£cients at the £nest scale are zero and only consider (estimate) the other 4096 coef£cients. In both examples, the coarsest scale is $j_0 = 3$, and the £nest scale is $j_1 = 6$. We use the mean of the posterior over $\boldsymbol{\theta}$ to perform the image reconstruction. The reconstruction error is de£ned as $||\hat{\boldsymbol{u}} \quad \boldsymbol{u}||_2/||\boldsymbol{u}||_2$, where $\hat{\boldsymbol{u}}$ is the reconstructed image and $\boldsymbol{u}$ is the original one.

In the £rst example, we choose 12 images from three different scenes. To reconstruct the image, we perform an inverse wavelet transform on the CS-estimated coef£cients. In Figure 8 (a) we show the reconstructed images with all 4096 measurements using linear reconstruction ($\boldsymbol{\theta} = \boldsymbol{\Phi}^T \boldsymbol{v}$), which is the best possible performance. Figure 8 (b)-(d) represent the reconstructed images by the DP-MT, MT*, and the ST algorithms, respectively, with the number of CS measurements $n = 1764$ (1700 measurements in the £ne scales and 64 in the coarse scale) for each task. The reconstruction errors for these four methods are compared in Table 1. We notice that the DP-MT algorithm reduces the reconstruction error compared to the ST method, which indicates that the multi-task CS inversion shares information among tasks and therefore requires less measurements than the single task learning does to achieve the same performance. In addition to the CS inversion, the DP-MT also yield task clustering, with this inferred simultaneously with the CS inversion; while this clustering is not the £nal product of interest, it is informative, with results shown in Figures 9. Note that the algorithms infer three clusters, each corresponding to a particular class of imagery. By contrast the MT* algorithm imposes complete sharing among all tasks, and the results in Table I indicate that this undermines performance.

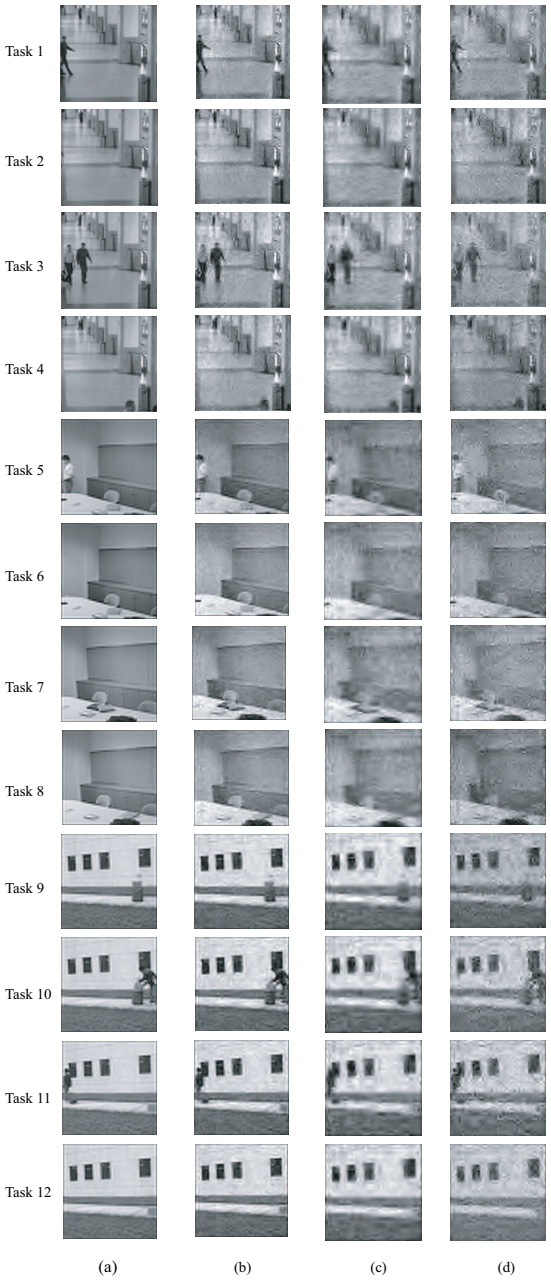


Figure 8. CS recon., (a) Linear, (b) DP-MT, (c) MT , (d) ST

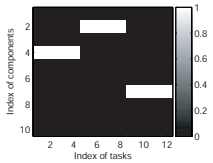


Figure 9. Sharing mechanism for 12 tasks in Figure 8 yielded by DP-MT CS.

Table 1. Reconstruction error (%) for the example in 8.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Task 11	Task 12
DP-MT	8.79	7.89	9.69	8.04	14.33	13.22	15.18	14.54	15.51	16.71	16.11	15.19
MT*	10.19	9.14	11.49	9.18	16.94	15.59	17.46	16.50	18.62	19.82	19.34	18.03
ST	10.28	10.37	12.81	10.28	18.37	16.18	18.65	17.67	20.77	22.24	21.19	19.59
Linear	6.66	6.20	7.08	6.14	12.41	11.70	12.43	11.99	13.83	14.41	14.10	13.53

In the second example we consider 11 images from three scenes. The reconstructed images are shown in Figure 10 by the linear reconstruction, DP-MT, MT* and ST algorithms; the reconstruction errors are listed in Table 2 for all four methods. As expected, the multi-task CS inversion algorithm yields smaller reconstruction error than the single task algorithm. The clustering result is shown in Figure 11, in which images 1-4 and 9-11 are clustered together by DP-MT. However, recall the simple example considered in Figure 7. The DP-based algorithm seeks to share the underlying sparseness of the images, even though the images themselves may appear distinct. In fact, the results in Figure 11 motivated the simple example considered in Figure 7.

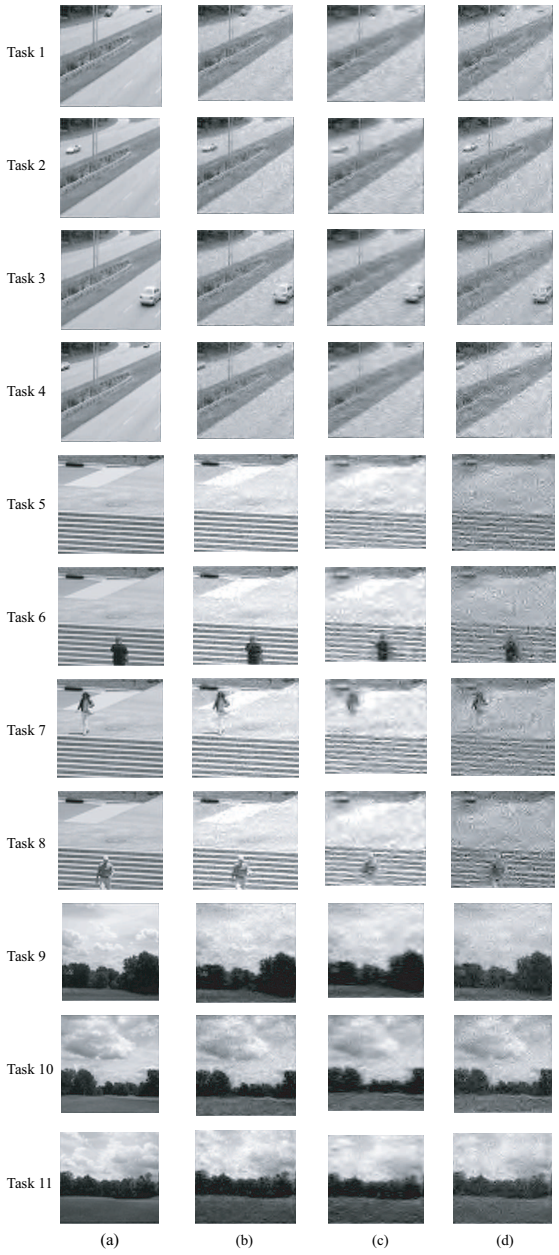


Figure 10. CS recon. (a) Linear, (b) DP-MT, (c) MT , (d) ST

5. Conclusions

Hierarchical Dirichlet process (DP) priors are considered for the imposition of sparseness on the transform coef£cients in the context of inverting multiple CS measurements. An independent zero-mean Gaussian prior is placed on each transform coef£cient of each CS task and the task-dependent precision parameters are assumed drawn from a distribution G, where G is drawn from a Dirichlet process (DP); the base distribution of the DP is a product of Gamma distributions. The DP framework imposes the belief that many of the tasks may share underlying sparseness properties, and the objective is to cluster the CS measurements, where each cluster constitutes a particular form of sparseness. The DP formulation is non-parametric, in the sense that the number of clusters is not set *a priori* and is inferred from the data. A computationally ef£cient variational Bayesian inference has been considered on all model parameters. For all examples considered, the DP-MT CS inversion performed at least as well as ST CS inversion and CS inversion based on global sharing. Especially when global sharing was inappropriate, the DP-based inversion is signi£cantly better.

In future research, we may consider correlation between spatially and spectrally adjacent transformation coef£cients and remove the assumption of exchangeability employed within the DP, which in practice may not be true.

Appendix: Update Equations in VB DP MT

The updated hyperparameters for all $q(\ )$ in Sec 3 are

- $\tilde{a} = a + \frac{1}{2}\sum_{i=1}^{M} n_i$ and $\tilde{b} = b + \frac{1}{2}\sum_{i=1}^{M}\left[tr(\mathbf{\Phi}_i\mathbf{\Gamma}_i^{-1}\mathbf{\Phi}_i^T)+(\mathbf{\Phi}_i\boldsymbol{\mu}_i \quad \boldsymbol{v}_i)^T(\mathbf{\Phi}_i\boldsymbol{\mu}_i \quad \boldsymbol{v}_i)\right]$.

- $\tilde{e} = e+K \quad 1$ and $\tilde{f} = f \quad \sum_{k=1}^{K-1}\left[\psi(\ _{2k}) \quad \psi(\ _{1k}+\ _{2k})\right]$, where $\psi(x) = \frac{\partial}{\partial x}\log\mathbf{\Gamma}(x)$.

Table 2. Reconstruction Error (%) for the example in Figure 10

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10	Task 11
DP-MT	6.50	6.41	6.89	6.86	15.81	15.09	15.91	14.74	7.74	8.05	8.50
MT*	7.79	7.76	8.12	8.32	18.17	17.70	18.66	17.13	8.87	9.16	9.97
ST	8.31	8.23	8.81	9.23	19.79	19.74	20.36	18.88	8.77	9.58	9.62
Linear	4.78	4.77	5.01	5.15	15.39	14.49	15.18	14.06	6.10	5.72	6.72

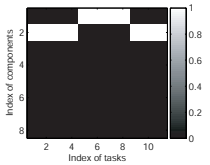


Figure 11. Sharing mechanism for 11 tasks in Figure 10 yielded by DP-MT

- $_{1k} = 1 + \sum_{i=1}^{M} \kappa_{i,k}$ and $_{2k} = \frac{\tilde{e}}{\tilde{f}} + \sum_{i=1}^{M}\sum_{l=k+1}^{K} \kappa_{i,k}$, where $\kappa_{i,k} = q(z_i = k)$.
- $\tilde{c}_{k,j} = c + \frac{1}{2}\sum_{i=1}^{M} \kappa_{i,k}$ and $\tilde{d}_{k,j} = d + \frac{1}{2}\sum_{i=1}^{M} \kappa_{i,k}(\sigma_{i,j} + {}^{2}_{i,k,j})$, where $[\sigma_{i,1}, \quad , \sigma_{i,m}]$ is the diagonal elements of $\mathbf{\Gamma}_i^{-1}$ and $_{i,k,j}$ is the j^{th} element of vector $\boldsymbol{\mu}_i$.
- $\mathbf{\Gamma}_i = \sum_{k=1}^{K} \kappa_{i,k}\mathbf{\Lambda}_k + \frac{\tilde{a}}{\tilde{b}}\mathbf{\Phi}_i^T\mathbf{\Phi}_i$ and $\boldsymbol{\mu}_i = \frac{\tilde{a}}{\tilde{b}}\mathbf{\Gamma}_i^{-1}\mathbf{\Phi}_i^T\boldsymbol{v}_i$, where $\mathbf{\Lambda}_k = diag(\tilde{c}_{k,1}/\tilde{d}_{k,1}, \quad , \tilde{c}_{k,m}/\tilde{d}_{k,m})$ is a diagonal matrix of $m \quad m$.
- $\kappa_{i,k} = \frac{e^{\lambda_{i,k}}}{\sum_{l=1}^{K} e^{\lambda_{i,l}}}$, where $\lambda_{i,k} = \sum_{l=1}^{k-1}\left[\psi(\ _{2l}) \quad \psi(\ _{1l} + \ _{2l})\right] + \left[\psi(\ _{1k}) \quad \psi(\ _{1k} + \ _{2k})\right] \frac{1}{2}\Big\{\sum_{j=1}^{m}\left[\ln 2 \quad \psi(\tilde{c}_{k,j}) + \ln(\tilde{d}_{k,j})\right] + tr(\mathbf{\Gamma}_i^{-1}\mathbf{\Lambda}_k) + \boldsymbol{\mu}_i^T\mathbf{\Lambda}_k\boldsymbol{\mu}_i\Big\}$.

References

Baron, D., Wakin, M. B., Duarte, M. F., Sarvotham, S., & Baraniuk, R. G. (2005). Distributed compressed sensing.

Beal, M. J. (2003). *Variational algorithms for approximate bayesian inference*. Doctoral dissertation, Gatsby Computational Neuroscience Unit, Univ. College London.

Blei, D., & Jordan, M. (2004). Variational methods for the dirichlet process. *ICML*.

Candes, E. (2006). Compressive sensing. *Proc. of ICM*, *3*, 1433–1452.

Candes, E., Romberg, J., & Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. on Inform. Theory*, *52*, 489–502.

Charilaos, C. (1999). Jpeg2000 tutorial. *ICIP*.

Donoho, D. L. (2006). Compressed sensing. *IEEE Trans. Information Theory*, *52*, 1289–1306.

Donoho, D. L., Tsaig, Y., Drori, I., & Starck, J.-C. (2006). Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit.

Escobar, M. D., & West, M. (1995). Bayesian density estimation and inference using mixtures. *JASA*, *90*, 577–588.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1996). Introducing markov chain monte carlo. In *Markov chain monte carlo in practice*. London, U.K.: Chapman Hall.

Ishwaran, H., & James, L. F. (2001). Gibbs sampling methods for stick-breaking priros. *JASA*, *96*, 161–173.

Ji, S., Dunson, D., & Carin, L. (2007a). Multi-task compressive sensing. *Submit to IEEE Trans. on SP*.

Ji, S., Xue, Y., & Carin, L. (2007b). Bayesian compressive sensing. *Accepted to IEEE Trans. on SP*.

MacKay, D. J. C. (1994). Bayesian methods for backpropagation networks. In E. Domany, van J. L. Hemmen and K. Schulten (Eds.), *Models of neural networks III*. New York: Springer-Verlag.

Neal, R. M. (1996). *Bayesian learning for neural networks*. Springer.

S. Chen, D. L. D., & Saunders, M. A. (1999). Atomic decomposition by basis pursuit. *SIAM Journal on Scienti£c Computing*, *20*, 33–61.

Sethuraman, J. (1994). A constructive de£nition of the dirichlet prior. *Statistica Sinica*, *2*, 639–650.

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *JMLR*, *1*, 211–244.

Tropp, J. A., & Gilbert, A. C. (2005). Signal recovery from partial information via orthogonal matching pursuit.

Tsaig, Y., & Donoho, D. L. (2006). Extensions of compressed sensing. *Signal Processing*, *86*, 549–571.

West, M., Muller, P., & Escobar, M. (1994). Hierarchical priors and mixture models with applications in regression and density estimation. In P. R. Freeman and A. F. Smith (Eds.), *Aspects of uncertainty*, 363–386. John Wiley.

Wipf, D. P., & Rao, B. D. (2007). An empirical bayesian strategy for solving the simultaneous sparse approximation problem. *IEEE Trans. on SP*, *55*, 3704–3716.

Estimating Labels from Label Proportions

Novi Quadrianto NOVI.QUAD@GMAIL.COM
Alex J. Smola ALEX.SMOLA@GMAIL.COM
Tiberio S. Caetano TIBERIO.CAETANO@GMAIL.COM
Statistical Machine Learning, NICTA and RSISE, Australian National University
Quoc V. Le QUOCLE@STANFORD.EDU
Computer Science Department, Stanford University

Abstract

Consider the following problem: given sets of unlabeled observations, each set with known label proportions, predict the labels of another set of observations, also with known label proportions. This problem appears in areas like e-commerce, spam filtering and improper content detection. We present consistent estimators which can reconstruct the correct labels with high probability in a uniform convergence sense. Experiments show that our method works well in practice.

1 Introduction

Assume that a web services company wants to increase its profit in sales. Obviously sending out discount coupons will increase sales, but sending coupons to customers who would have purchased the goods anyway decreases the margins. Alternatively, failing to send coupons to customers who would only buy in case of a discount reduces overall sales. We would like to identify the class of would-be customers who are most likely to change their purchase decision when receiving a coupon. The problem is that there is no direct access to a sample of would-be customers. Typically only a sample of people who buy regardless of coupons (those who bought when there was no discount) and a mixed sample (those who bought when there was discount) are available. The mixing proportions can be reliably estimated using random assignment to control and treatment groups. How can we use this information to determine the would-be customers?

Likewise, consider the problem of spam filtering. Datasets of spam are likely to contain almost pure spam (this is achieved e.g. by listing e-mails as spam bait), while user's inboxes typically contain a mix of spam and non-spam. We would like to use the inbox data to improve estimation of spam. In many cases it is possible to estimate the *proportions* of spam and non-spam in a user's inbox much more cheaply than the actual labels. We would like to use this information to categorize e-mails into spam and non-spam.

Similarly, consider the problem of filtering images with "improper content". Datasets of such images are readily accessible thanks to user feedback, and it is reasonable to assume that this labeling is highly reliable. However the rest of images on the web (those not labeled) is a far larger dataset, albeit without labels (after all, this is what we would like to estimate the labels for). That said, it is considerably cheaper to obtain a good estimate of the *proportions* of proper and improper content in addition to having one dataset of images being of likely improper content. We would like to obtain a classifier based on this information.

In this paper we present a method to estimate labels *directly* in such situations, assuming that only label proportions be known. In the above examples, this would be helpful in identifying potential customers, spam e-mails and improper images. We prove bounds indicating that the estimates obtained are close to those from a fully labeled scenario. The formal setting though is more general than the above examples might suggest: we do not require *any* label to be known, only their proportions within each of the involved datasets. Also we are not restricted to the binary case but instead can deal with large numbers of classes.

Problem Formulation Assume that we have n sets of observations $X_i = \{x_1^i, \ldots, x_{m_i}^i\}$ of respective sample sizes m_i (our calibration set) as well as a set $X = \{x_1, \ldots, x_m\}$ (our test set). Moreover, assume that we know the fractions π_{iy} of patterns of labels $y \in \mathcal{Y}$ ($|\mathcal{Y}| \leq n$) contained in each set X_i and assume that we also know the marginal probability $p(y)$ of the

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Table 1. Notation Conventions

X_i	i^{th} set of observations: $X_i = \{x_1^i, \ldots, x_{m_i}^i\}$
m_i	number of observations in X_i
X	test set of observations: $X = \{x_1, \ldots, x_m\}$
Y	test set of labels: $Y = \{y_1, \ldots, y_m\}$
m	number of observations in the test set X
π_{iy}	proportion of label y in set i
$\phi(x, y)$	map from (x, y) to a Hilbert Space

Table 2. Major quantities of interest in the paper

Numbers on the left represent the order in which the corresponding quantity is computed in the algorithm (letters denote the variant of the algorithm: 'a' for general feature map $\phi(x, y)$ and 'b' for factorizing feature map $\phi(x, y) = \psi(x) \otimes \varphi(y)$). Lowercase subscripts refer to model expectations, uppercase subscripts are sample averages.

Expectations with respect to the model:

$$\begin{aligned} \mu_{xy} &:= \mathbf{E}_{(x,y)\sim p(x,y)}[\phi(x, y)] \\ \mu_x^{\text{class}}[y, y'] &:= \mathbf{E}_{(x)\sim p(x|y)}[\phi(x, y')] \\ \mu_x^{\text{set}}[i, y'] &:= \mathbf{E}_{(x)\sim p(x|i)}[\phi(x, y')] \\ \mu_x^{\text{class}}[y] &:= \mathbf{E}_{(x)\sim p(x|y)}[\psi(x)] \\ \mu_x^{\text{set}}[i] &:= \mathbf{E}_{(x)\sim p(x|i)}[\psi(x)] \end{aligned}$$

Expectations with respect to data:

$$\begin{aligned} \mu_{XY} &:= \tfrac{1}{m} \textstyle\sum_{i=1}^m \phi(x_i, y_i) \\ \text{(1a)}\quad \mu_X^{\text{set}}[i, y'] &:= \tfrac{1}{m_i} \textstyle\sum_{x\in X_i} \phi(x, y') \text{ (\textbf{known})} \\ \text{(1b)}\quad \mu_X^{\text{set}}[i] &:= \tfrac{1}{m_i} \textstyle\sum_{x\in X_i} \psi(x) \text{ (\textbf{known})} \end{aligned}$$

Estimates:

$$\begin{aligned} \text{(2)}\quad \hat{\mu}_x^{\text{class}} &= (\pi^\top \pi)^{-1} \pi^\top \mu_X^{\text{set}} \\ \text{(3a)}\quad \hat{\mu}_{XY} &= \textstyle\sum_{y\in\mathcal{Y}} p(y) \hat{\mu}_x^{\text{class}}[y, y] \\ \text{(3b)}\quad \hat{\mu}_{XY} &= \textstyle\sum_{y\in\mathcal{Y}} p(y) \varphi(y) \otimes \hat{\mu}_x^{\text{class}}[y] \\ \text{(4)}\quad \hat{\theta}^* &\text{ solution of (5) for } \mu_{XY} = \hat{\mu}_{XY}. \end{aligned}$$

test set X.[1] It is our goal to design algorithms which are able to obtain conditional class probability estimates $p(y|x)$ solely based on this information. As an illustration, take the spam filtering example. We have $X_1 =$ "mail in spam box" (only spam) and $X_2 =$ "mail in inbox" (spam mixed with non-spam). The test set X then may be X_2 itself, for example. The goal is to find $p(\text{spam}|\text{mail})$ in X_2. Note that (for general π_{iy}) this is more difficult than transduction, where we have at least one dataset with actual labels plus an unlabeled test set where we might have an estimate as to what the relative fractions of class labels might be.

2 Mean Operators

Our idea relies on uniform convergence properties of the expectation operator and of corresponding risk functionals (Altun & Smola, 2006). In doing so, we are able to design estimators with the same performance guarantees in terms of uniform convergence as those with full access to the label information.

[1]Label dictionaries $\mathcal{Y}_i$ do not need to be the same across all sets i: define $\mathcal{Y} := \cup_i \mathcal{Y}_i$ and allow for $\pi_{iy} = 0$ as needed.

2.1 Exponential Families

Denote by $\mathcal{X}$ the space of observations and let $\mathcal{Y}$ be the space of labels. Moreover, let $\phi(x, y) : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$ be a feature map into a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ with kernel $k((x, y), (x', y'))$. In this case we may state conditional exponential models via

$$p(y|x, \theta) = \exp\left(\langle \phi(x, y), \theta \rangle - g(\theta|x)\right) \text{ with} \tag{1}$$

$$g(\theta|x) = \log \sum_{y\in\mathcal{Y}} \exp \langle \phi(x, y), \theta \rangle , \tag{2}$$

where the normalization g is called the log-partition function. For $\{(x_i, y_i)\}$ drawn iid from a distribution $p(x, y)$ on $\mathcal{X} \times \mathcal{Y}$ the conditional log-likelihood is

$$\begin{aligned} \log p(Y|X, \theta) &= \sum_{i=1}^m \left[\langle \phi(x_i, y_i), \theta \rangle - g(\theta|x_i)\right] \\ &= m \langle \mu_{XY}, \theta \rangle - \sum_{i=1}^m g(\theta|x_i) \end{aligned} \tag{3}$$

where μ_{XY} is defined as in Table 2. In order to avoid overfitting one commonly maximizes the log-likelihood penalized by a prior $p(\theta)$. This means that we need to solve the following optimization problem

$$\theta^* := \operatorname*{argmin}_\theta \left[-\log p(Y|X, \theta) p(\theta)\right]. \tag{4}$$

For instance, for a Gaussian prior on θ, i.e. for $-\log p(\theta) = \lambda \|\theta\|^2 + \text{const.}$ we have

$$\theta^* = \operatorname*{argmin}_\theta \left[\sum_{i=1}^m g(\theta|x_i) - m \langle \mu_{XY}, \theta \rangle + \lambda \|\theta\|^2\right]. \tag{5}$$

The problem is that in our setting we do not know the labels y_i, so the sufficient statistics μ_{XY} cannot be computed exactly. The only place where the labels enter the estimation process is via the mean μ_{XY}. Our strategy is to exploit the fact that this quantity, however, is statistically well behaved and converges under relatively mild technical conditions at rate $O(m^{-\frac{1}{2}})$ to its expected value (see Theorem 2)

$$\mu_{xy} := \mathbf{E}_{(x,y)\sim p(x,y)}[\phi(x, y)]. \tag{6}$$

Our goal therefore will be to estimate μ_{xy} and use it as a proxy for μ_{XY}, and only then solve (5) with the estimated $\hat{\mu}_{XY}$ instead of μ_{XY}. We will discuss explicit convergence guarantees in Section 3 after describing how to compute the mean operator in detail.

2.2 Estimating the Mean Operator

In order to obtain θ^* we would need μ_{XY}, which is impossible to compute exactly, since we do not have

Y. However, we know that μ_{XY} and μ_{xy} are close. Hence, if we are able to approximate μ_{xy} this, in turn, will be a good estimate for μ_{XY}.

Our quest is therefore as follows: express μ_{xy} as a linear combination over expectations with respect to the distributions on the datasets $X_1, \ldots, X_n$ (where $n \geq |\mathcal{Y}|$). Secondly, show that the expectations of the distributions having generated the sets X_i ($\mu_x^{\text{set}}[i, y']$, see Table 2) can be approximated by empirical means ($\mu_X^{\text{set}}[i, y']$, also see Table 2). Finally, we need to combine both steps to provide guarantees for μ_{XY}.

It will turn out that in certain cases some of the algebra can be sidestepped, in particular whenever we may be able to identify several sets with each other (e.g. the test set X is one of the training datasets X_i) or whenever $\phi(x, y)$ factorizes into $\psi(x) \otimes \varphi(y)$.

Mean Operator: Since μ_{xy} is a linear operator mapping $p(x, y)$ into a Hilbert Space we may expand μ_{xy}

$$\mu_{xy} = \sum_{y \in \mathcal{Y}} p(y) \mathbf{E}_{x \sim p(x|y)}[\phi(x, y)] = \sum_{y \in \mathcal{Y}} p(y) \mu_x^{\text{class}}[y, y]$$

where the shorthand $\mu_x^{\text{class}}[y, y]$ is defined in Table 2. This means that if we were able to compute $\mu_x^{\text{class}}[y, y]$ we would be able to "reassemble" μ_{xy} from its individual components. We now show that $\mu_x^{\text{class}}[y, y]$ can be estimated directly.

Key to our assumptions is that $p(x|y, i) = p(x|y)$. In other words, we assume that the *conditional* distribution of x is independent of the index i, as long as we know the label y. This yields the following:

$$p(x|i) = \sum_y p(x|y) \pi_{iy}. \tag{7}$$

This allows us define the following means

$$\mu_x^{\text{set}}[i, y'] := \mathbf{E}_{x \sim p(x|i)}[\phi(x, y')] \overset{(7)}{=} \sum_y \pi_{iy} \mu_x^{\text{class}}[y, y'].$$

Note that in order to compute $\mu_x^{\text{set}}[i, y']$ we do *not* need any label information with respect to $p(x|i)$. However, since we have at least $|\mathcal{Y}|$ of those equations and we assumed that π has full rank, they allow us to solve a linear system of equations and compute $\mu_x^{\text{class}}[y, y]$ from $\mu_x^{\text{set}}[i, y']$ for all i. That is, we may use

$$\mu_x^{\text{set}} = \pi \mu_x^{\text{class}} \text{ and hence } \mu_x^{\text{class}} = (\pi^\top \pi)^{-1} \pi^\top \mu_x^{\text{set}} \tag{8}$$

to compute $\mu_x^{\text{class}}[y, y]$ for all $y \in \mathcal{Y}$. Whenever $\pi \in \mathbb{R}^{n \times n}$ is invertible (8) reduces to $\mu_x^{\text{class}} = \pi^{-1} \mu_x^{\text{set}}$. With some slight abuse of notation we have μ_x^{class} and μ_x^{set} represent the *matrices* of terms $\mu_x^{\text{class}}[y, y']$ and $\mu_x^{\text{set}}[i, y']$ respectively.

Algorithm 1

Input datasets X, $\{X_i\}$, probabilities π_{iy} and $p(y)$
for $i = 1$ **to** n **and** $y' \in \mathcal{Y}$ **do**
  Compute empirical means $\mu_X^{\text{set}}[i, y']$
end for
Compute $\hat{\mu}_x^{\text{class}} = (\pi^\top \pi)^{-1} \pi^\top \mu_X^{\text{set}}$
Compute $\hat{\mu}_{XY} = \sum_{y \in \mathcal{Y}} p(y) \hat{\mu}_x^{\text{class}}[y, y]$
Solve the minimization problem

$$\hat{\theta}^* = \operatorname*{argmin}_\theta \left[\sum_{i=1}^m g(\theta | x_i) - m \left\langle \hat{\mu}_{XY}, \theta \right\rangle + \lambda \|\theta\|^2 \right]$$

Return $\hat{\theta}^*$.

Obviously we cannot compute $\mu_x^{\text{set}}[i, y']$ explicitly, since we only have *samples* from $p(x|i)$. However the same convergence results governing the convergence of μ_{XY} to μ_{xy} also hold for the convergence of $\mu_X^{\text{set}}[i, y']$ to $\mu_x^{\text{set}}[i, y']$. Hence we may use the empirical average $\mu_X^{\text{set}}[i, y']$ as the estimate for $\mu_x^{\text{set}}[i, y']$ and from that find an estimate for μ_{XY} (see Algorithm 1).

2.3 Special Cases

In some cases the calculations described in Algorithm 1 can be carried out more efficiently.

Minimal number of sets, i.e. $|\mathcal{Y}| = n$: Provided that π has full rank, $(\pi^\top \pi)^{-1} \pi^\top = \pi^{-1}$. This means that the inverse can be computed more directly.

Testing on one of the calibration sets, i.e. $X = X_i$: This means that X is one of the training sets. We only need one less set of observations. This is particularly useful for factorizing feature maps.

Special feature map $\phi(x, y) = \psi(x) \otimes \varphi(y)$: In this case the calculations of $\hat{\mu}_x^{\text{class}}[y, y']$ and $\mu_X^{\text{set}}[i, y']$ are greatly simplified, since we may pull the dependency on y out of the expectations. Defining $\mu_x^{\text{class}}[y]$, $\mu_x^{\text{set}}[i]$, and $\mu_X^{\text{set}}[i]$ as in Table 2 allows us to simplify

$$\hat{\mu}_{XY} = \sum_{y \in \mathcal{Y}} p(y) \varphi(y) \otimes \hat{\mu}_x^{\text{class}}[y] \tag{9}$$

$$\text{where } \hat{\mu}_x^{\text{class}} = (\pi^\top \pi)^{-1} \pi^\top \mu_X^{\text{set}}. \tag{10}$$

A significant advantage of (10) is that we only need to perform $O(n)$ averaging operations rather than $O(n \cdot |\mathcal{Y}|)$. Obviously the cost of computing $(\pi^\top \pi)^{-1} \pi^\top$ remains unchanged but the latter is negligible in comparison to the operations in Hilbert Space.

Binary classification: One may show that the feature map $\phi(x, y)$ takes on a particularly appealing form of $\phi(x, y) = y \psi(x)$ where $y \in \{\pm 1\}$. This follows since we can always re-calibrate $\langle \phi(x, y), \theta \rangle$ by an offset in-

dependent of y such that $\phi(x,1)+\phi(x,-1)=0$.

If we moreover assume that X_1 only contains class 1 and $X_2 = X$ contains a mixture of classes with labels 1 and -1 with proportions $p(1) =: \rho$ and $p(-1) = 1-\rho$ respectively, we obtain the mixing matrix

$$\pi = \begin{bmatrix} 1 & 0 \\ \rho & 1-\rho \end{bmatrix} \Rightarrow \pi^{-1} = \begin{bmatrix} 1 & 0 \\ \frac{-\rho}{1-\rho} & \frac{1}{1-\rho} \end{bmatrix}$$

Plugging this into (10) and the result in (9) yields

$$\begin{aligned}\hat{\mu}_{XY} &= \rho\mu_X^{\text{set}}[1] - (1-\rho)\left[\tfrac{-\rho}{1-\rho}\mu_X^{\text{set}}[1] + \tfrac{1}{1-\rho}\mu_X^{\text{set}}[2]\right] \\ &= 2\rho\mu_X^{\text{set}}[1] - \mu_X^{\text{set}}[2]. \end{aligned} \tag{11}$$

Consequently taking a simple weighted difference between the averages on two sets, e.g. one set containing spam whereas the other one containing an unlabeled mix of spam and non-spam allows one to obtain the sufficient statistics needed for estimation.

3 Convergence Bounds

The obvious question is how well $\hat{\mu}_{XY}$ manages to approximate μ_{XY} and secondly, how badly any error in estimating μ_{XY} would affect the overall quality of the solution. We approach this problem as follows: first we state the uniform convergence properties of μ_{XY} and similar empirical operators relative to μ_{xy}. Secondly, we apply those bounds to the cases discussed above, and thirdly, we show that the approximate minimizer of the log-posterior has a bounded deviation from what we would have obtained by knowing μ_{XY} exactly.

3.1 Uniform Convergence for Mean Operators

In order to introduce the key result we need to introduce Rademacher averages:

Definition 1 (Rademacher Averages) *Let $\mathcal{X}$ be a domain and p a distribution on $\mathcal{X}$ and assume that $X := \{x_1, \dots, x_m\}$ is drawn iid from p. Moreover, let $\mathcal{F}$ be a class of functions $\mathcal{X} \to \mathbb{R}$. Furthermore denote by σ_i Rademacher random variables, i.e. $\{\pm 1\}$ valued with zero mean. The Rademacher average is*

$$R_m(\mathcal{F}, p) := \mathbf{E}_X \mathbf{E}_\sigma \left[\sup_{f \in \mathcal{F}} \left| \frac{1}{m} \sum_{i=1}^{m} \sigma_i f(x_i) \right| \right]. \tag{12}$$

This quantity measures the flexibility of the function class $\mathcal{F}$ — in our case linear functions in $\phi(x,y)$.

Theorem 2 (Convergence of Empirical Means) *Denote by $\phi : \mathcal{X} \to \mathcal{B}$ a map into a Banach space $\mathcal{B}$, denote by $\mathcal{B}^*$ its dual space and let $\mathcal{F}$ the class of linear functions on $\mathcal{B}$ with bounded $\mathcal{B}^*$ norm by 1. Let $R > 0$ such that for all $f \in \mathcal{F}$ we have $|f(x)| \le R$. Moreover, assume that X is an m-sample drawn from p on $\mathcal{X}$. For $\bar{\epsilon} > 0$ we have that with probability at least $1 - \exp(-\bar{\epsilon}^2 m / 2R^2)$ the following holds:*

$$\|\mu_X - \mu_x\|_{\mathcal{B}} \le 2R_m(\mathcal{F}, p) + \bar{\epsilon} \tag{13}$$

For $k \ge 0$ we only have a failure probability of $1 - \exp(-\bar{\epsilon}^2 m / R^2)$.

Theorem 3 (Bartlett & Mendelson (2002)) *Whenever $\mathcal{B}$ is a Reproducing Kernel Hilbert Space with kernel $k(x,x')$ the Rademacher average can be bounded from above by $R_m(\mathcal{F}) \le m^{-\frac{1}{2}} \left[\mathbf{E}_x[k(x,x)]\right]^{\frac{1}{2}}$*

Our approximation error can be bounded as follows. From the triangle inequality we have:

$$\|\hat{\mu}_{XY} - \mu_{XY}\| \le \|\hat{\mu}_{XY} - \mu_{xy}\| + \|\mu_{xy} - \mu_{XY}\|.$$

For the second term we may employ Theorem 2 directly. To bound the first term note that by linearity

$$\epsilon := \hat{\mu}_{XY} - \mu_{xy} = \sum_y p(y) \left[(\pi^\top \pi)^{-1} \pi^\top \hat{\epsilon}\right]_{y,y} \tag{14}$$

where we define the matrix of coefficients

$$\hat{\epsilon}[i, y'] := \mu_x^{\text{set}}[i, y'] - \mu_X^{\text{set}}[i, y']. \tag{15}$$

Now note that all $\hat{\epsilon}[i, y']$ also satisfy the conditions of Theorem 2 since the sets X_i are drawn iid from the distributions $p(x|i)$ respectively. We may bound each term individually in this fashion and subsequently apply the union bound to ensure that all $n \cdot |\mathcal{Y}|$ components satisfy the constraints. Hence each of the terms needs to satisfy the constraint with probability $1 - \delta/(n|\mathcal{Y}|)$ to obtain an overall bound with probability $1-\delta$. To obtain bounds we would need to bound the linear operator mapping $\hat{\epsilon}$ into ϵ.

3.2 Special Cases

A closed form solution in the general case is not particularly useful. However, we give an explicit analysis for two special cases: firstly the situation where $\phi(x,y) = \psi(x) \otimes \varphi(y)$ and secondly, the binary classification setting where $\phi(x,y) = y\psi(x)$ and $X_i = X$, where much tighter bounds are available.

Special feature map We only need to deal with n rather than with $n \times |\mathcal{Y}|$ empirical estimates, i.e. $\mu_X^{\text{set}}[i]$ vs. $\mu_X^{\text{set}}[i, y']$. Hence (14) and (15) specialize to

$$\epsilon = \sum_y p(y) \sum_{i=1}^{n} \varphi(y) \otimes \left[(\pi^\top \pi)^{-1} \pi^\top\right]_{yi} \hat{\epsilon}[i] \tag{16}$$

$$\hat{\epsilon}[i] := \mu_x^{\text{set}}[i] - \mu_X^{\text{set}}[i]. \tag{17}$$

Assume that with high probability each $\hat{\epsilon}[i]$ satisfies $\|\hat{\epsilon}[i]\| \leq c_i$ (we will deal with the explicit constants c_i later). Moreover, assume for simplicity that $|\mathcal{Y}| = n$ and that π has full rank (otherwise we need to follow through on our expansion using $(\pi^\top\pi)^{-1}\pi^\top$ instead of π^{-1}). This implies that

$$\begin{aligned}\|\epsilon\|^2 &= \sum_{i,j} \langle \hat{\epsilon}[i], \hat{\epsilon}[j] \rangle \times \\ &\quad \sum_{y,y'} p(y)p(y')k(y,y') \left[\pi^{-1}\right]_{yi} \left[\pi^{-1}\right]_{y'j} \\ &\leq \sum_{i,j} c_i c_j \left| \left[\pi^{-1}\right]^\top K^{y,p} \pi^{-1} \right|_{ij} \end{aligned} \tag{18}$$

where $K^{y,p}_{y,y'} = k(y,y')p(y)p(y')$. Combining several bounds we have the following theorem:

Theorem 4 *Assume that we have n sets of observations X_i of size m_i, each of which drawn from distributions with probabilities π_{iy} of observing data with label y. Moreover, assume that $k((x,y),(x',y')) = k(x,x')k(y,y') \geq 0$ where $k(x,x) \leq 1$ and $k(y,y) \leq 1$. Finally, assume that $m = |X|$. In this case the mean operator μ_{XY} can be estimated by $\hat{\mu}_{XY}$ with probability at least $1-\delta$ with precision*

$$\begin{aligned}&\|\mu_{XY} - \hat{\mu}_{XY}\| \leq \left[2 + \sqrt{\log((n+1)/\delta)}\right] \times \\ &\left[m^{-\frac{1}{2}} + \left[\sum_{i,j} m_i^{-\frac{1}{2}} m_j^{-\frac{1}{2}} \left| \left[\pi^{-1}\right]^\top K^{y,p} \pi^{-1} \right|_{ij}\right]^{\frac{1}{2}}\right]\end{aligned}$$

Proof We begin our argument by noting that both for $\phi(x,y)$ and for $\psi(x)$ the corresponding Rademacher averages R_m for functions of RKHS norm bounded by 1 is bounded by $m^{-\frac{1}{2}}$. This is a consequence of all kernels being bounded by 1 in Theorem 3 and $k \geq 0$.

Next note that in Theorem 2 we may set $R = 1$, since for $\|f\| \leq 1$ and $k((x,y),(x,y)) \leq 1$ and $k(x,x) \leq 1$ it follows from the Cauchy Schwartz inequality that $|f(x)| \leq 1$. Solving $\delta \leq \exp -m\epsilon^2$ for ϵ yields $\epsilon \leq m^{-\frac{1}{2}}\left[2 + \sqrt{\log(1/\delta)}\right]$.

Finally, note that we have $n+1$ deviations which we need to bound: one between μ_{XY} and μ_{xy}, and n for each of the $\epsilon[i]$ respectively. Dividing the failure probability δ into $n+1$ cases yields bounds of the form $m^{-\frac{1}{2}}\left[2 + \sqrt{\log((n+1)/\delta)}\right]$ and $m_i^{-\frac{1}{2}}\left[2 + \sqrt{\log((n+1)/\delta)}\right]$ respectively. Plugging all error terms into (18) and summing over terms yields the claim and substituting this back into the triangle inequality proves the claim. ■

Binary Classification Next we consider the special case of binary classification where $X_2 = X$. Using (11) we see that the corresponding estimator is given by

$$\hat{\mu}_{XY} = 2\rho\mu_X^{\text{set}}[1] - \mu_X^{\text{set}}[2]. \tag{19}$$

Since $\hat{\mu}_{XY}$ shares a significant fraction of terms with μ_{XY} we are able to obtain tighter bounds as follows:

Theorem 5 *With probability $1-\delta$ (for $1 > \delta > 0$) the following bound holds:*

$$\|\hat{\mu}_{XY} - \mu_{XY}\| \leq 2\rho\left[2 + \sqrt{\log(2/\delta)}\right]\left[m_1^{-\frac{1}{2}} + m_+^{-\frac{1}{2}}\right]$$

m_+ is the number of observations with $y = 1$ in X_2.

Proof Denote by $\mu[X_+]$ and $\mu[X_-]$ the averages over the subsets of X_2 with positive and negative labels respectively. By construction we have that

$$\begin{aligned}\mu_{XY} &= \rho\mu[X_+] - (1-\rho)\mu[X_-] \\ \hat{\mu}_{XY} &= 2\rho\mu_X^{\text{set}}[1] - \rho\mu[X_+] - (1-\rho)\mu[X_-]\end{aligned}$$

Taking the difference yields $2\rho\left[\mu_X^{\text{set}}[1] - \mu[X_+]\right]$. To prove the claim note that we may use Theorem 2 both for $\left\|\mu_X^{\text{set}}[1] - \mathbf{E}_{x\sim p(x|y=1)}[\psi(x)]\right\|$ and for $\left\|\mu[X_+] - \mathbf{E}_{x\sim p(x|y=1)}[\psi(x)]\right\|$. Taking the union bound and summing over terms proves the claim. ■

The bounds we provided show that $\hat{\mu}_{XY}$ converges at the same rate to μ_{xy} as μ_{XY} does, assuming that the sizes of the sets X_i increase at the same rate as X.

3.3 Stability Bounds

To complete our reasoning we need to show that those bounds translate in guarantees in terms of the minimizer of the log-posterior. In other words, estimates using the correct mean μ_{XY} vs. its estimate $\hat{\mu}_{XY}$ do not differ by a significant amount. For this purpose we make use of (Altun & Smola, 2006, Lemma 17).

Lemma 6 *Denote by f a convex function on $\mathcal{H}$ and let $\mu, \hat{\mu} \in \mathcal{H}$. Moreover let $\lambda > 0$. Finally denote by $\theta^*, \in \mathcal{H}$ the minimizer of*

$$L(\theta, \mu) := f(\theta) - \langle \mu, \theta \rangle + \lambda \|\theta\|^2 \tag{20}$$

with respect to θ and $\hat{\theta}^*$ the minimizer of $L(\hat{\theta}, \hat{\mu})$ respectively. In this case the following inequality holds:

$$\left\|\theta^* - \hat{\theta}^*\right\| \leq \lambda^{-1} \|\mu - \hat{\mu}\|. \tag{21}$$

This means that a good estimate for μ immediately translates into a good estimate for the minimizer of the approximate log-posterior. This leads to the following bound on the risk minimizer.

Corollary 7 *The deviation between θ^*, as defined in (4) and $\hat{\theta}^*$, the minimizer of the approximate log-posterior using $\hat{\mu}_{XY}$ rather than μ_{XY}, is bounded by $O(m^{-\frac{1}{2}} + \sum_i m_i^{-\frac{1}{2}})$.*

Finally, we may use (Altun & Smola, 2006, Theorem 16) to obtain bounds on the quality of $\hat{\theta}^*$ when considering how well it minimizes the *true* negative log-posterior. Using the bound

$$L(\hat{\theta}^*, \mu) - L(\theta^*, \mu) \leq \left\|\hat{\theta}^* - \theta^*\right\| \|\hat{\mu} - \mu\| \tag{22}$$

yields the following bound for the log-posterior:

Corollary 8 *The minimizer $\hat{\theta}^*$ of the approximate log-posterior using $\hat{\mu}_{XY}$ rather than μ_{XY} incurs a penalty of at most $\lambda^{-1} \|\hat{\mu}_{XY} - \mu_{XY}\|^2$.*

4 Extensions

Note that our analysis so far focused on a specific setting, namely maximum-a-posteriori analysis in exponential families. While this is a common and popular setting, the derivations are by no means restricted to this. We have the entire class of (conditional) models described by Altun & Smola (2006); Dudík & Schapire (2006) at our disposition. They are characterized via

$$\underset{p}{\text{minimize}} -H(p) \text{ subject to } \|\mathbf{E}_{z\sim p}[\phi(z)] - \mu\| \leq \epsilon$$

Here p is a distribution, H is an entropy-like quantity defined on the space of distributions, and $\phi(z)$ is some evaluation map into a Banach space. This means that the optimization problem can be viewed as an approximate maximum entropy estimation problem, where we do not enforce exact moment matching of μ but rather allow ϵ slack. In both Altun & Smola (2006) and Dudík & Schapire (2006) the emphasis lay on *unconditional* density models: the dual of the above optimization problem. In particular, it follows that for H being the Shannon-Boltzmann entropy, the dual optimization problem is the maximum a posteriori estimation problem, which is what we are solving here.

In the conditional case, p denotes the collection of probabilities $p(y|x_i)$ and the operator $\mathbf{E}_{z\sim p}[\phi(z)] = \frac{1}{m}\sum_{i=1}^m \mathbf{E}_{y|p(y|x_i)}[\phi(x_i, y)]$ is the conditional expectation operator on the set of observations. Finally, $\mu = \frac{1}{m}\sum_{i=1}^m \phi(x_i, y_i)$, that is, it describes the empirical observations. We have two design parameters:

Function Space: Depending on which Banach Space norm we may choose to measure the deviation between μ and its expectation with respect to p in terms of e.g. the ℓ_2 norm, the ℓ_1 norm or the ℓ_∞ norm. The latter would lead to sparse coding and convex combinations.

Entropy and Regularity: Depending on the choice of entropy and divergence functionals we obtain a range of diverse estimators. For instance, if we were to choose the *unnormalized* entropy instead of the entropy, we would obtain AdaBoost style problems. We may also use Csiszar and Bregmann divergences.

The key point is that our reasoning of estimating μ_{XY} based on an aggregate of samples with unknown labels but known label proportions is still applicable. This means that it should be possible to design boosting algorithms and sparse coding methods which could operate on similarly restricted sets of observations.

5 Related Work and Alternatives

Transduction Gärtner et al. (2006) and Mann & McCallum (2007) performed transduction by enforcing a proportionality constraint on the unlabeled data via a Gaussian Process model. At first glance these methods might seem applicable for our problem but as stated in Section 1, they do require that we have at least some labeled instances of *all classes* at our disposition which need to be drawn in an unbiased fashion. This is clearly not the case in our setting.

Self consistent proportions Kück & de Freitas (2005) introduced a more informative variant of the binary multiple-instance learning, in which groups of instances are given along with estimates of the fraction of positively-labeled instances per group. This is then used to design a hierarchical probabilistic model which will generate consistent fractions. The optimization is solved via a MCMC sampler. While only described for a binary problem it could be easily extended to multi-class settings. Chen et al. (2006) and Musicant et al. (2007) use a similar approach with similar drawbacks, since we typically only have as many sets as classes.

Conditional Probabilities A seemingly valid alternative approach is to try building a classifier for $p(i|x)$ and subsequently recalibrating the probabilities to obtain $p(y|x)$. At first sight this may appear promising since this method is easily applicable in conjunction with most discriminative methods. The idea would be to reconstruct $p(y|x)$ by

$$p(y|x) = \sum_i \pi_{iy} p(i|x). \tag{23}$$

However, this is not a useful estimator in our setting for a simple reason: it assumes the conditional independence $y \perp\!\!\!\perp x \mid i$, which obviously does not hold.

A simple example illustrates the problem. Assume that $\mathcal{X}, \mathcal{Y} = \{1, 2\}$ and that $p(y = 1|x = 1) = p(y = 2|x = 2) = 1$. In other words, the estimation problem

is solvable since the classes are well separated. Moreover, assume that π is given by

$$\pi = \begin{bmatrix} 0.5 - \epsilon & 0.5 + \epsilon \\ 0.5 & 0.5 \end{bmatrix} \text{ for } 0 < \epsilon \ll 1.$$

Here, $p(i|x)$ is useless for estimating $p(y|x)$, since we will only exceed random guessing by at most ϵ.

Reduction to Binary For binary classification and real-valued classification scores we may resort to yet another fairly straightforward method: build a classifier which is able to distinguish between the sets X_1 and X_2 and subsequently threshold labels such that the appropriate fraction of observations in X_1 and X_2 respectively has its proper labels. Unfortunately, multi-class variants of this reduction are nontrivial and experiments show that even for the binary case this method is inferior to our approach.

Density Estimation Finally, one way of obtaining the probabilities $p(x, y|i)$ is to perform density estimation for each set of observations X_i. Subsequently we may use

$$p(x|y) = \sum_i \left[\pi^{-1}\right]_{yi} p(x, y|i) \tag{24}$$

to re-calibrate the probability estimates. Bayes' theorem is finally invoked to compute posterior probabilities. This tends to fail for high-dimensional data due to the curse of dimensionality in density estimation.

6 Experiments

Datasets: We use binary and three-class classification datasets from the UCI repository[2] and the LibSVM site.[3] If separate training and test sets are available, we merge them before performing nested 10-fold cross-validation. Since we need to generate as many splits as classes, we limit ourselves to three classes.

For the binary datasets we use half of the data for X_1 and the rest for X_2. We also remove all instances of class 2 from X_1. That is, the conditional class probabilities in X_2 match those from the repository, whereas in X_1 their counterparts are deleted.

For three-class datasets we investigate two different partitions. In scenario A we use class 1 exclusively in X_1, class 2 exclusively in X_2, and a mix of all three classes weighted by $(0.5 \cdot p(1), 0.7 \cdot p(2), 0.8 \cdot p(3))$ to generate X_3. In scenario B we use the following splits

$$\begin{bmatrix} c_1 \cdot 0.4 \cdot p(1) & c_1 \cdot 0.15 \cdot p(2) & c_1 \cdot 0.14 \cdot p(3) \\ c_2 \cdot 0.1 \cdot p(1) & c_2 \cdot 0.15 \cdot p(2) & c_2 \cdot 0.06 \cdot p(3) \\ c_3 \cdot 0.5 \cdot p(1) & c_3 \cdot 0.7 \cdot p(2) & c_3 \cdot 0.8 \cdot p(3) \end{bmatrix}$$

Here the constants c_1, c_2 and c_3 are chosen such that the probabilities are properly normalized. As before, X_3 contains half of the data.

Model Selection: As stated, we carry out a *nested* 10-fold cross-validation procedure: 10-fold cross-validation to assess the performance of the estimators; within each fold, 10-fold cross-validation is performed to find a suitable value for the parameters.

For supervised classification, i.e. discriminative sorting, such a procedure is quite straightforward because we can directly optimize for classification error. For kernel density estimation (KDE), we use the log-likelihood as our criterion.

Due to the high number of hyper-parameters (at least 8) in MCMC, it is difficult to perform *nested* 10-fold cross-validation. Instead, we choose the *best* parameters from a simple 10-fold crossvalidation run. In other words, we are giving the MCMC method an unfair advantage over our approach by reporting the best performance during the model selection procedure.

Finally, for the re-calibrated sufficient statistics $\hat{\mu}_{XY}$ we use the estimate of the log-likelihood on the validation set as the criterion for cross-validation, since no other quantity, such as classification errors is readily available for estimation.

Algorithms: For discriminative sorting we use an SVM with a Gaussian RBF kernel whose width is set to the median distance between observations (Schölkopf, 1997); the regularization parameter is chosen by cross-validation. The same strategy applies for our algorithm. For KDE, we use Gaussian kernels with diagonal densities. Cross-validation is performed over the kernel width. For MCMC, 10^3 samples are generated after a burn in period of 10^3 steps (Kück & de Freitas (2005)).

Optimization: Bundle methods (Smola et al., 2007; Teo et al. , 2007) are used to solve the optimization problem in Algorithm 1.

Results: The experimental results are summarized in Table 3. Our method outperforms KDE and discriminative sorting. In terms of computation, our approach is somewhat more efficient, since it only needs to deal with a smaller sample size (only X rather than the union of all X_i). The training time for our method is less than 2 minutes for all cases, whereas MCMC on average takes 15 minutes and maybe even much longer

[2] http://archive.ics.uci.edu/ml/

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

when the number of active kernels and/or observations are high. However, for large number of partitions n, the MCMC procedure might potentially have an edge over our method as we do not take full advantage of this setting. However, this can be achieved easily by optimizing the condition number of the pseudoinverse of the redundant system of linear equations.

Our method also performs well on multiclass datasets. As described in Section 3.2, the quality of our minimizer of the log-posterior depends on the mixing matrix and this is noticeable in the reduction of performance for the dense mixing matrix (scenario B) in comparison to the better conditioned sparse mixing matrix (scenario A). In other words, for ill conditioned π even our method has its limits, simply due to numerical considerations of effective sample size.

7 Conclusion

We have showed a rather surprising result, namely that it is possible to consistently reconstruct the labels of a dataset if we can only obtain information about the proportions of occurrence of each class (in at least as many data aggregates as there are classes). In particular, we prove that up to constants, our algorithm enjoys the same rates of convergence afforded to methods which have full access to all label information.

This has a range of potential applications in e-commerce, spam filtering and improper content detection. It also suggests that techniques used to anonymize observations, e.g. demographic data, may not be really safe. Experiments show our algorithm is fast and outperforms competitive methods.

References

Altun, Y., & Smola, A. (2006). Unifying divergence minimization and statistical inference via convex duality. *COLT'06*, LNCS, 139–153. Springer.

Bartlett, P. L., & Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *JMLR*, *3*.

Chen, B., Chen, L., Ramakrishnan, R., & Musicant, D. (2006). Learning from aggregate views. *ICDE'06*, 3–12.

Dudík, M., & Schapire, R. E. (2006). Maximum entropy distribution estimation with generalized regularization. *COLT'06*, Springer.

Gärtner, T., Le, Q., Burton, S., Smola, A., & Vishwanathan, S. (2006). Large-scale multiclass transduction. *NIPS'06*

Kück, H., & de Freitas, N. (2005). Learning about individuals from group statistics. In *UAI'05*, 332–339.

Mann, G., & McCallum, A. (2007). Simple, robust, scalable semi-supervised learning via expectation regularization. In *ICML'07*.

Musicant, D., Christensen, J., & Olson, J. (2007). Supervised learning by training on aggregate outputs. In *IEEE ICDM*.

Schölkopf, B. (1997). *Support Vector Learning*. Oldenbourg Verlag.

Smola, A., Vishwanathan, S. V. N., & Le, Q. (2007). Bundle methods for machine learning. In *NIPS'07*.

Teo, C.H., Le, Q.V, Smola, A.J., & Vishwanathan, S.V.N. (2007). A Scalable Modular Convex Solver for Regularized Risk Minimization. In *KDD'07*.

Table 3. Classification error on UCI/LibSVM datasets

Errors are reported in mean ± standard error. The best result and those not significantly worse than it, are highlighted in boldface. We use a one-sided paired t-test with 95% confidence.

`MM`: Mean Map (our method); `KDE`: Kernel Density Estimation; `DS`: Discriminative Sorting (only applicable for binary classification); `MCMC`: the sampling method; `BA`: Baseline, obtained by predicting the major class. †: Program fails (too high dimensional data - only `KDE`). ‡: Program fails (large datasets - only `MCMC`).

Data	MM	KDE	DS	MCMC	BA
iono	18.4±3.2	**17.5±3.2**	**12.2±2.6**	18.0±2.1	35.8
iris	**10.0±3.6**	**16.8±3.4**	**15.4±1.1**	**21.1±3.6**	29.9
optd	1.8±0.5	**0.7±0.4**	9.8±1.2	2.0±0.4	49.1
page	**3.8±2.3**	7.1±2.8	18.5±5.6	**5.4±2.8**	43.9
pima	**27.5±3.0**	34.8±0.6	34.4±1.7	**23.8±1.8**	34.8
tic	31.0±1.5	34.6±0.5	**26.1±1.5**	31.3±2.5	34.6
yeast	**9.3±1.5**	**6.5±1.3**	25.6±3.6	10.4±1.9	39.9
wine	**7.4±3.0**	**12.1±4.4**	18.8±6.4	**8.7±2.9**	40.3
wdbc	**7.8±1.3**	**5.9±1.2**	10.1±2.1	15.5±1.3	37.2
sonar	**24.2±3.5**	35.2±3.5	31.4±4.0	39.8±2.8	44.5
heart	**30.0±4.0**	38.1±3.8	**28.4±2.8**	**33.7±4.7**	44.9
brea	**5.3±0.8**	14.2±1.6	**3.5±1.3**	**4.8±2.0**	34.5
aust	**17.0±1.7**	33.8±2.5	**15.8±2.9**	30.8±1.8	44.4
svm3	**20.4±0.9**	27.2±1.3	25.5±1.5	24.2±0.8	23.7
adult	**18.9±1.2**	24.5±1.3	22.1±1.4	**18.7±1.2**	24.6
cleve	**19.1±3.6**	35.9±4.5	**23.4±2.9**	**24.3±3.1**	22.7
derm	**4.9±1.4**	27.4±2.6	**4.7±1.9**	14.2±2.8	30.5
musk	**25.1±2.3**	28.7±2.6	**22.2±1.8**	**19.6±2.8**	43.5
ger	**32.4±1.8**	41.6±2.9	37.6±1.9	**32.0±0.6**	32.0
cove	37.1±2.5	41.9±1.7	**32.4±1.8**	41.1±2.2	45.9
spli	**25.2±2.0**	35.5±1.5	**26.6±1.7**	28.8±1.6	48.4
giss	**10.3±0.9**	†	**12.2±0.8**	50.0±0.0	50.0
made	**44.1±1.5**	†	**46.0±2.0**	49.6±0.2	50.0
cmc	**37.5±1.4**	43.8±0.7	45.1±2.3	46.9±2.6	49.9
bupa	**48.5±2.9**	50.8±5.1	**40.3±4.9**	50.4±0.8	49.7
protA	**44.6±0.3**	60.2±0.1	N/A	65.3±1.9	61.2
protB	**45.7±0.6**	61.2±0.0	N/A	67.7±1.8	61.2
dnaA	**16.6±1.0**	30.7±0.8	N/A	37.7±0.8	40.5
dnaB	**29.1±1.0**	33.0±0.7	N/A	40.5±0.0	40.5
sensA	**19.8±0.1**	43.1±0.0	N/A	‡	43.2
sensB	**21.0±0.1**	43.1±0.0	N/A	‡	43.2

Acknowledgments We thank Hendrik Kück and Choon Hui Teo. NICTA is funded by the Australian Government's Backing Australia's Ability and the Centre of Excellence programs. We received funding of the FP7 Network of Excellence by the European Union.

Learning Diverse Rankings with Multi-Armed Bandits

Filip Radlinski FILIP@CS.CORNELL.EDU
Robert Kleinberg RDK@CS.CORNELL.EDU
Thorsten Joachims TJ@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

Algorithms for learning to rank Web documents usually assume a document's relevance is independent of other documents. This leads to learned ranking functions that produce rankings with redundant results. In contrast, user studies have shown that diversity at high ranks is often preferred. We present two online learning algorithms that directly learn a diverse ranking of documents based on users' clicking behavior. We show that these algorithms minimize abandonment, or alternatively, maximize the probability that a relevant document is found in the top k positions of a ranking. Moreover, one of our algorithms asymptotically achieves optimal worst-case performance even if users' interests change.

1. Introduction

Web search has become an essential component of the Internet infrastructure, and has hence attracted significant interest from the machine learning community (e.g. Herbrich et al., 2000; Burges et al., 2005; Radlinski & Joachims, 2005; Chu & Ghahramani, 2005; Metzler & Croft, 2005; Yue et al., 2007; Taylor et al., 2008). The conventional approach to this learning-to-rank problem has been to assume the availability of manually labeled training data. Usually, this data consists of a set of documents judged as relevant or not to specific queries, or of pairwise judgments comparing the relative relevance of pairs of documents. These judgments are used to optimize a ranking function offline, to a standard information retrieval metric, then deploying the learned function in a live search engine.

We propose a new learning to rank problem formulation that differs in three fundamental ways. First, unlike most previous methods, we learn from usage data rather than manually labeled relevance judgments. Usage data is available in much larger quantities and at much lower cost. Moreover, unlike manual judgments, which need to be constantly updated to stay relevant, usage data naturally reflects current users' needs and the documents currently available. Although some researchers have transformed usage data into relevance judgments, or used it to generate features (e.g. Joachims, 2002; Radlinski & Joachims, 2005; Agichtein et al., 2006), we go one step further by directly optimizing a usage-based metric.

Second, we propose an online learning approach for learning from usage data. As training data is being collected, it immediately impacts the rankings shown. This means the learning problem we address is regret minimization, where the goal is to minimize the total number of poor rankings displayed over *all* time. In particular, in this setting there is a natural tradeoff between exploration and exploitation: It may be valuable in the long run to present some rankings with unknown documents, to allow training data about these documents to be collected. In contrast, in the short run exploitation is typically optimal. With only few exceptions (e.g. Radlinski & Joachims, 2007), previous work does not consider such an online approach.

Third and most importantly, except for (Chen & Karger, 2006), previous algorithms for learning to rank have considered the relevance of each document independently of other documents. This is reflected in the performance measures typically optimized, such as Precision, Recall, Mean Average Precision (MAP) (Baeza-Yates & Ribeiro-Neto, 1999) and Normalized Discounted Cumulative Gain (NDCG) (Burges et al., 2006). In fact, recent work has shown that these measures do not necessarily correlate with user satisfaction (Turpin & Scholer, 2006). Additionally, it intuitively stands to reason that presenting many slight variations of the same relevant document in web search results may increase the MAP or NDCG score, yet would be suboptimal for users. Moreover, web queries often have different meanings for different users (a canonical example is the query *jaguar*) suggesting that a ranking with diverse documents may be preferable.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

We will show how clickthrough data can be used to learn rankings maximizing the probability that any new user will find at least one relevant document high in the ranking.

2. Related Work

The standard approach for learning to rank uses training data, in the form of judgments assessing the relevance of individual documents to a query, to learn parameters θ for a scoring function $f(q, d_i, \theta)$. Given a new query q, this function computes $f(q, d_i, \theta)$ for each document d_i *independently* and ranks documents by decreasing score (e.g. Herbrich et al., 2000; Joachims, 2002; Burges et al., 2005; Chu & Ghahramani, 2005). This also applies to recent algorithms that learn θ to maximize nonlinear performance measures such as MAP (Metzler & Croft, 2005; Yue et al., 2007) and NDCG (Burges et al., 2006; Taylor et al., 2008).

The theoretical model that justifies ranking documents in this way is the probabilistic ranking principle (Robertson, 1977). It suggests that documents should be ranked by their probablility of relevance to the query. However, the optimality of such a ranking relies on the assumption that there are no statistical dependencies between the probabilities of relevance among documents – an assumption that is clearly violated in practice. For example, if one document about jaguar cars is not relevant to a user who issues the query *jaguar*, other car pages become less likely to be relevant. Furthermore, empirical studies have shown that given a fixed query, the same document can have different relevance to different users (Teevan et al., 2007). This undermines the assumption that each document has a single relevance score that can be provided as training data to the learning algorithm. Finally, as users are usually satisfied with finding a small number of, or even just one, relevant document, the usefulness and relevance of a document does depend on other documents ranked higher.

As a result, most search engines today attempt to eliminate redundant results and produce *diverse* rankings that include documents that are potentially relevant to the query for different reasons. However, learning optimally diverse rankings using expert judgments would require document relevance to be measured for different possible meanings of a query. While the TREC interactive track[1] provides some documents labeled in this way for a small number of queries, such document collections are even more difficult to create than standard expert labeled collections.

[1] http://trec.nist.gov/data/t11_interactive/t11i.html

Several non-learning algorithms for obtaining a diverse ranking of documents from a non-diverse ranking have been proposed. One common one is Maximal Marginal Relevance (MMR) (Carbonell & Goldstein, 1998). Given a similarity (relevance) measure between documents and queries $sim_1(d, q)$ and a similarity measure between pairs of documents $sim_2(d_i, d_j)$, MMR iteratively selects documents by repeatedly finding $d_i = \text{argmax}_{d \in \mathcal{D}} \lambda sim_1(d, q) - (1 - \lambda) \max_{d_j \in S} sim_2(d, d_j)$ where S is the set of documents already selected and λ is a tuning parameter. In this way MMR selects the most relevant documents that are also different from any documents already selected.

Critically, MMR requires that the relevance function $sim_1(d, q)$, and the similarity function $sim_2(d_i, d_j)$ is known. It is usual to obtain sim_1 and sim_2 using algorithms such as those discussed above. The goal of MMR is to rerank an already learned ranking (that of ranking documents by decreasing sim_1 score) to improve diversity. All previous approaches of which we are aware that optimize diversity similarly require a relevance function to be learned prior to performing a diversification step (Zhu et al., 2007; Zhang et al., 2005; Zhai et al., 2003), with the exception of Chen and Karger (2006). Rather, they require that a model for estimating the probability a document is relevant, given a query and other non-relevant documents, is available. In contrast, we directly learn a diverse ranking of documents using users' clicking behavior.

3. Problem Formalization

We address the problem of learning an optimally diversified ranking of documents $\mathcal{D} = \{d_1, \ldots, d_n\}$ for one fixed query. Suppose we have a population of users, where each user u_i considers some subset of documents $A_i \subset \mathcal{D}$ as relevant to the query, and the remainder of the documents as non-relevant. Intuitively, users with different interpretations for the query would have different relevant sets, while users with similar interpretations would have similar relevant sets.

At time t, we interact with user u_t with relevant set A_t. We present an ordered set of k documents, $B_t = (b_1(t), \ldots, b_k(t))$. The user considers the results in order, and clicks on up to one document. The probability of user u_t clicking on document d_i (conditional on the user not clicking on a document presented earlier in the ranking) is assumed to be $p_{ti} \in [0, 1]$. We refer to the vector of probabilities $(p_{ti})_{i \in \mathcal{D}}$ as the *type* of user u_t. In the simplest case, we could take $p_{ti} = 1$ if $d_i \in A_t$ and 0 otherwise, in which case the user clicks on the first relevant document or does not click if no documents in B_t are relevant. However, in reality clicks tend to be noisy although more relevant docu-

Algorithm 1 Ranked Explore and Commit

input: Documents $(d_1, .., d_n)$, parameters ϵ, δ, k.
$x \leftarrow \lceil 2k^2/\epsilon^2 \log(2k/\delta) \rceil$
$(b_1, \ldots, b_k) \leftarrow k$ arbitrary documents.
for i=1 ... k **do** *At every rank*
  $\forall j.\ p_j \leftarrow 0$
  for counter=1 ... x **do** *Loop x times*
    for j=1 ... n **do** *over every document d_j*
      $b_i \leftarrow d_j$
      display $\{b_1, \ldots, b_k\}$ to user; record clicks
      if user clicked on b_i **then** $p_j \leftarrow p_j + 1$
    end for
  end for
  $j^* \leftarrow \text{argmax}_j\, p_j$ *Commit to best document at this rank*
  $b_i \leftarrow d_{j^*}$
end for

ments are more likely to be clicked on. In our analysis, we will take $p_{ti} \in [0, 1]$.

We get payoff 1 if the user clicks, 0 if not. The goal is to maximize the total payoff, summing over all time. This payoff represents the number of users who clicked on any result, which can be interpreted as the user finding at least one potentially relevant document (so long as p_{ti} is higher when $d_i \in A_t$ than when $d_i \notin A_t$).

The event that a user does not click is called *abandonment* since the user abandoned the search results. Abandonment is an important measure of user satisfaction because it indicates that users were presented with search results of no potential interest.

4. Learning Algorithms

We now present two algorithms that directly minimize the abandonment rate. At a high level, both algorithms learn a marginal utility for each document at each rank, displaying documents to maximize the probability that a new user of the search system would find at least one relevant document within the top k positions. The algorithms differ in their assumptions.

4.1. Ranked Explore and Commit

The first algorithm we present is a simple greedy strategy that assumes that user interests and documents do not change over time. As we will see, after T time steps this algorithm achieves a payoff of at least $(1-1/e-\epsilon)OPT - O(k^3 n/\epsilon^2 \ln(k/\delta))$ with probability at least $1-\delta$. OPT denotes the maximal payoff that could be obtained if the click probabilities p_{ti} were known ahead of time for all users and documents, and $(1-1/e)OPT$ is the best obtainable polynomial time approximation, as will be explained in Section 5.1.

As described in Algorithm 1, Ranked Explore and Commit (REC) iteratively selects documents for each rank. At each rank position i, every document d_j is presented a fixed number x times, and the number of clicks it receives during these presentations is recorded. After nx presentations, the algorithm permanently assigns the document that received the most clicks to the current rank, and moves on to the next rank.

Algorithm 2 Ranked Bandits Algorithm

initialize $\mathsf{MAB}_1(n), \ldots, \mathsf{MAB}_k(n)$ *Initialize MABs*
for t = 1 ... T **do**
  for i = 1 ... k **do** *Sequentially select documents*
    $\hat{b}_i(t) \leftarrow$ select-arm (MAB_i)
    if $\hat{b}_i(t) \in \{b_1(t), .., b_{i-1}(t)\}$ **then** *Replace repeats*
      $b_i(t) \leftarrow$ arbitrary unselected document
    else
      $b_i(t) \leftarrow \hat{b}_i(t)$
    end if
  end for
  display $\{b_1(t), \ldots, b_k(t)\}$ to user; record clicks
  for i = 1 ... k **do** *Determine feedback for MAB_i*
    if user clicked $b_i(t)$ **and** $\hat{b}_i(t) = b_i(t)$ **then**
      $f_{it} = 1$
    else
      $f_{it} = 0$
    end if
    update $(\mathsf{MAB}_i, arm = \hat{b}_i(t), reward = f_{it})$
  end for
end for

4.2. Ranked Bandits Algorithm

Ranked Explore and Commit is purely greedy, meaning that after each document is selected, this decision is never revisited. In particular, this means that if user interests or documents change, REC can perform arbitrarily poorly. In contrast, the Ranked Bandits Algorithm (RBA) achieves a combined payoff of $(1-1/e)OPT - O(k\sqrt{Tn \log n})$ after T time steps even if documents and user interests change over time.

This algorithm leverages standard theoretical results for multi-armed bandits. Multi-armed bandits (MAB) are modeled on casino slot machines (sometimes called one-armed bandits). The goal of standard MAB algorithms is to select the optimal sequence of slot machines to play to maximize the expected total reward collected. For further details, refer to (Auer et al., 2002a). The ranked bandits algorithm runs an MAB instance MAB_i for *each rank* i. Each of the k copies of the multi-armed bandit algorithm maintains a value (or index) for every document. When selecting the ranking to display to users, the algorithm MAB_1 is responsible for choosing which document is shown at rank 1. Next, the algorithm MAB_2 determines which

document is shown at rank 2, unless the same document was selected at the highest rank. In that case, the second document is picked arbitrarily. This process is repeated to select all top k documents.

Next, after a user considers up to the top k documents in order and clicks on one or none, we need to update the indices. If the user clicks on a document actually selected by an MAB instance, the reward for the arm corresponding to that document for the multi-armed bandit at that rank is 1. The reward for the arms corresponding to all other selected documents is 0. In particular, note that the RBA treats the bandits corresponding to each rank independently. Precise pseudocode for the algorithm is presented in Algorithm 2. A generalization of this algorithm, in an abstract setting without the application to Information Retrieval, was discovered independently by Streeter and Golovin (2007).

The actual MAB algorithm used for each MAB_i instance is not critical, and in fact any algorithm for the non-stochastic multi-armed bandit problem will suffice. Our theoretical analysis only requires that:

- The algorithm has a set S of n strategies.
- In each period t a payoff function $f_t : S \to [0, 1]$ is defined. This function is not revealed to the algorithm, and may depend on the algorithm's choices before time t.
- In each period the algorithm chooses a (random) element $y_t \in S$ based on the feedback revealed in prior periods.
- The feedback revealed in period t is $f_t(y_t)$.
- The expected payoffs of the chosen strategies satisfy:
$$\sum_{t=1}^{T} \mathbf{E}[f_t(y_t)] \;\geq\; \max_{y \in S} \sum_{t=1}^{T} \mathbf{E}[f_t(y)] \;-\; R(T)$$
where $R(T)$ is an explicit function in $o(T)$ which depends on the particular multi-armed bandit algorithm chosen, and the expectation is over any randomness in the algorithm. We will use the **Exp3** algorithm in our analysis, where $R(T) = O\left(\sqrt{Tn \log n}\right)$ (Auer et al., 2002b).

We will also later see that although these conditions are needed to bound worst-case performance, better practical performance may be obtained at the expense of worst-case performance if they are relaxed.

5. Theoretical Analysis

We now present a theoretical analysis of the algorithms presented in Section 4. First however, we discuss the offline version of this optimization problem.

5.1. The Offline Optimization Problem

The problem of choosing the optimum set of k documents for a given user population is NP-hard, even if all the information about the user population (i.e. the set of relevant documents for each user) is given offline and we restrict ourselves to $p_{ij} \in \{0, 1\}$. This is because selecting the optimal set of documents is equivalent to the maximum coverage problem: Given a positive integer k and a collection of subsets $S_1, S_2, \ldots, S_n$ of an m-element set, find k of the subsets whose union has the largest possible cardinality.

The standard greedy algorithm for the maximum coverage problem, translated to our setting, iteratively chooses the document that is relevant to the most users for whom a relevant document has not yet been selected. This algorithm is a $(1 - 1/e)$-approximation algorithm for this maximization problem (Nemhauser et al., 1978). The $(1 - 1/e)$ factor is optimal and no better worst-case approximation ratio is achievable in polynomial time unless $NP \subseteq DTIME\left(n^{\log \log n}\right)$ (Khuller et al., 1997).

5.2. Analysis of Ranked Bandits Algorithm

We start by analyzing the Ranked Bandits Algorithm. This algorithm works by simulating the offline greedy algorithm, using a separate instance of the multi-armed bandit algorithm for each step of the greedy algorithm. Except for the sublinear regret term, the combined payoff is as high as possible without violating the hardness-of-approximation result stated in the preceding paragraph.

To analyze the RBA, we first restrict ourselves to users who click on any given document with probability either 0 or 1. We refer to this restricted type of user as a *deterministic user*; we will relax the requirement later. Additionally, this analysis applies to a worst case (and hence fixed) sequence of users.

Further, it is useful to introduce some notation. For a set A and a sequence $B = (b_1, b_2, \ldots, b_k)$, let

$$G_i(A, B) = \begin{cases} 1 & \text{if } A \text{ intersects } \{b_1, \ldots, b_i\} \\ 0 & \text{otherwise} \end{cases}$$

$$g_i(A, B) = G_i(A, B) - G_{i-1}(A, B)$$

Recalling that A_t is the set of documents relevant to user u_t, we see that $G_k(A_t, B)$ is the payoff of presenting B to the user u_t. Let

$$B^* = \operatorname*{argmax}_B \sum_{t=1}^{T} G_k(A_t, B),$$

$$OPT = \sum_{t=1}^{T} G_k(A_t, B^*).$$

Recall that $(\hat{b}_1(t), \ldots, \hat{b}_k(t))$ is the sequence of documents chosen by the algorithms $\mathsf{MAB}_1, \ldots, \mathsf{MAB}_k$ at time t, and that $(b_1(t), \ldots, b_k(t))$ is the sequence of documents presented to the user. Define the feedback function f_{it} for algorithm MAB_i at time t, as follows:

$$f_{it}(b) = \begin{cases} 1 & \text{if } G_{i-1}(A_t, B_t) = 0 \text{ and } b \in A_t \\ 0 & \text{otherwise} \end{cases}.$$

Note that the value of f_{it} defined in the pseudocode for the Ranked Bandits Algorithms is equal to $f_{it}(\hat{b}_i(t))$.

Lemma 1. *For all i,*

$$\begin{aligned} &\mathbf{E}\left[\sum_{t=1}^{T} g_i(A_t, B_t)\right] \\ &\geq \frac{1}{k}\mathbf{E}\left[\sum_{t=1}^{T} (G_k(A_t, B^*) - G_{i-1}(A_t, B_t))\right] - R(T) \\ &= \frac{1}{k}OPT - \frac{1}{k}\mathbf{E}\left[\sum_{t=1}^{T} G_{i-1}(A_t, B_t)\right] - R(T). \end{aligned}$$

Proof. First, note that

$$g_i(A_t, B_t) \geq f_{it}(\hat{b}_i(t)). \tag{1}$$

This is trivially true when $f_{it}(\hat{b}_i(t)) = 0$. When $f_{it}(\hat{b}_i(t)) = 1$, $G_{i-1}(A_t, B_t) = 0$ and $\hat{b}_i(t) \in A_t$. This implies that $b_i(t) = \hat{b}_i(t)$ and that $g_i(A_t, B_t) = 1$.

Now using the regret bound for MAB_i we obtain

$$\begin{aligned} \sum_{t=1}^{T} \mathbf{E}[f_{it}(\hat{b}_i(t))] &\geq \max_b \sum_{t=1}^{T} \mathbf{E}[f_{it}(b)] - R(T) \\ &\geq \frac{1}{k}\mathbf{E}\left[\sum_{b \in B^*} \sum_{t=1}^{T} f_{it}(b)\right] - R(T). \end{aligned} \tag{2}$$

To complete the proof of the lemma, we will prove that

$$\sum_{b \in B^*} f_{it}(b) \geq G_k(A_t, B^*) - G_{i-1}(A_t, B_t). \tag{3}$$

The lemma follows immediately by combining (1)-(3). Observe that the left side of (3) is a non-negative integer, while the right side takes one of the values $\{-1, 0, 1\}$. Thus, to prove (3) it suffices to show that the left side is greater than or equal to 1 whenever the right side is equal to 1. The right side equals 1 only when $G_{i-1}(A_t, B_t) = 0$ and A_t intersects B^*. In this case it is clear that there exists at least one $b \in B^*$ such that $f_{it}(b) = 1$, hence the left side is greater than or equal to 1. □

Theorem 1. *The algorithm's combined payoff after T rounds satisfies:*

$$\mathbf{E}\left[\sum_{t=1}^{T} G_k(A_t, B_t)\right] \geq \left(1 - \frac{1}{e}\right) OPT - kR(T). \tag{4}$$

Proof. We will prove, by induction on i, that

$$OPT - \mathbf{E}\left[\sum_{t=1}^{T} G_i(A_t, B_t)\right] \leq \left(1 - \frac{1}{k}\right)^i OPT + iR(T). \tag{5}$$

The theorem follows by taking $i = k$ and using the inequality $\left(1 - \frac{1}{k}\right)^k < \frac{1}{e}$.

In the base case $i = 0$, inequality (5) is trivial. For the induction step, let

$$Z_i = OPT - \mathbf{E}\left[\sum_{t=1}^{T} G_i(A_t, B_t)\right].$$

We have

$$Z_i = Z_{i-1} - \mathbf{E}\left[\sum_{t=1}^{T} g_i(A_t, B_t)\right], \tag{6}$$

and Lemma 1 says that

$$\mathbf{E}\left[\sum_{t=1}^{T} g_i(A_t, B_t)\right] \geq \frac{1}{k} Z_{i-1} - R(T). \tag{7}$$

Combining (6) with (7), we obtain

$$Z_i \leq \left(1 - \frac{1}{k}\right) Z_{i-1} + R(T).$$

Combining this with the induction hypothesis proves (5). □

The general case, in which user u_i's type vector $(p_{ij})_{j \in \mathcal{D}}$ is an arbitrary element of $[0, 1]^{\mathcal{D}}$, can be reduced via a simple transformation to the case of deterministic users analyzed above. We replace user u_i with a *random* deterministic user $\hat{u}_i$ whose type vector $\hat{p}_i \in \{0, 1\}^{\mathcal{D}}$ is sampled using the following rule: the random variable $\hat{p}_{ij}$ has distribution

$$\hat{p}_{ij} = \begin{cases} 1 & \text{with probability } p_{ij} \\ 0 & \text{with probability } 1 - p_{ij}, \end{cases}$$

and these random variables are mutually independent. Note that the clicking behavior of user u_i when presented with a ranking B is *identical* to the clicking behavior observed when a random user type $\hat{u}_i$ is sampled from the above distribution, and the ranking B is presented to this random user. Thus, if we apply the specified transformation to users $u_1, u_2, \ldots, u_T$, obtaining a random sequence $\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_T$ of deterministic users, this transformation changes neither the algorithm's expected payoff nor that of the optimum ranking B^*. Thus, Theorem 1 for general users can be deduced by applying the same theorem to the random sequence $\hat{u}_1, \ldots, \hat{u}_T$ and taking the expectation of the left and right sides of (4) over the random choices involved in sampling $\hat{u}_1, \ldots, \hat{u}_T$.

Note also that B^* is defined as the optimal *subset* of k documents, and OPT is the payoff of presenting B^*, without specifying the order in which documents are presented. However, the Ranked Bandits Algorithm learns an order for the documents in addition to identifying a set of documents. In particular, given $k' < k$, RBA(k') would receive exactly the same feedback as the first k' instances of MAB_i receive when running RBA(k). Hence any k' sized prefix of the learned ranking also has the same performance bound with respect the appropriate smaller set B'^*.

Finally, it is worth noting that this analysis *cannot* be trivially extended to non-binary payoffs, for example when learning a ranking of web advertisements. In particular, the greedy algorithm on which RBA is based in the non-binary payoff case can obtain a payoff that is a factor of $k - \varepsilon$ below optimal, for any $\varepsilon > 0$.

5.3. Analysis of Ranked Explore and Commit

The analysis of the Ranked Explore and Commit (REC) algorithm is analogous to that of the Ranked Bandits algorithm, except that the equivalents of Lemma 1 and Theorem 1 are only true with high probability after $t_0 = nxk$ time steps of exploration have occurred. Let B denote the ranking selected by REC.

Lemma 2. *Let $x = 2k^2/\epsilon^2 \log(2k/\delta)$. Assume A_t is drawn i.i.d. from a fixed distribution of user types. For any i, with probability $1 - \delta/k$,*

$$\mathbf{E}\left[\sum_{t=t_0}^{T} g_i(A_t, B)\right] \geq \frac{1}{k}\mathbf{E}\left[\sum_{t=t_0}^{T} (G_k(A_t, B^*) - G_{i-1}(A_t, B))\right] - \frac{\epsilon}{k}T.$$

Proof Outline. First note that in this setting, B^* and OPT are defined in expectation over the A_t drawn. For any document, by Hoeffding's inequality, with probability $1 - \delta/2k$ the true payoff of that document explored at rank i is within $\epsilon/2k$ of the observed mean payoff. Hence the document selected at rank i is within ϵ/k of the payoff of the best document available at rank i. Now, the same proof as for Lemma 1 applies, although with a different regret $R(T)$. □

Theorem 2. *With probability $(1 - \delta)$, the algorithm's combined payoff after T rounds satisfies:*

$$\mathbf{E}\left[\sum_{t=1}^{T} G_k(A_t, B)\right] \geq \left(1 - \frac{1}{e}\right) OPT - \epsilon T - nkx \quad (8)$$

Proof Outline. Applying Lemma 2 for all $i \in \{1, .., k\}$, with probability $(1 - k\delta/k) = (1 - \delta)$ the conclusion of the Lemma holds for all i.

Next, an analogous proof as for Theorem 1 applies, except replacing $R(T)$ with $\frac{\epsilon}{k}T$ and noting that the regret during the nkx exploration steps is at most 1 for every time step. □

It is interesting to note that, in contrast to the Ranked Bandits Algorithm, this algorithm can be adapted to the case where clicked documents provide real valued payoffs. The only modification necessary is that documents should always be presented by decreasing payoff value. However, we do not address this extension further due to space constraints.

6. Evaluation

In this section, we evaluate the Ranked Bandits and Ranked Explore and Commit algorithms, as well as two variants of RBA, with simulations using a user and document model.

We chose a model that produces a user population and document distribution designed to be realistic yet allow us to evaluate the performance of the presented algorithms under different levels of noise in user clicking behavior. Our model first assigns each of 20 users to topics of interest using a Chinese Restaurant Process (Aldous, 1985) with parameter $\theta = 3$. This led to a mean of 6.5 unique topics, with topic popularity decaying according to a power law. Taking a collection of 50 documents, we then randomly assigned as many documents to each topic as there were users assigned to the topic, leading to topics with more users having more relevant documents. We set each document assigned to a topic as relevant to all users assigned to that topic, and all other documents as non relevant. The probabilities of a user clicking on relevant and non-relevant documents were set to constants p_R and p_{NR} respectively.

We tested by drawing one user uniformly from the user population at each time step, and presented this user with the ranking selected by each algorithm, using $k = 5$. We report the average number of time steps where the user clicked on a result, and the average number of time steps where at least one of the presented documents was relevant to the user. All numbers we report are averages over 1,000 algorithm runs.

6.1. Performance Without Click Noise

We start by evaluating how well the REC and RBA algorithms maximize the clickthrough rate in the simplest case when $p_R = 1$ and $p_{NR} = 0$. We also compare their performance to the clickthrough rate that the same users would generate if presented with a static system that orders documents by decreasing true prob-

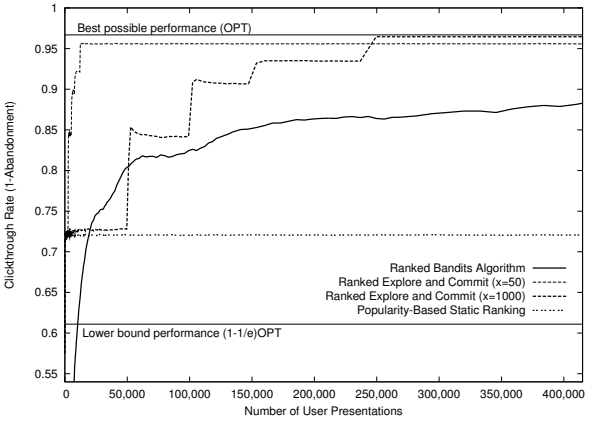


Figure 1. Clickthrough rate of the learned ranking as a function of the number of times the ranking was presented to users.

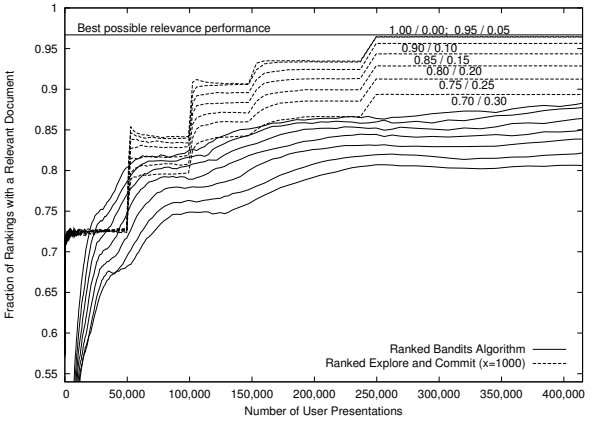


Figure 2. Effect of noise in clicking behavior on the quality of the learned ranking.

ability of relevance to the users assuming document relevances are independent. Figure 1 shows that both REC and RBA perform well above the static baseline and well above the performance guarantee provided by the theoretical results. This is not surprising, as the $(1 - 1/e)OPT$ bound is a worst-case bound. In fact, we see that REC with $x = 1000$ nearly matches the performance of the best possible ranking after finishing its initial exploration phase. We also see that the exploration parameter of REC plays a significant role in the performance, with lower exploration leading to faster convergence but slightly lower final performance. Note that despite REC performing best here, the ranking learned by REC is fixed after the exploration steps have been performed. If user interests and documents change over time, the performance of REC could fall arbitrarily. In contrast, RBA is guaranteed to remain near or above the $(1 - 1/e)OPT$ bound.

6.2. Effect of Click Noise

In Figure 1, the clickthrough rate and fraction of users who found a relevant document in the top k positions is identical (since users click if and only if they are presented with a relevant document). In contrast, Figure 2 shows how the fraction of users who find a relevant document decays as the probability of a user clicking becomes noisier. The figure presents the performance lines for REC and RBA across a range of click probabilities, from $(p_R = 1, p_{NR} = 0)$ to $(p_R = 0.7, p_{NR} = 0.3)$. We see that both algorithms decay gracefully: as the clicks become noisier noisy, the fraction of users presented with a relevant documents decays slowly.

6.3. Optimizing Practical Effectiveness

Despite the theoretical results shown earlier, it would be surprising if an algorithm designed for the worst case had best average case performance. Figure 3 shows the clickthrough rate (which the algorithms optimize), and fraction of users who find relevant documents (which is of more interest to information retrieval practitioners), for variants building on the insights of the ranked bandits idea. Specifically, two variants of RBA that have the best performance we could obtain in our simulation are shown. We found that using a UCB1-based multi-armed bandit algorithm (Auer et al., 2002a) in place of EXP3 improves the performance of RBA substantially when user interests are static. Note however, that UCB1 does not satisfy the constraints presented in Section 4.2 because it assumes rewards are identically distributed over time, an assumption violated in our setting when changes in the documents presented above rank i alter the reward distribution at rank i. Nevertheless, we see that this modification substantially improves the performance of RBA. We expect such an algorithm to perform best when few documents are prone to radical shifts in popularity.

7. Conclusions and Extensions

We have presented a new formulation of the learning to rank problem that explicitly takes into account the relevance of different documents being interdependent. We presented, analyzed and evaluated two algorithms and two variants for this learning setting. We have shown that the learning problem can be solved in a theoretically sound manner, and that our algorithms can be expected to perform reasonably in practice.

We plan to extend this work by addressing the non-binary document relevance settings, and perform empirical evaluations using real users and real documents. Furthermore, we plan to investigate how prior knowledge can be incorporated into the algorithms to improve speed of convergence. Finally, we plan to inves-

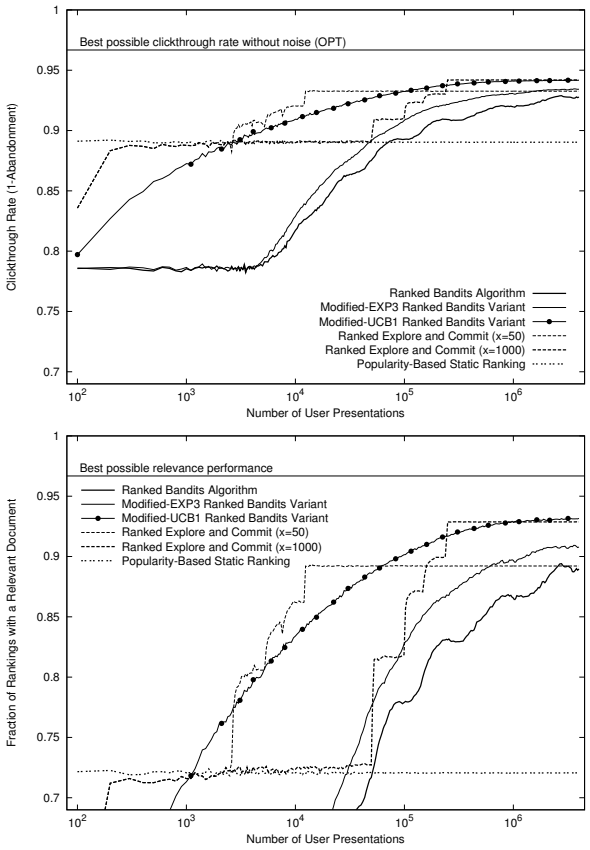


Figure 3. In a practical setting, it may be beneficial to use a variant of RBA to obtain improved performance at the cost of weaker theoretical guarantees. Performance is shown in realistic settings $p_R = 0.8$, $p_{NR} = 0.2$.

tigate if the bandits at different ranks can be coupled to improve the rate at which RBA converges.

Acknowledgments

We would like to thank the reviewers for helpful comments. This work was supported by NSF Career Award CCF-0643934, NSF Award CCF-0729102, NSF Career Award 0237381 and a gift from Google. The first author was supported by a Microsoft Research Fellowship.

References

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior. *In SIGIR* (pp. 19–26).

Aldous, D. J. (1985). Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII* (pp. 1–198).

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, *47*, 235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2002b). The non-stochastic multi-armed bandit problem. *SIAM Journal of Computing*, *32*, 48–77.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. New York, NY: Addison Wesley.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *In ICML* (pp. 89–96).

Burges, C. J. C., Ragno, R., & Le, Q. V. (2006). Learning to rank with nonsmooth cost functions. *In NIPS* (pp. 193–200). MIT Press.

Carbonell, J., & Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. *In SIGIR* (pp. 335–336).

Chen, H., & Karger, D. R. (2006). Less is more: Probabilistic models for retrieving fewer relevant documents. *In SIGIR* (pp. 429–436).

Chu, W., & Ghahramani, Z. (2005). Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, *6*, 1019–1041.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers* (pp. 115–132).

Joachims, T. (2002). Optimizing search engines using clickthrough data. *In KDD* (pp. 132–142).

Khuller, S., Moss, A., & Naor, J. (1997). The budgeted maximum coverage problem. *Information Processing Letters*, *70*, 39–45.

Metzler, D., & Croft, W. B. (2005). A markov random field model for term dependencies. *In SIGIR* (pp. 472–479).

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximation for maximizing submodular set functions. *Mathematical Programming*, *14*, 265–294.

Radlinski, F., & Joachims, T. (2005). Query chains: Learning to rank from implicit feedback. *In KDD* (pp. 239–248).

Radlinski, F., & Joachims, T. (2007). Active exploration for learning rankings from clickthrough data. *In KDD* (pp. 570–579).

Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation*, *33*, 294–304.

Streeter, M., & Golovin, D. (2007). *An online algorithm for maximizing submodular functions* (Technical Report CMU-CS-07-171). Carnegie Mellon University.

Taylor, M. J., Guiver, J., Robertson, S. E., & Minka, T. (2008). Softrank: Optimizing non-smooth ranking metrics. *In WSDM* (pp. 77–86).

Teevan, J., Dumais, S. T., & Horvitz, E. (2007). Characterizing the value of personalizing search. *In SIGIR* (pp. 757–758).

Turpin, A., & Scholer, F. (2006). User performance versus precision measures for simple search tasks. *In SIGIR* (pp. 11–18).

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimizing average precision. *In SIGIR* (pp. 271–278).

Zhai, C., Cohen, W. W., & Lafferty, J. (2003). Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. *In SIGIR* (pp. 10–17).

Zhang, B., Li, H., Liu, Y., Ji, L., Xi, W., Fan, W., Chen, Z., & Ma, W.-Y. (2005). Improving web search results using affinity graph. *In CIKM* (pp. 504–511).

Zhu, X., Goldberg, A. B., Gael, J. V., & Andrzejewski, D. (2007). Improving diversity in ranking using absorbing random walks. *Proceedings of NAACL HLT*.

Semi-supervised Learning of Compact Document Representations with Deep Networks

Marc'Aurelio Ranzato RANZATO@COURANT.NYU.EDU
Courant Institute, New York University, 719 Broadway 12th fl., New York NY 10003, USA

Martin Szummer SZUMMER@MICROSOFT.COM
Microsoft Research Cambridge, 7 J J Thomson Avenue, Cambridge CB3 0FB, UK

Abstract

Finding good representations of text documents is crucial in information retrieval and classification systems. Today the most popular document representation is based on a vector of word counts in the document. This representation neither captures dependencies between related words, nor handles synonyms or polysemous words. In this paper, we propose an algorithm to learn text document representations based on semi-supervised autoencoders that are stacked to form a deep network. The model can be trained efficiently on partially labeled corpora, producing very compact representations of documents, while retaining as much class information and joint word statistics as possible. We show that it is advantageous to exploit even a few labeled samples during training.

1. Introduction

Document representations are a key ingredient in all information retrieval and processing systems. The goal of the representation is to make certain aspects of the document readily accessible, e.g. the document topic. To identify a document topic, we cannot rely on specific words in the document, as it may use other synonymous words or misspellings. Likewise, the presence of a word does not warrant that the document is related to it, as it may be taken out of context, or polysemous, or unimportant to the document topic.

The most widespread representations for document classification and retrieval today are based on a vector of counts. These include various term-weighting retrieval schemes, such as tf-idf and BM25 (Robertson and Walker, 1994), and bag-of-words generative models such as naive Bayes text classifiers. The pertinent feature of these representations is that they represent individual words. A serious drawback of the basic tf-idf and BM25 representations is that all dimensions are treated as independent, whereas in reality word occurrences are highly correlated.

There have been many attempts at modeling word correlations by rotating the vector space and projecting documents onto principal axes that expose related words. Methods include LSI (Deerwester et al., 1990) and pLSI (Hofmann, 1999). These methods constitute a linear re-mapping of the original vector space, and while an improvement, still can only capture very limited relations between words. As a result they need a large number of projections in order to give an appropriate representation.

Other models, such as LDA (Blei et al., 2003), have shown superior performance over pLSI and LSI. However, inferring the representation is computationally expensive because of the "explaining away" effect that plagues all directed graphical models.

More recently, a number of authors have proposed undirected graphical models that can make inference efficient at the cost of more complex learning due to a global (rather than local) partition function whose exact gradient is intractable. These models build on Restricted Boltzmann Machines (RBMs) by adapting the conditional distribution of the input visible units to model discrete counts of words (Hinton and Salakhutdinov, 2006; Gehler et al., 2006; Salakhutdinov and Hinton, 2007a,b). These models have shown state-of-the-art performance in retrieval and clustering, and can be easily used as a building block for deep multi-layer networks (Hinton et al., 2006). This might allow the

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

top-level representation to capture high-order correlations that would be difficult to efficiently represent with similar but shallow models (Bengio and LeCun, 2007). Many authors have pointed out that RBMs are robust to uncorrelated noise in the input since they model the distribution of the input data, and they implicitly perform automatic model selection by not using unnecessary hidden units. But they are also somewhat cumbersome to train, relying on two disparate steps: unsupervised pre-training using an approximate sampling technique such as contrastive divergence (Hinton, 2000), followed by supervised back-propagation. It is rather difficult to predict when training can be stopped and how long the Markov Chain has to run. An alternative is to replace RBMs with autoencoders (Bengio et al., 2006), or special autoencoders that produce sparse representations (Ranzato et al., 2007b). According to these authors, the performance of RBMs and standard autoencoders is quite similar as long as the dimensionality of the latent space is smaller than the input. Seeking an algorithm that can be trained efficiently, and that can produce a representation with just a few matrix multiplications, we propose a deep network whose building blocks are autoencoders, with a specially designed first layer for modeling discrete counts of words.

Previously, deep networks have been trained either from fully labeled data, or purely unlabeled data. Neither method is ideal, as it is expensive to label large collections, whereas purely unsupervised learning may not capture the relevant class information in the data. Inspired by the experiments by Bengio, Lamblin et al. (2006), we learn the parameters of the model by using *both a supervised and an unsupervised objective*. In other words, we require the representation to produce good reconstructions of the input documents and, at the same time, to give good predictions of the document class labels. Besides demonstrating better accuracy in retrieval, we also extend the deep network framework to a *semi-supervised* setting where we deal with partially labeled collections of documents. This allows us to use relatively few labeled documents yet leverage language structure learned from large corpora, see Sec. 3.1.

Finally, we study the relative advantages of different deep models. For instance, we investigate when deep models are better than shallow ones. Our experiments in Sec. 3.2 show that for learning compact representations of documents, deep architectures greatly outperform shallow models. Compact representations are beneficial because they require less storage (an important consideration for large search engines), and they are more computationally efficient when used in indexing. We also explored the possibility to use deep networks to learn binary *high-dimensional* representations instead of *compact* representations. These high-dimensional representations were trained using the Symmetric Encoding Sparse Machine (SESM) (Ranzato et al., 2007b). However, the compact representations proved to be far more efficient in terms of memory usage and CPU time, as described in Sec. 3.3. Also, training is more computationally efficient than for related models such as RBMs.

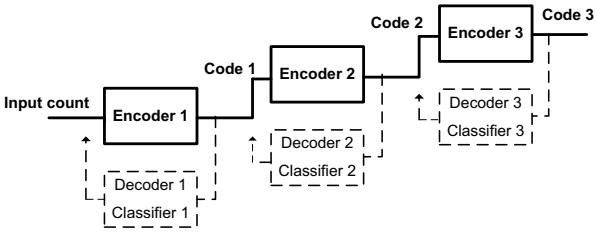


Figure 1. Architecture of a model with three stages. The system is trained layer by layer. During the training of the n-th layer, the n-th encoder is coupled with the n-th decoder and classifier (shown in dashed line). The n-th encoder will provide the codes to train the layer above. After training, the feedback decoding modules are discarded and the system is used to produce very compact codes by a feed-forward pass through the chain of encoders.

2. The model

The input to the system is a bag of words representation of each text document in the form of a count vector. The length of the vector equals the number of unique words in the collection, and its i-th entry stores the number of times the corresponding word occurs in the document. The goal of the system is to extract a *compact* representation from this very high-dimensional but sparse input vector. A compact representation is good because it requires less storage, and allows fast index lookup. Since the representation is produced by a deep multi-layer model, it can efficiently discover latent topics by grouping similar words and by activating features whenever some "interesting" combination of words is detected (see visualization in Sec. 3.4).

We propose a system that is composed of multiple layers. Each layer computes a weighted sum of its input followed by a logistic nonlinearity. Each layer can be seen as an *encoder* producing a representation, or *code*, from its input. This code will be propagated and used as the input to the next layer of the model. This architecture is quite similar to a neural network model, but is trained differently and has a special first layer able to encode discrete count data. The goal of training is to find the parameters in each layer.

In order to successfully learn the parameters we follow the strategy advocated by recent work (Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Bengio et al., 2006) on deep multi-layer models. Learning proceeds greedily layer by layer. When the parameters of one layer have been found, the data is fed through that layer and the output becomes the input for the next layer, which is trained subsequently.

Let us consider a generic layer in the model, and let $\boldsymbol{x}$ be its input and let $\boldsymbol{z}$ be the representation produced by the layer. In order to warrant the fidelity of the code $\boldsymbol{z}$, we attach a feedback module which aims to reconstruct the input $\boldsymbol{x}$ from the code $\boldsymbol{z}$. The reason is that if the model achieves a good reconstruction from the code $\boldsymbol{z}$, then we can be sure that the representation has preserved most of the information from $\boldsymbol{x}$. The original layer can then be interpreted as an *encoder* that computes a code from the input, while the feedback module can be seen as a *decoder* that reconstructs the input $\boldsymbol{x}$ from the code $\boldsymbol{z}$. Learning consists of minimizing a reconstruction error E_R with respect to the parameters in the encoder and decoder when the input $\boldsymbol{x}$ is drawn from the training dataset. Since we learn by stochastic gradient descent, any type of encoder and decoder is allowed as long as it is differentiable.

Some inputs may have labels specifying the class to which they belong. In order to incorporate this information, we add another module to the decoder. The feedback module now not only has a decoder reconstructing the input $\boldsymbol{x}$, but also a *classifier* predicting the label y from the code $\boldsymbol{z}$, see Fig. 1. During training the parameters of the encoder, classifier and decoder are learned by minimizing the loss

$$L = E_R + \alpha_c E_C, \tag{1}$$

where E_R and E_C are terms measuring the reconstruction and classification error respectively, and α_c is a coefficient balancing them. The first term is common to many unsupervised learning algorithms and makes the system model the structure and the dependencies among the input components of $\boldsymbol{x}$. The second term represents the supervised goal ensuring that codes are also going to be good for discriminating between classes.

For the classifier module we used a linear classifier trained by cross-entropy error E_C. Denoting with $(W_C)_i$ the i-th row of the classifier weight matrix, with b_{Ci} the i-th bias of the classifier, and with h_j the j-th output unit of the classifier passed through a soft-max:

$$h_j = \frac{\exp((W_C)_j \cdot \boldsymbol{z} + b_{Cj})}{\sum_i \exp((W_C)_i \cdot \boldsymbol{z} + b_{Ci})},$$

we define $E_C = -\sum_i y_i \log h_i$, where $\boldsymbol{y}$ is a 1-of-N encoding of the target class label.

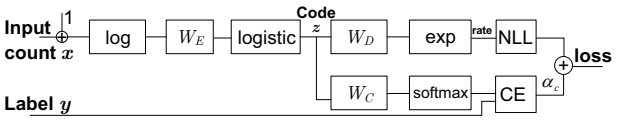


Figure 2. The architecture of the first stage has three components: (1) an encoder, (2) a decoder (Poisson regressor), and (3) a classifier. The loss is the weighted sum of cross-entropy (CE) and negative log-likelihood (NLL) under the Poisson model.

2.1. Training the First Stage

The first stage is special because the input $\boldsymbol{x}$ is a discrete vector of word counts, with x_i counting the number of occurrences of the i-th word in the document. The decoder is a *Poisson regression model* aiming to predict $\boldsymbol{x}$ from the code $\boldsymbol{z}$. A Poisson regressor is a log-linear model which assigns the following probability to an observed $\boldsymbol{x}$:

$$P(\boldsymbol{x}) = \prod_i P(x_i) = \prod_i \mathrm{e}^{-\lambda_i} \frac{\lambda_i^{x_i}}{x_i!}, \tag{2}$$

where the set of rates is given by $\lambda = \beta \mathrm{e}^{W_D \boldsymbol{z} + \boldsymbol{b}_D}$, with a decoder weight matrix W_D, decoder biases $\boldsymbol{b}_D$, and a constant β proportional to the document length. This normalization handles documents of different lengths and makes learning stable. The reconstruction error E_R minimized at the first stage (eq. 1) is the negative log-likelihood of the data

$$E_R = \sum_i (\beta \mathrm{e}^{((W_D)_i \cdot \boldsymbol{z} + b_{Di})} - x_i (W_D)_i \cdot \boldsymbol{z} - x_i b_{Di} + \log x_i!), \tag{3}$$

averaged over the samples $\boldsymbol{x}$ in the training dataset.

We design the encoder by "reverse-engineering" the decoder to make the machine symmetric. Since the decoder computes an exponential of a weighted sum, the encoder performs a weighted sum of the log-transformed input $\boldsymbol{x}$. In addition to this, the encoder applies a logistic nonlinearity. Hence, the code $\boldsymbol{z}$ is given by $\boldsymbol{z} = \boldsymbol{\sigma}(W_E \log(\boldsymbol{x}) + \boldsymbol{b}_E)$, where W_E and $\boldsymbol{b}_E$ are the weight matrix and the biases in the encoder, and $\boldsymbol{\sigma}$ is the logistic. Since many components in $\boldsymbol{x}$ are zero (because only a few dictionary words are actually present in a given document), and since the rate at which a word might appear is generally fairly low, this architecture is prone to numerical problems in the evaluation of the logarithm, and would possibly require large negative weights in W_D (in order to make the rate λ small). Thus, we shift the Poisson regression by adding one to the input. As a result, if a word does not occur in the document, the input to the encoder weight matrix W_E will be zero (and not minus infinity), and

if a word is rare its rate will be one forcing the corresponding weights in W_D to be close to zero (and not to minus infinity). Fig. 2 shows the final architecture of the first stage.

2.2. Training the Upper Stages

The outputs of earlier layers are fed as inputs of subsequent layers. The architecture of the subsequent layers differs from the first one in that the decoder uses a Gaussian regressor instead of a Poisson regressor. Accordingly, the encoder computes a weighted sum of its input and applies a logistic nonlinearity. This architecture is similar to an autoencoder neural network, but here the feedback layer also includes a supervised classifier. If $\boldsymbol{z}^{(n-1)}$ is the input to the n-th layer, the code $\boldsymbol{z}^{(n)}$ produced at this stage is $\boldsymbol{z}^{(n)} = \boldsymbol{\sigma}(W_E \boldsymbol{z}^{(n-1)} + \boldsymbol{b}_E)$. The reconstruction error E_R in the loss of eq. 1 can be written as $E_R = \|\boldsymbol{z}^{(n-1)} - W_D \boldsymbol{z}^{(n)} - \boldsymbol{b}_D\|_2^2$.

2.3. Training the Whole Model

Learning consists of determining the parameters at each layer of the deep model. The algorithm proceeds as follows:
(1) attach a Poisson regressor and a linear classifier to the first layer, and minimize the loss in eq. 1 with respect to the parameters $(W_E, b_E, W_D, b_D, W_C, b_C)$ by stochastic gradient descent;
(2) transform the training samples $\boldsymbol{x}$ into codes $\boldsymbol{z}^{(1)}$ using the trained encoder of the first layer;
(3) train the second layer by attaching a Gaussian regressor and a linear classifier to the encoder, using the codes $\boldsymbol{z}^{(1)}$ as input;
(4) use the trained encoder of the second layer to transform the codes $\boldsymbol{z}^{(1)}$ into the higher-level codes $\boldsymbol{z}^{(2)}$;
(5) repeat the previous two steps for as many layers as desired.

When the input sample is not accompanied by a label, the classifier is not updated and the loss function simply reduces to $L = E_R$. In order to minimize the loss with respect to the parameters we use stochastic gradient descent and we back-propagate the derivatives through the decoder, classifier and encoder (LeCun et al., 1998).

The learning algorithm is particularly efficient. The computational cost of learning is linear in the number of training samples (sublinear for redundant datasets, which are frequent). For each training document at any given layer, the cost is given by a forward and backward pass through encoder, decoder and classifier. Each pass is dominated by a matrix-vector multiplication whose complexity depends on the size of the matrix. Since at each layer we reduce the dimensionality of the input, the first layer dominates the computational cost. However, the sparsity of the input count vector can be exploited to speed-up the computation by taking into account only those rows in W_E that are involved in the computation. In general, the computational cost at a given layer scales as $4MN + 2NK$, where M is the dimensionality of the input, N is the dimensionality of the code, and K is the number of classes.

If we are interested in classification we can also use the trained classifier to predict the labels from the features (at any layer), without training a separate supervised system (see Sec. 3.1 for an example). Also, our experiments show that there is not much advantage in "fine-tuning" the parameters by doing global non-greedy supervised training of the machine as performed by Hinton et al. (2006). The label injection during the greedy training of each layer renders this final supervised training stage unnecessary. This saves a lot of time because it is expensive to do forward and backward propagation through a large and deep network.

Inference is also very efficient. Once the model is trained the encoders are stacked and the decoder and classifier modules are removed. A feature vector is computed by a forward propagation of the input sparse count vector through the sequence of encoders. This computation requires a few matrix vector multiplications, where the most expensive one is at the first layer, which can benefit further from a sparse computation.

3. Experiments

In our experiments we considered three standard datasets: 20 Newsgroups, Reuters-21578, and Ohsumed[1]. The 20 Newsgroups dataset contains 18845 postings taken from the Usenet newsgroup collection. Documents are partitioned into 20 topics. The dataset is split into 11314 training documents and 7531 test documents. Training and test articles are separated in time. Reuters has a predefined ModApte split of the data into 11413 training documents and 4024 test documents. Documents belong to one of 91 topics. The Ohsumed dataset has 34389 documents with 30689 words and each document might be assigned to more than one topic, for a total of 23 topics. The dataset is split into training and test by randomly selecting the 67% and the 33% of the data. Rainbow[2] was used to pre-process these datasets by stemming the documents,

[1] These corpora were downloaded from `http://people.csail.mit.edu/jrennie/20Newsgroups`, and `http://www.kyb.mpg.de/bs/people/pgehler/rap`

[2] Rainbow is available at `http://www.cs.cmu.edu/~mccallum/bow/rainbow`

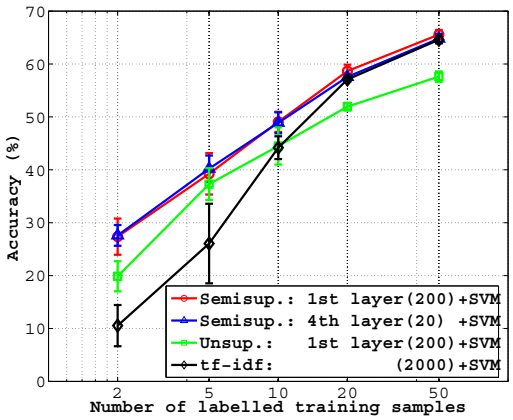


Figure 3. SVM classification of documents from the 20 Newsgroups dataset (2000 word vocabulary) trained with between 2 and 50 labeled samples per class. The SVM was applied to representations from the deep model trained in a semi-supervised or unsupervised way, and to the tf-idf representation. The numbers in parentheses denote the number of code units. Error bars indicate one standard deviation. The fourth layer representation has only 20 units, and is much more compact and computationally efficient than all the other representations.

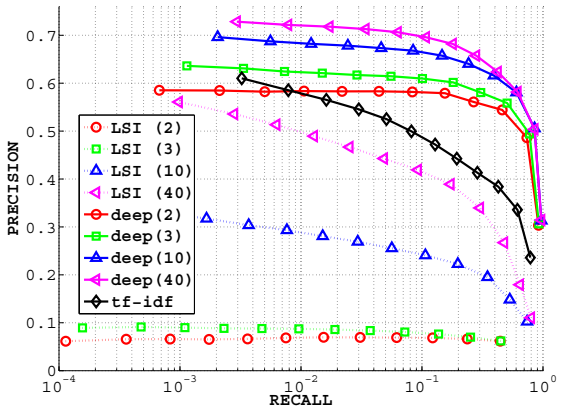


Figure 4. Precision-recall curves for the Reuters dataset comparing a linear model (LSI) to the nonlinear deep model with the same number of code units (in parentheses). Retrieval is done using the k most similar documents according to cosine similarity, with $k \in [1 \dots 4095]$.

removing stop words and words appearing less than three times or in only a single document, and retaining between 1000 and 30,000 words with the highest mutual information.

Unless stated otherwise, we trained each layer of the network for only 4 epochs over the whole training dataset. Convergence took only a couple of epochs, and was robust to the choice of the learning rate. This was set to about 10^{-4} when training the first layer, and to 10^{-3} when training the layers above. The learning rate was exponentially decreased by multiplying it by 0.97 every 1000 samples. A small L1 regularizer on the parameters was added to the loss. Each weight was randomly initialized, and was updated by taking a gradient step with a regularizer given by the value of the learning rate times $5 \cdot 10^{-4}$ the sign of the weight. The value of α_c in eq. 1 was set to the ratio between the number of input units in the layer and the number of classes in order to make the two error terms E_R and E_C comparable. Its exact value did not affect the performance as long as it had the right order of magnitude.

3.1. The Value of Labels

In order to assess whether semi-supervised training was better than purely unsupervised training, we trained the deep model on the 20 Newsgroup dataset using only 2, 5, 10, 20 and 50 samples per class. During training we showed the system 10 labeled samples every 100 examples by sweeping more often over the labeled data. This procedure was repeated at each layer during training. We trained 4 layers for 10 epochs with an architecture of 2000-200-100-50-20, denoting 2000 inputs, 200 hidden units at the first layer, 100 at the second, 50 at the third, and 20 at the fourth. Then, we trained a Support Vector Machine[3] (SVM) with a Gaussian kernel on (1) the codes that corresponded to the labeled documents, and we compared the accuracy of the semi-supervised model to the one achieved by a Gaussian SVM trained on the features produced by (2) the same model but trained in an unsupervised way, and by (3) the tf-idf representation of the same labeled documents. The SVM was generally tuned by five-fold cross validation on the available labeled samples (but two-fold cross validation when using only two samples per class). Fig. 3 demonstrates that the learned features gave much better accuracy than the tf-idf representation overall when labeled data was scarce. The model was able to exploit the very few labeled samples producing features that were easier to discriminate. The performance actually improved when the dimensionality of the code was reduced and only 2 or 5 labeled samples per class were available, probably because a more compact code implicitly enforces a stronger regularization. Semi-supervised training outperformed unsupervised training, and the gap widened as we increased the number of labeled samples, indicat-

[3]We used libsvm package available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

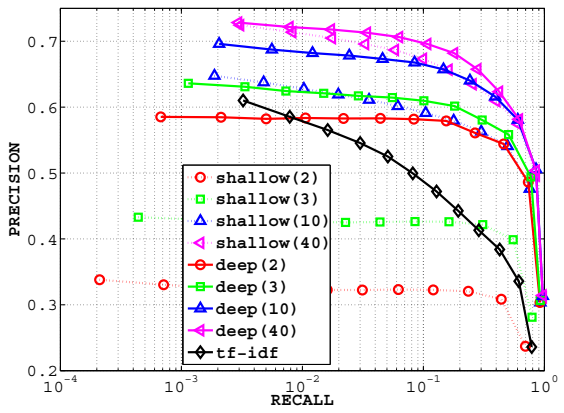


Figure 5. Precision-recall curves for the Reuters dataset comparing one-layer models (shallow) to deep models with the same numbers of code units. The deep models are more accurate overall when the codes are extremely compact. This also suggests that the number of hidden units has to be *gradually* decreased from layer to layer.

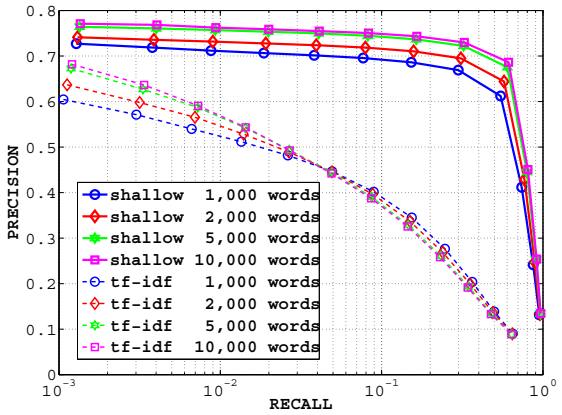


Figure 6. Precision-recall curves for the 20 Newsgroups dataset comparing the performance of tf-idf versus a one-layer shallow model for varying sizes of the word dictionary (from 1000 to 10000 words).

ing that the unsupervised method had failed to model information relevant for classification when compressing to a low-dimensional space.

Interestingly, if we classify the data using the classifier of the feedback module we obtain a performance similar to the one achieved by the Gaussian SVM. For example, when *all* training samples are labeled the classifier at the first stage achieves accuracy of 76.3% (as opposed to 75.5% of the SVM trained either on the learned representation or on tf-idf), while the one on the fourth layer achieves accuracy of 74.8%. Hence, the training algorithm provides an accurate classifier as a side product of the training, reducing the overall learning time.

3.2. Deep or Shallow?

In all the experiments discussed in this section the model was trained using fully labeled data (still, training also includes an unsupervised objective as discussed earlier). In order to retrieve documents after training the model, all documents are mapped into the latent low-dimensional space, the cosine similarity between each document in the test dataset and each document in the training dataset is measured, and the k most similar documents are retrieved. k is chosen to be equal to 1, 3, 7, ..., 4095. Based on the topic label of the documents, we assess the performance by computing the *recall* and the *precision* averaged over the whole test dataset.

In the first experiment, we compared the linear mapping produced by LSI to the nonlinear mapping produced by our model. We considered the Reuters dataset with a 12317 word vocabulary and trained a network with 3 layers. The first layer had 100 code units, the second layer had 40 units in one experiment and 10 in another, the third layer was trained with either 3 or 2 code units. As shown in Fig. 4, the nonlinear representation is more powerful than the linear one, when the representation is very compact.

Another interesting question is whether adding layers is useful. Fig. 5 shows that for a given dimensionality of the output latent space the deep architecture outperforms the shallow one. The deep architecture is capable of capturing more complex dependencies among the input variables than the shallow one, while the representation remains compact. The compactness allows us to efficiently handle very large vocabularies (more than 30,000 words for the Ohsumed, see Sec. 3.4). Fig. 6 shows that increasing the number of words (i.e. the dimensionality of the input) does give better retrieval performance.

3.3. Compact or Binary High-Dimensional?

The most popular representation of documents is tf-idf, a very high-dimensional and sparse representation. One might wonder whether we should learn a high-dimensional representation instead of a compact representation. Unfortunately, the autoencoder based learning algorithm forces us to map data into a lower-dimensional space at each layer, as without additional constraints (Ranzato et al., 2007a) the trivial identity function would be learned. We used the sparse encod-

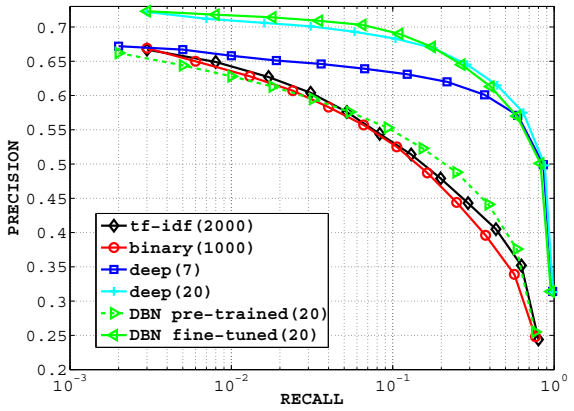


Figure 7. Precision-recall curves comparing compact representations vs. high-dimensional binary representations. Compact representations can achieve better performance using less memory and CPU time.

ing symmetric machine (SESM) (Ranzato et al., 2007b) as a building block for training a deep network producing sparse features. SESM is a symmetric autoencoder with a sparsity constraint on the representation, and it is trained unsupervised. In order to make the sparse representation at the final layer computationally appealing we thresholded it to make it binary. We trained a 2000-1000-1000 SESM network on the Reuters dataset. In order to make a fair comparison with our compact representation, we fixed the information content of the code in terms of precision[4] at $k = 1$. We measured the precision and recall of the binary representation of a test document by computing its Hamming distance from the representation of the training documents. We then trained our model with the following number of units 2000-200-100-7. The last number of units was set to match the precision of the binary representation at $k = 1$. Fig. 7 shows that our compact representation outperforms the high-dimensional and binary representation at higher values of k. Just 7 continuous units are able to achieve better retrieval than 1000 binary units! Storing the Reuters dataset with the compact representation takes less than half the memory space than using the binary representation, and comparing a test document against the whole training dataset is five times faster with the compact representation. The best accuracy for our model is given with a 20-unit representation. Fig. 7 shows the performance of a representation with the same number of units learned by a deep belief network (DBN) following Salakhutdinov and Hinton's constrained Poisson model (2007). Their model was greedily pre-trained for one epoch in an unsupervised way (200 pre-training epochs gave similar fine-tuned accuracy), and then fine-tuned with supervision for 100 epochs. While fine-tuning does not help our model, it significantly improves the DBN which eventually achieves the same accuracy as our model. Despite the similar accuracy, the computational cost of training a DBN (with our implementation using conjugate gradient on mini-batches) is several times higher due to this supervised training through a large and deep network. By looking at how words are mapped to the top-level feature space, we can get an intuition about the learned mapping. For instance, the code closest to the representation of the word "jakarta" corresponds to the word "indonesia", similarly,"meat" is closest to "beef" (table 1). As expected, the model implicitly clusters synonymous and related words.

[4]The entropy of the representation would be more natural, but its value depends on the quantization level.

Table 1. Neighboring word stems for the model trained on Reuters. The number of units is 2000-200-100-7.

Word stem	Neighboring word stems
livestock	beef, meat, pork, cattle
lend	rate, debt, bond, downgrad
acquisit	merger, stake, takeov
port	ship, port, vessel, freight
branch	stake, merger, takeov, acquisit
plantat	coffe, cocoa, rubber, palm
barrel	oil, crude, opec, refineri
subcommitte	bill, trade, bond, committe
coconut	soybean, wheat, corn, grain
meat	beef, pork, cattl, hog
ghana	cocoa, buffer, coffe, icco
varieti	wheat, grain, agricultur, crop
warship	ship, freight, vessel, tanker
edibl	beef, pork, meat, poultri

3.4. Visualization

The deep model can also be used to visualize documents. When the top layer is two-dimensional we can visualize high-dimensional nonlinear manifolds in the space of bags of words. Fig. 8 shows how documents in the Ohsumed test set are mapped to the plane. The model exposes clusters documents according to the topic class, and places similar topics next to each other. The dimensionality reduction is extreme in this case, from more than 30000 to 2.

4. Conclusions

We have proposed and demonstrated a simple and efficient algorithm to learn document representations from

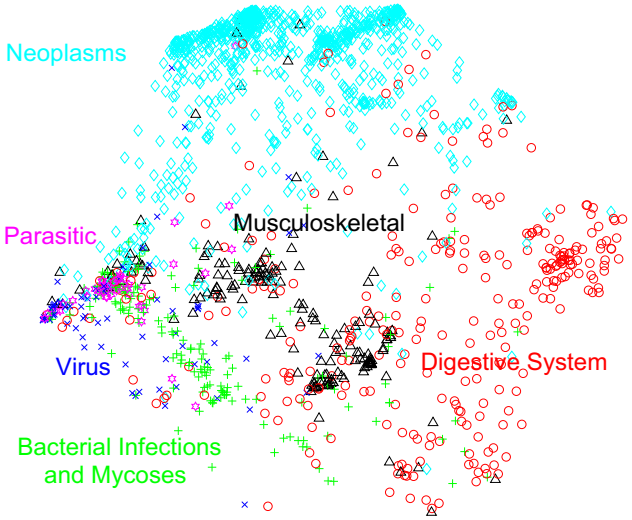


Figure 8. Two-dimensional codes produced by the deep model 30689-100-10-5-2 trained on the Ohsumed dataset (only the 6 most numerous classes are shown). The codes results from propagating documents in the test set through the four-layer network.

partially labeled datasets. The representation is rich in that it can model complex dependencies between words, which allows us to capture higher-level semantic aspects of documents than is possible with linear models. Capturing such complex structure would not be possible based on labeled data alone; by leveraging unlabeled documents we get access to a much larger amount data.

This algorithm trains faster than a similar model based on RBMs, and it finds more efficient representations than a network trained with SESMs that produce high-dimensional binary features. We have shown that these deep models greatly outperform similar but shallow models when the learning task is very hard, such as learning very compact representations. Compact representations are very important for search engines because they are cheap to store, and fast to compute and to compare. Also, we have shown that even a few labels can be exploited to make the features more discriminative.

For future work, we are interested in applying the representation for clustering and ranking. It would also be interesting to go beyond the bag of words model to capture word proximity.

Acknowledgments

The authors would like to thank Y. LeCun for his insights and suggestions.

References

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. In *NIPS*.

Bengio, Y. and LeCun, Y. (2007). *Scaling learning algorithms towards AI*. MIT press.

Blei, D., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet allocation. *JMLR*.

Deerwester, S., Dumais, S., Landauer, T., Furnas, G., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journ. of American Society of Information Science*, 41:391–407.

Gehler, P. V., Holub, A. D., and Welling, M. (2006). The rate adapting Poisson model for information retrieval and object recognition. In *ICML*.

Hinton, G. (2000). Training products of experts by minimizing contrastive divergence. Technical report, U. Toronto.

Hinton, G., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554.

Hinton, G. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hofmann, T. (1999). Probabilisitc latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence*.

LeCun, Y., Bottou, L., Orr, G., and Muller, K. (1998). Efficient backprop. In Orr, G. and K., M., editors, *Neural Networks: Tricks of the trade*. Springer.

Ranzato, M., Boureau, Y., Chopra, S., and LeCun, Y. (2007a). A unified energy-based framework for unsupervised learning. In *AI-STATS*.

Ranzato, M., Boureau, Y., and LeCun, Y. (2007b). Sparse feature learning for deep belief networks. In *NIPS*. MIT Press.

Robertson, S. and Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proc. ACM SIGIR*, pages 232–241.

Salakhutdinov, R. and Hinton, G. (2007a). Semantic hashing. In *ACM SIGIR workshop on Information Retrieval and Applications of Graphical Models*.

Salakhutdinov, R. and Hinton, G. (2007b). Using deep belief nets to learn covariance kernels for gaussian processes. In *NIPS 20*. MIT Press.

Message-passing for Graph-structured Linear Programs: Proximal Projections, Convergence and Rounding Schemes

Pradeep Ravikumar[†] PRADEEPR@STAT.BERKELEY.EDU
Alekh Agarwal[♯] ALEKH@EECS.BERKELEY.EDU
Martin J. Wainwright[†,♯] WAINWRIG@STAT.BERKELEY.EDU
Department of Statistics[†] and Department of Electrical Engineering and Computer Sciences[♯]
University of California, Berkeley

Abstract

A large body of past work has focused on the first-order tree-based LP relaxation for the MAP problem in Markov random fields. This paper develops a family of super-linearly convergent LP solvers based on proximal minimization schemes using Bregman divergences that exploit the underlying graphical structure, and so scale well to large problems. All of our algorithms have a double-loop character, with the outer loop corresponding to the proximal sequence, and an inner loop of cyclic Bregman divergences used to compute each proximal update. The inner loop updates are distributed and respect the graph structure, and thus can be cast as message-passing algorithms. We establish various convergence guarantees for our algorithms, illustrate their performance, and also present rounding schemes with provable optimality guarantees.

1. Introduction

A key computational challenge associated with discrete Markov random fields (MRFs) is the problem of *maximum a posteriori* (MAP) estimation: computing the most probable configuration(s). For general graphs, this MAP problem includes a large number of classical NP-complete problems, including MAX-CUT independent set, and satisfiability problems, among various others.

This intractability motivates the development and analysis of methods for obtaining approximate solutions. The ordinary max-product algorithm is a form of non-serial dynamic-programming, exact for trees, and also widely used as a heuristic for obtaining approximate solutions to the MAP problem, but it suffers from convergence failures, and despite some local optimality results (Freeman & Weiss, 2001), it has no general correctness guarantees. For certain MRFs arising in computer vision, Boykov et al. (2001) studied graph-cut based search algorithms that compute a local maximum over two classes of moves. A related class of methods are those based on various types of convex relaxations, in which the discrete MAP problem is relaxed some type of convex optimization problem over continuous variables. Examples include linear programming (LP) relaxations (Wainwright et al., 2005; Chekuri et al., 2005), as well as quadratic, semidefinite and other conic programming relaxations (Ravikumar & Lafferty, 2006; Kumar et al., 2006; Wainwright & Jordan, 2003).

Among convex relaxations, LP relaxation is the least computationally expensive and best understood. The primary focus of this paper is a well-known tree-based LP relaxation (Chekuri et al., 2005; Wainwright et al., 2005) of the MAP estimation problem for pairwise Markov random fields, based on optimizing over a set of locally consistent pseudomarginals on edges and vertices of the graph. In principle, this LP relaxation can be solved by any standard solver, including simplex or interior-point methods (Bertsimas & Tsitsiklis, 1997). However, such generic methods fail to exploit the graph-structured nature of the LP, and hence do not scale favorably to large-scale problems.

Wainwright et al. (2005) established a connection between this tree-based LP relaxation and the class of tree-reweighted max-product (TRW-MP) algorithms, showing that suitable TRW-MP fixed points specify optimal solutions to the LP relaxation. Subsequent work has extended this basic connection in various in-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

teresting ways. For instance, Kolmogorov (2005) developed a serial form of TRW-MP with some convergence properties but as with the ordinary TRW-MP updates, no guarantees of LP optimality. Weiss et al. (2007) connected convex forms of sum-product and exactness of reweighted max-product algorithms. Globerson and Jaakkola (2007) developed a convergent dual-ascent algorithm, but its fixed points are guaranteed to be LP-optimal only for binary problems, as is also the case for the TRW-MP algorithm (Kolmogorov & Wainwright, 2005), and the rate of convergence is not analyzed. Other authors (Komodakis et al., 2007; Feldman et al., 2002) have proposed subgradient methods, but such methods typically have sub-linear convergence rates.

The goal of this paper is to develop and analyze various classes of message-passing algorithms that always solve the LP, and are provably convergent with at least a geometric rate. The methods that we develop are flexible, in that new constraints can be incorporated in a relatively seamless manner, with new messages introduced to enforce them. All of the algorithms in this paper are based on the notion of *proximal minimization*: instead of directly solving the original linear program itself, we solve a sequence of so-called proximal problems, with the property that the sequence of associated solutions is guaranteed to converge to the LP solution. We describe different classes of algorithms, based on different choices of the proximal function: quadratic, entropic, and reweighted Bethe entropies. For all choices, we show how the intermediate proximal problems can be solved by message-passing updates on the graph, guaranteed to converge but with a distributed nature that scales favorably. An additional desirable feature, given the wide variety of lifting methods for further constraining LP relaxations (Wainwright & Jordan, 2003), is that additional constraints are easily incorporated within the framework.

2. Background

We begin by introducing some background on Markov random fields, and the LP relaxations that are the focus of this paper. Given a discrete space $\mathcal{X} = \{0, 1, 2, \ldots, m\}$, let $X = \{X_1, \ldots, X_p\} \in \mathcal{X}^p$ denote a $p-$dimensional discrete random vector. We assume that its distribution $\mathbb{P}$ is a Markov random field, meaning that it factors according to the structure of an undirected graph $G = (V, E)$, with each variable X_s associated with one node $s \in V$, in the following way. Letting $\theta_s : \mathcal{X} \rightarrow$ and $\theta_{st} : \mathcal{X} \times \mathcal{X} \rightarrow$ be singleton and edgewise potential functions respectively, we assume that the distribution takes the form $\mathbb{P}(x; \theta) \propto \exp\big\{\sum_{s\in V} \theta_s(x_s) + \sum_{(s,t)\in E} \theta_{st}(x_s, x_t)\big\}$.

The problem of *maximum a posteriori* (MAP) estimation is to compute a configuration with maximum probability—i.e., an element

$$x^* \in \arg\max_{x\in\mathcal{X}^p} \Big\{ \sum_{s\in V} \theta_s(x_s) + \sum_{(s,t)\in E} \theta_{st}(x_s, x_t)\Big\} \quad (1)$$

This problem is an integer program, since it involves optimizing over the discrete space $\mathcal{X}^p$. The functions $\theta_s(\cdot)$ and $\theta_{st}(\cdot)$ can always be represented in the form

$$\theta_s(x_s) = \sum_{j\in\mathcal{X}} \theta_{s;j} \mathbb{I}[x_s = j] \quad (2a)$$

$$\theta_{st}(x_s, x_t) = \sum_{j,k\in\mathcal{X}} \theta_{st;jk} \mathbb{I}[x_s = j;\, x_t = k], \quad (2b)$$

where the m-vectors $\{\theta_{s;j}, j \in \mathcal{X}\}$ and $m \times m$ matrices $\{\theta_{st;jk},\ (j,k) \in \mathcal{X} \times \mathcal{X}\}$ parameterize the problem.

The basic linear programming (LP) relaxation of this problem is based on a set of pseudomarginals μ_s and μ_{st}, associated with the nodes and vertices of the graph. These pseudomarginals are constrained to be non-negative, as well to normalize and be locally consistent in the following sense:

$$\sum_{x_s} \mu_s(x_s) = 1, \qquad \text{for all } s \in V \quad (3a)$$

$$\sum_{x_t} \mu_{st}(x_s, x_t) = \mu_s(x_s) \qquad \text{for all } (s,t) \in E.$$

The polytope defined in this way is denoted LOCAL(G), or $\mathbb{L}(G)$ for short. The LP relaxation is based on solving maximizing the linear function

$$\sum_s \sum_{x_s} \theta_s(x_s)\mu_s(x_s) + \sum_{(s,t)\in E} \sum_{x_s, x_t} \theta_{st}(x_s, x_t)\mu_{st}(x_s, x_t),$$

subject to the constraint $\mu \in \mathbb{L}(G)$. In the sequel, we write this LP more compactly in the form $\max_{\mu\in\mathbb{L}(G)} \theta^T\mu$. By construction, this relaxation is guaranteed to be exact for any problem on a tree-structured graph (Wainwright et al., 2005), so that it can be viewed as a tree-based relaxation. The main goal of this paper is to develop efficient and distributed algorithms for solving this LP relaxation, as well as strengthenings based on additional constraints. For instance, one natural strengthening is by "lifting": view the pairwise MRF as a particular case of a more general MRF with higher order cliques, define higher-order pseudomarginals on these cliques, and use them to impose higher-order consistency constraints. This particular progression of tighter relaxations underlies the Bethe to Kikuchi (sum-product to generalized sum-product) hierarchy.

3. Proximal Minimization Schemes

We begin by defining the notion of a proximal minimization scheme, and the Bregman divergences that we use to define our proximal sequences. Instead of referring to the maximization problem $\max_{\mu \in \mathbb{L}(G)} \theta^T \mu$, it is convenient to consider the equivalent minimization problems $\min_{\mu \in \mathbb{L}(G)} -\theta^T \mu$.

3.1. Proximal Minimization

The class of methods that we develop are based on the notion of proximal minimization (Bertsekas & Tsitsiklis, 1997). Instead of attempting to solve the LP directly, we solve a sequence of problems of the form

$$\mu^{n+1} = \arg \min_{\mu \in \mathbb{L}(G)} \left\{ -\theta^T \mu + \frac{1}{\omega^n} D_f(\mu \,\|\, \mu^n) \right\}, \quad (4)$$

where for each $n = 0, 1, 2, \ldots$, μ^n denotes current iterate, $\{\omega^n\}$ denotes a sequence of positive weights, and D_f is a certain type of generalized distance, known as the proximal function. The purpose of introducing the proximal function is to convert the original LP—a convex optimization problem but non-differentiable in dual space —into a strictly convex optimization problem that can be solved relatively easily. This scheme appears similar to an annealing scheme, in that it involves a choice of weights $\{\omega^n\}$. However, although the weights $\{\omega^n\}$ can be adjusted for faster convergence, they can also be set to a constant, unlike for annealing procedures, which would typically require that $1/\omega^n \to 0$. The reason is that $D_f(\mu \,\|\, \mu^{(n)})$, as a generalized distance, itself converges to zero when the method gets closer to the optimum, thus providing an "adaptive" annealing. For appropriate choice of weights and proximal functions, these proximal minimization schemes converge to the LP optimum with at least geometric and possibly superlinear rates (Bertsekas & Tsitsiklis, 1997; Iusem & Teboulle, 1995).

In this paper, we focus exclusively on proximal functions that are Bregman divergences (Censor & Zenios, 1997), a class that includes various well-known divergences (e.g., quadratic norm, Kullback-Leibler divergence etc.). More specifically, we say that a function f is a Bregman function if it is continuously differentiable, strictly convex, and has bounded level sets. It then induces a Bregman divergence

$$D_f(\mu \,\|\, \nu) := f(\mu) - f(\nu) - \langle \nabla f(\nu), \mu - \nu \rangle \quad (5)$$

This function satisfies $D_f(\mu \,\|\, \nu) \geq 0$ with equality iff $\mu = \nu$, but need not be symmetric or satisfy the triangle inequality, so it is known as a generalized distance.

We study the sequence $\{\mu^n\}$ of proximal iterates (4) for the following choices of Bregman divergences.

Quadratic Distances: This choice is the simplest, corresponding to the quadratic norm across nodes and edges

$$Q(\mu \,\|\, \nu) := \sum_{s \in V} \|\mu_s - \nu_s\|^2 + \sum_{(s,t) \in E} \|\mu_{st} - \nu_{st}\|^2, \quad (6)$$

where we have used the shorthand

$$\|\mu_s - \nu_s\|^2 = \sum_{x_s} |\mu_s(x_s) - \nu_s(x_s)|^2,$$

and similarly for the edges. The Bregman function this corresponds to is the quadratic function,

$$f(\mu) = \frac{1}{2} \left\{ \sum_{s, x_s} \mu_s^2(x_s) + \sum_{s,t,x_s,x_t} \mu_{st}^2(x_s, x_t) \right\} \quad (7)$$

Weighted Entropic Distances: Here we consider a (possibly weighted) sum of Kullback-Leibler (KL) divergences across the nodes and edges:

$$D(\mu \,\|\, \nu) = \sum_{s \in V} \rho_s D(\mu_s \,\|\, \nu_s) + \sum_{s,t} \rho_{st} D(\mu_{st} \,\|\, \nu_{st}) \quad (8)$$

where $D(p \,\|\, q) := \sum_x \big(p(x) \log \frac{p(x)}{q(x)} - \big[p(x) - q(x) \big] \big)$ is the KL divergence, and $\{\rho_s, \rho_{st}\}$ are positive node and edge weights, respectively. An advantage of the KL distance, in contrast to the quadratic norm, is that it automatically acts to enforce non-negativity constraints on the pseudomarginals. The Bregman function this corresponds to is the entropy function,

$$f(\mu) = \sum_s H_s(\mu_s) + \sum_{s,t} H_{st}(\mu_{st}) \quad (9)$$

where H_s and H_{st} are singleton and edge-based entropies, respectively.

An extension of define a Bregman function based on a convex combination of tree-structured entropy functions (Wainwright & Jordan, 2003), and using expressions such as the reweighted Bethe entropy which are equivalent to the convex combination of tree entropies within the local polytope, we can derive an iterative procedure involving tree-reweighted message passing to solve the outer proximal steps. We defer further details to a full-length version.

3.2. Proximal Sequences via Bregman Projection

The key in designing an efficient proximal minimization scheme is ensuring that the proximal sequence $\{\mu^n\}$ can be computed efficiently. In this section, we

first describe how the sequence of each proximal minimization can be reformulated as a particular Bregman projection. We then describe how this Bregman projection can itself be computed iteratively, in terms of a sequence of cyclic Bregman projections based on a decomposition of the constraint set LOCAL(G). In the sequel, we then show how this cyclic Bregman projections reduce to very simple message-passing updates.

Given a Bregman divergence D, the *Bregman projection* of the vector ν onto a convex set C is given by

$$\widehat{\mu} \quad := \quad \arg\min_{\mu \in C} D_f(\mu \,\|\, \nu) \tag{10}$$

By taking derivatives and using standard conditions for optima over convex sets (Bertsekas & Tsitsiklis, 1997), the defining optimality condition for $\widehat{\mu}$ is

$$\big(\nabla f(\widehat{\mu}) - \nabla f(\nu)\big)^T \big(\mu - \widehat{\mu}\big) \quad \geq \quad 0 \tag{11}$$

for all $\mu \in C$. Now consider the proximal minimization problem to be solved at step n, namely the strictly convex problem

$$\min_{\mu \in \mathbb{L}(G)} \left\{ -\theta^T \mu + \frac{1}{\omega^n} D_f(\mu \,\|\, \mu^n) \right\}. \tag{12}$$

By taking derivatives and using the same convex optimality, we see that the optimum μ^{n+1} is defined by the conditions

$$\big(\nabla f(\mu^{n+1}) - \nabla f(\mu^n) - \omega^n \theta\big)^T (\mu - \mu^{n+1}) \quad \geq \quad 0$$

for all $\mu \in C$. Note that these optimality conditions are of the same form as the Bregman projection conditions (11), with the vector $\nabla f(\mu^n) + \omega^n \theta$ taking the role of $\nabla f(\nu)$; in other words, with $(\nabla f)^{-1}(\nabla f(\mu) + \omega^n \theta)$ being substituted for ν. Consequently, efficient algorithms for computing the Bregman projection (11) can be leveraged to compute the proximal update (12). In particular, our algorithms leverage the fact that Bregman projections can be computed efficiently in a *cyclic manner*—that is, by decomposing the constraint set $C = \cap_i C_i$ into an intersection of simpler constraint sets, and then performing a sequence of projections onto these simple constraint sets (Censor & Zenios, 1997).

To simplify notation, for any Bregman function f, let us define the operator

$$\mathrm{J}_f(\mu, \nu) = (\nabla f)^{-1}(\nabla f(\mu) + \nu)$$

and for any Bregman divergence D with Bregman function f and any convex set C, define the projection operator

$$\Pi_{D_f}(\gamma; C) \quad := \quad \arg\min_{\mu \in C} D_f(\mu \,\|\, \gamma)$$

With this notation, we can write the proximal update in a compact manner as a type of projection

$$\mu^{n+1} \quad = \quad \Pi_{D_f}\left(\mathrm{J}_f(\mu^n, \omega^n \theta); \mathbb{L}(G)\right).$$

Now consider a decomposition of the constraint set as an intersection—say $\mathbb{L}(G) = \cap_{k=1}^T \mathbb{L}_k(G)$. By the method of cyclic Bregman projections (Censor & Zenios, 1997), we can compute μ^{n+1} in an iterative manner, by performing the sequence of projections onto the simpler constraint sets, initializing $\mu^{n,0} = \mu^n$ and updating from $\mu^{n,\tau} \mapsto \mu^{n,\tau+1}$ by projecting $\mu^{n,\tau}$ onto constraint set $\mathbb{L}_{i(\tau)}(G)$, where $i(\tau) = \tau \mod T$, for instance. This procedure is summarized in Algorithm 1.

Algorithm 1 Basic proximal-Bregman LP solver

Given a Bregman distance D, weight sequence $\{\omega^n\}$ and problem parameters θ:

- Initialize $\mu_s^{(0)}(x_s) = \frac{1}{m}$, $\mu_{st}^{(0)}(x_s, x_t) = \frac{1}{m^2}$.
- **Outer loop:** For iterations $n = 0, 1, 2, \ldots$, update $\mu^{n+1} = \Pi_D\left(\mathrm{J}_f(\mu^n, \omega^n\theta); \mathbb{L}(G)\right)$.
  - Solve outer loop via **Inner loop:**
    (a) Initialize $\mu^{n,0} = \mathrm{J}_f(\mu^n, \omega^n\theta)$.
    (b) For $\tau = 0, 1, 2, \ldots$, set $i(\tau) = \tau \mod T$.
    (c) Set $\mu^{n,\tau+1} = \Pi_D\left(\mu^{n,\tau}; \mathbb{L}_{i(\tau)}(G)\right)$.

As shown in the following sections, by using a decomposition of $\mathbb{L}(G)$ over the edges of the graph, the inner loop steps correspond to local message-passing updates, slightly different in nature depending on the choice of Bregman distance. Iterating the inner and outer loops yields a provably convergent message-passing algorithm for the LP. Convergence follows from the convergence properties of proximal minimization (Bertsekas & Tsitsiklis, 1997), combined with convergence guarantees for cyclic Bregman projections (Censor & Zenios, 1997). In the following section, we derive the message-passing updates corresponding to various Bregman functions of interest. We also give rates of convergence for the cyclic projection updates in the inner loop.

3.3. Quadratic Projections

Consider the proximal sequence with the quadratic distance Q from equation (6); the Bregman function inducing this distance is the quadratic function $f(y) = \frac{1}{2}y^2$, whose gradient is given by $\nabla f(y) = y$.

The Map $\nu = \mathbf{J}_f(\mu, \omega\theta)$: In this case, it can

be derived as,

$$\nabla f(\nu) = \nabla f(\mu) + \omega\theta \quad (13)$$
$$\Rightarrow \nu = \mu + \omega\theta \quad (14)$$

whence we get the initialization in Equation 18.

The Projections $\mu^{n,\tau+1} = \Pi_Q(\mu^{n,\tau}, \mathbb{L}_i(G))$: onto the individual constraints $\mathbb{L}_i(G)$; the associated local update takes the form

$$\mu^{n,\tau+1} = \min_{\alpha \in \mathbb{L}_i(G)} \{f(\alpha) - \alpha^\top \nabla f(\mu^{n,\tau})\} \quad (15)$$

Consider the edge marginalization constraint for edge (s,t), $\mathbb{L}_i(G) \equiv \sum_{x_t} \mu_{st}(x_s, x_t) = \mu_s(x_s)$. Denoting the dual (Lagrange) parameter corresponding to the constraint by $\lambda_{st}(x_s)$, the KKT conditions for (15) are given by

$$\begin{aligned}
\nabla f(\mu_{st}^{n,\tau+1}(x_s,x_t)) &= \nabla f(\mu_{st}^{n,\tau}(x_s,x_t)) + \lambda_{st}(x_s) \\
\nabla f(\mu_{s}^{n,\tau+1}(x_s)) &= \nabla f(\mu_{s}^{n,\tau}(x_s)) - \lambda_{st}(x_s) \\
\mu_{st}^{n,\tau+1}(x_s,x_t) &= \mu_{st}^{n,\tau}(x_s,x_t) + \lambda_{st}(x_s) \\
\mu_{s}^{n,\tau+1}(x_s) &= \mu_{s}^{n,\tau}(x_s) - \lambda_{st}(x_s),
\end{aligned}$$

while the constraint itself gives

$$\sum_{x_t} \mu_{st}^{n,\tau+1}(x_s,x_t) = \mu_s^{n,\tau}(x_s) \quad (17)$$

Solving for $\lambda_{st}(x_s)$ yields equation (20). The node marginalization follows similarly, so that overall, we obtain message-passing algorithm (2) for the inner loop.

3.4. Entropic Projections

Consider the proximal sequence with the Kullback-Leibler distance $D(\mu \,\|\, \nu)$ defined in equation (8); the Bregman function inducing the distance is a sum of negative entropy functions $f(\mu) = \mu \log \mu$, and its gradient is given by $\nabla f(\mu) = \log(\mu) + \mathbf{1}$.

The derivation of the updates mirrors the previous section, and defering the details to a full-length version, we get the message passing algorithm (3) for the inner loop.

There are also interesting similarities between our corresponding dual updates and sum-product updates—which are updates to the dual parameters—details of which we defer to a full-length version of this paper due to lack of space.

3.5. Reweighted Entropy Projections

The message passing updates here are "reweighted" versions of those in the previous section for the unweighted entropy induced Kullback-Leibler divergence proximal iterates.

Algorithm 2 Quadratic Messages for μ^{n+1}

Initialization:

$$\mu_{st}^{(n,0)}(x_s,x_t) = \mu_{st}^{(n)}(x_s,x_t) + w^n\theta_{st}(x_s,x_t) \quad (18)$$
$$\mu_{s}^{(n,0)}(x_s) = \mu_{s}^{(n)}(x_s) + w^n\theta_{s}(x_s) \quad (19)$$

repeat
for each edge $(s,t) \in E$ **do**

$$\mu_{st}^{(n,\tau+1)}(x_s,x_t) = \mu_{st}^{(n,\tau)}(x_s,x_t) + (1/L+1)\left(\mu_s^{(n,\tau)}(x_s) - \sum_{x_t}\mu_{st}^{(n,\tau)}(x_s,x_t)\right) \quad (20)$$

$$\mu_{s}^{(n,\tau+1)}(x_s) = \mu_{s}^{(n,\tau)}(x_s) + (1/L+1)\left(-\mu_s^{(n,\tau)}(x_s) + \sum_{x_t}\mu_{st}^{(n,\tau)}(x_s,x_t)\right) \quad (21)$$

end for
for each node $s \in V$ **do**

$$\mu_s^{(k+1)}(x_s) = \mu_s^{(k)}(x_s) + \frac{1}{L}(1 - \sum_{x_s}\mu_s^{(k)}(x_s))$$
$$\mu_s^{(k+1)}(x_s) = \max(0, \mu_s^{(k+1)}(x_s))$$

end for
until convergence

Initialization of Proximal Steps:

$$\mu_{st}^{(n,0)}(x_s,x_t) = \mu_{st}^{(n)}(x_s,x_t)\exp(\omega^n/\rho_{st}\theta_{st}(x_s,x_t))$$
$$\mu_{s}^{(n,0)}(x_s) = \mu_{s}^{(n)}(x_s)\exp(\omega^n/\rho_{s}\theta_{s}(x_s)).$$

Projections: The node normalization update remains the same as in the previous section, while the marginalization update changes as,

$$\mu_{st}^{(n,\tau+1)}(x_s,x_t) = \mu_{st}^{(n,\tau)}(x_s,x_t)\left(\frac{\mu_s^{(n,\tau)}(x_s)}{\sum_{x_t}\mu_{st}^{(n,\tau)}(x_s,x_t)}\right)^{\frac{\rho_s}{\rho_s+\rho_{st}}}$$
$$\mu_{s}^{(n,\tau+1)}(x_s) = \mu_{s}^{(n,\tau)}(x_s)^{\frac{\rho_s}{\rho_s+\rho_{st}}}\left(\sum_{x_t}\mu_{st}^{(n,\tau)}(x_s,x_t)\right)^{\frac{\rho_{st}}{\rho_s+\rho_{st}}}$$

4. Rounding with Optimality Certificates

A key practical issue in applying LP relaxation is how round the fractional solution; a standard approach is

Algorithm 3 Entropic Messages for μ^{n+1}

Initialization:

$$\mu_{st}^{(n,0)}(x_s, x_t) = \mu_{st}^{(n)}(x_s, x_t) \exp(\omega^n \theta_{st}(x_s, x_t))$$
$$\mu_s^{(n,0)}(x_s) = \mu_s^{(n)}(x_s) \exp(\omega^n \theta_s(x_s))$$

repeat
for each edge $(s,t) \in E$ **do**

$$\mu_{st}^{(n,\tau+1)}(x_s, x_t) = \mu_{st}^{(n,\tau)}(x_s, x_t) \sqrt{\frac{\mu_s^{(n,\tau)}(x_s)}{\sum_{x_t} \mu_{st}^{(n,\tau)}(x_s, x_t)}}$$
$$\mu_s^{(n,\tau+1)}(x_s) = \sqrt{\mu_s^{(n,\tau)}(x_s) \sum_{x_t} \mu_{st}^{(n,\tau)}(x_s, x_t)}$$

end for
for each node $s \in V$ **do**

$$\mu_s^{(n,\tau+1)}(x_s) = \frac{\mu_s^{(n,\tau)}(x_s)}{\sum_{x_s} \mu_s^{(n,\tau)}(x_s)}$$

end for
until convergence

to round the node marginals to the nearest integer solution. However, in general, such rounding procedures need not always output the optimal integer configuration. An attractive feature of our proximal Bregman procedures is the existence of rounding schemes which, assuming that the LP relaxation is tight, can produce the LP integral optimum and certify that it is correct, even before the pseudomarginals converge to the LP solution. Here we describe two rounding schemes, and state the optimality certificate associated with each.

Node-based Rounding: This method applies to any of our proximal schemes. Given the vector μ^n of pseudomarginals at iteration n, define an integer configuration x^n by choosing, for each vertex $s \in V$, a value $x_s^n \in \arg\max_{x_s} \mu_s^n(x_s)$. Say that such a rounding is *edgewise-consistent* if for all edges $(s,t) \in E$, we have $(x_s^n, x_t^n) \in \arg\max_{(x_s, x_t)} \mu_{st}^n(x_s, x_t)$.

Tree-based Rounding: We describe this method in application to the unweighted entropic proximal updates. Let $T_1, \ldots, T_M$ be a set of spanning trees that cover the graph (meaning that each edge appears in at least one tree); for each edge (s,t), define the edge weight $\alpha_{st} = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}[(s,t) \in T_i]$. Then for each tree $i = 1, \ldots, M$:

(a) Define the tree-structured energy function $E_i(x) := \sum_s \mu^n(x_s) + \sum_{(s,t) \in E(T_i)} \frac{1}{\alpha_{st}} \mu_{st}^n(x_s, x_t)$.

(b) Run the ordinary max-product problem on energy $E_i(x)$ to find a MAP-optimal configuration $x^n(T_i)$.

Say that such a rounding is *tree-consistent* if the tree MAP solutions $\{x^n(T_i), i = 1, \ldots, M\}$ are all equal.

The following result characterizes the optimality guarantees associated with these rounding schemes:

Theorem 1 (Rounding with optimality certificates)**.** *At any iteration $n = 1, 2, \ldots$, any edge-consistent configuration obtained from node-rounding, or any tree-consistent configuration obtained from tree-rounding is guaranteed to be MAP optimal for the original problem.*

The proof is based on a certain energy-invariance property of the proximal updates; in particular, at any iteration n, the pseudomarginals μ^n have an associated function $F(x; \mu^n)$ which is proportional to the energy $E(\theta; x) = \sum_s \theta_s(x_s) + \sum_{st} \theta_{st}(x_s, x_t)$ of the graphical model. For instance, for the entropic proximal scheme, at any iteration n, the function $F(x; \mu^n) := \prod_{s \in V} \mu_s^n(x_s) \prod_{(s,t) \in E} \mu_{st}^n(x_s, x_t)$ is proportional to the exponential of $E(\theta; x)$. (See the technical report (Ravikumar et al., 2008) for full details.)

Both rounding schemes require relatively little computation. Of course, the node-rounding scheme is purely local, and so trivial to implement. With reference to the tree-rounding scheme, many graphs can be covered with a small number M of trees (e.g. $M = 2$ for grid graphs). Consequently, the tree-rounding scheme requires running the ordinary max-product algorithm twice, certainly more expensive than node-rounding but doable. In practice, we find that tree-rounding tends to find LP optima more quickly than node rounding.

5. Convergence Rates

The convergence of our message passing updates follows from two sets of results: (a) convergence of proximal algorithms (Bertsekas & Tsitsiklis, 1997) and (b) convergence of cyclic Bregman projections (Censor & Zenios, 1997). Our outer loop is a proximal algorithm; which has been well-studied in the optimization literature. A sequence $\mu^{(t)}$ is said to have superlinear convergence to the optimum μ^* if $\lim_{k \to \infty} \frac{\|\mu^{(t+1)} - \mu^*\|}{\|\mu^{(t)} - \mu^*\|} = 0$. Note that such convergence is faster than a multiplicative contraction ($\lim_{k \to \infty} \frac{\|\mu^{(t+1)} - \mu^*\|}{\|\mu^{(t)} - \mu^*\|} \leq \alpha < 1$). Bertsekas and Tsistiklis (1997) show that a proximal

algorithm with a quadratic proximity has a superlinear convergence under mild conditions, whereas Iusem and Teboulle (1995) show the same for the entropy proximity. Under the assumption that inner loops are solved exactly, these convergence results then show that our outer iterates converge superlinearly. Our inner loop message updates use cyclic Bregman projections; Censor and Zenios (1997) show that with dual feasibility correction, projections onto general convex sets are convergent. For Euclidean (quadratic) projections onto linear constraints (half-spaces), Deutsch et al. (2006) establish a geometric rate of convergence, dependent on angles between the half-spaces. The intuition is that the more orthogonal the half-spaces are, the faster the convergence; for instance, a single iteration suffices for completely orthogonal constraints. Our inner updates thus converge geometrically to an $\epsilon-$suboptimal solution for any outer proximal step. As noted earlier, the proximal convergence results assume that the inner loop has been solved exactly, while the Bregman projection results yield geometric convergence to but an $\epsilon-$suboptimal solution. While with ϵ small enough, e.g. 10^{-6} as in our experiments, this issue might not be practically relevant, there has been some recent work, e.g. (Solodov & Svaiter, 2001), showing that under mild conditions, superlinear rates still hold for $\epsilon-$suboptimal proximal iterates.

6. Experiments

We performed experiments on a 4-nearest neighbor grid graphs with sizes varying from $p = 100$ to $p = 900$, in all cases using models with 5 labels. The edge potentials were set to Potts functions, $\theta_{st}(x_s, x_t) = \beta_{st}\, \mathbb{I}[x_s = x_t]$, which penalize disagreement of labels by β_{st}. The Potts weights on edges β_{st} were chosen randomly as Uniform$(-1, +1)$, while the node potentials $\theta_s(x_s)$ were set as Uniform$(-\,\mathrm{SNR}, \mathrm{SNR})$, where the parameter SNR ≥ 0 controls the ratio of node to edge strengths, and thus corresponds roughly to a signal-to-noise ratio.

Figure 1 shows plots of the logarithmic distance between the current iterate μ^n and the LP optimum μ^n for grids of different sizes. In all cases, note how the curves have an inverted quadratic shape, corresponding to superlinear convergence.

Figure 2 shows the fraction of edges for which the node-based rounding is edgewise inconsistent for grids of different sizes. Note how the fraction converges to zero in a small number of iterations. Figure 3 shows the fraction of the energy of the rounded solution to the energy of the MAP optimum, or the suboptimality factor. Note again, the small number of iterations for convergence.

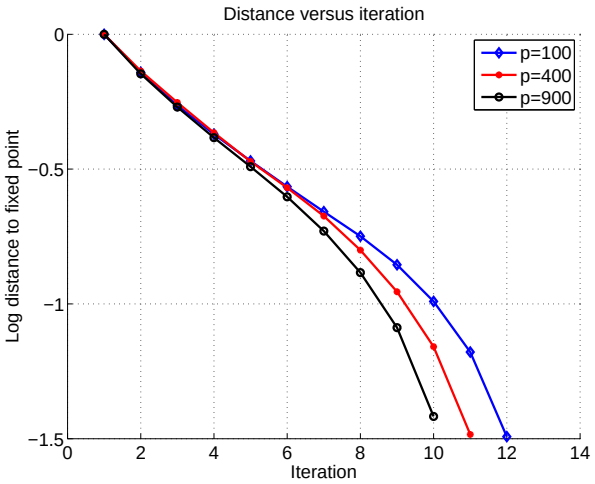


Figure 1. Plot of distance $\log_{10} \|\mu^n - \mu^*\|_2$ between the current iterate μ^n and the LP optimum μ^* versus iteration number for Potts models on grids with $p \in \{100, 400, 900\}$ vertices, and SNR $= 1$. Note the superlinear rate of convergence.

7. Discussion

In this paper, we have developed distributed algorithms, based on the notion of proximal sequences, for solving graph-structured linear programming (LP) relaxations. Our methods respect the graph structure, and so can be scaled to large problems, and they exhibit a superlinear rate of convergence. We also developed rounding schemes that can be used to generate integral solutions along with a certificate of optimality. These optimality certificates allow the algorithm to be terminated in a finite number of iterations.

The structure of our algorithms naturally lends itself to incorporating additional constraints, both linear and other types of conic constraints. It would be interesting to develop an adaptive version of our algorithm, which selectively incorporated new constraints as necessary, and then used the same proximal schemes to minimize the new conic program.

Acknowledgements Work supported by NSF grants CCF-0545862 and DMS-0528488. We thank the anonymous reviewers for helpful comments.

References

Bertsekas, D. P., & Tsitsiklis, J. N. (1997). *Parallel and Distributed Computation: Numerical Methods.* Boston, MA: Athena Scientific.

Bertsimas, D., & Tsitsikilis, J. (1997). *Introduction to linear optimization.* Belmont, MA: Athena Scientific.

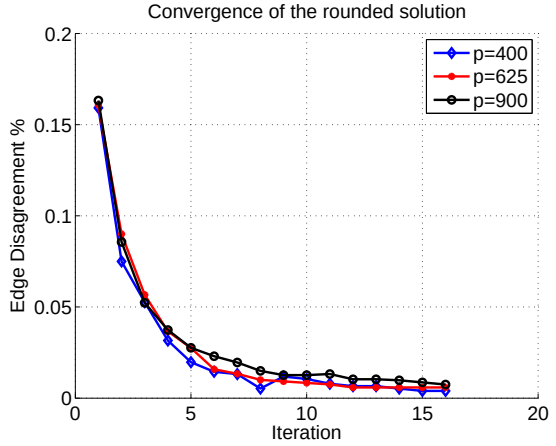


Figure 2. Plots of the fraction of edges for which the node-based rounding is edgewise inconsistent for grids of different sizes. Recall that when the fraction is zero, we recover the MAP optimum.

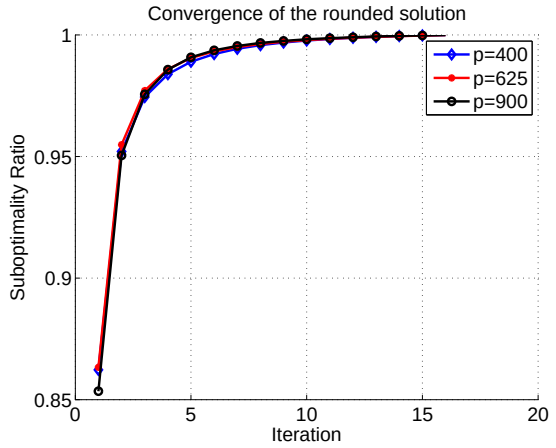


Figure 3. Plots of the fraction of the energy of the rounded solution to the energy of the MAP optimum. Note the small number of iterations for convergence.

Boykov, Y., Veksler, O., & .Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, *23*, 1222–1239.

Censor, Y., & Zenios, S. A. (1997). *Parallel optimization - theory, algorithms and applications*. Oxford University Press.

Chekuri, C., Khanna, S., Naor, J., & Zosin, L. (2005). A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal on Discrete Mathematics*, *18*, 608–625.

Deutsch, F., & Hundal, H. (2006). The rate of convergence for the cyclic projections algorithm i: Angles between convex sets. *Journal of Approximation Theory*, *142*, 36 55.

Feldman, J., Karger, D. R., & Wainwright, M. J. (2002). Linear programming-based decoding of turbo-like codes and its relation to iterative approaches. *Proc. 40th Annual Allerton Conf. on Communication, Control, and Computing*.

Freeman, W. T., & Weiss, Y. (2001). On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Trans. Info. Theory*, *47*, 736–744.

Globerson, A., & Jaakkola, T. (2007). Fixing max-product: Convergent message passing algorithms for map lp-relaxations. *Neural Information Processing Systems* (p. To appear). Vancouver, Canada.

Iusem, A. N., & Teboulle, M. (1995). Convergence rate analysis of nonquadratic proximal methods for convex and linear programming. *Mathematics of Operations Research*, *20(3)*, 657–677.

Kolmogorov, V. (2005). Convergent tree-reweighted message-passing for energy minimization. *International Workshop on Artificial Intelligence and Statistics*.

Kolmogorov, V., & Wainwright, M. J. (2005). On optimality properties of tree-reweighted message-passing. *UAI*.

Komodakis, N., Paragios, N., & Tziritas, G. (2007). MRF optimization via dual decomposition: Message-passing revisited. *ICCV*. Rio di Janeiro, Brazil.

Kumar, P., Torr, P., & Zisserman, A. (2006). Solving markov random fields using second order cone programming. *IEEE CVPR*.

Ravikumar, P., Agarwal, A., & Wainwright, M. J. (2008). *Message-passing for graph-structured linear programs: Proximal projections, convergence and rounding schemes* (Technical Report). UC Berkeley.

Ravikumar, P., & Lafferty, J. (2006). Quadratic programming relaxations for metric labeling and markov random field map estimation. *ICML '06* (pp. 737–744).

Solodov, M., & Svaiter, B. (2001). A unified framework for some inexact proximal point algorithms. *Numerical Functional Analysis and Optimization*, *22*, 1013–1035.

Wainwright, M., Jaakkola, T., & Willsky, A. (2005). Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, *51*, 3697–3717.

Wainwright, M. J., & Jordan, M. I. (2003). *Graphical models, exponential families, and variational inference* (Technical Report). UC Berkeley, Department of Statistics, No. 649.

Weiss, Y., Yanover, C., & Meltzer, T. (2007). Map estimation, linear programming and belief propagation with convex free energies. *UAI*.

Bayesian Multiple Instance Learning: Automatic Feature Selection and Inductive Transfer

Vikas C. Raykar VIKAS.RAYKAR@SIEMENS.COM
Balaji Krishnapuram BALAJI.KRISHNAPURAM@SIEMENS.COM
Jinbo Bi JINBO.BI@SIEMENS.COM
Murat Dundar MURAT.DUNDAR@SIEMENS.COM
R. Bharat Rao BHARAT.RAO@SIEMENS.COM
CAD and Knowledge Solutions (IKM CKS), Siemens Medical Solutions Inc., Malvern, PA 19355 USA

Abstract

We propose a novel Bayesian multiple instance learning (MIL) algorithm. This algorithm automatically identifies the relevant feature subset, and utilizes inductive transfer when learning multiple (conceptually related) classifiers. Experimental results indicate that the proposed MIL method is more accurate than previous MIL algorithms and selects a much smaller set of useful features. Inductive transfer further improves the accuracy of the classifier as compared to learning each task individually.

1. Multiple Instance Learning

In a single instance learning scenario we are given a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ containing N instances, where $x_i \in \mathcal{X}$ is an instance (the feature vector) and $y_i \in \mathcal{Y} = \{0, 1\}$ is the corresponding known label. The task is to learn a classification function $f : \mathcal{X} \rightarrow \mathcal{Y}$.

In the *multiple instance learning* framework the training set consists of *bags*. A bag contains many instances. All the instances in a bag share the same bag-level label. A bag is labeled positive if it contains *at least* one positive instance. A negative bag means that *all* instances in the bag are negative. The goal is to learn a classification function that can predict the labels of unseen instances and/or bags.

MIL is a natural framework to model many real-life tasks like drug activity prediction (Dietterich et al., 1997), image retrieval (Andrews et al., 2002), face detection (Viola et al., 2006), scene classification, text categorization, etc and is often found to be superior than a conventional supervised learning approaches (Ray & Craven, 2005). The concept of MIL was first introduced by (Dietterich et al., 1997) in the context of drug activity prediction. (Maron & Lozano-Perez, 1998) proposed a framework called Diverse Density algorithm. Since then various variants of standard single instance learning algorithms like Boosting (Xin & Frank, 2004; Viola et al., 2006), SVMs (Andrews et al., 2002; Fung et al., 2007), Logistic Regression (Ray & Craven, 2005; Settles et al., 2008), nearest neighbor (Wang & Zucker, 2000) etc. have been modified to adapt to the MIL scenario.

Our motivation for this work comes from the area of computer aided diagnosis (CAD) (see section § 10)– where the task is to build a classifier to predict whether a suspicious region (instance) on a computed tomography (CT) scan is a pulmonary embolism/nodule/lesion or not. This was proposed as a MIL problem by (Fung et al., 2007) by recognizing the fact that all instances which are within a certain distance to a radiologist mark (ground truth) can be considered as a positive bag. A requirement is that run time of the classifier during testing be as small as possible. Hence we would like the final classifier to use *as few features as possible*.

In this paper we propose a novel multiple instance algorithm which performs *automatic feature selection* and *classifier design* jointly. In particular we start out with the well-known logistic regression as our classifier and demonstrate how it can be modified for the MIL framework (§ 3–6). We use the feature selection method originally proposed for the *relevance vector machine* (RVM) (Tipping, 2001) single-instance classifier, in a manner that is optimal for multiple-instance classification(§ 7). We extend the algorithm to handle multi-task learning in § 8.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Novel Contributions

Our method differs from the substantial body of previous literature on MIL in the following crucial aspects.

a. **Baseline-model:** We use Logistic Regression as our baseline (single instance) classifier, similar to two previous papers. However, our model for combining the positive instances is quite different from the soft-max (Ray & Craven, 2005) or the averaging approach (Xin & Frank, 2004) used by others. We directly enforce the definition that at least one of the instances in a positive bag is positive.

b. **Feature selection:** Relying on the Bayesian automatic relevance determination paradigm, our learning algorithm selects the relevant subset of features that is most useful for accurate *multiple instance* classification. Experimental results demonstrate that the number of features chosen for optimizing the accuracy of multiple-instance classification *is much smaller* than that selected in a corresponding single instance learning algorithm. While MI Boost (Xin & Frank, 2004) also does feature selection, results indicate that our our approach is more accurate than MI Boost.

c. **Inductive transfer:** The proposed method is easily extended to statistically exploit information from other data sets while learning multiple related classifiers. This inductive-transfer approach results in substantial improvements in accuracy in real-life problems with limited training data. We are not aware of previous work which accomplishes inductive transfer in the context of multiple-instance classification.

3. Notation

We represent an instance as a feature vector $x \in \mathbb{R}^d$. A bag which contains K instances is denoted by boldface $\mathbf{x} = \{x_j \in \mathbb{R}^d\}_{j=1}^K$. The label of a bag is denoted by $y \in \{0, 1\}$.

Training Data The training data $\mathcal{D}$ consists of N bags $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i = \{x_{ij} \in \mathbb{R}^d\}_{j=1}^{K_i}$ is a bag containing K_i instances that share the same label $y_i \in \{0, 1\}$.

Classifier We consider the family of linear discriminating functions: $\mathcal{F} = \{f_w\}$, where for any $x, w \in \mathbb{R}^d$, $f_w(x) = w^T x$. The final classifier can be written in the following form

$$y = \begin{cases} 1 & \text{if } w^T x > \theta \\ 0 & \text{if } w^T x < \theta \end{cases} . \qquad (1)$$

Ties are resolved by flipping a fair coin. The *threshold parameter* θ determines the operating point of the classifier. The ROC curve is obtained as θ is swept from $-\infty$ to ∞. A bag is labeled positive if at least one instance is positive and negative if all instances are negative. Learning a classifier implies choosing the weight vector w given the training data $\mathcal{D}$.

4. Logistic Generalized Linear Model

The posterior probability for the positive class is modeled as a *logistic sigmoid* acting on the linear classifier f_w, *i.e.*, $p(y = 1|x) = \sigma(w^\top x)$. The logistic sigmoid function is defined as $\sigma(z) = 1/(1 + e^{-z})$. This classification model is known as *logistic regression* in the statistics community. Also $p(y = 0|x) = 1 - p(y = 1|x) = 1 - \sigma(w^\top x)$.

4.1. Logistic Model for MIL

In the MIL framework we have the concept of bags– where all the examples in a bag share the same label. A positive bag means at least one example in the bag is positive. The probability that a bag contains at least one positive instance is one minus the probability that all of them are negative. Hence the posterior probability for the positive bag can be written as

$$p(y = 1|\mathbf{x}) = 1 - \prod_{j=1}^{K} \left[1 - \sigma(w^\top x_j)\right], \qquad (2)$$

where the bag $\mathbf{x} = \{x_j\}_{j=1}^K$ contains K examples.This model is sometimes referred to as the $noisy-OR$ and has been previously used by (Viola et al., 2006) in a boosting framework and (Maron & Lozano-Perez, 1998) in the Diverse Density algorithm. We use this model for Logistic Regression. A negative bag means that *all* examples in the bag are negative. Hence

$$p(y = 0|\mathbf{x}) = \prod_{j=1}^{K} \left[1 - \sigma(w^\top x_j)\right]. \qquad (3)$$

5. Maximum Likelihood Estimator

Given the training data $\mathcal{D}$ the maximum likelihood (ML) estimate for w is given by

$$\widehat{w}_{\text{ML}} = \arg\max_w p(\mathcal{D}/w) = \arg\max_w \left[\log p(\mathcal{D}/w)\right]. \quad (4)$$

Define $p_i = p(y_i = 1|\mathbf{x}_i) = 1 - \prod_{j=1}^{K_i} \left[1 - \sigma(w^\top x_{ij})\right]$– the probability that the i^{th} bag $\mathbf{x}_i$ is positive. Assuming that the training bags are independent the log-likelihood can be written as

$$\log p(\mathcal{D}/w) = \sum_{i=1}^{N} y_i \log p_i + (1 - y_i) \log(1 - p_i). \quad (5)$$

A similar likelihood was also maximized by (Viola et al., 2006) using the AnyBoost framework, which views boosting as gradient descent in function space.

6. The MAP Estimator

The ML solution in practice can exhibit severe overfitting especially for high-dimensional data. This can be addressed by using a prior on w.

Prior We will assume zero mean Gaussian prior ($\mathcal{N}(w|0, \mathbf{A}^{-1})$) on the weights w with inverse covariance matrix $\mathbf{A} = diag(\alpha_1 \dots \alpha_d)$ (also referred to as the *hyper-parameters*).

$$p(w) = (2\pi)^{-d/2}|\mathbf{A}^{-1}|^{-1/2}\exp\left(-\frac{w^\top \mathbf{A} w}{2}\right). \quad (6)$$

This encapsulates our prior belief that the individual weights in w are independent and close to zero with a variance parameter $1/\alpha_i$.

Posterior Once we observe the training data $\mathcal{D}$ we will update the prior to compute the posterior $p(w/\mathcal{D})$, which can be written as follows (using Bayes's rule)– $p(w/\mathcal{D}) = p(\mathcal{D}/w)p(w)/\int p(\mathcal{D}/w)p(w)dw$. This posterior can then be used to compute predictive distributions, which will typically involve high dimensional integrals. For computational efficiency we could use point estimates of w. In particular the *maximum a-posteriori* (MAP) estimate is given by

$$\widehat{w}_{\text{MAP}} = \arg\max_w \left[\log p(\mathcal{D}/w) + \log p(w)\right]. \quad (7)$$

Substituting for the log likelihood and the prior we have $\widehat{w}_{\text{MAP}} = \arg\max_w L(w)$, where

$$L(w) = \left[\sum_{i=1}^{N} y_i \log p_i + (1-y_i)\log(1-p_i)\right] - \frac{w^\top \mathbf{A} w}{2}. \quad (8)$$

Optimization Due to the non-linearity of the sigmoid we do not have a closed form solution and we have to use gradient based optimization methods. We use the Newton-Raphson update given by $w^{t+1} = w^t - \eta \mathbf{H}^{-1}\mathbf{g}$, where $\mathbf{g}$ is the gradient vector, $\mathbf{H}$ is the Hessian matrix, and η is the step length. The gradient is given by

$$\mathbf{g}(w) = \sum_{i=1}^{N}[y_i\beta_i - (1-y_i)]\sum_{j=1}^{K_i} x_{ij}\sigma(w^\top x_{ij}) - \mathbf{A}w, \quad (9)$$

where $\beta_i = \frac{1-p_i}{p_i}$. Note that $\beta_i = 1$ corresponds to the derivatives of the standard logistic regression updates. These term β_i can be thought of as the bag weight by which each instance weight gets modified. The Hessian matrix is given by

$$\mathbf{H}(w) = \sum_{i=1}^{N}[y_i\beta_i - (1-y_i)]\sum_{j=1}^{K_i} x_{ij}x_{ij}^T\sigma(w^\top x_{ij})\sigma(-w^\top x_{ij}) - \sum_{i=1}^{N} y_i\beta_i(\beta_i+1)\left[\sum_{j=1}^{K_i} x_{ij}\sigma(w^\top x_{ij})\right]\left[\sum_{j=1}^{K_i} x_{ij}\sigma(w^\top x_{ij})\right]^T - \mathbf{A}. \quad (10)$$

Note that the Hessian matrix depends on the class labels also–unlike in regular logistic regression.

7. Bayesian MIL: Feature Selection

We imposed a prior of the form $p(w) = \mathcal{N}(w|0, \mathbf{A}^{-1})$, parameterized by d hyper-parameters $\mathbf{A} = diag(\alpha_1 \dots \alpha_d)$. Clearly, as the precision $\alpha_k \to \infty$, *i.e*, the variance for w_k tends to zero (thus concentrating the prior sharply at zero). Hence, regardless of the evidence of the training data, the posterior for w_k will also be sharply concentrated on zero, thus that feature will not affect the classification result-hence, it is effectively removed out via feature selection. Therefore, the discrete optimization problem corresponding to feature selection (should each feature be included or not?), can be more easily solved via an easier continuous optimization over hyper-parameters . If one could maximize the marginal likelihood $p(\mathcal{D}|\mathbf{A})$ this would perform optimal feature selection. This approach is also known as the *type-II maximum likelihood* method in the Bayesian literature.

We choose the hyper-parameters to maximize the marginal likelihood.

$$\widehat{\mathbf{A}} = \arg\max_{\mathbf{A}} p(\mathcal{D}|\mathbf{A}) = \arg\max_{\mathbf{A}} \int p(\mathcal{D}|w)p(w|\mathbf{A})dw. \quad (11)$$

Since this integral is not easy to compute for our MIL model we use an approximation to the marginal likelihood via the Taylor series expansion. The marginal likelihood $p(\mathcal{D}|\mathbf{A})$ can be written as $p(\mathcal{D}|\mathbf{A}) = \int e^{\Psi(w)}dw$, where $\Psi(w) = \log p(\mathcal{D}|w) + \log p(w|\mathbf{A})$. Approximating Ψ using a second order Taylor series around $\widehat{w}_{\text{MAP}}$,

$$\Psi(w) \approx \Psi(\widehat{w}_{\text{MAP}}) + \frac{1}{2}(w-\widehat{w}_{\text{MAP}})\mathbf{H}(\widehat{w}_{\text{MAP}}, \mathbf{A})(w-\widehat{w}_{\text{MAP}})^\top. \quad (12)$$

Hence we have the following approximation to the

Input: $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i = \{x_{ij} \in \mathbb{R}^d\}_{j=1}^{K_i}$ is a bag containing K_i instances that share the same label $y_i \in \{0, 1\}$.

Output: A list of selected features and weight vector w for the linear classifier.

Initialize $\alpha_i = 1$ and $w_i = 0$ for $i = 1, \ldots, d$.
repeat
  If $\alpha_i > \tau$ remove feature w_i.
  MAP estimate using the selected features.
  repeat
    Compute the gradient vector $\mathbf{g}$. (Eq. 9)
    Compute the Hessian matrix $\mathbf{H}$. (Eq. 10)
    Determine η using a line search.
    Update $w \leftarrow w - \eta \mathbf{H}^{-1}\mathbf{g}$.
  until $\|\mathbf{g}\|/d < \epsilon_1$
  Update the hyper-parameters using $\alpha_i \leftarrow 1/(w_i^2 + \Sigma_{ii})$. (Eq. 17)
until $\max_i |\log \alpha_i^{curr} - \log \alpha_i^{prev}| < \epsilon_2$
In the experiments reported in this paper we use $\tau = 10^{12}$, $\epsilon_1 = 10^{-5}$, and $\epsilon_2 = 10^{-3}$.

Algorithm 1: The proposed algorithm

marginal likelihood

$$\begin{aligned} & p(\mathcal{D}|\mathbf{A}) \\ \approx\ & e^{\Psi(\widehat{w}_{\mathrm{MAP}})} \int e^{\frac{1}{2}(w-\widehat{w}_{\mathrm{MAP}})\mathbf{H}(\widehat{w}_{\mathrm{MAP}},\mathbf{A})(w-\widehat{w}_{\mathrm{MAP}})^\top} dw \\ \approx\ & p(\mathcal{D}|\widehat{w}_{\mathrm{MAP}}) p(\widehat{w}_{\mathrm{MAP}}|\mathbf{A})(2\pi)^{d/2} |-\mathbf{H}^{-1}(\widehat{w}_{\mathrm{MAP}}, \mathbf{A})|^{1/2} \end{aligned} \quad (13)$$

Using the prior $p(w|\mathbf{A}) = \mathcal{N}(w|0, \mathbf{A}^{-1})$, the log marginal likelihood can be written as

$$\begin{aligned} \log p(\mathcal{D}|\mathbf{A}) \ &\approx\ \log p(\mathcal{D}|\widehat{w}_{\mathrm{MAP}}) - \frac{1}{2}\widehat{w}_{\mathrm{MAP}}^\top \mathbf{A} \widehat{w}_{\mathrm{MAP}} \\ &+\ \frac{1}{2}\log|\mathbf{A}| - \frac{1}{2}\log|-\mathbf{H}(\widehat{w}_{\mathrm{MAP}}, \mathbf{A})|. \end{aligned} \quad (14)$$

The hyper-parameters $\mathbf{A}$ are found by maximizing this approximation to the log marginal likelihood. There is no closed-form solution for this. Hence we use a iterative re-estimation method by setting the first derivative to zero. The derivative can be written as

$$\begin{aligned} \frac{\partial \log p(\mathcal{D}|\mathbf{A})}{\partial \mathbf{A}} \ &=\ -\frac{1}{2}\widehat{w}_{\mathrm{MAP}}\widehat{w}_{\mathrm{MAP}}^\top + \frac{1}{2}\mathbf{A}^{-1} \\ &-\ \frac{1}{2}\mathbf{H}^{-1}(\widehat{w}_{\mathrm{MAP}}, \mathbf{A}). \end{aligned} \quad (15)$$

Since $\mathbf{A} = diag(\alpha_1 \ldots \alpha_d)$, we can further simplify

$$\frac{\partial \log p(\mathcal{D}|\mathbf{A})}{\partial \alpha_i} = -\frac{1}{2}\widehat{w}_i^2 + \frac{1}{2\alpha_i} - \frac{1}{2}\Sigma_{ii}, \quad (16)$$

where Σ_{ii} is the i^{th} diagonal element of $\mathbf{H}^{-1}(\widehat{w}_{\mathrm{MAP}}, \mathbf{A})$. Assuming Σ_{ii} does not depend on $\mathbf{A}$ a simple update rule for the hyper-parameters can be written by equating the first derivative to zero.

$$\alpha_i^{\mathrm{new}} = \frac{1}{w_i^2 + \Sigma_{ii}}. \quad (17)$$

The final algorithm has two levels of iterations (see Algorithm 1): in an outer loop we update the hyper-parameters α_i and in an inner loop we find the MAP estimator $\widehat{w}_{\mathrm{MAP}}$ given the hyper-parameters. After a few iterations we find that the hyper-parameters–the inverse variances of the priors–for several features tend to infinity causing numerical problems in implementation. This means that those $w_i \to 0$ we can simply remove those irrelevant features from further consideration in future iterations. The proposed algorithm with feature selection can be considered as the extension of the Relevance Vector Machine (RVM) (Tipping, 2001) to multiple-instance learning framework.

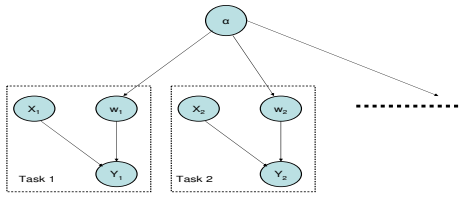


Figure 1. In multi-task learning tasks share the same prior.

8. Multi-task Learning

We are often faced with a shortage of training data for learning classifiers for a task. However we may have additional data for closely related, albeit non-identical tasks. For example in our CAD applications where we have to identify early stage cancers from CT scans, our data set includes images from CT scanners with two different reconstruction kernels–B50 and B60.

While training the classifier we could ignore this information and pool all the data together. However, there are some systematic differences that make the feature distributions slightly different. Alternatively, we could train a separate classifier for each kernel, but a large part of our data set is from one particular kernel (B60) and we have a smaller data set for the other (B50).

Here, we discuss another approach–*multi-task learning* (Caruana, 1997)– that tries to estimate models w^j for several classification tasks j in a joint manner. Multi-task learning can compensate for small sample size by using additional samples from related tasks, and exchanging statistical information between tasks. In a hierarchical Bayesian approach, the clas-

Table 1. Datasets used in our MIL experiments. d is the number of features.

Dataset	d	positive		negative	
		examples	bags	examples	bags
Musk1	166	207	47	269	45
Musk2	166	1017	39	5581	63
Elephant	230	762	100	629	100
Tiger	230	544	100	676	100

sifiers share a common prior $p(w^j|\mathbf{A})$ (See Figure 1). A separate classifier is trained for each task. However the optimal hyper-parameters of the shared prior are estimated from all the data sets simultaneously during the training. The update equation becomes (in place of Eq. 17): $\alpha_i^{\text{new}} = 1/\sum_{\text{task}j}(\widehat{w}_i^j)^2 + \Sigma_{ii}^j$.

9. Experimental Results

9.1. Datasets

Experiments were performed on four common benchmark data sets from the MIL literature (see Table 1).

Musk1 and Musk2 (Asuncion & Newman, 2007) The task is to predict whether a new drug molecule will bind to a target protein. However each molecule (bag) can have many different low energy shapes (instances) of which only one can actually bind with the target.

Elephant and Tiger The task is to search a repository to find images that contain objects of interest. An image is represented as a bag. An instance in a bag corresponds to a segment in the image; the object of interest is contained in at least one segment.

9.2. Competing Algorithms

Various learning algorithms have been adapted to the multiple learning scenario. We compare our proposed algorithm with a variant of Boosting, SVM, and Logistic Regression. Specifically we perform our experimental comparison for the following algorithms.

MI RVM The proposed multiple-instance algorithm with feature selection. This is completely automatic and does not require tuning any parameters.

RVM The proposed algorithm without multiple instance learning. This is same as MI RVM but every example is assigned to a unique bag.

MI The proposed multiple-instance algorithm without feature selection. We set $\mathbf{A} = \lambda\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and λ is chosen by five-fold cross-validation.

MI Boost (Xin & Frank, 2004) This is a variant of the AdaBoost algorithm adapted for the multiple instance learning scenario. We boosted for 50-100 rounds.

Table 2. The AUC for different algorithms and datasets.

Set	MIRVM	RVM	MIBoost	MILR	MISVM	MI
Musk1	0.942	**0.951**	0.899	0.846	0.899	0.922
Musk2	**0.987**	0.985	0.964	0.795	-	0.982
Elephant	0.962	**0.979**	0.828	0.814	0.959	0.953
Tiger	**0.980**	0.970	0.890	0.890	0.945	0.956

Table 3. The average number of features selected per fold by different algorithms.

Dataset	Number of features	selected by RVM	selected by MI RVM	selected by MI Boost
Musk1	166	39	**14**	33
Musk2	166	90	**17**	32
Elephant	230	42	**16**	33
Tiger	230	56	**19**	37

MI SVM (Andrews et al., 2002) This is a bag-level SVM variant for MIL. We used the implementation publicly available at (Yang, 2006). We used a linear kernel and the regularization parameters was chosen by 5-fold cross-validation.

MI LR (Settles et al., 2008; Ray & Craven, 2005) This is a variant of Logistic Regression which uses the soft-max function to combine posterior probabilities over the instances of a bag. We used $\alpha = 3$ in the soft-max function. Non-linear conjugate gradient with tolerance set at 10^{-3} was used as the optimization routine.

Of all the above algorithms only our proposed method and the boosting one does automatic feature selection. We are not aware of any other multiple-instance algorithms which does automatic feature selection.

9.3. Evaluation Procedure

The results are shown for a 10-fold stratified cross-validation. The folds are split at the positive bag level, so that examples in the same positive bag will not be split. We plot the Receiver Operating Characteristics (ROC) curve for various algorithms (see Figure 2). The ROC curve is a plot of False Positive Rate (FPR) vs True Positive Rate (TPR) as the decision threshold of the classifier θ is varied from ∞ to $-\infty$. The TPR is computed on a bag level–*i.e.*, a bag is predicted as positive if at least one on the instances in classified as positive. The ROC curve is plotted by pooling the prediction of the algorithm across all folds as in (Ray & Craven, 2005). We also report the area under the ROC curve (AUC) in Table 2.

9.4. Results

Comparison with other methods. From Figure 2 and Table 2 we see that among other MIL algorithms the ROC for the proposed method clearly dominates

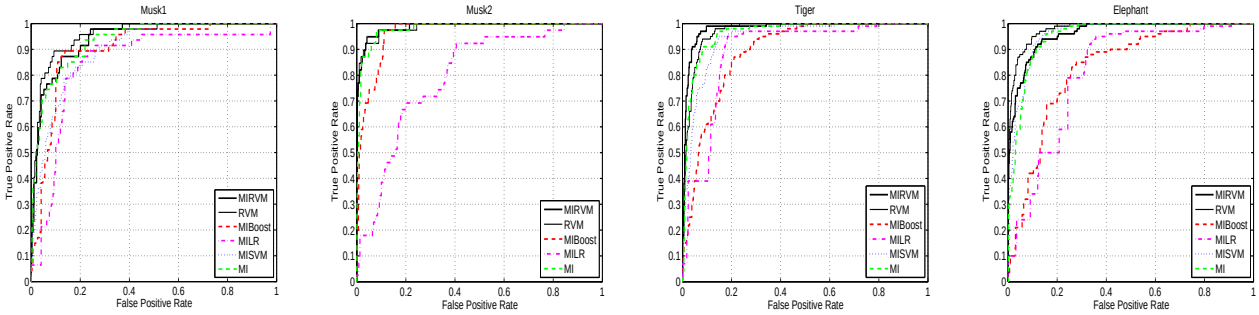


Figure 2. The ROC Curves for the different data sets and the different algorithms.

the other methods. However it is interesting to note that plain RVM is better than MI RVM for Musk 1 and Elephant data sets. This confirms the surprising observation in (Ray & Craven, 2005) that for some MIL benchmarks standard supervised learning algorithm may be more accurate than MIL algorithms.

Number of features selected. Table 3 compares the number of features selected by MI RVM, RVM, and the MI Boost algorithm. It can be seen that the proposed MI RVM algorithm selects the least number of features. Selecting features in a multiple instance setting reduces the number of features selected by half.

Does feature selection help? From Table 2 we see that the AUC with feature selection is higher than that without feature selection. Thus we are able to achieve better performance and at the same time use a smaller set of features.

Runtime The proposed algorithm and the MI Boost are orders of magnitude faster than other MIL methods. As a result MI SVM and MI LR could not be run on our CAD experiments described in the next section. Also the proposed method has no free parameters to tune. The runtime of our algorithm scales as $\mathcal{O}(d^3)$ with the number of features. This is because we need to compute the inverse of the $d \times d$ Hessian matrix.

10. Computer Aided Diagnosis

In computer aided diagnosis (CAD) the goal is to detect potentially malignant nodules, tumors, emboli, or lesions in medical images like computed tomography (CT), X-ray, MRI etc. A CAD system aids the radiologist by marking the location of likely anomaly on a medical image. Figure 3 shows two pulmonary emboli (PE) in a CT scan. PE (blood clots in the lung), is a potentially life-threatening condition. An early and accurate diagnosis is the key to survival. Computed tomography angiography (CTA) has emerged as an accurate diagnostic tool.

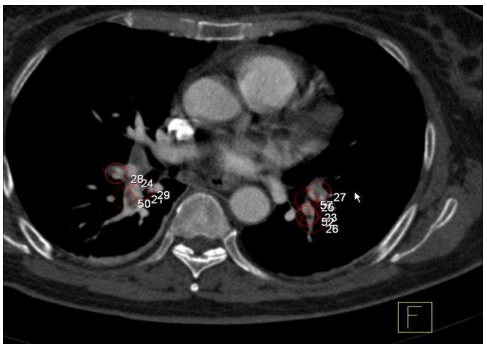


Figure 3. Sample pulmonary emboli in a Lung CT scan along with the candidates which point to it.

Most CAD systems consist of the following three steps–(1) *Candidate generation*–this step identifies potentially unhealthy regions of interest. While this step can detect most of the anomalies, the number of false positives will be extremely high (60-100 false positives/ patient). (2) *Feature computation*–computation of a set of descriptive morphological features for each of the candidates. (3) *Classification*–labeling of each candidate as a nodule or not by a classifier. The goal of the classifier is to reduce the number of false positives without appreciable decrease in the sensitivity.

In order to train a classifier, a set of CT scans is collected from hospitals. These scans are then read by expert radiologists who mark the pulmonary emboli locations–this constitutes our ground truth for learning. The candidate generation step generates a lot of potential candidates. Any candidate which is close to the radiologist mark (for example within a certain distance) is considered a positive example for training and the rest of the candidates are considered as negative examples. Based on a set of features computed for these candidates we intend to train a classifier.

Table 4. Datasets used in our PE CAD MIL experiments.

Dataset	Features	positive		negative
		examples	bags	examples
Training	134	514	312	4619
Validation	134	305	214	3246

10.1. Multiple Instance Learning for CAD

The candidate generation step very often produces a lot of candidates which are spatially close to each other (See Figure 3 for two PE appearing in a CT scan). All these candidates point to the same ground truth and can considered to share the same label for training. A single instance classifier can be trained using the labeled candidates. In this work we use the multiple instance learning algorithm by recognizing the fact that all candidates which point to the same radiologist mark can be considered as a positive bag (Fung et al., 2007). There is another important reason why MIL is a natural framework for CAD. The candidate generation algorithm produces a lot of spatially close candidates. Even if one of these is highlighted to the radiologist and other adjacent or overlapping candidates are missed, the underlying embolism would still have been detected. Hence while evaluating the performance of CAD systems we use the bag level sensitivity, *i.e.*, a classifier is successful in detecting an embolism if at least one of the candidates pointing to it is predicted as a PE. MIL naturally lends itself to model our desired accuracy measure during training itself.

Another important requirement is that run time of the classifier during testing should be as small as possible. The candidate generation step generally produces thousands of candidates for a CT scan. Computing all the features can be very time-consuming. Hence it is imperative that the final classifier uses as few features as possible without any decrease in the sensitivity. The proposed classifier automatically selects features for multiple-instance classification.

10.2. Experiments

Table 4 summarizes the PE CAD data sets we use in our experiments. Note that unlike the previous four data sets we do not have negative bags. Every negative example is considered a negative bag. The classifier is trained on the training set and tested on a separate validation set. Since we are interested in the number of False Positives per volume (patient) we plot Free response ROC (FROC) curves for the validation set (see Figure 4). MI RVM gives a substantial improvement over the single instance RVM approach and also the MI Boost method. The MI SVM and MI LR could not be run for the large CAD data set. MI RVM algorithm selected 21 features in contrast to the 34 features selected by the single instance RVM algorithm

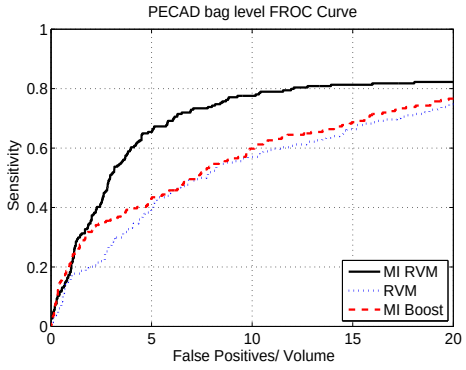


Figure 4. The bag level FROC curve for the PECAD validation set.

10.3. Multi-task Learning Experiments

Lung cancer is a leading cause of cancer related death in western countries. However early detection can substantially improve survival. Automatic CAD systems can be developed to identify suspicious regions such as solid nodules or ground-glass opacities (GGO) in CT scans of the lung. A solid nodule is defined as an area of increased opacity more than 5mm in diameter which completely obscures underlying vascular marking. Ground-glass opacity(GGO) is defined as an area of a slight, homogenous increase in density, which did not obscure underlying bronchial and vascular markings. Figure 5 shows an example nodule and GGO.

Detecting nodules and GGOs are two closely related tasks although each has its own respective characteristics. Hence multi-task learning is likely to be beneficial, even when building a specific model for each task. To train such a system we used 15 CT scans which included GGOs and 23 CT scans that included nodules. The model accuracy was validated on a held out set of 86 CT scans that included nodules. Figure 6 compares the FROC curves for nodule detection system designed in two different ways. (1) Single task learning: the classifier was learnt only using nodule data. (2) Multi-task learning: the classifier was learnt using nodule data and GGO data. As Figure 6 shows, inductive transfer using the proposed scheme the improves accuracy of the multiple instance learning system, when we have a limited amount of training data.

11. Conclusion

In this paper we proposed a novel MIL algorithm that automatically selects the features relevant for *multi-*

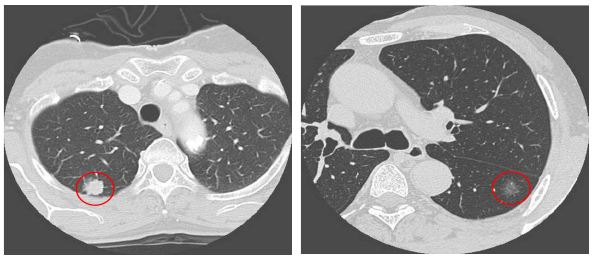


Figure 5. Lung CT image showing a sample (a)nodule and (b) GGO. Figure reprinted from (Suzuki et al., 2006).

ple instance classification. The proposed algorithm is more accurate than other competing MIL methods, both on benchmark data sets and on real life CAD problems. Our experiments also validate the previous observation of (Ray & Craven, 2005) that on some multiple instance benchmarks the single instance classifier is slightly more accurate. For all domains, the number of features selected by our algorithm is much smaller than that for the corresponding single instance classifier. Inductive transfer improves accuracy in data poor CAD applications.

Acknowledgements

We would like to thank Sangmin Park, Vikram Anand, Anna Jerebko, and Marcos Salgonicoff for help with the CAD data sets. We would also like to thank the reviewers for their suggestions which helped to improve the overall quality of the paper.

References

Andrews, S., Tsochantaridis, I., & Hofmann, T. (2002). Support vector machines for multiple-instance learning. In *Advances in neural information processing systems 15*.

Asuncion, A., & Newman, D. J. (2007). UCI machine learning repository, `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Caruana, R. (1997). Multi-task learning. *Machine Learning*, *28*, 41–75.

Dietterich, T. G., Lathrop, R. H., & Lozano Perez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Fung, G., Dundar, M., Krishnapuram, B., & Rao, R. B. (2007). Multiple instance learning for computer aided diagnosis. In *Advances in neural information processing systems 19*, 425–432.

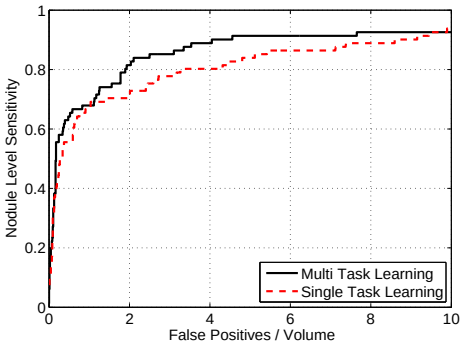


Figure 6. Multi-task learning experiments The bag level FROC curve for the validation set.

Maron, O., & Lozano-Perez, T. (1998). A framework for multiple-instance learning. In *Advances in neural information processing systems 10*, 570–576.

Ray, S., & Craven, M. (2005). Supervised versus multiple instance learning: an empirical comparison. *ICML '05: Proceedings of the 22nd international conference on Machine learning* (pp. 697–704).

Settles, B., Craven, M., & Ray, S. (2008). Multiple-instance active learning. In *Advances in neural information processing systems 20*.

Suzuki, K., Kusumoto, M., Watanabe, S.-i., Tsuchiya, R., & Asamura, H. (2006). Radiologic Classification of Small Adenocarcinoma of the Lung: Radiologic-Pathologic Correlation and Its Prognostic Impact. *Ann Thorac Surg*, *81*, 413–419.

Tipping, M. E. (2001). Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, *1*, 211–244.

Viola, P., Platt, J., & Zhang, C. (2006). Multiple instance boosting for object detection. In *Advances in neural information processing systems 18*, 1417–1424.

Wang, J., & Zucker, J. (2000). Solving the multiple-instance problem: A lazy learning approach. In *Proceedings of the 17th international conference on machine learning*, 1119–1125.

Xin, X., & Frank, E. (2004). Logistic regression and boosting for labeled bags of instances. *Proc 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 272–281).

Yang, J. (2006). MILL: A multiple instance learning library. `http://www.cs.cmu.edu/~juny/MILL`.

Online Kernel Selection for Bayesian Reinforcement Learning

Joseph Reisinger JOERAII@CS.UTEXAS.EDU
Peter Stone PSTONE@CS.UTEXAS.EDU
Risto Miikkulainen RISTO@CS.UTEXAS.EDU
Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712

Abstract

Kernel-based Bayesian methods for Reinforcement Learning (RL) such as Gaussian Process Temporal Difference (GPTD) are particularly promising because they rigorously treat uncertainty in the value function and make it easy to specify prior knowledge. However, the choice of prior distribution significantly affects the empirical performance of the learning agent, and little work has been done extending existing methods for prior model selection to the online setting. This paper develops Replacing-Kernel RL, an online model selection method for GPTD using sequential Monte-Carlo methods. Replacing-Kernel RL is compared to standard GPTD and tile-coding on several RL domains, and is shown to yield significantly better asymptotic performance for many different kernel families. Furthermore, the resulting kernels capture an intuitively useful notion of prior state covariance that may nevertheless be difficult to capture manually.

1. Introduction

Bayesian methods are a natural fit for Reinforcement Learning (RL) because they represent prior knowledge compactly and allow for rigorous treatment of value function uncertainty. Modeling such uncertainty is important because it offers a principled solution for balancing exploration and exploitation in the environment. One particularly elegant Bayesian RL formulation is Gaussian Process Temporal Difference (GPTD) (Engel et al., 2005). GPTD is an efficient adaptation of Gaussian processes (GPs) to the problem of online value-function estimation. In GPs, prior knowledge in the form of value covariance across states is represented compactly by a Mercer kernel (Rasmussen & Williams, 2006), offering a conceptually simple method for biasing learning.

An important open question for Bayesian RL is how to perform model selection efficiently and online. In GPTD, model selection determines the particular form of the prior covariance function and the settings of any hyperparameters. This paper contributes towards answering this question in two ways: (1) It demonstrates empirically the importance of model selection in Bayesian RL; and (2) it outlines *Replacing-Kernel Reinforcement Learning* (RKRL), a simple and effective sequential Monte-Carlo procedure for selecting the model online. RKRL not only improves learning in several domains, but does so in a way that cannot be matched by any choice of standard kernels.

Although conceptually similar to methods combining evolutionary algorithms and RL (Whiteson & Stone, 2006), RKRL is novel for two reasons: (1) The sequential Monte-Carlo technique employed is simpler and admits a clear *empirical Bayesian* interpretation (Bishop, 2006), (2) Since GPs are nonparametric, it is possible to replace kernels online during learning without discarding any previously acquired knowledge, simply by maintaining the dictionary of saved training examples between kernel swaps. This online replacing procedure significantly improves performance over previous methods, because learning does not need to start from scratch for new kernels.

This paper is divided into seven main sections: Section 2 introduces GPTD, Section 3 describes RKRL, Section 4 details the experimental setup using Mountain Car, Ship Steering and Capture Go as example domains, and the last three sections give results, future work and conclusions.

2. Gaussian Process Reinforcement Learning

In RL domains with large or infinite state spaces, function approximation becomes necessary as it is impractical or impossible to store a table of all state values (Sutton & Barto, 1998). Gaussian Processes (GPs) have emerged as a principled method for solving regression problems in Machine Learning (Rasmussen & Williams, 2006), and have recently been extended to performing function approxima-

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

tion in RL as well (Engel et al., 2005). In this section, we briefly review GPs and their application to temporal difference learning.

GPs are a class of statistical generative models for Bayesian nonparametric inference. Instead of relying on a fixed functional form as in parametric model, GPs are defined directly in some (infinite-dimensional) function space (Rasmussen & Williams, 2006). Specifically, an indexed collection of random variables $V : \mathcal{X} \rightarrow \Re$ over a common probability space is a GP if the distribution of any finite subset of V is Gaussian. Gaussian processes are completely specified by prior mean and covariance functions. In this paper, the mean function is assumed to be identically zero and the prior covariance function is specified as a Mercer kernel $k(\cdot, \cdot)$. Mechanistically, a GP is composed of the set of training data and a prior covariance function (kernel) that defines how to interpolate between those points.

Following the formulation of (Engel, 2005), consider a statistical generative model of the form

$$R(\mathbf{x}) = \mathbf{H}V(\mathbf{x}) + N(\mathbf{x}), \tag{1}$$

where V is the unknown function to be estimated, N is a noise model, $\mathbf{H}$ is a linear transformation, and R is the observed regression function. Given a set of data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=0}^{t}$, the model reduces to a system of linear equations $R_t = \mathbf{H}_t V_t + N_t$, where $R_t = (R(\mathbf{x}_0), \ldots, R(\mathbf{x}_t))^\top$, $V_t = (V(\mathbf{x}_0), \ldots, V(\mathbf{x}_t))^\top$, and $N_t = (N(\mathbf{x}_0), \ldots, N(\mathbf{x}_t))^\top$.

Assuming that $V \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_t)$ is a zero-mean GP with $[\mathbf{K}_t]_{i,j} \stackrel{\text{def}}{=} k(\mathbf{x}_i, \mathbf{x}_j)$ for $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{D}$ and $N \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_t)$, then the Gauss-Markov theorem gives the posterior distribution of V conditional on the observed R:

$$\hat{V}_t(\mathbf{x}) = \mathbf{k}_t(\mathbf{x})^\top \boldsymbol{\alpha}_t, \tag{2}$$

$$P_t(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_t(\mathbf{x})^\top \mathbf{C}_t \mathbf{k}_t(\mathbf{x}), \tag{3}$$

where

$$\boldsymbol{\alpha}_t = \mathbf{H}_t^\top (\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \mathbf{\Sigma})^{-1} \mathbf{r}_{t-1},$$

$$\mathbf{C}_t = \mathbf{H}_t^\top (\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \mathbf{\Sigma})^{-1} \mathbf{H}_t,$$

and $\mathbf{k}_t(\mathbf{x}) \stackrel{\text{def}}{=} (k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_t))^\top$. This closed-form posterior can be used to calculate the predicted value of V at some new test point $\mathbf{x}^*$. In RL, the sequence of observed reward values are assumed to be related by some (possibly stochastic) environment dynamics that are captured through the matrix $\mathbf{H}$.

In order to adapt GPs to RL, the standard Markov Decision Process (MDP) framework needs to first be formalized as follows. Let $\mathcal{X}$ and $\mathcal{U}$ be the state and action spaces, respectively. Define $R : \mathcal{X} \rightarrow \Re$ to be the reward function and let $p : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \rightarrow [0, 1]$ be the state transition probabilities. A policy $\mu : \mathcal{X} \times \mathcal{U} \rightarrow [0, 1]$ is a mapping from states to action selection probabilities. The *discounted return* for a state $\mathbf{x}$ under policy μ is defined as

$$D(\mathbf{x}) = \sum_{i=0}^{\infty} \gamma^i R(\mathbf{x}_i) | \mathbf{x}_0 = \mathbf{x},$$

where $\mathbf{x}_{i+1} \sim p^\mu(\cdot|\mathbf{x}_i)$, the policy-dependent state transition probability distribution, and $\gamma \in [0, 1]$ is the discount factor. The goal of RL is to compute a *value function* that estimates the discounted reward for each state under a policy μ, $V(\mathbf{x}) = \mathbf{E}_\mu[D(\mathbf{x})]$.

GPs can be used to model the latent value function given a sequence of observed rewards and an appropriate noise model. Reward is related to value by

$$R(\mathbf{x}) = V(\mathbf{x}) - \gamma V(\mathbf{x}') + N(\mathbf{x}, \mathbf{x}'),$$

where $\mathbf{x}' \sim p^\mu(\cdot|\mathbf{x})$. Extending this model temporally to a series of states $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t$ yields the system of equations $R_{t-1} = \mathbf{H}_t V_t + N_t$, where R and V are defined as before, and

$$N_t = (N(\mathbf{x}_0, \mathbf{x}_1), \ldots, N(\mathbf{x}_{t-1}, \mathbf{x}_t))^\top,$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & 0 & \ldots & 0 \\ 0 & 1 & -\gamma & \ldots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \ldots & 1 & -\gamma \end{bmatrix}$$

and $N_t \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_t)$. If the environment dynamics are assumed to be deterministic, the covariance of the state-dependent noise can be modeled as $\mathbf{\Sigma}_t = \sigma^2 \mathbf{I}$. For stochastic environments, the noise model $\mathbf{\Sigma}_t = \sigma^2 \mathbf{H}_{t+1} \mathbf{H}_{t+1}^\top$ is more suitable. See (Engel, 2005) for a complete derivation for these models.

Given $\mathbf{H}_t$, $\mathbf{\Sigma}_t$, and a sequence of states and reward values, the posterior moments $\hat{V}_t$ and P_t can be computed to yield value function estimates. Thus GPs fit naturally into the RL framework: Learning is straightforward and does not require setting unintuitive parameters such as α or λ; prior knowledge of the problem can be built in through the covariance function; the full distribution of the posterior is available, making it possible to select actions in more compelling ways, e.g. via interval estimation. Furthermore, the value estimator $\hat{V}_t$ and covariance P_t can be computed incrementally online as each new state action pair is sampled, without having to invert a $t \times t$ matrix at each step. For details of this procedure, see (Engel et al., 2005).

One issue with using GPs for RL is that the size of $\mathbf{K}_t$, $\mathbf{k}(\cdot)$, $\mathbf{H}_t$ and $\mathbf{r}_t$ each grow linearly with the number of states visited, yielding a computational complexity of $O(|\mathcal{D}|^2)$ for each step. Since it is not practical to remember every single experience in online settings, the GP dictionary size must

be limited in some way. To this end, Engel et al. derive a *kernel sparsification* procedure based on an approximate linear dependence (ALD) test. As the number of observed training examples tends to infinity, the number of examples that need to be saved tends to zero (Engel et al., 2005). A matrix $\mathbf{A}_t$ contains approximation coefficients for the ALD test and a parameter ν controls how "novel" a particular training example must be before it is remembered by the GP, making it possible to tune how compact and computationally efficient the value function representation is.

Finally, note that GPTD can be extended to the case where no environment model is available, simply by defining the covariance function over state-action pairs $k(\mathbf{x}, \mathbf{u}, \mathbf{x}', \mathbf{u}')$. This procedure will be termed GP-SARSA in this paper.

GPTD has been shown to be successful, but in practice performance relies on a good choice of kernel. The next section will focus on a particular online method for performing such kernel selection.

3. Online Model Selection

A common requirement in RL is that learning take place *online*, e.g. the learner must maximize total reward accrued. However, traditional model selection techniques applied to GPs, such as cross-validation, or Bayesian Model Averaging, are not designed to address this constraint. The main contribution of this paper is to introduce *Replacing-Kernel Reinforcement Learning* (RKRL), an online procedure for model selection in RL. In section 3.1 an online sequential Monte-Carlo method developed and used to implement RKRL, as described in section 3.2.

3.1. Sequential Monte-Carlo Methods

Given a set of kernels $\{k_{\boldsymbol{\theta}}(\cdot,\cdot)|\boldsymbol{\theta} \in \mathcal{M}\}$ parameterized by a random variable $\boldsymbol{\theta}$ and a prior $p(\boldsymbol{\theta})$ over these parameterizations, a fully Bayesian approach to learning involves integrating over all possible settings of $\boldsymbol{\theta}$, yielding the posterior distribution

$$p(R|\mathcal{D}) = \iint p(R|V, \mathcal{D}, \boldsymbol{\theta}) p(V|\mathcal{D}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) dV d\boldsymbol{\theta}.$$

The integration over V given $\boldsymbol{\theta}$ is carried out implicitly when using GPs, however, the remaining integral over $\boldsymbol{\theta}$ is generally intractable for all but the most simple cases. Instead of integrating over all possible model settings $\boldsymbol{\theta}$, we can use the data distribution $\mathcal{D}$ to infer reasonable settings for $\boldsymbol{\theta}$ via $p(\boldsymbol{\theta}|\mathcal{D})$. Such *evidence approximation* can is more computationally efficient and is an example of an empirical Bayes approach, where likelihood information is used to guide prior selection (Bishop, 2006).

Monte-Carlo methods can be used to sample from $p(\boldsymbol{\theta}|\mathcal{D})$, however, such methods assume that this distribution is stationary (Bishop, 2006). In the RL case, stationarity implies that when evaluating $\boldsymbol{\theta}$, previous data acquired while evaluating $\boldsymbol{\theta}'$ cannot be used. To avoid this inefficiency, we instead employ a *sequential* Monte-Carlo (SMC) method adapted from (Gordon et al., 1993) that relaxes the stationarity assumption.

Algorithm 1 Sequential Monte Carlo

Parameters: n, μ, τ, Λ

1: Draw $\{\boldsymbol{\theta}_i^{(0)}\}_{i=1}^n \sim p(\boldsymbol{\theta})$
2: **for** $t = 0, 1, \ldots$ **do**
3: Calculate $\{w_i^{(t)}\}_{i=1}^n$ from equation 4.
4: Draw $\{\tilde{\boldsymbol{\theta}}_i^{(t+1)}\}_{i=1}^n$ by resampling $\{(\boldsymbol{\theta}_i^{(t)}, w_i^{(t)})\}_{i=1}^n$.
5: $\boldsymbol{\theta}_i^{(t+1)} \leftarrow \tilde{\boldsymbol{\theta}}_i^{(t+1)} + (c_0\phi_0, \ldots, c_k\phi_k)^\top$ where $c_k \sim$ Bernoulli(μ) and $\phi_k \sim \mathcal{N}(0,1)$.
6: **end for**

SMC approximates the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ at time t empirically via a set of n samples and weights $\{(\boldsymbol{\theta}_i^{(t)}, w_i^{(t)})\}_{i=1}^n$ where

$$w_i \stackrel{\text{def}}{=} \frac{p(\mathcal{D}|\boldsymbol{\theta}_i^{(t)}) p(\boldsymbol{\theta}_i^{(t)})}{\sum_m p(\mathcal{D}|\boldsymbol{\theta}_m^{(t)}) p(\boldsymbol{\theta}_m^{(t)})}, \tag{4}$$

where $p(\mathcal{D}|\boldsymbol{\theta}_i^{(t)})$ is the likelihood of $\boldsymbol{\theta}_i^{(t)}$ and $p(\boldsymbol{\theta}_i^{(t)})$ is the prior. Inference proceeds sequentially with samples for time $t+1$ drawn from the empirical distribution

$$p(\boldsymbol{\theta}^{(t+1)}|\mathcal{D}) = \sum_l w_l^{(t)} p(\boldsymbol{\theta}^{(t+1)}|\boldsymbol{\theta}_l^{(t)}), \tag{5}$$

where $p(\boldsymbol{\theta}^{(t+1)}|\boldsymbol{\theta}^{(t)})$ is the *transition kernel*, defining how the hyperparameter space should be explored. In this paper, the prior over models $p(\boldsymbol{\theta})$ is the uniform distribution over $[0, 1]$ for each of the kernel hyperparameters (listed in table 1) and the transition kernel $p(\boldsymbol{\theta}^{(t+1)}|\boldsymbol{\theta}^{(t)})$ is defined mechanistically as $\boldsymbol{\theta}^{(t+1)} \leftarrow \tilde{\boldsymbol{\theta}}^{(t+1)} + (c_0\phi_0, \ldots, c_k\phi_k)^\top$ where $c_k \sim$ Bernoulli(μ) and $\phi_k \sim \mathcal{N}(0,1)$. Pseudocode for this procedure is given in algorithm 1.

3.2. Replacing-Kernel Reinforcement Learning

In Replacing-Kernel Reinforcement Learning (RKRL), SMC is used to select good kernel hyperparameter settings. Rather than calculating the true model likelihood $p(\mathcal{D}|\boldsymbol{\theta}_i)$ in equation 4, RKRL instead weights models based on their relative *predictive likelihood*, $\tilde{p}(\mathcal{D}|\boldsymbol{\theta}_i)$, where

$$\log \tilde{p}(\mathcal{D}|\boldsymbol{\theta}_i) \stackrel{\text{def}}{=} \tau^{-1} \sum_t r_t,$$

and $(r_0, r_1, \ldots)$ is the sequence of rewards obtained by evaluating the hyperparameter setting $\boldsymbol{\theta}_i$ for Λ episodes.

Table 1. Basic kernel functions and the corresponding extended parameterizations.

KERNEL	BASIC	EXTENDED
NORM	$k(\mathbf{x},\mathbf{x}') = 1 - \frac{\|\|\mathbf{x}-\mathbf{x}'\|\|^2}{\alpha}$	$k(\mathbf{x},\mathbf{x}') = 1 - \sum_i w_i(x_i - x_i')^2$
GAUSSIAN	$k(\mathbf{x},\mathbf{x}') = \exp\left[\frac{-\|\|\mathbf{x}-\mathbf{x}'\|\|^2}{\sigma^2}\right]$	$k(\mathbf{x},\mathbf{x}') = \exp\left[-\sum_i w_i(x_i - x_i')^2\right]$
POLYNOMIAL	$k(\mathbf{x},\mathbf{x}') = (\langle \mathbf{x},\mathbf{x}'\rangle + 1)^d$	$k(\mathbf{x},\mathbf{x}') = (\sum_i w_i x_i x_i' + 1)^d$
TANH NORM	$k(\mathbf{x},\mathbf{x}') = \tanh(v\|\|\mathbf{x}-\mathbf{x}'\|\|^2 - c)$	$k(\mathbf{x},\mathbf{x}') = \tanh(\sum_i w_i(x_i - x_i')^2 - 1)$
TANH DOT	$k(\mathbf{x},\mathbf{x}') = \tanh(v\langle \mathbf{x},\mathbf{x}'\rangle - c)$	$k(\mathbf{x},\mathbf{x}') = \tanh(\sum_i w_i x_i x_i' - 1)$

The parameter τ is introduced to control how strongly model search should focus on hyperparameter settings that yield high reward. Maximizing predictive ability directly is preferable as it is more closely related to the goal of learning than maximizing the fit to the observed data. In tabular methods these two approaches indeed coincide in the limit of large data, however when using function approximation, they may differ.

When using GPTD, the current value function estimate is formed from the combination of the kernel parameterization $\boldsymbol{\theta}$ determining the prior covariance function and the dictionary $\tilde{\mathcal{D}} \subseteq \mathcal{D}$ gathered incrementally from observing state transitions. In this paper we consider two variants of RKRL: Standard RKRL and *Experience-Preserving* RKRL (EP-RKRL) that differ based on their treatment of the saved experience $\tilde{\mathcal{D}}$. In Standard RKRL, $\tilde{\mathcal{D}}$ is discarded at the start of each new kernel evaluation (making $p(\boldsymbol{\theta}|\mathcal{D})$ stationary). In contrast, in EP-RKRL each kernel parameterization sample $\boldsymbol{\theta}^{(t)}$ inherits $\tilde{\mathcal{D}}$[1] from the sample $\boldsymbol{\theta}^{(t-1)}$ that generated it in equation 5.

RKRL naturally spends more time evaluating hyperparameter settings that correspond to areas with high predictive likelihood, i.e. maximizes online reward. Each sampling step increases information about the predictive likelihood in the sample (exploitation), while sampling from the transition kernel reduces such information (exploration).

4. Experimental Setup

Standard RKRL and EP-RKRL are compared against GP-SARSA on three domains. This section gives the parameter settings, kernel classes and domains used.

4.1. Parameters

In all experiments, the TD discount factor was fixed at $\gamma = 1.0$ and ϵ-greedy action selection was employed with $\epsilon = 0.01$. The GP-SARSA parameters for prior noise variance (σ) and dictionary sparsity (ν) were $\sigma = 1.0$ and $\nu = 0.001$. For each RKRL evaluation, GP-SARSA is run for Λ episodes using the specified kernel parameterization. The RKRL parameters were set to $n = 25$, $\mu = 0.01$ and $\tau = 0.5$. Performance of RKRL is insensitive to changes doubling Λ or μ. Higher settings of n improve the initial performance, but reduce the total number of epochs possible given a fixed number of episodes. The setting of τ significantly impacts performance, although the main results of this paper are insensitive for $0.25 \leq \tau \leq 1.0$. All kernels are extended to functions of both the state and action, with actions treated as extra state variables.

[1]For efficiency the sufficient statistics $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\mathbf{C}}$ for sparsified GP-SARSA are also inherited, though they can be recalculated. The matrix of approximation coefficients $\mathbf{A}_t$ is not recalculated, although doing so should lead to be better performance in general.

4.2. Kernels

Although RKRL automates the choice of kernel hyperparameters, there is still a need to choose a set of kernels that represents the search space for RKRL. General kernel classes are derived from basic classes commonly found in the literature (table 1) by replacing the standard inner products and norms with weighted variants (cf. *automatic relevance determination*), yielding kernel classes with significantly more hyperparameters. Setting these hyperparameters is the model selection task; as more hyperparameters are added, the model becomes more general, but the corresponding difficulty of inferring the model parameters increases as well.

In order to give a fair baseline GP-SARSA comparison, the best hyperparameter setting for each basic kernel class was derived manually for each domain using grid search. Note that although they are common, the hyperbolic tangent kernels are not positive semi-definite; however they still yield good performance in practice (Smola & Schölkopf, 2004).

4.3. Test Domains

GP-SARSA, RKRL and EP-RKRL are compared across three domains: Mountain Car, Ship Steering and Capture Go. Each domain highlights a different aspect of complexity found in RL problems: Mountain Car and Ship Steering have continuous state spaces and thus require function approximation, Ship Steering also has a large (discrete) action space, and Capture Go is stochastic with a high-dimensional state space.

4.3.1. MOUNTAIN CAR

In *Mountain Car*, the learning agent must drive an underpowered car up a steep hill (Sutton & Barto, 1998). The available actions are $a \in \{-1, 0, 1\}$, i.e., brake, neutral and accelerate. The state $\mathbf{x}_t = (x_t, \dot{x}_t) \in \Re^2$ is comprised of the position and velocity. The environment is deterministic with state evolution governed by

$$\begin{aligned} x_{t+1} &= x_t + \dot{x}_{t+1} \\ \dot{x}_{t+1} &= \dot{x}_t + 0.001a_t + -0.0025\cos(3x_t) \end{aligned}$$

where $-1.2 \leq x \leq 0.5$ and $|\dot{x}| \leq 0.07$. Reward is -1 for each time step the car has not passed the goal at $x = 0.5$. In all RKRL experiments with Mountain Car, $\Lambda = 100$ (100 episodes per epoch) and each episode is limited to 1000 steps to reduce computation time.

4.3.2. SHIP STEERING

In *Ship Steering*, the learning agent must properly orient a sailboat to a specific heading and travel as fast as possible (White, 2007). Actions are two-dimensional rudder position (degrees) and thrust (Newtons), $\mathbf{a}_t = (r_t, T_t) \in [-90, 90] \times [-1, 2]$. Possible rudder settings are discretized at 3-degree increments and thrust increments are 0.5 Newtons, yielding 427 possible actions. The state is a 3-tuple consisting of the heading, angular velocity and velocity $\mathbf{x}_t = (\theta_t, \dot{\theta}_t, \dot{x}_t) \in \Re^3$. State evolution is described by

$$\begin{aligned} \dot{x}_{t+1} &= \dot{x}_t + \frac{1}{250}(30T_t - 2\dot{x}_t - 0.03\dot{x}_t(5\theta_t + r_t^2)) \\ \dot{\theta}_{t+1} &= \dot{\theta}_t + \frac{\dot{x}_t r_t + \dot{x}_t}{1000} \\ \theta_{t+1} &= \theta_t + 0.5(\dot{\theta}_{t+1} + \dot{\theta}_t) \end{aligned}$$

Reward at time step t is equal to $\dot{x}_t$ if $|\theta_t| < 5$ and zero otherwise. In all RKRL experiments with Ship Steering, $\Lambda = 1$. By comparing results in Ship Steering to Mountain Car, it is possible to elucidate how the learner's performance depends on the size of the action space.

4.3.3. CAPTURE GO

The third domain used in this paper is *Capture Go*, a simplified version of Go played where the first player to make a capture wins.[2] The learner plays against a fixed random opponent on a 5×5 board, and receives reward of -1 for a loss and +1 for a win. The board state $\mathbf{x}_t \in \{-1, 0, 1\}^{25}$ is encoded as a vector where -1 entries correspond to opponent pieces, 0 entries correspond to blank territory and 1 entries correspond to the agent's pieces. The agent is given knowledge of *afterstates*, that is, knowledge of how its moves affect the state. In all RKRL experiments using Capture Go, $\Lambda = 1000$. This domain was chosen because it has a high dimensional state vector and stochastic dynamics.

[2] http://www.usgo.org/teach/capturegame.html

Table 2. Asymptotic performance on Mountain car. Bold numbers represent statistical significance.

KERNEL	GP-SARSA	RKRL	EP-RKRL
POLYNOMIAL	-67.6 ±0.3	-149 ±5.5	**-63.7 ±0.3**
GAUSSIAN	-230 ±16	-521 ±84	**-66.9 ±0.9**
TANH NORM	-638 ±72	-569 ±41	**-130 ±8.7**
TANH DOT	-482 ±37	-532 ±113	**-97.0 ±2.0**

Table 3. Asymptotic performance ($\times 10^2$) on Ship Steering.

KERNEL	GP-SARSA	RKRL	EP-RKRL
POLYNOMIAL	34.0 ±1.0	3.3 ±0.7	**171 ±241**
GAUSSIAN	2.5 ±0.3	5.0 ±0.6	12.9 ±9.1
TANH NORM	2.1 ±0.8	4.5 ±0.7	**662 ±183**
TANH DOT	2.9 ±0.6	3.0 ±0.8	**19.5 ±15.3**

Table 4. Asymptotic performance (% wins) on Capture Go.

KERNEL	GP-SARSA	RKRL	EP-RKRL
NORM	90.9 ±0.2	76.1 ±4.4	**94.3 ±0.5**
POLYNOMIAL	89.7 ±0.4	69.5 ±1.1	**92.6 ±1.3**
GAUSSIAN	90.3 ±0.5	78.3 ±0.7	**93.3 ±0.1**
TANH NORM	55.7 ±0.2	78.7 ±3.7	**94.5 ±0.6**
TANH DOT	62.4 ±2.8	70.5 ±1.5	**89.1 ±1.1**

5. Results

GP-SARSA, RKRL and EP-RKRL were applied to three RL domains. Section 5.1 summarizes asymptotic performance in the three domains, Section 5.2 compares asymptotic dictionary sizes, Section 5.3 evaluates the learned kernel performance as a stand-alone static kernel and Section 5.4 analyzes the learned kernel hyperparameter settings in Capture Go.

5.1. Asymptotic Reward

Asymptotic performance is evaluated across three domains: Mountain Car, Ship Steering and Capture Go. In each domain EP-RKRL significantly outperforms both GP-SARSA and RKRL over most kernel classes.

5.1.1. MOUNTAIN CAR

In Mountain Car, learning trials are run for 125,000 episodes and asymptotic performance is measured as the average reward over the last 100 episodes. EP-RKRL significantly outperforms both GP-SARSA and RKRL across all kernel classes asymptotically (table 2). GP-SARSA performance using the POLYNOMIAL kernel reaches a peak at -51.6 after 33 episodes, which is significantly better than

EP-RKRL ($p < 10^{-5}$). However, performance degrades significantly with more episodes. This is a phenomenon common to neural-network based function approximators (Sutton & Barto, 1998).

The Mountain Car problem has been studied extensively in RL literature. The best asymptotic results from the Reinforcement Learning Library stand at -53.92 (White, 2007). NEAT+Q, a similar method for combining TD and evolutionary algorithms, achieves -52.0 (Whiteson & Stone, 2006). However, in the former case, different values for ϵ and γ are used and in the latter case, the learner is run for significantly more episodes, making direct comparison difficult. Running Mountain Car using a standard tile-coding function approximator (Sutton & Barto, 1998) with 8 tilings and the same RL parameter settings yields asymptotic performance of -108.9, significantly better than GP-SARSA across all kernels except POLYNOMIAL, but significantly worse than EP-RKRL under all kernels except TANH NORM.

Note that EP-RKRL significantly outperforms RKRL because it discards less experience over the course of learning. Since kernels can only describe smoothness properties of the value functions, the data points themselves become more important for learning; hence discarding them at each model selection step significantly reduces performance. This contrasts with Whiteson's NEAT+Q work precisely because neural networks are more expressive.

5.1.2. SHIP STEERING

In Ship Steering, each learner trains for 2500 episodes (1000 steps each) and the asymptotic performance is measured as the average reward obtained in the last 10 episodes. EP-RKRL significantly outperforms GP-SARSA and RKRL in all kernel classes except GAUSSIAN (table 3). In the remaining three cases, however, EP-RKRL outperforms both methods by several orders of magnitude. Tile coding with 8 tilings yields asymptotic performance of 0.17, significantly higher performance than GP-SARSA in all cases[3] except for the POLYNOMIAL kernel class ($p < 10^{-9}$), but significantly worse than EP-RKRL in all cases.

5.1.3. CAPTURE GO

In Capture Go, each learner trains for $3.75 \cdot 10^6$ episodes, and asymptotic performance is measured as the average number of wins over the last 1000 episodes. EP-RKRL outperforms GP-SARSA and RKRL across all kernel classes (table 4). GP-SARSA's average reward peaks early and declines under the TANH DOT kernel, achieving a maximum of 78.7% wins after 10,000 episodes, still significantly lower than EP-RKRL ($p < 10^{-6}$).

[3]Performance values in table 3 are scaled by a factor of 100.

Table 5. Asymptotic dictionary size for Mountain Car. Bold numbers indicate statistical significance.

KERNEL	GP-SARSA	RKRL	EP-RKRL
POLYNOMIAL	21.6 ±0.4	**10.3 ±0.5**	13.6 ±0.2
GAUSSIAN	29.6 ±0.5	**7.2 ±0.6**	12.5 ±0.2
TANH NORM	**2.5 ±0.1**	8.5 ±0.8	12.6 ±0.2
TANH DOT	7.7 ±0.7	**5.1 ±0.4**	11.7 ±0.4

Table 6. Asymptotic dictionary size for Ship steering.

KERNEL	GP-SARSA	RKRL	EP-RKRL
POLYNOMIAL	15.3 ±0.8	**4.0 ±0.1**	6.2 ±0.3
GAUSSIAN	**12.3 ±0.5**	15.5 ±0.3	13.7 ±0.1
TANH NORM	**3.8 ±0.6**	5.1 ±0.2	12.3 ±1.0
TANH DOT	7.7 ±0.8	**3.2 ±0.1**	5.3 ±0.2

Table 7. Asymptotic dictionary size for Capture Go.

KERNEL	GP-SARSA	RKRL	EP-RKRL
NORM	28.5 ±0.2	5.9 $\pm\epsilon$	**3.0** $\pm\epsilon$
POLYNOMIAL	147.1 ±3.3	**25.2 ±1.4**	40.4 ±1.4
GAUSSIAN	**66.1 ±0.5**	71.7 ±3.4	91.9 ±6.1
TANH NORM	62.0 ±1.1	19.6 ±0.4	**17.5 ±0.7**
TANH DOT	329.6 ±14.4	**25.6 ±1.5**	28.7 ±1.2

5.2. Dictionary Size

RKRL and GP-SARSA can be compared in terms of computational complexity by measuring the final dictionary sizes $|\tilde{\mathcal{D}}|$ of each learning agent. At each decision point, the computational complexity of GPTD is $O(|\tilde{\mathcal{D}}|^2)$, arising from matrix-vector multiplications and partitioned matrix inversion (Engel, 2005). Furthermore, in practice the $O(|\tilde{\mathcal{D}}|)$ cost of computing $\mathbf{k}(\mathbf{x}) \stackrel{\text{def}}{=} (k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_t))^\top$ for $\mathbf{x}_i \in \tilde{\mathcal{D}}$ can carry a high constant overhead for complex kernels. Thus, keeping $|\tilde{\mathcal{D}}|$ small is critical for online performance.

The dictionary sizes for each kernel and learning algorithm pair is given in table 5. In eight of the thirteen cases, EP-RKRL kernels generate significantly smaller dictionaries for $\nu = 0.001$ than GP-SARSA kernels, and likewise in ten of the thirteen cases RKRL generates significantly smaller dictionaries. Thus in the majority of cases employing model selection yields faster learning both in terms of episodes and in terms of computation. However, these dictionary sizes never differ by more than a single order of magnitude, with the largest difference being between RKRL and GP-SARSA the TANH DOT kernel for Capture Go.

5.3. Generalization

How well a particular kernel hyperparameter setting found by EP-RKRL performs depends on what training examples it encounters during learning. Determining to what degree

Table 8. Generalization performance of EP-RKRL kernel parameterizations for Capture Go. In all cases asymptotic performance declines after the original dictionary is discarded.

KERNEL	BASELINE	RELEARNED
NORM	94.3 ±0.5	68.6 ±1.7
POLYNOMIAL	92.6 ±1.3	72.5 ±0.8
GAUSSIAN	93.3 ±0.1	81.1 ±1.0
TANH NORM	94.5 ±0.6	90.7 ±1.9
TANH DOT	89.1 ±1.1	74.6 ±0.7

this performance depends on the exact set of saved training examples yields a notion of how general the kernel parameterization is. In order to evaluate this generalization in EP-RKRL, final kernel parameterizations for Capture Go were saved and all stored training examples were discarded. Learning was then restarted with the kernel parameterization fixed. Table 8 summarizes the results. Across all kernels, the asymptotic performance decreases significantly. Because most training examples are acquired in the first several hundred episodes, this result indicates that the kernel hyperparameter settings are perhaps overfitting to the particular dictionary of saved experience. In other words RKRL is exploiting the acquired data to pick a highly biased covariance function that has low generalization error given that particular set of stored experience.

Although kernels learned through RKRL overfit the dictionary, this is not a serious problem as it improves generalization performance. Overfitting in this nonparametric case simply means that the learned parameterizations are not transferable between agents with different experience. Thus the saved dictionary should be considered part of the model parameters being optimized.

5.4. Analysis of Learned Kernels

Because the hyperparameter space for kernels in Capture Go is high dimensional, RKRL can exploit many kinds of symmetries and patterns in the value function. It is enlightening to analyze whether the learned kernel parameter settings correspond to intuitively meaning covariance functions. Figure 1 plots the 25 hyperparameter values for the TANH NORM kernel averaged over the entire sample for both RKRL and EP-RKRL. The average pattern of weights at the final epoch differs significantly from the expected average over the prior $p(\boldsymbol{\theta})$. Furthermore, the pattern learned in EP-RKRL has a regular structure: Each weight corresponding to a particular board location takes the opposite sign of its neighbors, yielding a regular "checkerboard" pattern of positive and negative weights. This pattern enforces a simple strategy whereby the learner plays to capture isolated stones. Such a strategy is effective against opponents that do not pay attention to stones in danger of being captured.

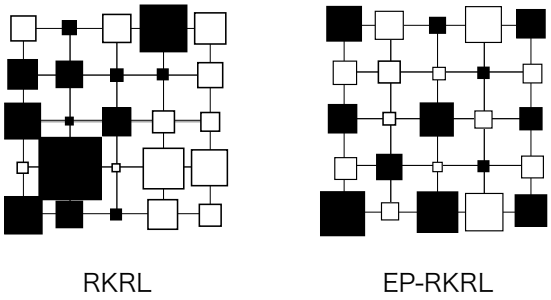


Figure 1. "Hinton diagram" of the average learned hyperparameters by board position for the TAHN NORM kernel in Capture Go. Filled boxes correspond to positive weights and white boxes to negative; box area is proportional to the weight. The parameterization generated by EP-RKRL shows a significant amount of structure, biasing play towards moves that surround single stones.

To further elucidate this result, a second trial was run using the same kernel class, but with only two hyperparameters, corresponding to the positive and negative parameter settings observed above. This *translation invariant* TANH NORM kernel $k(\mathbf{x}, \mathbf{x}') = \tanh(\sum_{i=0}^{25} w_{i \bmod 2}(x_i - x'_i)^2 - 1)$ can express the same alternating positive and negative weights in a more compact form, thus trading off generality compared to the original parameterization. Under the same experimental setup, the translation invariant kernel only reaches an asymptotic performance of 88.6% wins, compared to 94.5% wins with the more general parameterization ($p < 10^{-7}$). Furthermore, the resulting dictionary size of the translation invariant kernel is 32.4, significantly larger than more general parameterization (17.4; $p < 0.001$). The general TANH NORM kernel parameterization also outperforms a variant with built-in rotational symmetry. The *rotationally symmetric* TANH NORM kernel obtains asymptotic performance of 89.3% wins ($p < 10^{-4}$). Taken together these results highlight some of the difficulties of manually building in prior knowledge.

6. Related and Future Work

This paper has presented an efficient and conceptually simple online method for selecting kernels in Bayesian RL. There is a growing body of model selection literature in machine learning and statistics, both on theory and applications, e.g. (Hastie et al., 2001; Seeger, 2001). In RL, model selection has been performed previously using regularization (Jung & Polani, 2006; Loth et al., 2007) and evolutionary methods (Whiteson & Stone, 2006). RKRL is most similar to the latter approach, differing in its use of GPs, ability to save training data across lineages, and simpler hyperparameter optimization procedure.

There are several interesting areas of future work. First, RKRL can be naturally applied to more general kernel

classes, e.g. the Matérn kernels, that admit many basic kernels as special cases (Genton, 2002). In particular, the tradeoff between kernel class complexity and performance should be explored.

Second, the learned kernel parameterizations acquired under one dictionary do not perform well under different dictionaries, indicating that the dictionaries themselves can be thought of as hyperparameters to be optimized. Developing such a theory of nonparametric sparsification in RL may lead to significantly better value function approximations.

Third, it is possible to derive a *full kernel-replacing* procedure where all covariance function evaluations share the same accumulated learning, updated after every fitness evaluation of every individual. Such an approach would further reduce the sample complexity of RKRL, and also makes possible the use of *Bayesian Model Averaging* techniques within a single learning agent, i.e. averaging over an ensemble of GP value functions weighted by their predictive likelihoods (Hastie et al., 2001).

Finally, there is a deep connection between action kernels and the concept of Relocatable Action Models (Leffler et al., 2007). It may be possible to cast the latter in terms of state-independent prior covariance functions, yielding a powerful framework for model selection.

7. Conclusion

This paper developed RKRL, a simple online procedure for improving the performance of Gaussian process temporal difference learning by automatically selecting the prior covariance function. In several empirical trials, RKRL yielded significantly higher asymptotic reward than the best hand-picked parameterizations for common covariance functions, even in cases where a large number of hyperparameters must be adapted. Furthermore, the learned covariance functions exhibit highly structured knowledge of the task that would have been difficult to build in *a priori* without significant knowledge of the domain. Overall these initial results are promising, and suggest that leveraging work in statistical model selection will significantly improve online learning.

Acknowledgements

Thanks to the anonymous reviewers for providing very poignant feedback on the original draft, also thanks to Yaakov Engel for help with some implementation details and Tobias Jung and Bryan Silverthorn for helpful discussions. This work was supported in part by an NSF Graduate Research Fellowship to the first author and NSF CAREER award IIS-0237699.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Engel, Y. (2005). *Algorithms and representations for reinforcement learning*. Doctoral dissertation, Hebrew University.

Engel, Y., Mannor, S., & Meir, R. (2005). Reinforcement learning with gaussian processes. *Proc. of ICML-05* (pp. 201–208). New York, NY, USA: ACM Press.

Genton, M. G. (2002). Classes of kernels for machine learning: a statistics perspective. *Journal of Machine Learning Research*, *2*, 299–312.

Gordon, N. J., Salmond, D. J., & Smith, A. F. M. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, *140*, 107–113.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning*. Springer.

Jung, T., & Polani, D. (2006). Least squares svm for least squares td learning. *ECAI* (pp. 499–503). IOS Press.

Leffler, B. R., Littman, M. L., & Edmunds, T. (2007). Efficient reinforcement learning with relocatable action models. *Proc. of AAAI-07* (pp. 572–577). Menlo Park, CA, USA: The AAAI Press.

Loth, M., Davy, M., & Preux, P. (2007). Sparse temporal difference learning using lasso. *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. Hawaii, USA.

Rasmussen, C. E., & Williams, C. K. (2006). *Gaussian processes for machine learning*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press.

Seeger, M. (2001). Covariance kernels from bayesian generative models. *NIPS* (pp. 905–912). MIT Press.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*, 199–222.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning*. Cambridge, MA, USA: MIT Press.

White, A. (2007). The University of Alberta Reinforcement Learning Library. `http://rlai.cs.ualberta.ca/RLR/`. Edmonton, Alberta: University of Alberta.

Whiteson, S., & Stone, P. (2006). Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, *7*, 877–917.

The Dynamic Hierarchical Dirichlet Process

Lu Ren LR@EE.DUKE.EDU
Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA

David B. Dunson DUNSON@STAT.DUKE.EDU
Department of Statistical Science, Duke University, Durham, NC 27708, USA

Lawrence Carin LCARIN@EE.DUKE.EDU
Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA

Abstract

The dynamic hierarchical Dirichlet process (dHDP) is developed to model the time-evolving statistical properties of sequential data sets. The data collected at any time point are represented via a mixture associated with an appropriate underlying model, in the framework of HDP. The statistical properties of data collected at consecutive time points are linked via a random parameter that controls their probabilistic similarity. The sharing mechanisms of the time-evolving data are derived, and a relatively simple Markov Chain Monte Carlo sampler is developed. Experimental results are presented to demonstrate the model.

1. Introduction

The Dirichlet process (DP) mixture model (Escobar & West, 1995) has been widely used to perform density estimation and clustering, by generalizing finite mixture models to (in principle) infinite mixtures. In order to "share statistical strength" across different groups of data, the hierarchical Dirichlet process (HDP) (Teh et al., 2005) has been proposed to model the dependence among groups through sharing the same set of discrete parameters ("atoms"), and the mixture weights associated with different atoms are varied as a function of the data group. In the HDP, it is assumed that the data groups are exchangeable. However, in many real applications, such as seasonal market analysis and gene investigation for disease, data are measured in a sequential manner, and there is information in this temporal character that should ideally be exploited; this violates the aforementioned assumption of exchangeability.

Developing models for time-evolving data has recently been the focus of significant interest, and researchers have proposed various solutions directed toward specific applications. An early example is the order-based dependent DP (Griffin & Steel, 2006), in which the model is time-reversible but is not Markovian, and it requires one to specify how the mixture weights change through time. Another related work is the time-varying Dirichlet process mixture model (Caron et al., 2007) based on a modified Polya urn scheme (Blackwell & MacQueen, 1973), implemented by changing the number and locations of clusters over time. This method is easy to understand intuitively but has computational challenges for large data sets. To examine the temporal dynamics of scientific topics, latent Dirichlet allocation (Blei et al., 2003) (Griffiths & Steyvers, 2004) has been used as a generative model for analysis of documents. In order to explicitly model the dynamics of the underlying topics, Blei (Blei & Lafferty, 2006) proposed a dynamic topic model, in which the parameter at the previous time $t-1$ is the expectation for the distribution of the parameter at the next time t, and the correlation of the samples at adjacent times is controlled through adjusting the variance of the conditional distribution. Unfortunately, the non-conjugate form of the conditional distribution requires approximations in the model inference.

Recently Dunson (Dunson, 2006) proposed a Bayesian dynamic model to learn the latent trait distribution through a mixture of DPs, in which the latent variable density changes dynamically in location and shape across levels of predictors. This dynamic structure is considered in this paper to extend HDP to incorpo-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

rate time dependence, and has the following features: (*i*) two data samples drawn at proximate times have a higher probability of sharing the same underlying model parameters (atoms) than parameters drawn at disparate times; and (*ii*) there is a possibility that temporally distant data samples may also share model parameters, thereby accounting for possible distant repetition in the data.

2. Dynamic HDP

2.1. Background

A Dirichlet process is a measure on a measure G and is parameterized as $G \sim DP(\alpha_0, G_0)$, in which G_0 is a base measure and α_0 is a positive "precision" parameter. To provide an explicit form for a G drawn from $DP(\alpha_0, G_0)$, Sethuraman (Sethuraman, 1994) developed a stick-breaking construction:

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{\theta}_k^*}, \quad \pi_k = \tilde{\pi}_k \prod_{i=1}^{k-1} (1 - \tilde{\pi}_i) \tag{1}$$

where $\{\boldsymbol{\theta}_k^*\}_{k=1}^{\infty}$ represent a set of atoms drawn i.i.d. from G_0 and $\{\pi_k\}_{k=1}^{\infty}$ represent a set of weights, with the constraint $\sum_{k=1}^{\infty} \pi_k = 1$; each $\tilde{\pi}_k$ is drawn i.i.d. from $Be(1, \alpha_0)$. According to the construction in (1), a draw G from a $DP(\alpha_0, G_0)$ is discrete with probability one. Based on this important property, Teh (Teh et al., 2005) proposed a hierarchical Dirichlet process (HDP) to link the group-specific Dirichlet processes, learning the models jointly across multiple data sets.

Assume we have J groups of data and the j^{th} data set (group) is denoted as $\{x_{j,i}\}_{i=1,\ldots,N_j}$. For each of these data sets, $x_{j,i}$ is drawn from the model $x_{j,i} \overset{ind}{\sim} F(\boldsymbol{\theta}_{j,i})$ with parameters $\boldsymbol{\theta}_{j,i} \overset{iid}{\sim} G_j$, and the parameters $\{\boldsymbol{\theta}_{j,i}\}_{i=1,\ldots,N_j}$ are likely to assume the atoms $\boldsymbol{\theta}_k^*$ for which the associated sticks $\pi_{j,k}$ are large, as a consequence of the form of G_j given by (1); for the J data sets, different group-specific G_j are drawn from $DP(\alpha_{j0}, G_0)$, in which G_0 is drawn from another DP. The generative model for HDP is represented as:

$$\begin{gathered} x_{j,i} \overset{ind}{\sim} F(\boldsymbol{\theta}_{j,i}) \\ \boldsymbol{\theta}_{j,i} \overset{iid}{\sim} G_j \\ G_j \overset{ind}{\sim} DP(\alpha_{j0}, G_0) \\ G_0 \sim DP(\gamma, H) \end{gathered} \tag{2}$$

where $j = 1, \ldots, J$ and $i = 1, \ldots, N_j$.

Under this hierarchical structure, not only can different observations $x_{j,i}$ and $x_{j,i'}$ in the same group share the same parameters $\boldsymbol{\theta}^*$ based on the stick weights represented by G_j, but also the observations across different groups might share parameters as a consequence of the discrete form of G_0 (all G_j are composed of the same set of atoms $\{\boldsymbol{\theta}_k^*\}_{k=1}^{\infty}$). The clusters in each group j, assumed by the set $\{\boldsymbol{\theta}_{j,i}\}_{i=1,\ldots,N_j}$, are inferred via the posterior density function on the parameters, with the likelihood function selecting the set of discrete parameters $\{\boldsymbol{\theta}_k^*\}_{k=1}^{\infty}$ most consistent with the data $\{x_{j,i}\}_{i=1,\ldots,N_j}$. Meanwhile, clusters (and, hence, associated cluster parameters $\{\boldsymbol{\theta}_k^*\}_{k=1}^{\infty}$) are shared across multiple data sets, as appropriate.

Although the HDP introduces a dependency between the J groups, the data sets are assumed exchangeable. However, in many applications, the data may be collected sequentially, and one may have a prior belief that sharing of data is more probable when the data sets are collected at similar points in time. The purpose of this paper is to extend the HDP to account for such temporal information.

Before proceeding, it will prove useful to consider an alternative form of the HDP model, as derived in (Teh et al., 2005). Specifically, each draw G_j may be expressed as:

$$\begin{gathered} G_j = \sum_{k=1}^{\infty} \pi_{j,k} \delta_{\boldsymbol{\theta}_k^*} \\ \boldsymbol{\pi}_j \overset{ind}{\sim} DP(\alpha_{0j}, \boldsymbol{\beta}) \\ \boldsymbol{\beta} \sim Stick(\gamma) \\ \boldsymbol{\theta}_k^* \overset{iid}{\sim} H \end{gathered} \tag{3}$$

where $Stick(\gamma)$ stochastically generates an infinite set of sticks $\{\beta_1, \beta_2, \ldots\}$, based on a stick-breaking process of the form in (1), here with parameter γ, satisfying the constraint $\sum_{i=1}^{\infty} \beta_i = 1$.

2.2. Bayesian Dynamic Structure

Similar to HDP, we again consider J data sets but now using an explicit assumption that the data sets are collected sequentially, with $\{x_{1,i}\}_{i=1,\ldots,N_1}$ collected first, $\{x_{2,i}\}_{i=1,\ldots,N_2}$ collected second, and with $\{x_{J,i}\}_{i=1,\ldots,N_J}$ collected last. Since our assumption is that a time evolution exists between adjacent data groups, the distribution G_{j-1}, from which $\{\boldsymbol{\theta}_{j-1,i}\}_{i=1,\ldots,N_{j-1}}$ are drawn, is likely related to G_j, from which $\{\boldsymbol{\theta}_{j,i}\}_{i=1,\ldots,N_j}$ are drawn.

To specify explicitly the dependence between G_{j-1} and G_j, Dunson (Dunson, 2006) proposed a Bayesian dynamic mixture DP (DMDP), in which G_j shares features with G_{j-1} but some innovation may also occur. The DMDP has the drawback that mixture components can only be added over time, so that one ends up with more components at later times as an artifact of the model.

In the dHDP, we have

$$G_j = (1 - \tilde{w}_{j-1})G_{j-1} + \tilde{w}_{j-1}H_{j-1} \tag{4}$$

where $G_1 \sim DP(\alpha_{01}, G_0)$, H_{j-1} is called an innovation distribution drawn from $DP(\alpha_{0j}, G_0)$, and $\tilde{w}_{j-1} \sim Be(a_{w(j-1)}, b_{w(j-1)})$. In this way, G_j is modified from G_{j-1} by introducing a new innovation distribution H_{j-1}, and the random variable $\tilde{w}_{j-1}$ controls the probability of innovation (*i.e.*, it defines the mixture weights). As a result, the relevant atoms adjust with time, and it is probable that proximate data will share the same atoms, but with the potential for transient innovation.

Additionally, we assume that $G_0 \sim DP(\gamma, H)$ as in the HDP to enforce that G_0 is discrete, which manifests another important aspect of the dynamic HDP: the same atoms are used for *all* G_j, but with different time-evolving weights. Consequently, the model encourages sharing between temporally proximate data, but it is also possible to share between data sets widely separated in time.

Providing now more model details, the discrete base distribution drawn from $DP(\gamma, H)$ may be expressed as:

$$G_0 = \sum_{k=1}^{\infty} \beta_k \delta_{\boldsymbol{\theta}_k^*} \tag{5}$$

where $\{\boldsymbol{\theta}_k^*\}_{k=1,2,\ldots,\infty}$ are the global parameter components (atoms), drawn independently from the base distribution H and $\{\beta_k\}_{k=1,2,\ldots,\infty}$ are drawn from a stick-breaking process $\boldsymbol{\beta} \sim Stick(\gamma)$, defined as:

$$\beta_k = \tilde{\beta}_k \prod_{l<k}(1 - \tilde{\beta}_l) \qquad \tilde{\beta}_k \overset{iid}{\sim} Be(1, \gamma) \tag{6}$$

We also have J groups of data. G_j represents the prior for the mixture distribution associated with the global components in group j, H_{j-1} represents the associated prior for the innovation mixture distribution, and this yields the explicit priors used in (4):

$$\begin{gathered} G_1 = \sum_{k=1}^{\infty} \pi_{1,k} \delta_{\boldsymbol{\theta}_k^*}, H_1 = \sum_{k=1}^{\infty} \pi_{2,k} \delta_{\boldsymbol{\theta}_k^*}, \ldots, \\ H_{J-1} = \sum_{k=1}^{\infty} \pi_{J,k} \delta_{\boldsymbol{\theta}_k^*} \end{gathered} \tag{7}$$

where, analogous to the discussion at the end of Section 2.1, the different weights $\boldsymbol{\pi}_j$ are independent given $\boldsymbol{\beta}$ since $G_1, H_1, \ldots, H_{J-1}$ are independent given G_0; the relationship between $\boldsymbol{\pi}_j$ and $\boldsymbol{\beta}$ is proven (Teh et al., 2005) to be

$$\boldsymbol{\pi}_j | \alpha_{0j}, \boldsymbol{\beta} \sim DP(\alpha_{0j}, \boldsymbol{\beta}) \tag{8}$$

To further develop the dynamic relationship from G_1 to G_J, we extend the mixture structure in (4) from group to group:

$$\begin{aligned} G_j &= (1 - \tilde{w}_{j-1})G_{j-1} + \tilde{w}_{j-1}H_{j-1} \\ &= \prod_{l=1}^{j-1}(1 - \tilde{w}_l)G_1 + \sum_{l=1}^{j-1}\{\prod_{m=l+1}^{j-1}(1 - \tilde{w}_m)\}\tilde{w}_l H_l \\ &= w_{j1}G_1 + w_{j2}H_1 + \ldots + w_{jj}H_{j-1} \end{aligned} \tag{9}$$

where $w_{jl} = \tilde{w}_{l-1}\prod_{m=l}^{j-1}(1 - \tilde{w}_m)$, for $l = 1, 2, \ldots, j$, with $\tilde{w}_0 = 1$. It can be easily verified that $\sum_{l=1}^{j} w_{jl} = 1$ for each $\mathbf{w}_j$, which is the prior probability that the data in group j will be drawn from the mixture distribution: $G_1, H_1, \ldots, H_{j-1}$. If all $\tilde{w}_j = 0$, all of the groups share the same mixture distribution G_1 and the model reduces to a Dirichlet mixture model, and if all $\tilde{w}_j = 1$ the model reduces to the HDP. Therefore, the dynamic HDP is more general than both DP and HDP, with each a special case. A visual representation of the model is depicted in Figure 1.

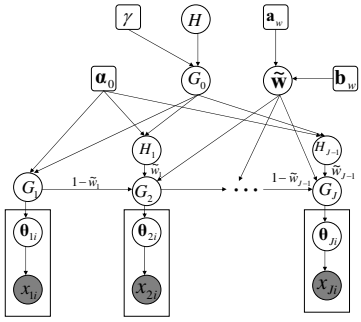


Figure 1. General graphical model for the dynamic HDP.

According to (9), the observation $x_{j,i}$ will choose a mixture distribution from $\boldsymbol{\pi}_{1:j}$ based on $Mult(\mathbf{w}_j)$ to be drawn from the global parameter components $\{\boldsymbol{\theta}_k^*\}_{k=1}^{\infty}$. We let $r_{j,i}$ be a variable to indicate which mixture distribution is taken from $\boldsymbol{\pi}_{1:j}$ to draw the observation $x_{j,i}$; $z_{j,i}$ is a parameter component indicator variable. An alternative form of the dHDP model is represented as:

$$\begin{array}{cc} \boldsymbol{\theta}_k^* | H \sim H, & \boldsymbol{\beta} | \gamma \sim Stick(\gamma) \\ \tilde{w}_j | a_{wj}, b_{wj} \sim Be(\tilde{w}_j | a_{wj}, b_{wj}), & r_{j,i} | \tilde{\mathbf{w}} \sim \mathbf{w}_j \\ \boldsymbol{\pi}_j | \alpha_{0j}, \boldsymbol{\beta} \sim DP(\alpha_{0j}, \boldsymbol{\beta}), & z_{j,i} | \boldsymbol{\pi}_{1:j}, r_{j,i} \sim \boldsymbol{\pi}_{r_{j,i}} \\ x_{j,i} | z_{j,i}, (\boldsymbol{\theta}_k^*)_{k=1}^{\infty} \sim F(\boldsymbol{\theta}_{z_{j,i}}^*), & \end{array} \tag{10}$$

and a graphical representation is shown in Figure 2, in which we add a gamma prior for γ and for the components of the vector $\boldsymbol{\alpha}_0$: $Pr(\gamma) = Ga(\gamma; \gamma_{01}, \gamma_{02})$ and $Pr(\boldsymbol{\alpha}_0) = \prod_{j=1}^{J} Ga(\alpha_{0j}; c_0, d_0)$. The form of the parametric model $F(\cdot)$ may be varied depending on the application.

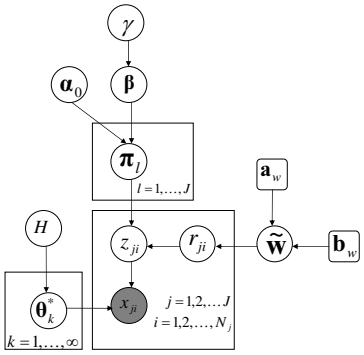


Figure 2. Graphical representation of the dHDP from a stick-breaking view.

2.3. Sharing Properties

To see the mixture structure in a discrete partition space $\mathcal{A} = (A_1, \ldots, A_K)$, we consider

$$\begin{aligned} &G_j(A_1,\ldots,A_K)|G_{j-1},\tilde{w}_{j-1} \sim \\ &(1-\tilde{w}_{j-1})G_{j-1}(A_1,\ldots,A_K)+\tilde{w}_{j-1}H_{j-1}(A_1,\ldots,A_K) \\ &\triangleq G_{j-1}(A_1,\ldots,A_K)+\triangle_j(A_1,\ldots,A_K) \end{aligned} \tag{11}$$

where $\triangle_j(A_1,\ldots,A_K) = \tilde{w}_{j-1}\{H_{j-1}(A_1,\ldots,A_K) - G_{j-1}(A_1,\ldots,A_K)\}$ is the random deviation from G_{j-1} to G_j.

Theorem 1. Given any discrete partition $\mathcal{A}$, we have:

$$\begin{aligned} &E\{\triangle_j(\mathcal{A})|G_{j-1},\tilde{w}_{j-1},H,\gamma,\alpha_{0j}\} \\ =&\tilde{w}_{j-1}\{H(\mathcal{A})-G_{j-1}(\mathcal{A})\} \end{aligned} \tag{12}$$

$$\begin{aligned} &V\{\triangle_j(\mathcal{A})|G_{j-1},\tilde{w}_{j-1},H,\gamma,\alpha_{0j}\} \\ =&\tilde{w}_{j-1}^2\frac{(1+\gamma+\alpha_{0j})H(\mathcal{A})(1-H(\mathcal{A}))}{(1+\alpha_{0j})(1+\gamma)} \end{aligned} \tag{13}$$

According to Theorem 1, given the previous mixture distribution G_{j-1}, the expectation of the deviation from G_{j-1} to G_j is controlled by $\tilde{w}_{j-1}$. Meanwhile, the variance of the deviation is both related with $\tilde{w}_{j-1}$ and the precision parameters γ, α_{0j}. To consider limiting cases, we observe the following:

- if $\tilde{w}_{j-1} \to 0$, $G_j = G_{j-1}$;
- if $G_{j-1} \to H$, $E(G_j(\mathcal{A})|G_{j-1},\tilde{w}_{j-1},H,\gamma,\alpha_{0j}) = G_{j-1}(\mathcal{A})$;
- if $\gamma \to \infty$ and $\alpha_{0j} \to \infty$, $V(\triangle_j(\mathcal{A})|G_{j-1},\tilde{w}_{j-1},H,\gamma,\alpha_{0j}) \to 0$.

These limiting cases yield insights on the underlying dependence between adjacent groups.

Theorem 2. The correlation coefficient of the distributions between two adjacent groups G_{j-1} and G_j for $j = 2,\ldots,J$ is

$$\begin{aligned} &Corr(G_{j-1},G_j) \\ =&\frac{E\{G_j(\mathcal{A})G_{j-1}(\mathcal{A})\}-E\{G_j(\mathcal{A})\}E\{G_{j-1}(\mathcal{A})\}}{[V\{G_j(\mathcal{A})\}V\{G_{j-1}(\mathcal{A})\}]^{1/2}} \\ =&\frac{\sum_{l=1}^{j-1}\frac{w_{jl}w_{j-1,l}}{1+\alpha_{0l}}\cdot\frac{\alpha_{0l}+\gamma+1}{\gamma+1}}{[\sum_{l=1}^{j}\frac{w_{jl}^2}{1+\alpha_{0l}}\cdot\frac{\alpha_{0l}+\gamma+1}{\gamma+1}]^{1/2}[\sum_{l=1}^{j-1}\frac{w_{j-1,l}^2}{1+\alpha_{0l}}\cdot\frac{\alpha_{0l}+\gamma+1}{\gamma+1}]^{1/2}} \end{aligned} \tag{14}$$

To compare the similarity of two data groups, the correlation coefficient defined in Theorem 2 can be calculated from the posterior expectation of $\mathbf{w}$, $\boldsymbol{\alpha}_0$ and γ as a local similarity measure.

2.4. Posterior Computation

A modification of the block Gibbs sampler (Ishwaran & James, 2001) is proposed for dHDP inference. Since in practice the $\{\pi_k\}_{k=1}^{\infty}$ in (1) diminish quickly with increasing k, a truncated stick-breaking process (Ishwaran & James, 2001) is employed here, with a large truncation level K, to approximate the infinite stick breaking process. In the dHDP, the second level of DPs associated with the dynamic structure is the only part different from HDP (see Figure 2). Due to the limited space, we only give the conditional posterior distributions for $\tilde{\mathbf{w}}$, $\tilde{\boldsymbol{\pi}}$, $\mathbf{r}$ and $\mathbf{z}$.

The conditional distribution of $\tilde{w}_l$, for $l = 1,\ldots,J-1$ has the simple form:

$$(\tilde{w}_l|\cdots) \sim Be(a_w + \sum_{j=l+1}^{J} n_{j,l+1}, b_w + \sum_{j=l+1}^{J}\sum_{h=1}^{l} n_{jh}) \tag{15}$$

where $n_{jh} = \sum_{i=1}^{N_j}\delta(r_{ji} = h)$. In (15) and in the results that follow, for simplicity, the distributions $Be(a_{wj}, b_{wj})$ are set with fixed parameters $a_{wj} = a_w$ and $b_{wj} = b_w$ for all time samples.

The conditional distribution of $\tilde{\pi}_{lk}$, for $l = 1,\ldots,J$ and $k = 1,\ldots,K$, is updated under the conjugate prior: $\tilde{\pi}_{lk} \sim Be(\alpha_{0,l}\beta_k, \alpha_{0,l}(1-\sum_{m=1}^{k}\beta_m))$, which is specified in (Teh et al., 2005). Then the conditional posterior of $\tilde{\pi}_{lk}$ has the form

$$\begin{aligned} &(\tilde{\pi}_{lk}|\cdots) \sim Be(\alpha_{0l}\beta_k + \sum_{j=1}^{J}\sum_{i=1}^{N_j}\delta(r_{ji}=l,z_{ji}=k), \\ &\alpha_{0l}(1-\sum_{l=1}^{k}\beta_l) + \sum_{j=1}^{J}\sum_{i=1}^{N_j}\sum_{k'=k+1}^{K}\delta(r_{ji}=l,z_{ji}=k')) \end{aligned} \tag{16}$$

The update of the indicator variables r_{ji} and z_{ji}, for $j = 1,\ldots,J$ and $i = 1,\ldots,N_j$ are completed by generating samples from multinomial distributions with

entries as follows:

$$Pr(r_{ji} = l|\cdots) \propto \tilde{w}_{l-1} \prod_{m=l}^{j-1}(1-\tilde{w}_m) \cdot \tilde{\pi}_{lz_{ji}} \prod_{q=1}^{z_{ji}-1}(1-\tilde{\pi}_{lq}) \cdot Pr(x_{ji}|\boldsymbol{\theta}^*_{z_{ji}}) \tag{17}$$

where $l = 1, \ldots, j$. The posterior probability $Pr(r_{ji} = l)$ is normalized so that $\sum_{l=1}^{j} Pr(r_{ji} = l) = 1$.

$$Pr(z_{ji} = k|\cdots) \propto \tilde{\pi}_{r_{ji}k} \prod_{k'=1}^{k-1}(1-\tilde{\pi}_{r_{ji}k'}) \cdot Pr(x_{ji}|\boldsymbol{\theta}^*_k) \tag{18}$$

where $k = 1, \ldots, K$ and the posterior is also normalized by a constant $\sum_{k=1}^{K} Pr(z_{ji} = k)$.

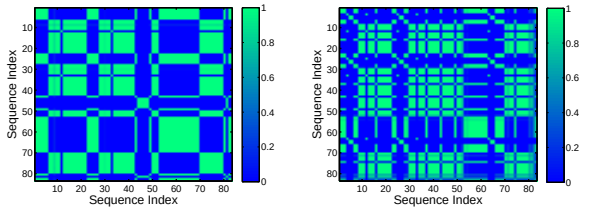
The remaining variables specified in (10) are sampled in the same ways as in HDP (Teh et al., 2005). The component parameters $\boldsymbol{\theta}^*_k$ for $k = 1, \ldots, K$ are considered for different model forms depending on the specific applications. For the results that follow, it is of interest to consider a hidden Markov model (HMM) mixture (Qi et al., 2007) and Gaussian mixture model (GMM), in which $\boldsymbol{\theta}^*_k$ respectively represent the state-transition matrix, the observation matrix, the initial-state distribution for the HMM and the mean vector and covariance matrix for GMM. For more details about sampling for such models, see (Qi et al., 2007) and (Escobar & West, 1995). The Gibbs sampling algorithm was tested carefully under different initializations and the diagnostic method in (Raftery & Lewis, 1992) is used to demonstrate rapid convergence and good mixing (for the results considered, convergence based on this method was observed for a burn-in of 200 samples, followed by a subsequent 4000 samples).

3. Experimental Results

3.1. Music Segmentation

It is of interest to segment music, to infer inter-relationships between different parts of a given piece, as well as between different pieces. Here we consider segmentation of music, where a given piece is divided into contiguous subsequences, with each subsequence modeled via a hidden Markov model (HMM). The dHDP model is useful in this application in enforcing the idea that contiguous subsequences are likely to be within the same music segment, and therefore are likely to share HMM parameters. However, when the segment changes, these changes are detected via innovation within the dHDP.

The music under consideration is the first movement "Largo - Allegro" from the Beethoven piano Sonata No. 17 (Newman, 1972). As is widely employed for analysis of such audio data, MFCC features are extracted and discretized with vector quatization (Qi et al., 2007); each of the aforementioned subsequences corresponds to a sequence of codewords (we here employ a discrete HMM). The basic form of the Bayesian representation of a discrete HMM is as discussed in (Qi et al., 2007). The piece is transformed into 4980 discrete symbols, divided into 83 subsequences of equal length (the codebook has 16 codes, and 8 states are employed for each HMM); each subsequence corresponds to 6 secs in the music. To model the time dependence between adjacent subsequences, each subsequence corresponds to one group in the dHDP HMM mixture and will choose one set of HMM parameters according to the corresponding mixture weights. In the dHDP framework, one subsequence can share the old DP mixture distributions with the previous ones or it might be drawn from an innovation DP mixture, which may be also shared by the following time series in a similar manner. To encourage that adjacent subsequences be shared, the prior for $\tilde{\mathbf{w}}$ is specified as $E(\tilde{\mathbf{w}}) < 0.5$. The product of most interest here is the segmentation of the music, with the specific HMM parameters of secondary importance.



(a) (b)

Figure 3. Similarity matrix $E(z'z)$ from HMM mixture modeling of the Sonata. (a) dHDP-HMM, (b) HDP-HMMs.

To represent the time dependence of the piece, the similarity measure $E(\mathbf{z}'\mathbf{z})$ (see $\mathbf{z}$ in Eq. (18)) is computed across each pair of subsequences, as shown in Figure 3, in which larger values represent higher probability of the two corresponding subsequences being shared during parameter inference. Based upon a discussion in (Newman, 1972), the movement alternates seeming peacefulness with sudden turmoil (1st-6th subsequences), after some time expanding into a haunting "storm" in which the peacefulness is lost (7th-21st subsequences). After the recurrence of the same pattern (22nd-42nd subsequences) and a small transition, the movement starts a long recitative section in a slow tone (53rd-69th subsequences). Then through the crescendo, previous disturbed tones come back again until the music goes to the peaceful epilogue

(after the 70th subsequence). See (Newman, 1972) for more details on the Sonata. This is deemed to be an interesting piece for study because it is well characterized in the music literature, as briefly summarized above, and because it is anticipated to have repeated segments over the length of the piece. In Figures 3(a) and (b) we compare the dHDP and HDP, respectively, the latter computed by fixing all $\tilde{\boldsymbol{w}} = 1$ in the dHDP model. The dHDP and HDP yield related results, but the former yields a smoother segmentation, in good agreement with the music theory discussed above.

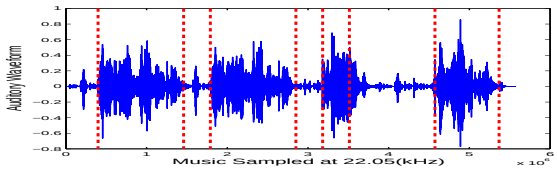


Figure 4. Segmentation of the Beethoven piano music from the dHDP HMMs (red dash lines represent segment positions and blue curves represent the auditory waveform).

Based on the results from the dHDP HMM, which effectively yields a model with smoothly time-evolving statistics, we segment the music and present the associated auditory waveform in Figure 4. By examining the waveform and the results in Figure 3, we note that the dHDP segments the music into dominant auditory phenomena, but it is less sensitive to noticeable but temporally localized events in the music, yielding a segmentation that is consistent with the music theory. By contrast, the HDP results in Figure 3(b) are evidently more sensitive to these local temporal bursts in the waveform.

3.2. Gene Expression Data

As a second example, we consider the time-evolving characteristics of gene-expression data, here for a Dengue virus study (Hibberd et al., 2006). Concerning a model for the gene-expression data at one time snapshot, Dunson (Dunson, 2006) proposed a latent response model based on a linear regression structure; we extend this model for time-evolving gene-expression data via dHDP (with comparison as well to HDP).

Assume $\mathbf{y}_{ji}$ is a feature vector with dimension p for $j = 1, \ldots, J$ and $i = 1, \ldots, N_j$ (index j corresponds to time, i represents a particular cell from which a sample is collected, and p denotes the number of genes being modeled). Each $\mathbf{y}_{ji}$ is represented as

$$\mathbf{y}_{ji} = \boldsymbol{\mu} + \boldsymbol{\lambda}\eta_{ji} + \boldsymbol{\varepsilon}_{ji} \tag{19}$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)'$ is the intercept vector and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_p)'$ represents factor loadings. We define a hidden variable η_{ji} underlying the observation $\mathbf{y}_{ji}$ to be associated with the i^{th} sample at time t_j. The error term $\boldsymbol{\varepsilon}_{ji}$ is also a vector of dimension p and each coefficient $\varepsilon_{ji,d}$ is independently drawn from a Student-t distribution. To eliminate the problem of model identifiability, we incorporate the constraints that $\mu_1 = 0$ and $\lambda_1 = 1$, as (Dunson, 2006) discusses. In the present model, one cannot explicitly associate $\boldsymbol{\eta}$ exclusively with the virus; however, since these are cell data, it is anticipated that the virus represents the dominant phenomena.

We have access to expressions of thousands of genes from each sample (cell) for multiple consecutive times $t_1, t_2, \ldots, t_J$. For each time t_j, there are N_j samples measured from different cells (Hibberd et al., 2006). Although these samples may have different observations in gene expressions at the same time, due to individual diversity, the hidden variable $\boldsymbol{\eta}$ (see (19)) underlying the observations may have similar characteristics. Based on this consideration, the $\boldsymbol{\eta}$ underlying the observations in one group corresponding to one time are assumed to be drawn from a Gaussian mixture model. They may also share the same mixture distribution for proximate time points, under the assumption of the dHDP model.

The Dengue gene expression data (Hibberd et al., 2006) are divided into six groups of samples measured at different times and the number of samples in each group are 10, 12, 12, 10, 12, 9 (the specific time points associated with these data are respectively 3, 6, 12, 24, 48 and 72 hours); each sample has 19,143 genes. To deal with such high-dimensional data, the Fisher score (Duda & Hart, 1973) is used to preliminarily select $p = 5000$ genes as being the most relevant (variable across time and cell), and then we use the dHDP mixture model discussed above to analyze the time evolution existing in these gene samples.

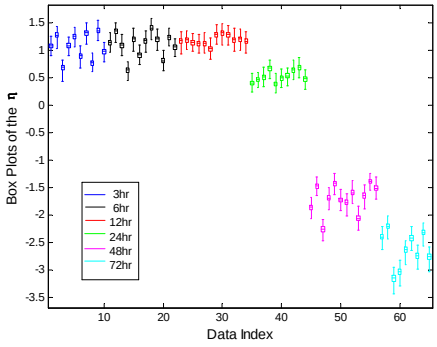


Figure 5. Median values and associated uncertainty based on posterior distributions of the hidden variables $\boldsymbol{\eta}$.

Based on the samples collected from the Gibbs sampling after burn-in, the posterior distributions (includ-

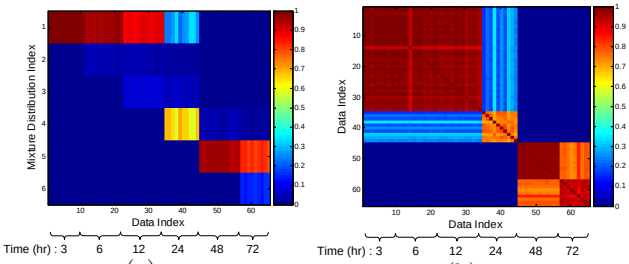


Figure 6. The dHDP GMM modeling for the gene expression data. (a) The posterior distribution of $\mathbf{r}$. (b) The similarity matrix $E[z'z]$.

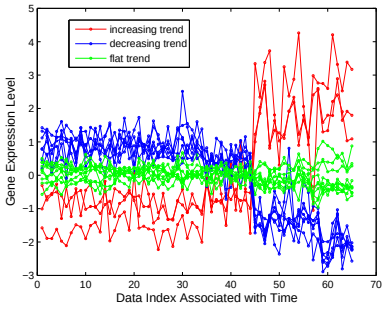


Figure 7. The first ten inferred important genes (color red and blue) and the relatively unrelated genes (color green).

ing the minimum, median, maximum, 25th and 75th percentiles of the values) for all components of $\boldsymbol{\eta}$ underlying these samples at different times are shown in Figure 5. Time points 3hr, 6hr and 12hr appear to share a similar pattern, but the $\boldsymbol{\eta}_{t=12}$ seem to have smaller diversity among different samples. From 24hrs, $\boldsymbol{\eta}$ drops slightly to a new pattern and they drop significantly again at 48hr. The posterior of indicator $\mathbf{r}$ is plotted in Figure 6(a) to show the mixture-distribution sharing relationship across different groups. Figure 6(b) shows the similarity measure $E(\mathbf{z}'\mathbf{z})$ across every pair of samples; here z_{ji} is the indicator variable for the η_{ji} associated with time t_j (see Eq. (18)).

Consider the factor loadings vector $\boldsymbol{\lambda}$, which has components linked to the p genes under consideration. The larger the value of $|\lambda_d|$, the more influence the pattern contained in $\boldsymbol{\eta}$ has on the corresponding gene at the d^{th} dimension. Therefore, according to the posterior mean of $|\lambda_d|$ for all d from the Gibbs sampling we rank the genes based on their importance.

In Figure 7 we plot the expression levels over time for the 10 most important and 10 least important genes. The red and blue curves show two different time patterns and their values have either an increasing or a decreasing trend with time, depending on whether the associated λ is positive or negative. The green curves represent the genes with no apparent relation to the virus (as determined by the analysis) due to the lack of a systematic trend over time.

As discussed in Section 2.2, if all $\tilde{w}_j$ are set to one for $j = 1, \ldots, J-1$, the dHDP reduces to HDP and all the temporal groups are conditionally exchangeable. It is of interest to compare the dHDP with HDP both in the sharing mechanism and parameter estimation. In practice, acquisition of the gene-expression data is expensive, and it is desirable to reduce the number of samples required. To consider this issue, we reduced the samples size to four at each time point, and plot the data similarity matrix $E[\boldsymbol{z}'\boldsymbol{z}]$ for HDP and dHDP respectively in Figures 8(a) and (b).

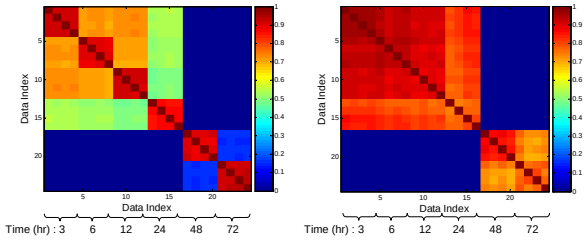


Figure 8. Similarity matrix $E[z'z]$ with four samples for each temporal group. (a) HDP, (b) dHDP.

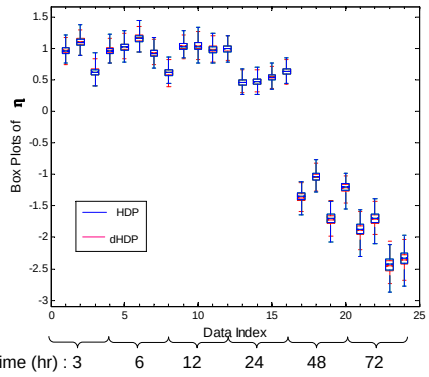
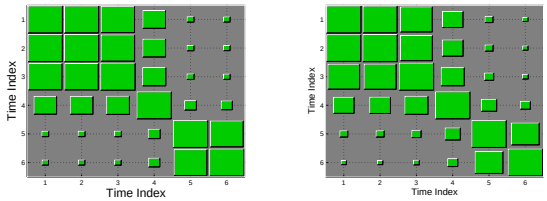


Figure 9. Comparison of dHDP and HDP with box plots of the hidden variables $\boldsymbol{\eta}$ as the sample size is reduced to four for each temporal group (the standard deviation based on dHDP is 12.1% reduced on average relative to HDP; the means are very similar).

Compared with HDP, dHDP has more sharing between the related groups (as expected from model construction), and despite the reduced data samples the dHDP yields an inter-relationship between the different times that is consistent with that in Figure 6(b) which employs all of the available data. In Figure 9 we compare dHDP and HDP estimation of $\boldsymbol{\eta}$ based on four samples per time point. These results show that dHDP has a smaller estimation uncertainty for most $\boldsymbol{\eta}$ relative to HDP, which is attributed to proper temporal

sharing explicitly imposed by dHDP. As the sample size is increased, the differences between dHDP and HDP diminish.

Finally, correlation coefficients between two groups are calculated from the samples drawn from the Gibbs sampler, according to (14) and plotted as a matrix in Figure 10; this representation is an additional benefit of the dynamic structure explicitly imposed within dHDP (of potential biological interest). The size of each small block at the i^{th} row and j^{th} column is proportional to the value of the correlation coefficient associated with group i and group j. We note based on Figure 10 that such inference appears to be accurate (or at least consistent) even with diminished sample size.



(a) (b)

Figure 10. Similarity matrix between data at different time points based on the correlation coefficients (14), as computed from the dHDP posterior. (a) using all available data, (b) using four samples for each temporal group.

4. Conclusions

The proposed dynamic hierarchical Dirichlet process (dHDP) extends the HDP (Teh et al., 2005), imposing a dynamic time dependence so that the initial mixture model and the subsequent time-dependent mixtures share the same set of components (atoms). The experiments indicate that the dHDP is an effective model for analysis of time-evolving data. Concerning future research, more efficient inference methods will be considered, such as collapsed sampling (Welling et al., 2007) and variational Bayesian inference (Blei & Jordan, 2004).

References

Blackwell, D., & MacQueen, J. B. (1973). Ferguson distributions via polya urn schemes. *Ann. Statist.*, *1*, 353–355.

Blei, D. M., & Jordan, M. I. (2004). Variational methods for the dirichlet process. *Proceedings of the International Conference on Machine Learning.*

Blei, D. M., & Lafferty, J. D. (2006). Dynamic topic models. *Proceedings of the International Conference on Machine Learning.*

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Caron, F., Davy, M., & Doucet, A. (2007). Generalized poly urn for time-varying dirichlet process mixtures. *Proceedings of the International Conference on Uncertainty in Artificial Intelligence(UAI).*

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* Wiley.

Dunson, D. B. (2006). Bayesian dynamic modeling of latent trait distributions. *Biostatistics*, *7*, 551–568.

Escobar, M. D., & West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, *90*, 577–588.

Griffin, J. E., & Steel, M. F. J. (2006). Order-based dependent dirichlet processes. *Journal of the American Statistical Association*, *101*, 179–194.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proc Natl Acad Sci U S A*, *101, Suppl 1*, 5228–5235.

Hibberd, M. L., Vasudevan, S. G., Ling, L., & George, J. (2006). *Time course expression data of human cell lines infected with dengue virus serotype2 ngc* (Technical Report). Genome Institute of Singapore.

Ishwaran, H., & James, L. F. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, *96*, 161–173.

Newman, A. S. (1972). *Sonata in the classic era (a history of the sonata idea).* W. W. Norton.

Qi, Y., Paisley, J. W., & Carin, L. (2007). Music analysis using hidden markov mixture models. *IEEE Transactions on Signal Processing*, *55*, 5209–5224.

Raftery, A. E., & Lewis, S. (1992). How many iterations in the gibbs sampler? *Bayesian Stat.*, *4*, 763–773.

Sethuraman, J. (1994). A constructive definition of dirichlet priors. *Statistica Sinica*, *2*, 639–650.

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2005). *Hierarchical dirichlet processes* (Technical Report). Dept. of Computer Science, National University of Singapore.

Welling, M., Porteous, I., & Bart, E. (2007). Infinite state bayes-nets for structured domains. *Proceedings of the International Conference on Neural Information Processing Systems.*

Closed-Form Supervised Dimensionality Reduction with Generalized Linear Models

Irina Rish, Genady Grabarnik, Guillermo Cecchi {RISH,GENADY,GCECCHI}@US.IBM.COM
IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA

Francisco Pereira FPEREIRA@PRINCETON.EDU
CSBMB, Green Hall, Princeton University, Princeton, NJ 08540 USA

Geoffrey J. Gordon GGORDON@CS.CMU.EDU
Carnegie Mellon University, School of Computer Science, Machine Learning Dept., Pittsburgh, PA 15213 USA

Abstract

We propose a family of supervised dimensionality reduction (SDR) algorithms that combine feature extraction (dimensionality reduction) with learning a predictive model in a unified optimization framework, using data- and class-appropriate generalized linear models (GLMs), and handling both classification and regression problems. Our approach uses simple closed-form update rules and is provably convergent. Promising empirical results are demonstrated on a variety of high-dimensional datasets.

1. Introduction

Dimensionality reduction (DR) is a popular data-processing technique that serves the following two main purposes: it helps to provide a meaningful interpretation and visualization of the data, and it also helps to prevent overfitting when the number of dimensions greatly exceeds the number of samples, thus working as a form of regularization.

When our goal is prediction rather than an (unsupervised) exploratory data analysis, *supervised* dimensionality reduction (SDR) that combines DR with *simultaneously* learning a predictor can significantly outperform a simple combination of unsupervised DR with a subsequent learning of a predictor on the resulting low-dimensional representation (Pereira & Gordon, 2006; Sajama and Alon Orlitsky, 2005). The problem of supervised dimensionality reduction can be viewed as finding a *predictive structure*, such as a low-dimensional representation, which captures the information about the class label contained in the high-dimensional feature vector while ignoring the "noise".

However, existing SDR approaches are often restricted to specific settings, and can be viewed as jointly learning a *particular mapping* (most commonly, a linear one) from the feature space to a low-dimensional hidden-variable space, together with a *particular predictor* that maps the hidden variables to the class label. For example, SVDM (Pereira & Gordon, 2006) learns a linear mapping from observed to hidden variables, effectively assuming Gaussian features when minimizing sum-squared reconstruction loss; on the prediction side, it focuses on SVM-like binary classification using hinge loss. SDR-MM method of (Sajama and Alon Orlitsky, 2005) treats various types of features (e.g., binary and real-valued) but is limited to discrete classification problems, i.e. is not suitable for regression. Recent work on distance metric learning (Weinberger et al., 2005; Weinberger & Tesauro, 2007) treats both classification and regression settings, but is limited, like SVDM, to Gaussian features and linear mappings when learning Mahalanobis distances.

This paper approaches SDR in a more general framework that views both features and class labels as exponential-family random variables, and allows to mix-and-match data- and label-appropriate *generalized linear models*, thus handling both classification and regression, with both discrete and real-valued data[1]. It can be viewed as a discriminative learn-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] In other words, the proposed framework provides nonlinear dimensionality reduction methods based on minimizing Bregman divergences that correspond to particular

ing based on minimization of conditional probability of class given the hidden variables, while using as a regularizer the conditional probability of the features given the low-dimensional hidden-variable "predictive" representation.

The main advantage of our approach, besides being more general, is using simple *closed-form* update rules when performing its alternate minimization procedure. This method yields a short Matlab code, fast performance, and is guaranteed to converge. The convergence property, as well as closed form update rules, result from using appropriate auxiliary functions bounding each part of the objective function (i.e., reconstruction and prediction losses). We exploit the additive property of auxiliary functions in order to combine bounds on multiple loss functions.

We perform a variety of experiments, both on simulated and real-life problems. Results on simulated datasets convincingly demonstrate that our SDR approach can discover underlying low-dimensional structure in high-dimensional noisy data, while outperforming SVM and SVDM, often by far, and practically always beating the unsupervised DR followed by learning a predictor. On real-life datasets, SDR approaches continue to beat the unsupervised DR by far, while often matching or somewhat outperforming SVM and SVDM.

2. SDR-GLM: Hidden-Variable Model

Let X be an $N \times D$ data matrix with entries denoted X_{nd} where N is the number of i.i.d. samples, and n-th sample is a D-dimensional row vector denoted $\mathbf{x}_n$. Let Y be an $N \times K$ matrix of class labels for K separate prediction problems (generally, we will consider $K > 1$), where j-th column, $1 \leq j \leq K$, provides a set of class labels for the j-th prediction problem. Our supervised dimensionality approach relies on the assumption that each data point $\mathbf{x}_n$, $n = 1, ..., N$, is a noisy version of some "true" data point $\theta_\mathbf{n}$ which lives in a low-dimensional space, and that this hidden representation of the noisy data is actually predictive about the class label.

We will also assume, following ePCA (Collins et al., 2001), $(GL)^2M$(Gordon, 2002), logistic PCA (Schein et al., 2003) and related extensions of PCA, that noise in the features follows exponential-family distributions with natural parameters $\theta_\mathbf{n}$, with different members of the exponential family used for different dimensions[2], and that the noise is applied independently to each coordinate of $\mathbf{x}_n$. Namely, it is assumed that $N \times D$ parameter matrix Θ can be represented by a linear model in an L-dimensional ($L < D$) space:

$$\Theta_{nd} = \sum_{l=1}^{L} U_{nl} V_{ld} + \Delta_{X_d},$$

where the rows of the $L \times D$ matrix V correspond to the basis vectors, the columns of the $N \times L$ matrix U correspond to the coordinates of the "true points" θ_n, $n = 1, ...N$ in the L-dimensional space spanned by those basis vectors, and Δ_X is the bias vector (corresponding to the empirical mean in PCA). However, to simplify the notation, we will include Δ_X as the $(L+1)$'s row of the matrix V (i.e., $V_{L+1} = \Delta_X$), and add the $(L+1)$'s column of 1's to U, so that we can write Θ_X as a product of two matrices, $\Theta_{X_{nd}} = (UV)_{nd} = \sum_{l=1}^{L+1} U_{nl} V_{ld}$.

Given the natural parameter Θ_{nd}, an exponential-family noise distribution is defined for each X_{nd} by

$$\log P(X_{nd}|\Theta_{X_{nd}}) = X_{nd}\Theta_{X_{nd}} - G(\Theta_{X_{nd}}) + \log P_0(X_{nd}),$$

where $G_x(\Theta_{nd})$ is the *cumulant* or *log-partition* function that ensures that $P(X_{nd}|\Theta_{nd})$ sums (or integrates) to 1 over the domain of X_{nd}. This function uniquely defines a particular member of the exponential family, e.g., Gaussian, multinomial, Poisson, etc.

We can now view each row U_n as a "compressed" representation of the corresponding data sample $\mathbf{x}_n$ that will be used to predict the class labels. We will again assume a noisy linear model for each class label Y_k (column-vector in Y) where the natural parameter is represented by linear combination

$$\Theta_{Y_{nk}} = \sum_{l=1}^{L} U_{nl} W_{lk} + \Delta_{Y_k}$$

with linear coefficients $\mathbf{w} = (w_1, ..., w_L)$ and K-dimensional bias vector Δ_Y. As for $\Theta_{X_{nd}}$, we will simplify the notation by including Δ_Y as the $(L+1)$'s row of the matrix W, and write $\Theta_{Y_{nk}} = (UW)_{nk} = \sum_{l=1}^{L+1} U_{nl} W_{lk}$. Using an appropriate type of exponential-family noise $P(Y_{nk}|\Theta_{Y_{nk}})$, we can handle both classification and regression problems. For

members of the exponential family of distribution, including linear (PCA-like) dimensionality reduction as a particular case for Gaussian variables.

[2] It is important to note that in case of standard (linear) PCA corresponding to Gaussian noise the parameters θ_n live in a linear subspace in the original data space, but for other types of exponential-family noise the low-dimensional space of parameters is typically a nonlinear surface in the original data space.

example, in case of binary classification, we can model Y_{nk} as a Bernoulli variable with parameter p_{nk} and the corresponding natural parameter $\Theta_{nk} = \log(\frac{p_{nk}}{1-p_{nk}})$, and use logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ to write the Bernoulli distribution for Y_{nk} as

$$P(Y_{nk}|\Theta_{Y_{nk}}) = \sigma(\Theta_{Y_{nk}})^{Y_{nk}}\sigma(-\Theta_{Y_{nk}})^{1-Y_{nk}}.$$

In case of regression, Y_{nk} will be a Gaussian variable with the expectation parameter coinciding with the natural parameter $\Theta_{Y_{nk}}$.

In other words, we will use a *generalized linear model (GLM)* $E(X_d) = f_d^{-1}(UV_d)$ for each feature X_d (d-th column in X, $1 \le d \le D$), and yet another set of GLMs $E(Y_k) = f_k^{-1}(UW)$ for each class label Y_k (k-th column in Y, $1 \le k \le K$), with possibly different *link functions* f_d and f_k (e.g., the logit link function $f(\mu) = \ln\frac{\mu}{1-\mu} = \sigma^{-1}(\theta)$ is used for binary classification, and identity link function $f(\mu) = \mu$ is used for real-valued regression with Gaussian output).

SDR-GLM optimization problem. We formulate SDR as joint optimization of two objective functions corresponding to the reconstruction accuracy (data likelihood) and the prediction accuracy (class likelihood):

$$\mathcal{L}_X(\Theta_X) = \sum_{nd} \log P(X_{nd}|\Theta_{X_{nd}}), \quad (1)$$

$$\mathcal{L}_Y(\Theta_Y) = \sum_{nk} \log P(Y_{nk}|\Theta_{Y_{nk}}), \quad (2)$$

where $\Theta_X = UV$, $\Theta_Y = UW$, and the likelihoods above correspond to particular members of exponential family. Then the SDR optimization problem is

$$\max_{U,V,W} \alpha\mathcal{L}_X(UV) + \mathcal{L}_Y(UW) \quad (3)$$

where the data likelihood can be viewed as a regularization added on top of the class likelihood maximization objective, with the regularization constant α controlling the trade-off between the two likelihoods [3].

Comparison with SVDM. Note that the idea of combining loss functions for SDR was also proposed before in SVDM (Pereira & Gordon, 2006), where, similarly to SVD, quadratic loss $||X - UV||_2^2$ was used for data reconstruction part of the objective, while the hinge loss was used for the prediction part (using U as the new data matrix). Herein, we generalize SVDM's sum-squared reconstruction loss to log-likelihoods of exponential family, similarly to ePCA(Collins et al., 2001) and G^2L^2M(Gordon, 2002), replace hinge loss by the loglikelihoods corresponding to exponential-family class variables, and provide closed-form update rules rather than perform optimization at each iteration, which results into a significant speed up when compared with SVDM.

[3] In our implementation, an L_2-norm regularization was also added on all matrices U, V, W with small regularization constants (0.01), effectively corresponding to assuming Gaussian priors on those matrices.

3. Combining Auxiliary Functions

Since the above problem (eq. 3) is not jointly convex in all parameters, finding a globally optimal solution is nontrivial. Instead, we can employ the auxiliary function approach commonly used in EM-style algorithms, and using auxiliary function of a particular form, derive *closed-form* iterative update rules that are guaranteed to converge to a local minimum. We show that an auxiliary function for the objective in eq. 3 can be easily derived for an arbitrary pair of $\mathcal{L}_X$ and $\mathcal{L}_Y$ provided that we know their corresponding auxiliary functions.

Auxiliary functions. Given a function $\mathcal{L}(\theta)$ to be maximized, a function $Q(\hat{\theta}, \theta)$ is called an *auxiliary function* for $\mathcal{L}(\theta)$ if $\mathcal{L}(\theta) = Q(\theta, \theta)$ and $\mathcal{L}(\hat{\theta}) \ge Q(\hat{\theta}, \theta)$ for all $\hat{\theta}$. It is easy to see that $\mathcal{L}(\theta)$ is non-decreasing under the update

$$\theta^{t+1} = \arg\max_{\hat{\theta}} Q(\hat{\theta}, \theta^t),$$

i.e., $\mathcal{L}(\theta^{t+1}) \ge \mathcal{L}(\theta^t)$, and thus an iterative application of such updates is guaranteed to converge to a local maximum of $\mathcal{L}$ under mild conditions on $\mathcal{L}$ and Q.

We will make use of the following properties of auxiliary functions:

Lemma 1 *Let $Q_1(\hat{\theta}, \theta)$ and $Q_2(\hat{\theta}, \theta)$ be auxiliary functions for $\mathcal{L}_1(\theta)$ and $\mathcal{L}_2(\theta)$, respectively. Then*

$$Q(\hat{\theta}, \theta) = \alpha_1 Q_1(\hat{\theta}, \theta) + \alpha_2 Q_2(\hat{\theta}, \theta) \quad (4)$$

is an auxiliary function for $\mathcal{L}(\theta) = \alpha_1\mathcal{L}_1(\theta) + \alpha_2\mathcal{L}_2(\theta)$, where $\alpha_i > 0$ for $i = 1, 2$.

Proof. $Q(\hat{\theta}, \theta) = \alpha_1 Q_1(\hat{\theta}, \theta) + \alpha_2 Q_2(\hat{\theta}, \theta) \le \alpha_1\mathcal{L}_1(\hat{\theta}) + \alpha_2\mathcal{L}_2(\hat{\theta}) = \mathcal{L}(\hat{\theta})$, and $Q(\theta, \theta) = \alpha_1 Q_1(\theta, \theta) + \alpha_2 Q_2(\theta, \theta) = \alpha_1\mathcal{L}_1(\theta) + \alpha_2\mathcal{L}_2(\theta) = \mathcal{L}(\theta)$. ■

Also, it is obvious that a function is an auxiliary for itself, i.e. $Q(\hat{\theta}, \theta) = \mathcal{L}(\hat{\theta})$ is an auxiliary function for $\mathcal{L}(\theta)$. This observations allows us to combine various dimensionality reduction approaches with appropriate predictive loss functions, given appropriate auxiliary functions for both (next section discusses two such combinations). When optimization of an auxiliary functions yields an analytical solution (e.g., for

quadratic functions), it is easy to obtain closed-form update rules for alternating minimization.

4. Iterative Update Rules

Recall that natural parameters for X_{nd} and Y_{nk} variables are represented by $\Theta_{X_{nd}} = \sum_{l=1}^{L+1} U_{nl}V_{ld}$ and $\Theta_{Y_{nk}} = \sum_{l=1}^{L+1} U_{nl}W_{lk}$. Let $Q_X(\hat{\Theta}_X, \Theta_X)$ and $Q_Y(\hat{\Theta}_Y, \Theta_Y)$ be the auxiliary functions for the corresponding loglikelihoods in eq. 2, then $Q(\hat{\Theta}_X, \Theta_X, \hat{\Theta}_Y, \Theta_Y) =$

$$= \alpha Q_X(\hat{\Theta}_X, \Theta_X) + Q_Y(\hat{\Theta}_Y, \Theta_Y) \quad (5)$$

is an auxiliary function for the combined log-likelihood in eq. 3 when we fix two out of three parameters and optimize over the remaining one. The update rules for $\hat{U}_{nl}$, $\hat{V}_{ld}$ and $\hat{W}_{nk}$ are obtained by solving

$$\frac{\partial Q}{\partial \hat{U}_{nl}} = 0,\ \frac{\partial Q_X}{\partial \hat{V}_{ld}} = 0,\ \frac{\partial Q_Y}{\partial \hat{W}_{nk}} = 0,\ where$$

$$\frac{\partial Q}{\partial \hat{U}_{nl}} = \alpha \sum_d \frac{\partial Q}{\partial \hat{\Theta}_{X_{nd}}} \frac{\hat{\Theta}_{X_{nd}}}{\partial \hat{U}_{nl}} + \sum_k \frac{\partial Q}{\partial \hat{\Theta}_{Y_{nk}}} \frac{\hat{\Theta}_{Y_{nk}}}{\partial \hat{U}_{nl}}.$$

Note that we get simpler expressions for V and W since they appear only in Q_X or only in Q_Y parts of eq. 5, respectively.

Auxiliary functions for Bernoulli and Gaussian log-likelihoods. For a Bernoulli variable s with natural (log odds) parameter θ we use the variational bound on log-likelihood $\mathcal{L}(\theta) = \log P(s|\theta)$ proposed by (Jaakkola & Jordan, 1997) and used later by (Schein et al., 2003)

$$\mathcal{L}(\hat{\theta}) \geq Q(\hat{\theta}, \theta) = \log 2 - \log\cosh(\theta/2) + + \frac{T\theta^2}{4} + \frac{(2s-1)\hat{\theta}}{2} - \frac{T\hat{\theta}^2}{4}, \quad (6)$$

where $T = \frac{tanh(\theta/2)}{\theta}$. Note that the auxiliary function is quadratic in θ and taking its derivatives leads to simple closed-form iterative update rules for U, V and W. For multinomial variables, one can use a recent extension of the above bound to multinomial logistic regression proposed by (Bouchard, 2007).

For a Gaussian variable s with natural parameter θ that coincides with the mean parameter (identity link function) we do not really have to construct an auxiliary function, since we can simply use the negative squared loss proportional to the Gaussian loglikelihood as an auxiliary function itself, i.e.

$$\mathcal{L}(\hat{\theta}) = Q(\hat{\theta}, \theta) = -c(s-\theta)^2, \quad (7)$$

where $c = (2\sigma)^{-1}$ is a constant, assuming a fixed standard deviation that will not be a part of our estimation procedure here, similarly to the approach of (Collins et al., 2001) and related work; c can be ignored since it will be subsumed by the parameter α.

Using the above auxiliary functions, we can combine them into joint auxiliary functions as in eq. 5 for various combinations of Bernoulli and Gaussian variables X_{nd} and Y_{nk}. Namely, assuming all X_{nd} are Bernoulli, we get (Schein et al., 2003):

$$Q_X^{Ber}(\hat{\Theta}_{X_{nd}}, \Theta_{X_{nd}}) = \sum_{nd} \log\cosh(\Theta_{X_{nd}}/2) + + \frac{T\Theta_{X_{nd}}^2}{4} + \frac{(2X_{nd}-1)\hat{\Theta}_{X_{nd}}}{2} - \frac{T\hat{\Theta}_{X_{nd}}^2}{4}, \quad (8)$$

while assuming all X_{nd} are Gaussian, we get

$$Q_X^{Gauss}(\hat{\Theta}_{X_{nd}}, \Theta_{X_{nd}}) = -\sum_{nd}(X_{nd} - \hat{\Theta}_{X_{nd}})^2. \quad (9)$$

Note that $Q_Y(\hat{\Theta}_{Y_{nk}}, \Theta_{Y_{nk}})$ for all-Bernoulli or all-Gaussian Y_{nk} is obtained by replacing X with Y, V with W, and d with k in eq. 8 and 9, respectively.

Due to lack of space, we omit the derivation of the iterative update rules for the four versions of SDR-GLM that we experimented with (for more detail, see (Rish et al., 2008)): *Gaussian-SDR*, that assumes Gaussian X_{nd} and Bernoulli Y_{nk}, *Bernoulli-SDR* in case of Bernoulli X_{nd} and Bernoulli Y_{nk}, *Gaussian-SDR-Regression* and *Bernoulli-SDR-Regression* in case of Gaussian Y_{nk} (appropriate for real-valued label, i.e. for the regression problem).

Prediction step. Once we learn the parameters U, V and W, and thus Θ_X and Θ_Y, we can use them on test data in order to predict labels for previously unseen data. Given the test data X^{test}, we only need to find the corresponding low-dimensional representation U^{test} by performing the corresponding iterative updates only with respect to U, keeping V and W fixed. Once U^{test} is found, we predict the class labels as $Y^{test} = U^{test}W$.

5. Empirical Evaluation

We evaluated our SDR-GLM methods on both simulated and real-life datasets, in both classification and regression settings. We varied the low-dimensionality parameter L from 2 to 10, and evaluated a range of regularization parameters $\alpha = 0.0001, 0.001, 0.01, 0.1, 1, 10, 100$, selecting those giving the best average error among several dimensions[4].

[4]Selecting the best α separately for each dimension can only improve the results, but would be more computationally intensive.

5.1. Classification Problems

In the classification setting, we compared *Bernoulli-SDR* and *Gaussian-SDR* versus linear SVM[5] and versus unsupervised dimensionality reduction followed by SVM and by logistic regression, which we refer to as SVM-UDR and *Logistic-UDR*, respectively. In both cases, unsupervised dimensionality reduction is performed first using the data-appropriate DR method (i.e., PCA for real-valued data and Logistic-PCA for binary data; this is equivalent to removing the prediction loss in eq. 3); then SVM or logistic regression are performed on the low-dimensional data representation U. For datasets with real-valued features, we also compared the above methods to SVDM (Pereira & Gordon, 2006). We performed k-fold cross-validation with $k = 10$ on the datasets with less than 1000 dimensions, and with $k = 5$ otherwise.

Simulated data. In order to test our methods first on the data with controllable low-dimensional structure, we simulated high-dimensional datasets that indeed were noisy versions of some underlying easily-separable low-dimensional data. Particularly, a set of $N = 100$ samples from two classes was generated randomly in two-dimensional space so that the samples were linearly separable with a large margin.

Next, we simulated two sets of exponential-family random variables X_{nd}, a Bernoulli set and a Gaussian set, using the coordinates of the above points for natural parameters Θ_{nd}, where the number of samples was $N = 100$ and the dimensionality of a "noisy" dataset varied from $D = 100$ to $D = 1000$. We then combined the data with the labels generated in the low-dimensional space and provided them as an input to our algorithms.

Simulated data: Bernoulli noise. Figures 1a summarize the results for Bernoulli-noise dataset. Supervised DR methods (Bernoulli-SDR and Gaussian-SDR) versus SVM and versus unsupervised DR followed by learning a predictor (Logistic-UDR and SVM-UDR); the reduced dimensionality parameters is set to $L = 2$ and the regularization constant is $\alpha = 0.0001$ (the choice of those parameters is discussed below). We can see that Bernoulli-SDR significantly outperforms all other methods, including SVM, and both Bernoulli-SDR and Gaussian-SDR also outperform the unsupervised DR followed by either logistic regression or SVM. Apparently, Bernoulli-SDR is able to reconstruct correctly the underlying separable two-dimensional dataset and is robust to noise, as its error remains zero for up to 700 dimensions, and only increases slightly up to 0.05 for 1000 dimensions. On the other hand, SVM has zero-error prediction only in the lowest-dimensional case ($D = 100$), and is much more sensitive to noise when the dimensionality of the data increases, making incorrect predictions in up to 14% to 21% cases when the dimensionality increases above $D = 300$. Apparently, SVM was not able to exploit the underlying separable low-dimensional structure disturbed by high-dimensional noise, while supervised dimensionality reduction easily detected this structure.

Also, using the Bernoulli model instead of Gaussian when features are binary is clearly beneficial, and thus, as noted previously, proper extensions of PCA to exponential-family data must be used (Collins et al., 2001; Schein et al., 2003). However, previous work on logistic PCA by (Schein et al., 2003) demonstrated advantages of using the Bernoulli vs Gaussian assumption only for *reconstruction* of the original data, while this paper investigates the impact of such assumptions on the *generalization* error in supervised learning case. This is less obvious, since a good fit to the training data does not necessarily imply a good generalization ability, as shown by our experiments with unsupervised dimensionality reduction followed by learning a classifier.

Regarding the choice of the regularization parameter α, we experimented with a range of values from 0.0001 to 10, and concluded that the smallest value was the most beneficial for both SDR algorithms; this is intuitive since it effectively puts most of the weight on the predictive loss. There is a clear trend (Figure 1b) in error decrease with the parameter decrease, where sufficiently low values of $\alpha \leq 0.1$ yield quite similar low errors, but increasing α further, especially above 1, leads to a drastic increase in the classification error, especially in higher-dimensional cases. Note, however, that such tendency is not present in the other datasets we experimented with, where the effect of regularization constant can be non-monotonic, and thus underscores the importance of using cross-validation or other parameter-tuning approaches[6].

Regarding the choice of the reduced-dimensionality parameter L, for low values of α, we did not observe any significant variation in the results with increasing this parameter up to $L = 10$, e.g. the results for different L were practically identical when $\alpha \leq 0.1$, while for higher values variance was more significant.

[5]We used the SVM code by A. Schwaighofer available at http://ida.first.fraunhofer.de/~anton/software.html.

[6]Bayesian approach to selecting regularization parameter may prove beneficial, as shown, for example, in (Y. Lin and D. Lee, 2006)

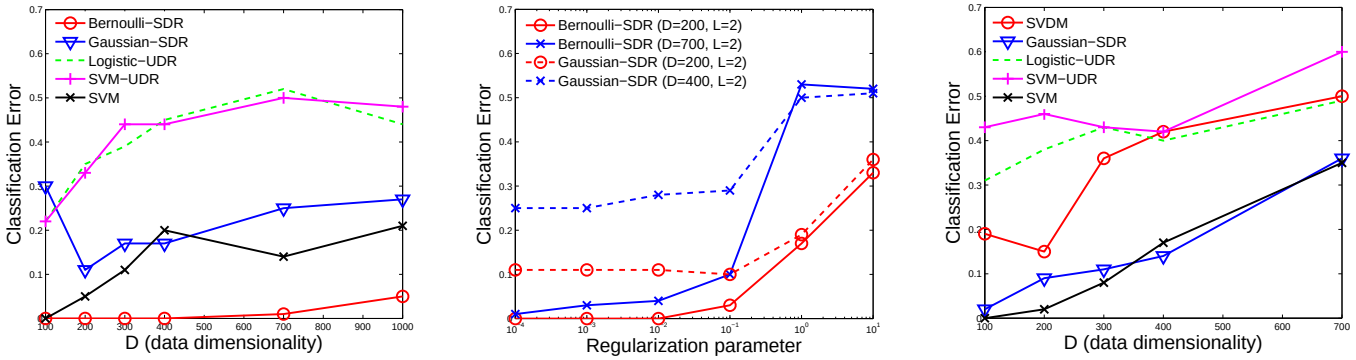


Figure 1. (a) Results for Bernoulli noise. (b) Effects of the regularization parameter α (the weight on the data reconstruction loss) - Bernoulli noise. (c) Results for Gaussian noise.

Simulated data: Gaussian noise. Figure 1c shows the results for the Gaussian noise: supervised dimensionality reduction provides a clear advantage over unsupervised ones, although the performance of SDR versus SVM is somewhat less impressive, as Gaussian-SDR is comparable to SVM. However, Gaussian-SDR seems to outperform considerably another supervised dimensionality method, SVDM, proposed by (Pereira & Gordon, 2006). SVDM was used with its regularization constant set to 100 since it provided the best SVDM performance among the same values of α as before. Interestingly, the performance of SVDM does not show any monotonic dependence on this parameter.

Real-life datasets. First, we considered several datasets with *binary features*, such as a 41-dimensional *Sensor network* dataset where the data represent connectivity (0 or 1) between all pairs of sensor nodes in a network of 41 light sensors (see Table 1). Note that Bernoulli-SDR with $L = 10$ and regularization parameter $\alpha = 0.1$ outperformed SVM, Gaussian-SDR with $L = 8, 10$ and same $\alpha = 0.1$ matched SVM performance, while both the unsupervised dimensionality reduction methods followed by SVM and logistic regression - *SVM-UDR* and *Logistic-UDR*, respectively - performed worse. (Best results for each method are shown in the boldface.) Also, using the Bernoulli model for binary data instead of Gaussian seems to pay off: Bernoulli-SDR performs somewhat better than Gaussian-SDR. It is interesting to note that for really low dimensionality $L = 2$, all of the above methods have same error of 0.2, while increasing the dimensionality allows for much better performance, although this effect is non-monotonic.

Another dataset related to network performance management, of somewhat larger dimensionality ($N = 169$, $D = 168$), contains pairwise end-to-end network latency (ping round-trip times) collected by the PlanetLab measurement project (http://www.pdos.lcs.mit.edu/∼simstrib/pl_app) discretized by applying a threshold as follows: above the average latency is considered "bad" (1) while the below-average latency is considered "good" (0). We selected the first column (latencies of all 169 nodes towards the node 1) as the label, and predicted them given the remaining data. The results are shown in Table 2. The regularization parameters α selected by cross-validation were different here for different SDR methods: for Bernoulli-SDR, $\alpha = 100$ turned out to be the best, while Gaussian-SDR performed better with $\alpha = 1$. Overall, the results for different methods and

Table 1. Results for *Sensor network* dataset ($N = 41, D = 41$): classification errors of different methods for different reduced dimension parameter, L.

method\ L	2	4	6	8	10
Bernoulli-SDR	0.20	0.24	0.27	0.15	**0.12**
Gaussian-SDR	0.20	0.22	0.22	**0.17**	0.17
Logistic-UDR	**0.20**	0.22	0.20	0.20	0.24
SVM-UDR	**0.20**	0.27	0.22	0.27	0.24
SVM	0.17				

Table 2. Results for *PlanetLab* dataset ($N = 169, D = 168$): classification errors of different methods for different reduced dimension parameter, L.

method\ L	2	4	6	8	10
Bernoulli-SDR	0.10	**0.07**	0.10	0.12	0.15
Gaussian-SDR	0.10	0.11	0.08	**0.07**	0.08
Logistic-UDR	0.11	0.10	**0.08**	0.09	0.10
SVM-UDR	0.10	0.09	0.08	0.08	**0.07**
SVM	0.10				

Table 3. Results for *Mass-spectrometry* dataset ($N = 50, D = 38573$): classification errors of different methods for different reduced dimension parameter, L.

method\ L	2	4	6	8	10
Gaussian-SDR	0.04	0.02	0.02	0.02	0.02
Logistic-UDR	0.5	0.18	0.08	0.04	0.02
SVM-UDR	0.54	0.2	0.02	0.06	0.02
SVDM	0.42	0.04	0.02	0.06	0.04
SVM	0.02				

Table 4. Results for *fMRI* dataset ($N = 84, D = 14043$): classification errors of different methods for different reduced dimension parameter, L.

method\ L	5	10	15	20	25
Gaussian-SDR	0.21	0.26	0.23	**0.20**	0.23
Logistic-UDR	0.44	0.42	**0.29**	0.30	0.26
SVM-UDR	**0.49**	0.52	0.56	0.57	0.55
SVDM	0.32	0.25	**0.21**	0.23	0.23
SVM	0.21				

varying dimensions L were surprisingly similar to each other, with both SDR methods achieving the lowest error of 0.07 for $L = 4$ and $L = 8$, respectively, that was also achieved by SVM-UDR at $L = 10$, slightly outperforming SVM (0.10 error). Very similar results (not shown here due to lack of space) were also obtained on the *Advertisement* dataset from UCI ML repository.

We experimented next with several extremely high-dimensional datasets from biological experiments that had *real-valued features*. The first dataset, called here *Proteomics data*, containing mass-spectrometry data for $D = 38573$ proteins (features), showing their "expression levels" in $N = 50$ female subjects (examples), 25 of which were pregnant (class 1), and the others were not (class 0). The results are shown in Table 3, comparing Gaussian-SDR with $\alpha = 0.001$ (that yields lowest average SVDM error (among all dimensions) on this dataset) versus SVM, Logistic-UDR, SVM-UDR (both using the Gaussian assumption, i.e. PCA, for dimensionality reduction), and SVDM with its best-performing parameter 0.01. Despite its very high dimensionality, this dataset turned out to be easy: both SVM and Gaussian-SDR achieved an error of 0.02, and Gaussian-SDR used only $L = 4$ dimensions to achieve it. On the contrary, unsupervised DR followed by predictor learning (Logistic-UDR and SVM-UDR), suffered from really high errors at low dimensions, and only managed to achieve same low error at $L = 10$. SVDM was able to reach its lowest error (same 0.02) a bit earlier ($L = 6$), although for $L = 2$ it incurred a huge error of 0.42, while Gaussian-SDR had 0.04 error at that same level of reduced dimensionality.

Another truly high-dimensional dataset we used contained the fMRI recordings of subject's brain activity (measured using changes in the brain oxygenation levels) while the subject was observing on a screen words of two types, representing tools or buildings (see (Pereira & Gordon, 2006) for details). The task was to learn a mind-reading classifier that would predict, given the fMRI data, what type of the word the subject was looking at. The features here correspond to fMRI voxels ($D = 14043$ voxels were selected after some preprocessing of the data, as described in (Pereira & Gordon, 2006)), and there are $N = 84$ samples (i.e., word instances presented to a subject). This dataset was clearly more challenging then the previous one (both SVM's and SDR's errors were around 0.2).

The results for all methods are shown in Table 4; for Gaussian-SDR we used $\alpha = 0.0001$, while SVDM was best at 0.001 (as before, we used the average error over all dimensions to select best α). For those values of α parameter, Gaussian-SDR matches SVM's errors of 0.21 using just five dimensions ($L = 5$), while SVDM reaches same error at $L = 15$ dimensions. Again, learning a predictor on "compressed" data obtained via *unsupervised* dimensionality reduction was consistently worse than both supervised methods.

5.2. Regression Problems

Finally, we did some preliminary experiments with the regression version of our SDR approach that makes Gaussian assumption about the label (response variable) Y and thus uses sum-squared predictive $\mathcal{L}_Y(UW)$ loss in equation 3, comparing it with the state-of-art sparse-regression technique called the *Elastic Net*, which was shown to improve over both *Lasso* and *ridge* regression using a convex combination of the $L1$- and $L2$-norm regularization terms (Zou & Hastie, 2005).

We used the fMRI data from the 2007 Pittsburgh Brain Activity Interpretation Competition (PBAIC)(Pittsburgh EBC Group, 2007), where the fMRI data were recorded while subjects were playing a videogame, and the task was to predict several real-valued response variables, such as level of anxiety the subject was experiencing etc. We used the data for the two runs (games) by the first subject, and for the *Instructions* response variable, learning from run 1 and predicting on run 2. The dataset contained $N = 704$ samples (measurements over time) and approximately $D = 33,000$ features (voxels). Figure 2 compares the performance of Elastic Net and Gaussian-SDR-Regression, where the L parameter denotes the number of active variables (voxels selected)

by the sparse EN regression. We can see that SDR regression is comparable with the state-of-the art EN (or even slightly better) when the number of hidden dimensions is not too low.

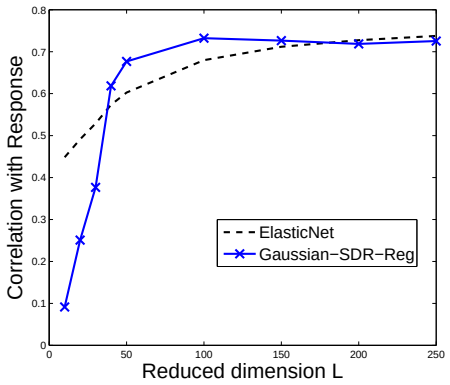


Figure 2. Results for the fMRI dataset from PBAIC. Performance is measured by correlation between the actual and predicted response variable (*Instructions*, in this case).

6. Conclusions and Future Work

This paper proposes a family of SDR algorithms that use generalized linear models to handle various types of features and labels, thus generalizing previous approaches in a unifying framework. Our SDR-GLM approach is fast and simple: it uses closed-form update rules at each iteration of alternating minimization procedure, and is always guaranteed to converge. Experiments on a variety of datasets show that this approach is clearly promising, although more empirical investigation is needed in case of SDR-regression.

Although we only tested our approach with Gaussian and Bernoulli variables, it can be easily extended to multinomial variables (and thus to multiclass classification) using a recent extension of the variational bound proposed in (Jaakkola & Jordan, 1997) to multinomial logistic regression (soft-max)(Bouchard, 2007). Deriving closed-form update rules for other members of the exponential family (e.g., Poisson) remains a direction of future work. Another possible extension is to obtain closed-form SDR update rules for alternative DR methods, such as non-negative matrix factorization (NMF)(Lee & Seung, 2000), simply plugging in NMF's auxiliary function instead of (unconstrained) PCA-like quadratic-loss.

Other potential applications of our approach include dimensionality reduction on mixed-type data, weighted dimensionality reduction schemes (e.g., assigning different weight to reconstruction error of different coordinates in PCA and similar DR techniques), multitask learning, as well as semi-supervised learning, including only the reconstruction loss part of the objective for unlabeled date, while keeping both reconstruction and prediction losses for the labeled ones. Finally, a more principled selection of the regularization constant α (e.g., using Bayesian approaches) is another open research direction.

References

Bouchard, G. (2007). Efficient bounds for the softmax and applications to approximate inference in hybrid models. In *Nips workshop on approximate inference in hybrid models*.

Collins, M., Dasgupta, S., & Schapire, R. (2001). A generalization of principal component analysis to the exponential family. *NIPS2001*.

Gordon, G. (2002). Generalized2 Linear2 Models. *NIPS2002* (pp. 577–584).

Jaakkola, T., & Jordan, M. (1997). A variational approach to bayesian logistic regression problems and their extensions. *Sixth International Workshop on Artificial Intelligence and Statistics*.

Lee, D. D., & Seung, S. H. (2000). Algorithms for non-negative matrix factorization. *NIPS* (pp. 556–562).

Pereira, F., & Gordon, G. (2006). The Support Vector Decomposition Machine. *ICML2006* (pp. 689–696).

Pittsburgh EBC Group (2007). PBAIC Homepage: http://www.ebc.pitt.edu/2007/competition.html.

Rish, I., Grabarnik, G., Cecchi, G., Pereira, F., & Gordon, G. (2008). *Closed-Form Supervised Dimensionality Reduction with Generalized Linear Models* (Technical Report). IBM T.J. Watson Research Center.

Sajama and Alon Orlitsky (2005). Supervised dimensionality reduction using mixture models. *ICML '05: Proceedings of the 22nd International Conference on Machine learning* (pp. 768–775). New York, NY, USA: ACM.

Schein, A., Saul, L., & Ungar, L. (2003). A generalized linear model for principal component analysis of binary data. *Ninth International Workshop on Artificial Intelligence and Statistics*.

Weinberger, K., Blitzer, J., & Saul, L. (2005). Distance Metric Learning for Large Margin Nearest Neighbor Classification. *NIPS2005*.

Weinberger, K., & Tesauro, G. (2007). Metric learning for kernel regression. In M. Meila and X. Shen (Eds.), *Eleventh international conference on artificial intelligence and statistics*, 608–615. Puerto Rico: Omnipress.

Y. Lin and D. Lee (2006). Bayesian L1-norm sparse learning. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2006)*.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society, Series B*, *67*, 301–320.

Bi-Level Path Following for Cross Validated Solution of Kernel Quantile Regression

Saharon Rosset SAHARON@POST.TAU.AC.IL
Department of Statistics and Operations Research, The Raymond and Beverly Sackler School of Mathematical Sciences, Tel Aviv University, Israel

Abstract

Modeling of conditional quantiles requires specification of the quantile being estimated and can thus be viewed as a parameterized predictive modeling problem. Quantile loss is typically used, and it is indeed parameterized by a quantile parameter. In this paper we show how to follow the path of cross validated solutions to regularized kernel quantile regression. Even though the bi-level optimization problem we encounter for every quantile is non-convex, the manner in which the optimal cross-validated solution evolves with the parameter of the loss function allows tracking of this solution. We prove this property, construct the resulting algorithm, and demonstrate it on data. This algorithm allows us to efficiently solve the whole family of bi-level problems.

1. Introduction

In the standard predictive modeling setting, we are given a *training sample* of n examples $\{\mathbf{x}_1, y_1\}, ..., \{\mathbf{x}_n, y_n\}$ drawn i.i.d from a joint distribution $P(X, Y)$, with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ for regression, $y_i \in \{0, 1\}$ for two-class classification. We aim to utilize these data to build a model $\hat{Y} = \hat{f}(X)$ to describe the relationship between X and Y, and later use it to predict the value of Y given new X values. This is often done by defining a family of models $\mathcal{F}$ and finding (exactly or approximately) the model $f \in \mathcal{F}$ which minimizes an *empirical loss function*: $\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))$. Examples of such algorithms include linear and logistic regression, empirical risk minimization in classification and others.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

If $\mathcal{F}$ is complex, it is often desirable to add *regularization* to control model complexity and overfitting. The generic regularized optimization problem can be written as:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$

where $J(f)$ is an appropriate model complexity penalty and λ is the regularization parameter. Given a loss and a penalty, selection of a good value of λ is a *model selection* problem. Popular approaches that can be formulated as regularized optimization problems include all versions of support vector machines, ridge regression, the Lasso and many others. For an overview of predictive modeling, regularized optimization and the algorithms mentioned above, see for example Hastie et al. (2001).

In this paper we are interested in a specific setup where we have a family of regularized optimization problems, defined by a parameterized loss function and a regularization term. A major motivating example for this setting is regularized quantile regression (Koenker, 2005):

$$\hat{\beta}(\tau, \lambda) = \arg\min_{\beta} \sum_{i=1}^{n} L_\tau(y_i - \beta^\mathsf{T} \mathbf{x}_i) + \lambda \|\beta\|_q^q \quad (1)$$
$$\text{for } 0 < \tau < 1,\ 0 \leq \lambda < \infty$$

where L_τ, the parameterized quantile loss function, has the form:

$$L_\tau(r) = \begin{cases} r\tau & r \geq 0 \\ -r(1-\tau) & r < 0 \end{cases}$$

and is termed *τ-quantile loss* because its population optimizer is the appropriate quantile (Koenker, 2005):

$$\arg\min_{c} E(L_\tau(Y - c)|X) = \text{quantile } \tau \text{ of } P(Y|X) \quad (2)$$

Because quantile loss has this optimizer, the solution of the quantile regression problems for the whole range

$0 < \tau < 1$ has often been advocated as an approach to estimating the full conditional probability of $P(Y|X)$ (Koenker, 2005; Perlich et al., 2007). Much of the interesting information about the behavior of $Y|X$ may lie in the details of this conditional distribution, and if it is not *nicely behaved* (i.i.d Gaussian noise being the most commonly used concept of nice behavior), just estimating a conditional mean or median is often not sufficient to properly understand and model the mechanisms generating Y. The importance of estimating a complete conditional distribution, and not just a central quantity like the conditional mean, has long been noted and addressed in various communities, like Econometrics, Education and Finance (Koenker, 2005; Buchinsky, 1994; Eide & Showalter, 1998). There has also been a surge of interest in the Machine Learning community in conditional quantile estimation in recent years (Meinshausen, 2006; Takeuchi et al., 2006). Figure 1 shows a graphical representation of L_τ for several values of τ, and a demonstration of the conditional quantile curves in a univariate regression setting, where the linear model is correct for the median, but the noise has a non-homoscedastic distribution.

On the penalty side, we typically use the ℓ_q norm of the parameters with $q \in \{1, 2\}$. Adding a penalty can be thought of as shrinkage, complexity control or putting a prior to express our expectation that the β's should be small.

As has been noted in the literature (Rosset & Zhu, 2007; Hastie et al., 2004; Li et al., 2007) if $q \in \{1, 2\}$ and if we fix $\tau = \tau_0$, we can devise *path following* (AKA parametric programming) algorithms to efficiently generate the 1-dimensional curve of solutions $\{\hat{\beta}(\tau_0, \lambda) \ : \ 0 \le \lambda < \infty\}$. Although it has not been explicitly noted by most of these authors, it naturally follows that similar algorithms exist for the case that we fix $\lambda = \lambda_0$ and are interested in generating the curve $\{\hat{\beta}(\tau, \lambda_0) \ : \ 0 < \tau < 1\}$.

In addition to parameterized quantile regression, there are other modeling problems in the literature which combine a parameterized loss function problem with the existence of efficient path following algorithms. These include Support vector regression (SVR, Smola and Schölkopf (2004), see Gunther and Zhu (2005) for path following algorithm) with ℓ_1 or ℓ_2 regularization, where the parameter ϵ determines the width of the *don't care* region around 0.

An important extension of the ℓ_2-regularized optimization problem is to *non-linear* fitting through kernel embedding (Schölkopf & Smola, 2002). The kernelized version of problem (1) is:

$$\hat{f}(\tau, \lambda) = \arg\min_f \sum_i L_\tau(y_i - f(\mathbf{x}_i)) + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_K} \quad (3)$$

where $\|\cdot\|_{\mathcal{H}_K}$ is the norm induced by the positive-definite kernel K in the Reproducing Kernel Hilbert Space (RKHS) it generates. The well known *representer theorem* (Kimeldorf & Wahba, 1971) implies that the solution of problem (3) lies in a low dimensional subspace spanned by the representer functions $\{K(\cdot, \mathbf{x}_i), i \in 1, ..., n\}$. Following the ideas of Hastie et al. (2004) for the support vector machine, Li et al. (2007) have shown that λ-path of solutions to problem (3) when τ is fixed can also be efficiently generated.

It is important to note the difference in the roles of the two parameters τ, λ. The former defines a family of loss functions, in our case leading to estimation of different quantiles. Thus we would typically want to build and use a model for every value of τ. The latter is a regularization parameter, controlling model complexity with the aim of generating a better model and avoiding overfitting, and is not part of the prediction objective (at least as long as we avoid the Bayesian view). We would therefore typically want to generate a set of models $\beta^*(\tau)$ (or $f^*(\tau)$ in the kernel case), by selecting a good regularization parameter $\lambda^*(\tau)$ for every value of τ, thus obtaining a family of good models for estimating the range of conditional quantiles, and consequently the whole conditional distribution.

This problem, of model selection to find a good regularization parameter, is often handled through *cross-validation*. In its simplest form, cross-validation entails having a second, independent set of data $\{\tilde{\mathbf{x}}_i, \tilde{y}_i\}_{i=1}^N$ (often referred to as a *validation set*), which is used to evaluate the performance of the models and select a good regularization parameter. For a fixed τ, we can write our model selection problem as a *Bi-level programming* extension of problems (1, 3), where $f^*(\tau) = \hat{f}(\tau, \lambda^*)$ and λ^* solves:

$$\begin{aligned} \min_{\lambda} \quad & \sum_{i=1}^{N} L_\tau(\tilde{y}_i, \hat{f}(\tau, \lambda)^\mathsf{T}\tilde{\mathbf{x}}_i) \\ & \text{s.t.} \quad \hat{f}(\tau, \lambda) \text{ solves problem (3)} \end{aligned} \quad (4)$$

The objective of this minimization problem is not convex as a function of λ. A similar non-convex optimization problem has been tackled by Kunapuli et al. (2007). The fundamental difference between their setting and ours is that they had a single bi-level optimization problem, while we have a family of such problems, parameterized by τ. This allows us to take advantage of internal structure to solve the bi-level

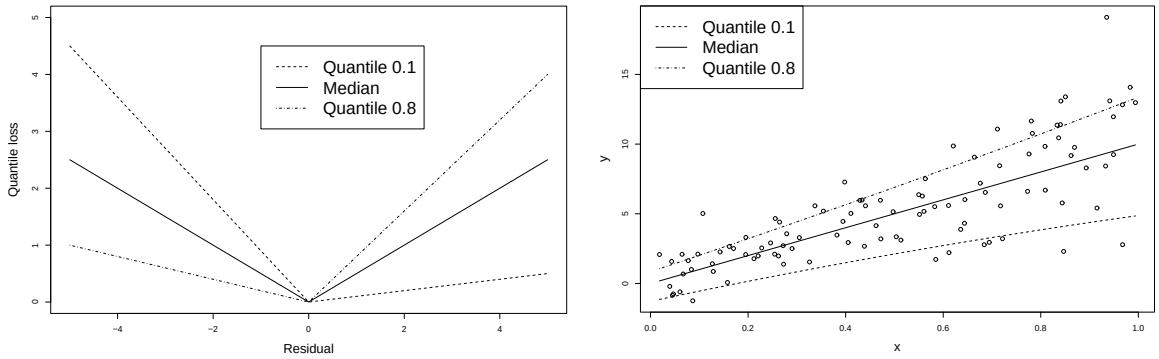


Figure 1. Quantile loss function for some values of τ (left) and an example where the median of Y is linear in X but the quantiles of $P(Y|X)$ are not because the noise is not identically distributed (right).

problem for all values of τ *simultaneously* (or more accurately, in one run of our algorithm).

The concept of a parameterized family of bi-level regularized quantile regression problems is demonstrated in Figure 2, where we see the cross-validation curves of the objective of (4) as a function of λ for several values of τ on the same dataset. As we can see, the optimal level of regularization varies with the quantile, and correct choice of the regularization parameter can have a significant effect on the success of our quantile prediction model.

The main goal of this paper is to devise algorithms for following the bi-level optimal solution path $f^*(\tau)$ as a function of τ, and demonstrate their practicality. We show that this non-convex family of bi-level programs can be solved exactly, as the optimum among the solutions of $O(n+N)$ standard (convex) path-following problems, with some additional twists. This result is based on a characterization of the evolution of the solution path $\hat{f}(\tau,\cdot)$ as τ varies, and on an understanding of the properties of optimal solutions of the bi-level problem, which can only occur at a limited set of points. We combine these insights to formulate an actual algorithm for solving this family of bi-level programs via path-following.

The rest of this paper is organized as follows. In Section 2 we discuss the properties of the quantile regression solution paths $\hat{f}(\tau,\lambda)$ and their evolution as τ changes. We then discuss in Section 3 the properties of the bi-level optimization problem (4) and demonstrate that the solutions change predictably with τ. Bringing together all our insights leads us to formulate an algorithm in Section 4, which allows us to follow the path of solutions $\{f^*(\tau)\ ,\ 0<\tau<1\}$ and only requires following a large but manageable number of solution paths of problem (3) simultaneously. We demonstrate our methods with a simulated data study in Section 5, where we demonstrate the computational efficiency of our approach and the ability of KQR to capture non-standard conditional distributions $P(Y|X)$.

This paper is a short version of Rosset (2008), and we defer the proofs, some of the technical details and much of the discussion to that version.

2. Quantile Regression Solution Paths

We concentrate our discussion on the kernel quantile regression (KQR) formulation in (3), with the understanding that it subsumes the linear formulation (1) with ℓ_2 regularization by using the *linear* kernel $K(\mathbf{x},\tilde{\mathbf{x}})=\mathbf{x}^\mathsf{T}\tilde{\mathbf{x}}$.

We briefly survey the results of Li et al. (2007) regarding the properties of $\hat{f}(\tau,\cdot)$, the optimal solution path of (3), with τ fixed. The representer theorem (Kimeldorf & Wahba, 1971) implies that the solution can be written as:

$$\hat{f}(\tau,\lambda)(\mathbf{x})=\frac{1}{\lambda}\left[\hat{\beta}_0+\sum_{i=1}^{n}\hat{\theta}_i K(\mathbf{x},\mathbf{x}_i)\right] \qquad (5)$$

For a proposed solution $f(\mathbf{x})$ define:

- $\mathcal{E}=\{i: y_i-f(\mathbf{x}_i)=0\}$ (points on *elbow* of L_τ)
- $\mathcal{L}=\{i: y_i-f(\mathbf{x}_i)<0\}$ (left of elbow)
- $\mathcal{R}=\{i: y_i-f(\mathbf{x}_i)>0\}$ (right of elbow)

Then Li et al. (2007) show that the Karush-Kuhn-Tucker (KKT) conditions for optimality of a solution $\hat{f}(\tau,\lambda)$ of problem (3) can be phrased as:

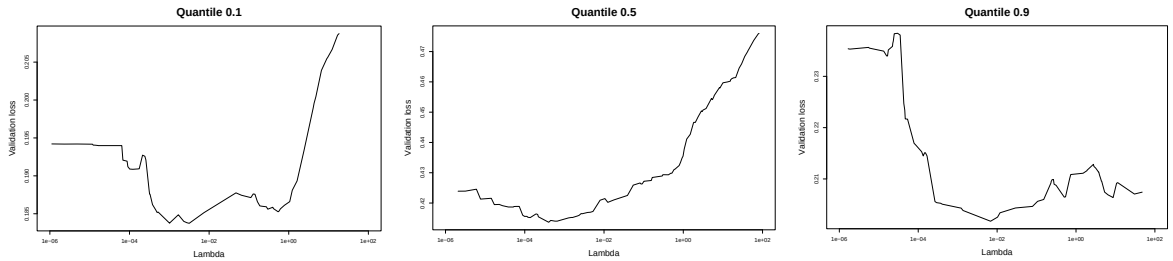


Figure 2. Estimated prediction error curves of Kernel Quantile Regression for some quantiles on one dataset. The errors are shown as a function of the regularization parameter λ

- $i \in \mathcal{E} \;\Rightarrow\; -(1-\tau) \le \hat{\theta}_i \le \tau$
- $i \in \mathcal{L} \;\Rightarrow\; \hat{\theta}_i = -(1-\tau)$
- $i \in \mathcal{R} \;\Rightarrow\; \hat{\theta}_i = \tau$
- $\sum_i \hat{\theta}_i = 0$

with some additional algebra they show that for a fixed τ, there is a series of *knots*, $0 = \lambda_0 < \lambda_1 < ... < \lambda_m < \infty$ such that for $\lambda \ge \lambda_m$ we have $\hat{f}(\tau, \lambda) =$ constant and for $\lambda_{k-1} < \lambda \le \lambda_k$ we have:

$$\hat{f}(\tau,\lambda)(\mathbf{x}) = \frac{1}{\lambda}\left(\lambda_k \hat{f}(\tau,\lambda_k)(\mathbf{x}) + (\lambda-\lambda_k)h_k(\mathbf{x})\right) \quad (6)$$

where $h_k(\mathbf{x}) = b_0^k + \sum_{i\in\mathcal{E}_k} b_i^k K(\mathbf{x},\mathbf{x}_i)$ can be thought of as the *direction* in which the solution is moving for the region $\lambda_{k-1} < \lambda \le \lambda_k$. The knots λ_k are points on the path where an observation passes between $\mathcal{E}$ and either $\mathcal{L}$ or $\mathcal{R}$, that is $\exists i \in \mathcal{E}$ such that exactly $\theta_i = \tau$ or $\theta_i = -(1-\tau)$. These insights lead Li et al. (2007) to an algorithm for incrementally generating $\hat{f}(\tau,\lambda)$ as a function of λ for fixed τ, starting from $\lambda = \infty$ (where the solution only contains the intercept β_0).

Although Li et al. (2007) suggest it is a topic for further study, it is in fact a reasonably straight forward extension of their results to show that a similar scenario holds when we fix λ and allow τ only to change, and also when both τ, λ are changing together *along a straight line*, i.e., a 1-dimensional subspace of the (τ,λ) space (this has been observed by Wang et al. (2006) for SVR, which is very similar from an optimization perspective). The explicit result is given in (Rosset, 2008), but we omit it here for brevity, given its marginal novelty.

Armed with this result, we next show the main result of this section: that the knots themselves move in a (piecewise) straight line as τ changes, and can therefore be *tracked* as τ and the regularization path change. Fix a quantile τ_0 and assume that λ_k is a knot in the λ-solution path for quantile τ_0. Further, let i_k be the observation that is passing in or out of the elbow at knot λ_k. Assume WLOG that $\hat{\theta}_{i_k}(\tau_0,\lambda_k) = \tau_0$, i.e. it is on the boundary between $\mathcal{R}_k$ and $\mathcal{E}_k$. Let $\tilde{\mathbf{K}}_{\mathcal{E}_k}$ be the matrix $\mathbf{K}_{\mathcal{E}_k}$ with the i_k column removed, and $\tilde{\mathbf{b}}^k = \mathbf{b}^k$ with index i_k removed. Let $s_i = \sum_{j\in\mathcal{R}\cup\mathcal{L}\cup\{i_k\}} K(\mathbf{x}_i,\mathbf{x}_j)$ for $i \in \mathcal{E}_k$. Let $\mathbf{s}_{\mathcal{E}_k}$ be the vector of all these values.

Theorem 1 *Any knot λ_k moves linearly as τ changes. That is, there exists a constant c_k such that for quantile $\tau_0 + \delta$ there is a knot in the λ-solution path at $\lambda_k + c_k\delta$, for $\delta \in [-\epsilon_k, \nu_k]$, a non-empty neighborhood of 0. c_k is determined through the solution of another set of $|\mathcal{E}_k|+1$ linear equations with $|\mathcal{E}_k|+1$ unknowns:*

$$\mathbf{B}^k \begin{pmatrix} \tilde{\mathbf{b}}^k \\ c_k \end{pmatrix} = \begin{pmatrix} -(|\mathcal{R}|+|\mathcal{L}|+1) \\ -\mathbf{s}_{\mathcal{E}_k} \end{pmatrix}$$

with

$$\mathbf{B}^k = \begin{pmatrix} 0 & \mathbf{1}^\mathsf{T} & 0 \\ \mathbf{1} & \tilde{\mathbf{K}}_{\mathcal{E}_k} & -\mathbf{y}_{\mathcal{E}_k} \end{pmatrix}$$

And the fit at this knot progresses as:

$$\hat{f}(\tau_0+\delta, \lambda_k + c_k\delta) = \quad (7)$$
$$= \frac{1}{\lambda_k + c_k\delta}\left(\lambda_k \hat{f}(\lambda_k,\tau_0)(\mathbf{x}) + \delta h_k(\mathbf{x})\right)$$
$$h_k(\mathbf{x}) = \tilde{b}_0^k + \sum_{i\in\mathcal{E}_k - i_k} \tilde{b}_i^k K(\mathbf{x},\mathbf{x}_i) + \quad (8)$$
$$+ \sum_{i\in\mathcal{L}\cup\mathcal{R}\cup\{i_k\}} K(\mathbf{x},\mathbf{x}_i)$$

This theorem tells us that we can in fact track the knots in the solution efficiently as τ changes. We still have to account for various types of *events* that can change the direction the knot is moving in. The value θ_i for a point in $\mathcal{E}_k - \{i_k\}$ can reach τ or $-(1-\tau)$, or a point in $\mathcal{L}\cup\mathcal{R}$ may reach the elbow $\mathcal{E}$. These events correspond to *knots crossings*, i.e., the knot λ_k is encountering another knot (which is tracking the other event). There are also *knot birth* events, and *knots*

merge events, which are possible but rare, and somewhat counter-intuitive. The details of how these are identified and handled can be found in the detailed algorithm description (Rosset, 2008). When any of these events occurs, the set of knots has to be updated and their directions have to be re-calculated using Theorem 1 and the new identity of the sets $\mathcal{E}, \mathcal{L}, \mathcal{R}$ and the observation i_k. This in essence allows us to map the whole 2-dimensional solution surface $\hat{f}(\tau, \lambda)$.

3. The Bi-Level Optimization Problem

Our next task is to show how our ability to track the knots as τ changes allows us to track the solution of the bi-level optimization problem (4), as τ changes. The key to this step is the following result.

Theorem 2 *Any minimizer*[1] *of (4) is always either at a knot in the λ-path for this τ or a point where a validation observation crosses the elbow. In other words, one of the two following statements must hold:*

- *λ^* is a knot: $\exists i \in \{1...n\}$ s.t. $\hat{f}(\tau, \lambda^*(\tau))(\mathbf{x}_i) = y_i$ and $\theta_i \in \{\tau, -(1-\tau)\}$, or*
- *λ^* is a* validation crossing*: $\exists i \in \{1...N\}$ s.t. $\hat{f}(\tau, \lambda^*(\tau))(\tilde{\mathbf{x}}_i) = \tilde{y}_i$*

Corollary 1 *Given the complete solution path for $\tau = \tau_0$, the solutions of the bi-level problem (4) for a range of quantiles around τ_0 can be obtained by following the paths of the knots and the validation crossings only, as τ changes.*

To implement this corollary in practice, we have two main issues to resolve: 1. How do we follow the paths of the validation crossings? 2. How do we determine which one of the knots and validation crossings is going to be optimal for every value of τ?

The first question is easy to answer when we consider the similarity between the knot following problem we solve in Theorem 1 and the validation crossing following problem. In each case we have a set of *elbow* observations whose fit must remain fixed as τ changes, but whose $\hat{\theta}$ values may vary; sets $\mathcal{L}, \mathcal{R}$ whose $\hat{\theta}$ are changing in a pre-determined manner with τ, but whose fit may vary freely; and one special observation which *characterizes* the knot or validation crossing. The only difference is that in a knot this is a *border* observation from the training set, so both its fit and its $\hat{\theta}$ are predetermined, while in the case of validation crossing it is a *validation* observation, whose fit must remain fixed (at the *elbow*), but which does not even have a $\hat{\theta}$ value. Taking all of this into account, it is easy to show a result similar to Theorem 1 for the validation crossings. We refer the reader to Rosset (2008) for the details.

The second question we have posed requires us to explicitly express the validation loss (i.e., L_τ on the validation set) at every knot and validation crossing in terms of δ, so we can compare them and identify the optimum at every value of δ. Using the representation in (7) we can write the *validation loss* for a knot k :

$$\sum_{i=1}^{N} \quad L_\tau(\tilde{y}_i, \hat{f}(\tau_0 + \delta, \lambda_k + c_k\delta)(\tilde{\mathbf{x}}_i)) = \ldots = \\ = \frac{\text{quadratic in } \delta}{\lambda_k + c_k\delta} \tag{9}$$

see Rosset (2008) for details, and note that similar expressions can naturally be derived for validation crossings. These are rational functions of δ with quadratic expressions in the numerator and linear expressions in the denominator. Our cross-validation task can be re-formulated as the identification of the minimum of these rational functions among all knots and validation crossings, for every value of τ in the current *segment*, where the directions h_k, h_v of all knots and validation crossings are fixed (and therefore so are the coefficients in the rational functions). This is a *lower-envelope* tracking problem, which has been extensively studied in the literature (Sharir and Agarwal (1995) and references therein).

To calculate the meeting point of two elements with neighboring scores we find the zeros of the cubic equation obtained by requiring equality for the two rational functions of the form (9) corresponding to the two elements. The smallest non-negative solution for δ is the one we are interested in.

4. Algorithm Overview

Bringing together all the elements from the previous sections, we now give (Algorithm 1) a succinct overview of the resulting algorithm. Since there is a multitude of details, we defer a detailed pseudo-code description of our algorithm to Rosset (2008).

The algorithm follows the knots of the λ-solution path as τ changes using the results of Section 2, and keeps track of the cross-validated solution using the results of Section 3. Every time an *event* happens (like a knot crossing), the direction in which two of the knots are moving has to be changed, or knots have to be added or deleted. Between these events, the evolution of the cross-validation objective at all knots and validation crossings has to be sorted and followed. Their order

[1] In pathological cases there may be a "segment" of minimizers. In this case it can be shown that such a segment will always be flanked by points described in the theorem.

is maintained, and updated whenever crossings occur between them.

4.1. Approximate Computational Complexity

Looking at Algorithm 1, we should consider the number of steps of the two loops and the complexity of the operations inside the loops. Even for a "standard" λ-path following problem for fixed τ, it is in fact impossible to rigorously bound the number of steps in the general case, but it has been argued and empirically demonstrated by several authors that the number of knots in the path behaves as $O(n)$, the number of samples (Rosset & Zhu, 2007; Hastie et al., 2004; Li et al., 2007). In our case the outer loop of Algorithm 1 implements a 2-dimensional path following problem, that can be thought of as following $O(n)$ 1-dimensional paths traversed by the knots of the path. It therefore stands to reason (and we confirm it empirically below) that the outer loop typically has $O(n^2)$ steps where *events* happen. The events in the inner loop, in turn, have to do with N validation observations meeting the $O(n)$ knots. So a similar logic would lead us to assume that the number of meeting events (counted by the inner loop) should be at most $O(nN)$ total for the whole running of the algorithm (i.e., many iterations of the outer loop may have no events happening in the inner loop). Each iteration of either loop requires a re-calculation of up to three directions (of knots or validation crossings), using Theorem 1. These calculations involve updating and inversion of matrices that are roughly $|\mathcal{E}| \times |\mathcal{E}|$ in size (where $|\mathcal{E}|$ is the number of observations in the elbow). However note that only one row and column are involved in the updating, leading to a complexity of $O(n + |\mathcal{E}|^2)$ for the whole direction calculation operation, using the Sherman-Morrison formula for updating the inverse. In principle, $|\mathcal{E}|$ can be equal to n, although it is typically much smaller for most of the steps of the algorithm, on the order of $\sqrt{n}$ or less. So we assume here that the loop cost is between $O(n)$ and $O(n^2)$.

Putting all of these facts and assumptions together, we can estimate the algorithm's complexity's *typical* dependence on the number of observations in the training and validation set as ranging between $O(n^2 \cdot \max(n, N))$ and $O(n^3 \cdot \max(n, N))$. Clearly, this estimation procedure falls well short of a formal "worst case" complexity calculation, but we offer it as an intuitive guide to support our experiments below and get an idea of the dependence of running time on the amount of data used.

We have not considered the complexity of the lower envelope tracking problem in our analysis, because it is expected to have a much lower complexity (number of order changes $O(\max(n, N) \log(\max(n, N)))$ and each order change involves $O(1)$ work).

5. Experiments

We demonstrate two main aspects of our methodology: The computational behavior as a function of the amount of training data used; and the ability of KQR to capture the form of the conditional distribution $P(Y|X)$, both in standard (i.i.d error) situations and when the error is not homoscedastic and asymmetric. We limit the experiments to simple simulated data, where we know the truth and can understand the behavior of KQR. This is due to shortage of space, and since our main contribution is not a new method that should be compared to competitors on real data, but rather a new algorithmic approach for an existing method.

Our simulation setup starts from univariate data $x \in [0, 1]$ and a "generating" function $f(x) = 2 \cdot \left(\exp(-30 \cdot (x - 0.25)^2) + \sin(\pi \cdot x^2)\right)$ (see Figure 3). We then let $Y = f(x) + \epsilon$, where the errors ϵ are independent, with a distribution that can be either:

1. $\epsilon \sim N(0, 1)$, i.e., i.i.d standard normal errors
2. $\epsilon + (x+1)^2 \sim \exp(1/(x+1)^2)$, which gives us errors that are still independent and have mean 0, but are asymmetric and have non-constant variance, with small signal-to-noise ratio on the higher values of x (see Figure 4).

Figure 3 demonstrates the results of the algorithm with i.i.d normal errors, 200 training samples and 200 validation samples and Gaussian kernel with parameter $\sigma = 0.2$. We see that the quantile estimates all capture the general shape of the true curve, with some "smoothing" due to regularization.

Next we consider the computational complexity of the algorithm, and its dependence on the number of training samples (with 200 validation samples). We compare it to the 1-dimensional KQR algorithm of Li et al. (2007), who have already demonstrated that their algorithm is significantly more efficient than grid-based approaches for generating 1-dimensional paths for fixed τ. Table 1 shows the number of steps of the main (outer) loop of Algorithm 1 and the total run time of our algorithm for generating the complete set of cross-validated solutions for $\tau \in [0.1, 0.9]$ as a function of the number of training samples (with validation sample fixed at 200). Also shown is the run time for the algorithm of Li et al. (2007), when we use it on a

Algorithm 1 Main steps of our bi-level path following algorithm

Input: The entire λ-solution path for quantile τ_0; the bi-level optimizer $\lambda^*(\tau_0)$
Output: Cross-validated solutions $f^*(\tau)$ for $\tau \in [\tau_0, \tau_{\text{end}}]$
Initialization: identify all knots and validation crossings in the solution path for τ_0;
Find direction of each knot according to Theorem 1
Find direction of each validation crossing
Create a list M of knots and validation crossings sorted by their validation loss
Let $\lambda^*(\tau_0)$ be the one at the bottom of the list M, and $f^*(\tau_0)$ accordingly
Calculate future meeting of each pair of neighbors in M by solving the cubic equation implied by (9)
Set $\tau_{\text{now}} = \tau_0$
while $\tau_{\text{now}} < \tau_{\text{end}}$ **do**
  Find value $\tau_1 > \tau_{\text{now}}$ where first knot crossing occurs
  Find value $\tau_2 > \tau_{\text{now}}$ where first knot merge occurs
  Find value $\tau_3 > \tau_{\text{now}}$ where first knot birth occurs
  Set $\tau_{\text{new}} = \min(\tau_1, \tau_2, \tau_3)$
  while $\tau_{\text{now}} < \tau_{\text{new}}$ **do**
    Find value $\tau_4 > \tau_{\text{now}}$ where first future meeting (order change) in M occurs
    Find value $\tau_5 > \tau_{\text{now}}$ where first validation crossing birth occurs
    Find value $\tau_6 > \tau_{\text{now}}$ where first validation crossing cancelation occurs
    Set $\tau_{\text{next}} = \min(\tau_4, \tau_5, \tau_6)$
    Update $\lambda^*(\tau), f^*(\tau)$ for $\tau \in (\tau_{\text{now}}, \tau_{\text{next}})$ as the evolution of the knot or validation crossing attaining the minimal L_τ in M (i.e., the one at $\lambda^*(\tau_{\text{now}})$)
    Update M according to the first event (order change, birth, cancelation)
    Update the future meetings of the affected elements using (9)
    Set $\tau_{\text{now}} = \tau_{\text{next}}$
  end while
  Update the list of knots according to the first event (knot crossing, birth, merge)
  Update the directions of affected knots using Theorem 1
end while

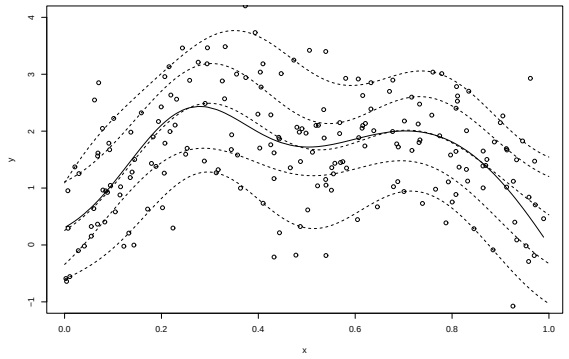


Figure 3. The function $f(x)$ (solid), data points drawn from it with i.i.d normal error, and our cross-validated estimates of quantiles $0.1, 0.25, 0.5, 0.75, 0.9$ (dahsed lines, from bottom to top).

grid of 8000 evenly spaced τ values in $[0.1, 0.9]$ and find the best cross validated solution by enumerating the candidates as identified in Section 3. Our conjecture that the number of knots in the 2-dimensional path behaves like $O(n^2)$ seems to be consistent with these results, as is the hypothesized overall time complexity dependence of $O(n^3)$.

Finally, we demonstrate the ability of KQR to capture the quantiles with "strange" errors from model 2. Figure 4 shows a data sample generated from this model and the $(0.25, 0.5, 0.75)$ quantiles of the conditional distribution $P(Y|X)$ (solid), compared to their cross-validated KQR estimates (dashed), using 500 samples for learning and 200 for validation (more data is needed for learning because of the very large variance at values of x close to 1). As expected, we can see that estimation is easier of the lower quantiles and at smaller values of x, because the distribution $P(Y|X = x)$ has long right tails everywhere and has much larger variance when x is big.

Table 1. Number of steps and run times of our algorithm and of Li et al. (2007), for the whole path from $\tau = 0.1$ to $\tau = 0.9$.

NTRAIN	NSTEPS	TIME(BI-LEVEL)	TIME(LI ET AL.)
200	29238	931 SEC.	2500 SEC.
100	12269	99 SEC.	900 SEC.
50	2249	23 SEC.	480 SEC.

6. Conclusions and Extensions

In this paper we have demonstrated that the family of bi-level optimization problems (4) defined by the

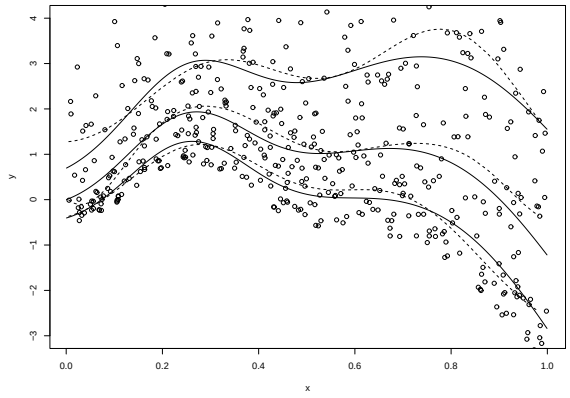


Figure 4. Quantiles of $P(Y|X)$ (solid), and their estimates (dashed) for quantiles $(0.25, 0.5, 0.75)$ with the exponential error model.

family of loss functions L_τ can be solved via a *path following* approach which essentially maps the whole surface of solutions $\hat{f}(\tau, \lambda)$ as a function of both τ and λ and uses insights about the possible locations of the bi-level optima to efficiently find them. This leads to a closed-form algorithm for finding $f^*(\tau)$ for all quantiles. We see two main contributions in this work: a. Identification and solution of a family of non-convex optimization problem of great practical interest which can be solved using solely convex optimization techniques; and b. Formulation of a practical algorithm for generating the full set of cross-validated solutions for the family of kernel quantile regression problems.

Our algorithm as presented here can easily be adapted to bi-level path following of cross validated solutions of SVR, as the size ϵ of the *don't-care* region changes (see Rosset (2008) for details). However, it should be noted that the statistical motivation for solving quantile regression for multiple quantiles does not really carry through to ϵ-SVR, as the parameter ϵ and the loss function it parameterizes do not have a natural interpretation in the spirit of τ.

There are many other interesting aspects of our work, which we have not touched on here, including: development of further optimization shortcuts to improve algorithmic efficiency; investigation of the range of applicability of our algorithmic approach beyond KQR and SVR; analysis of the use of various kernels for KQR and how the kernel parameters and kernel properties interact with the solutions; implementation of in-sample model selection approaches such as SIC (Koenker, 2005; Li et al., 2007) instead of cross validation in our framework (which should require minimal changes).

References

Buchinsky, M. (1994). Changes in the u.s. wage structure 1963-1987: Application of quantile regression. *Econometrica*, *62*, 405–458.

Eide, E., & Showalter, M. (1998). The effect of school quality on student performance: A quantile regression approach. *Economics Letters*, *58*, 345–350.

Gunther, L., & Zhu, J. (2005). Efficient computation and model selection for the support vector regression. *Neural Computation*, *19*.

Hastie, T., Rosset, S., Tibshirani, R., & Zhu, J. (2004). The complete regularization path of the support vector machine. *JMLR*, *5*, 1391–1415.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *Elements of statistical learning*. Springer.

Kimeldorf, G., & Wahba, G. (1971). Some results on chebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, *33*, 82–95.

Koenker, R. (2005). *Quantile regression*. New York : Cambridge University Press.

Kunapuli, G., Bennett, K. P., Hu, J., & Pang, J.-S. (2007). Bilevel model selection for support vector machines. CRM Proceedings and Lecture Notes. American Mathematical Society, to appear.

Li, Y., Liu, Y., & Zhu, J. (2007). Quantile regression in reproducing kernel hilbert spaces. *JASA*, *102*.

Meinshausen, N. (2006). Quantile regression forests. *JMLR*, *7*, 983–999.

Perlich, C., Rosset, S., Lawrence, R., & Zadrozny, B. (2007). High quantile modeling for customer wallet estimation with other applications. *Proceedings of the Twelfth International Conference on Data Mining, KDD-07*.

Rosset, S. (2008). Bi-level path following for cross validated solution of kernel quantile regression. In preparation, evolving draft available at www.tau.ac.il/~saharon/papers/cvpath.pdf.

Rosset, S., & Zhu, J. (2007). Piecewise linear regularized solution paths. *Annals of Statistics*, *35*.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. MIT Press, Cambridge, MA.

Sharir, M., & Agarwal, P. K. (1995). *Davenport-schinzel sequences and their geometric applications*. Cambridge University Press.

Smola, A., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*, 199–222.

Takeuchi, I., Le, Q., Sears, T., & Smola, A. (2006). Nonparametric quantile estimation. *JMLR*, *7*, 1231–1264.

Wang, G., Yeung, D.-Y., & Lochovsky, F. H. (2006). Two-dimensional solution path for support vector regression. *Proceedings of the 23rd international conference on Machine learning* (pp. 993–1000).

The Group-Lasso for Generalized Linear Models: Uniqueness of Solutions and Efficient Algorithms

Volker Roth VOLKER.ROTH@UNIBAS.CH
Computer Science Department, University of Basel, Bernoullistr. 16, CH-4056 Basel, Switzerland

Bernd Fischer BERND.FISCHER@INF.ETHZ.CH
Institute of Computational Science, ETH Zurich, Universitaetstrasse 6, CH-8092 Zurich, Switzerland

Abstract

The Group-Lasso method for finding important explanatory factors suffers from the potential non-uniqueness of solutions and also from high computational costs. We formulate conditions for the uniqueness of Group-Lasso solutions which lead to an easily implementable test procedure that allows us to identify all potentially active groups. These results are used to derive an efficient algorithm that can deal with input dimensions in the millions and can approximate the solution path efficiently. The derived methods are applied to large-scale learning problems where they exhibit excellent performance and where the testing procedure helps to avoid misinterpretations of the solutions.

1. Introduction

In many practical learning problems we are not only interested in low prediction errors but also in identifying important explanatory factors. These explanatory factors can often be represented as groups of input variables. Common examples are k-th order polynomial expansions of the inputs where the groups consist of products over combinations of variables up to degree k. Such expansions compute explicit mappings into feature spaces induced by polynomial kernel functions of the form $k(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x} \cdot \boldsymbol{y})^k$. Another popular example are categorical variables that are represented as groups of dummy variables.

A method for variable selection which has gained particular attention is the Lasso (Tibshirani, 1996) which exploits the idea of using ℓ_1-constraints in fitting problems. The Group-Lasso (Yuan & Lin, 2006) extends the former in the sense that it finds solutions that are sparse on the level of *groups* of variables, which makes this method a good candidate for situations described above. The Group-Lasso estimator, however, has several drawbacks: (i) in high-dimensional spaces, the solutions may not be unique. The potential existence of several solutions that involve different variables seriously hampers the interpretability of "identified" explanatory factors; (ii) existing algorithms can handle input dimensions up to thousands (Kim et al., 2006) or even several thousands (Meier et al., 2008), but in practical applications with high-order interactions or polynomial expansions these limits are easily exceeded; (iii) contrary to the standard Lasso, the solution path (i.e. the evolution of the individual group norms as a function of the constraint) is not piecewise linear, which precludes the application of efficient optimization methods like *least angle regression* (LARS) (Efron et al., 2004).

In this paper we address all these issues: (i) we derive conditions for the *completeness* and *uniqueness* of Group-Lasso estimates, where we call a solution *complete*, if it includes all groups that might be relevant in other solutions (meaning that we cannot have "overlooked" relevant groups). Based on these conditions we develop an easily implementable *test procedure*. If a solution is not complete, this procedure *identifies all other groups* that may be included in alternative solutions with identical costs. (ii) These results allow us to formulate a *highly efficient active-set algorithm* that can deal with input dimensions in the millions. (iii) The *solution path* can be approximated on a fixed grid of constraint values with almost no additional computational costs. Large-scale applications using both synthetic and real data illustrate the excellent performance of the developed concepts and algorithms. In particular, we demonstrate that the proposed completeness test successfully detects ambiguous solutions and thus avoids the misinterpretation of "identified" explanatory factors.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Characterization of Group-Lasso Solutions for Generalized Linear Models

This section largely follows (Osborne et al., 2000), with the exception that here we address the *Group*-Lasso problem and a more general class of likelihood functions.

According to (McCullaghand & Nelder, 1983), a generalized linear model (GLM) consists of three elements:
(i) a random component $f(y;\mu)$ specifying the stochastic behavior of a response variable Y;
(ii) a systematic component $\eta = \boldsymbol{x}^\top\boldsymbol{\beta}$ specifying the variation in the response variable accounted for by known covariates $\boldsymbol{x}$; and
(iii) a link function $g(\mu) = \eta$ specifying the relationship between the random and systematic components.
The random component $f(y;\mu)$ is typically an exponential family distribution

$$f(y;\theta,\phi) = \exp(\phi^{-1}(y\theta - b(\theta)) + c(y,\phi)), \qquad (1)$$

with natural parameter θ, sufficient statistics y/ϕ, log partition function $b(\theta)/\phi$ and a scale parameter $\phi > 0$.

Note that in the model (1) the mean of the responses $\mu = E_\theta[y]$ is related to the natural parameter θ by $\mu = b'(\theta)$. The link function g can be any strictly monotone differentiable function. In the following, however, we will consider only *canonical* link functions for which $g(\mu) = \eta = \theta$. We will thus use the parametrization $f(y;\eta,\phi)$.

From a technical perspective, an important property of this framework is that $\log f(y;\eta,\phi)$ is strictly concave in η. This follows from the fact that the one-dimensional sufficient statistics y/ϕ is necessarily *minimal*, which implies that the log partition function $b(\eta)/\phi$ is strictly convex, see (Brown, 1986; Wainwright et al., 2005).

The standard linear regression model is a special case derived from the normal distribution with $\phi = \sigma^2$, the identity link $\eta = \mu$ and $b(\eta) = (1/2)\eta^2$. Other popular models include logistic regression (binomial distribution), Poisson regression for count data and gamma- (or exponential-, Weibull-) models for cost- or survival analysis.

Given an i.i.d. data sample $\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, arranged as rows of the data matrix X, and a corresponding vector of responses $\boldsymbol{y} = (y_1,\ldots,y_n)^\top$, we will consider the problem of minimizing the negative log-likelihood

$$\begin{aligned} l(\boldsymbol{y},\boldsymbol{\eta},\boldsymbol{\phi}) &= -\sum_i \log f(y_i;\eta_i,\phi) \\ &= -\sum_i \phi^{-1}(y_i\eta_i - b(\eta_i)) + c(y_i,\phi). \end{aligned} \qquad (2)$$

We assume that the scale parameter is known, and for the sake of simplicity we assume $\phi = 1$. Since $\eta = x^T\beta$, the gradient of l can be viewed as a function in either $\boldsymbol{\eta}$ or $\boldsymbol{\beta}$:

$$\begin{aligned} \nabla_{\boldsymbol{\eta}} l(\boldsymbol{\eta}) &= -(\boldsymbol{y} - g^{-1}(\boldsymbol{\eta})), \\ \nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta}) &= -X^\top \nabla_{\boldsymbol{\eta}} l(\boldsymbol{\eta}) = -X^\top(\boldsymbol{y} - g^{-1}(X\boldsymbol{\beta})), \end{aligned} \qquad (3)$$

where $g^{-1}(\boldsymbol{\eta}) := (g^{-1}(\eta_1),\ldots,g^{-1}(\eta_n))^\top$. The corresponding Hessians are

$$H_{\boldsymbol{\eta}} = W, \quad H_{\boldsymbol{\beta}} = X^\top W X, \qquad (4)$$

where W is diagonal with elements $W_{ii} = (g^{-1})'(\eta_i) = 1/(g'(\mu_i)) = \mu'(\eta_i) = b''(\eta_i)$.

For the following derivation, it is convenient to partition X, $\boldsymbol{\beta}$ and $\boldsymbol{h} := \nabla_{\boldsymbol{\beta}} l$ into J subgroups: $X = (X_1,\ldots,X_j)$,

$$\boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_J \end{pmatrix}, \quad \boldsymbol{h} = \begin{pmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_J \end{pmatrix} = \begin{pmatrix} X_1^\top \nabla_{\boldsymbol{\eta}} l \\ \vdots \\ X_J^\top \nabla_{\boldsymbol{\eta}} l \end{pmatrix}. \qquad (5)$$

As stated above, b is strictly convex in $\theta = \eta$, thus $b''(\eta_i) > 0$ which in turn implies that $H_{\boldsymbol{\eta}} \succ 0$ and $H_{\boldsymbol{\beta}} \succeq 0$. This means that l is a strictly convex function in $\boldsymbol{\eta}$. For general matrices X it is convex in $\boldsymbol{\beta}$, and it is strictly convex in $\boldsymbol{\beta}$ if X has full rank and $d \leq n$.

Given X and $\boldsymbol{y}$, the Group-Lasso minimizes the negative log-likelihood viewed as a function in $\boldsymbol{\beta}$ under a constraint on the sum of the ℓ_2-norms of the subvectors $\boldsymbol{\beta}_j$:

$$\text{minimize } l(\boldsymbol{\beta}) \quad \text{s.t.} \quad g(\boldsymbol{\beta}) \geq 0, \qquad (6)$$

$$\text{where} \quad g(\boldsymbol{\beta}) = \kappa - \textstyle\sum_{i=1}^J \|\boldsymbol{\beta}_j\|. \qquad (7)$$

Here $g(\boldsymbol{\beta})$ is implicitly a function of the fixed parameter κ.

Considering the *unconstrained* problem, the solution is not unique if the dimensionality exceeds n: every $\boldsymbol{\beta}^* = \boldsymbol{\beta}^0 + \boldsymbol{\xi}$ with $\boldsymbol{\xi}$ being an element of the null space $\boldsymbol{N}(X)$ is also a solution. By defining the unique value

$$\kappa_0 := \min_{\boldsymbol{\xi}\in\boldsymbol{N}(X)} \textstyle\sum_{i=1}^J \|\boldsymbol{\beta}_j^0 + \boldsymbol{\xi}_j\|, \qquad (8)$$

we will require that the constraint is active i.e. $\kappa < \kappa_0$. Note that the minimum κ_0 is unique, even though there might exist several vectors $\boldsymbol{\xi} \in \boldsymbol{N}(X)$ which attain this minimum. Enforcing the constraint to be active is essential for the following characterization of solutions. Although it might be infeasible to ensure this activeness by computing κ_0 and selecting κ accordingly, practical algorithms will not suffer from this problem: given a solution, we can always check if the constraint was active. If this was not the case, then the uniqueness question reduces to checking if $d \leq n$ (if X has full rank). In this case the solutions are usually not sparse, because the feature selection mechanism has been switched off. To produce a sparse solution,

one can then try smaller κ-values until the constraint is active. In section 3 we propose a more elegant solution to this problem in the form of an algorithm that approximates the solution path, i.e. the evolution of the group norms when relaxing the constraint. This algorithm can be initialized with an arbitrarily small constraint value κ^0 which typically ensures that the constraint is active in the first optimization step. Activeness of the constraint in the following steps can then be monitored by observing the decay of the Lagrange parameter when increasing κ, cf. Eq. (14) below.

Under the assumption $l > -\infty$ a minimum of (6) is guaranteed to exist, since l is continuous and the region of feasible vectors β is compact. The assumption $l > -\infty$ simply means that the likelihood is finite ($f < +\infty$) for all parameter values θ which is usually satisfied for models of practical importance (see (Wedderburn, 1973) for a detailed discussion), and we will restrict our further analysis to models of this kind[1]. Since we assume that the constraint is active, any solution $\widehat{\boldsymbol{\beta}}$ will lie on the boundary of the constraint region. It is easily seen that $\sum_{j=1}^{J} \|\boldsymbol{\beta}_j\|$ is convex which implies that $g(\boldsymbol{\beta})$ is concave. Thus, the region of feasible values defined by $g(\boldsymbol{\beta}) \geq 0$ is convex. If $d \leq n$, the objective function l will be strictly convex if X has full rank, which additionally implies that the minimum is unique. In summary, we can state the following theorem:

Theorem 1. *If $\kappa < \kappa_0$ and X has maximum rank, then the following holds: (i) A solution $\widehat{\boldsymbol{\beta}}$ exists and $\sum_{i=1}^{J} \|\widehat{\boldsymbol{\beta}}_j\| = \kappa$ for any such solution. (ii) If $d \leq n$, the solution is unique.*

The Lagrangian for problem (6) reads

$$\mathcal{L}(\boldsymbol{\beta}, \lambda) = l(\boldsymbol{\beta}) - \lambda g(\boldsymbol{\beta}). \tag{9}$$

For a given $\lambda > 0$, $\mathcal{L}(\boldsymbol{\beta}, \lambda)$ is a convex function in $\boldsymbol{\beta}$. Under the assumption $l > -\infty$ a minimum is guaranteed to exist, since g goes to infinity if $\|\boldsymbol{\beta}\| \to \infty$.

The vector $\widehat{\boldsymbol{\beta}}$ minimizes $\mathcal{L}(\boldsymbol{\beta}, \lambda)$ iff the d-dimensional null-vector $\mathbf{0}_d$ is an element of the subdifferential $\partial_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}, \lambda)$. Let d_j denote the dimension of the j-th subvector $\boldsymbol{\beta}_j$ (i.e. the size of the j-th subgroup). The subdifferential is

$$\partial_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}, \lambda) = \nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta}) + \lambda \boldsymbol{v} = X^\top \nabla_{\boldsymbol{\eta}} l(\boldsymbol{\eta}) + \lambda \boldsymbol{v}, \tag{10}$$

with $\boldsymbol{v} = (\boldsymbol{v}_1, \dots \boldsymbol{v}_J)^\top$ defined by

$$\begin{aligned} \boldsymbol{v}_j &= \frac{\boldsymbol{\beta}_j}{\|\boldsymbol{\beta}_j\|}, \text{ if } \boldsymbol{\beta}_j \neq \mathbf{0}_{d_j} \text{ and} \\ \boldsymbol{v}_j &\in \{\boldsymbol{a} \in \mathbb{R}^{d_j} : \|\boldsymbol{a}\| \leq 1\}, \text{ else.} \end{aligned} \tag{11}$$

Thus, $\widehat{\boldsymbol{\beta}}$ is a minimizer for λ fixed iff

$$\mathbf{0}_d = X^\top \nabla_{\boldsymbol{\eta}} l(\boldsymbol{\eta})|_{\boldsymbol{\eta}=\widehat{\boldsymbol{\eta}}} + \lambda \boldsymbol{v} \qquad (\text{with } \widehat{\boldsymbol{\eta}} = X\widehat{\boldsymbol{\beta}}), \tag{12}$$

[1]Technically we require that the domain of l is $\mathbb{R}^d$, which implies that Slater's condition holds.

for some $\boldsymbol{v}$ of the form described above. Hence, for all j with $\widehat{\boldsymbol{\beta}}_j \neq \mathbf{0}_{d_j}$ it holds that

$$\|X_j^\top \nabla_{\boldsymbol{\eta}} l(\boldsymbol{\eta})|_{\boldsymbol{\eta}=\widehat{\boldsymbol{\eta}}}\| = \lambda. \tag{13}$$

For all other j with $\widehat{\boldsymbol{\beta}}_j = \mathbf{0}_{d_j}$ it holds that $\|X_j^\top \nabla_{\boldsymbol{\eta}} l(\boldsymbol{\eta})|_{\boldsymbol{\eta}=\widehat{\boldsymbol{\eta}}}\| \leq \lambda$ which implies

$$\lambda = \max_j \|X_j^\top \nabla_{\boldsymbol{\eta}} l(\boldsymbol{\eta})|_{\boldsymbol{\eta}=\widehat{\boldsymbol{\eta}}}\|. \tag{14}$$

Lemma 1. *Let $\widehat{\boldsymbol{\beta}}$ be a solution of (6). Let $\lambda = \lambda(\widehat{\boldsymbol{\beta}})$ be the associated Lagrangian multiplier. Then λ and $\widehat{\boldsymbol{h}} = \nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta})|_{\boldsymbol{\beta}=\widehat{\boldsymbol{\beta}}}$ are constant across all solutions $\widehat{\boldsymbol{\beta}}_{(i)}$ of (6).*

Proof. Since the value of the objective function $l(\boldsymbol{\eta}_{(i)}) = l_*$ is constant across all solutions and l is strictly convex in $\boldsymbol{\eta} = X\boldsymbol{\beta}$ and convex in $\boldsymbol{\beta}$, it follows that $\widehat{\boldsymbol{\eta}}$ must be constant across all solutions $\widehat{\boldsymbol{\beta}}_{(i)}$, which implies that $\nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta})|_{\boldsymbol{\beta}=\widehat{\boldsymbol{\beta}}} = X^\top \nabla_{\boldsymbol{\eta}} l(\boldsymbol{\eta})|_{\boldsymbol{\eta}=\widehat{\boldsymbol{\eta}}}$ is constant across all solutions. Uniqueness of λ follows now from (14). □

Theorem 2. *Let λ be the Lagrangian parameter associated with some (any) solution $\widehat{\boldsymbol{\beta}}$ of (6) and let $\widehat{\boldsymbol{h}}$ be the unique gradient vector at the optimum. Let $\mathcal{B} = \{j_1, \dots, j_p\}$ be the unique set of indices for which $\|\widehat{\boldsymbol{h}}_j\| = \lambda$. Then $\widehat{\boldsymbol{\beta}}_j = \mathbf{0}_{d_j}$ $\forall j \notin \mathcal{B}$ across all solutions $\widehat{\boldsymbol{\beta}}_{(i)}$ of (6).*

Proof. A solution with $\widehat{\boldsymbol{\beta}}_j \neq \mathbf{0}_{d_j}$ for at least one $j \notin \mathcal{B}$ would contradict (13). □

Assume that an algorithm has found a solution $\widehat{\boldsymbol{\beta}}$ of (6) with the set of "active" groups $\mathcal{A} := \{j : \widehat{\boldsymbol{\beta}}_j \neq \mathbf{0}\}$. If $\mathcal{A} = \mathcal{B} = \{j : \|\widehat{\boldsymbol{h}}_j\| = \lambda\}$, then there cannot exist any other solution with an active set $\mathcal{A}'$ with $|\mathcal{A}'| > |\mathcal{A}|$. Thus, $\mathcal{A} = \mathcal{B}$ implies that all relevant groups are contained in the solution $\widehat{\boldsymbol{\beta}}$. Otherwise, the additional elements in $\mathcal{B}$ which are not contained in $\mathcal{A}$ define all possible groups that potentially become active in alternative solutions.

Note that $\mathcal{A} = \mathcal{B}$ guarantees that we cannot have "overlooked" relevant groups, which is typically sufficient in practical applications. We will call such a solution *complete*. However, $\mathcal{A}$ might still contain redundant groups, and we might be additionally interested if we have found a *unique* (an thus minimal) set $\mathcal{A}$. The following theorem characterizes a simple test for uniqueness under a further rank assumption of the data matrix X.

Theorem 3. *Assume that every $n \times n$ submatrix of X has full rank. Let $\mathcal{A}$ be the active set corresponding to some solution $\widehat{\boldsymbol{\beta}}$ of (6) and let $X_{\mathcal{A}}$ be the $n \times s$ submatrix of X composed of all active groups. Assume further that $\mathcal{A}$ is complete, i.e. $\mathcal{A} = \mathcal{B}$. Then, if $s \leq n$, $\widehat{\boldsymbol{\beta}}$ is the unique solution of (6).*

Proof. Since the set $\mathcal{B}$ is unique, the assumption $\mathcal{A} = \mathcal{B}$ implies that the search for the optimal solution can be restricted to the space $\mathbb{S} = \mathbb{R}^s$. If $s \leq n$, $X_\mathcal{A}$ must have full rank by assumption. Thus, $l(\beta_\mathbb{S})$ is a strictly convex function on $\mathbb{S}$ which is minimized over the convex constraint set. Thus, $\widehat{\boldsymbol{\beta}}_\mathbb{S}$ is the unique minimizer on $\mathbb{S}$. Since all other $\widehat{\boldsymbol{\beta}}_{j:j\notin\mathcal{A}}$ must be zero, $\widehat{\boldsymbol{\beta}}$ is unique on the whole space. □

In practice, it might be difficult to guarantee the rank condition in the above theorem. Note, however, that for a given set $\mathcal{A}$ and associated matrix $X_\mathcal{A}$ it is sufficient to check if *rank*$(X_\mathcal{A}) = s$ via SVD or QR-decomposition.

3. An Efficient Active-Set Algorithm

The characterization of optimal solutions presented above is now used to build a highly efficient algorithm, which is a straight-forward generalization of the subset algorithm for the standard Lasso problem presented in (Osborne et al., 2000). Similar ideas for the standard Lasso have also been introduced in (Shevade & Keerthi, 2003). The algorithm starts with only one active group. The selection of further active groups (or their removal) is guided by observing Lagrangian violations. Testing for completeness of the active set will then identify all groups that could have nonzero coefficients in alternative solutions.

A: Initialize set $\mathcal{A} = \{j_0\}$, $\boldsymbol{\beta}_{j_0}$ arbitrary with $\|\boldsymbol{\beta}_{j_0}\| = \kappa$.
B: Optimize over the current active set $\mathcal{A}$. Define set $\mathcal{A}^+ = \{j \in \mathcal{A} : \|\boldsymbol{\beta}_j\| > 0\}$ (some $\boldsymbol{\beta}_j$ could have vanished during optimization). Define $\lambda = \max_{j\in\mathcal{A}^+} \|\boldsymbol{h}_j\|$. Adjust the active set $\mathcal{A} = \mathcal{A}^+$.
C: Lagrangian violation. $\forall j \notin \mathcal{A}$, check if $\|\boldsymbol{h}_j\| \leq \lambda$. If this is the case, we have found a global solution. Otherwise, include the group with the largest violation to $\mathcal{A}$ and go to **B**.
D: Completeness and uniqueness. $\forall j \notin \mathcal{A}$, check if $\|\boldsymbol{h}_j\| = \lambda$. If so, there might exist other solutions with identical costs that include these groups in the active set. Otherwise, the active set is *complete* in the sense that it contains all relevant groups. If X_a has full rank $s \leq n$, *uniqueness* can be checked additionally via theorem 3. Note that step **D** requires (almost) no additional computations, since it is a by-product of step **C**.

The above algorithm is easily extended to practical optimization routines in which we stop the fitting process at a predefined tolerance level: testing for "completeness within a ϵ-range" ($|\|\boldsymbol{h}_j\| - \lambda| < \epsilon$ in **D** with ϵ being the maximum deviation of gradient norms from λ in the active set) will then identify all potentially active groups in alternative solutions with costs close to the actual costs.

The minimization in step **B** can be performed efficiently by the projected gradient method introduced in (Kim et al., 2006), which is applicable for all continuous convex cost functions. Finding the projection is typically the computational bottleneck in methods of this kind. For our special case, however, the projection can be found very efficiently. We refer the reader to (Kim et al., 2006) for details.
Iterate:
B1: Gradient. At time $t-1$, set $\boldsymbol{b} = \boldsymbol{\beta}^{t-1} - s\nabla_{\boldsymbol{\beta}} l(\boldsymbol{\beta}^{t-1})$ and $\mathcal{A}^+ = \mathcal{A}$, where s is a step-size parameter.
B2: Projection. For all $j \in \mathcal{A}^+$ define $M_j := \|\boldsymbol{b}_j\| + (\kappa - \sum_j \|\boldsymbol{b}_j\|)/|\mathcal{A}^+|$. If $M_j \geq 0\, \forall j \in \mathcal{A}^+$, go to **B3**. Else update the active set $\mathcal{A}^+ = \{j : M_j > 0\}$ and repeat **B2**.
B3: New solution. For all $j \in \mathcal{A}^+$ set $\boldsymbol{\beta}_j^t = \boldsymbol{b}_j M_j / \|\boldsymbol{b}_j\|$. For all other $j \in \mathcal{A}, j \notin \mathcal{A}^+$ set $\boldsymbol{\beta}_j^t = 0$.

Note that during the whole algorithm, access to the full set of variables is only necessary in steps **C** and **D**, which are outside the core optimization routine. Thus, in large-scale applications where not all groups can be hold in the main memory, we still have a rather efficient method, even if we have to access external storage in steps **C/D**.

Computing the Solution Path. Contrary to the standard Lasso, the Group-Lasso does not exhibit a piecewise linear solution path. Algorithms like LARS (Efron et al., 2004) are therefore not applicable. Despite this problem, we can still approximate the solution path on a grid of constraint values with almost no additional costs: starting with a very small $\kappa^{(0)}$ (which will result in a small active set), we iteratively relax the constraint, resulting in a series of increasing values $\kappa^{(i)}$. Note that at the i-th step, the previous solution $\boldsymbol{\beta}(\kappa^{(i-1)})$ is a feasible initial estimate since $\kappa^{(i)} > \kappa^{(i-1)}$. Typically only few further iterations are needed to find $\boldsymbol{\beta}(\kappa^{(i)})$. Completeness/uniqueness can be tested efficiently at every step i. In practical applications we observed that the stepwise approximation of the solution path up to some final $\kappa^{(f)}$ is usually *faster* than directly computing the solution for $\kappa^{(f)}$, probably because the stepwise procedure allows the use of larger stepsizes.

4. Applications

As a first application example we use synthetic data generated by a script that has been used in the context of the NIPS'03 feature selection workshop (follow the link "dataset description" on the workshop webpage www.clopinet.com/isabelle/Projects/NIPS2003/#challenge). We reproduced the XOR example explained in the above cited document: there are two classes, each of which is composed of two Gaussian clusters. Two "useful" features are drawn from $N(0,1)$ for each cluster. Some covariance is added by multiplying by a random matrix. The clusters are placed in an XOR configuration and 3200 "useless" features are added, drawn from $N(0,1)$. All the features are shifted and rescaled randomly. Random noise is then added according to $N(0, 0.1)$. Finally, 1% of the labels are ran-

domly flipped. We construct a training set of size 2000 and a test set of size 6000.

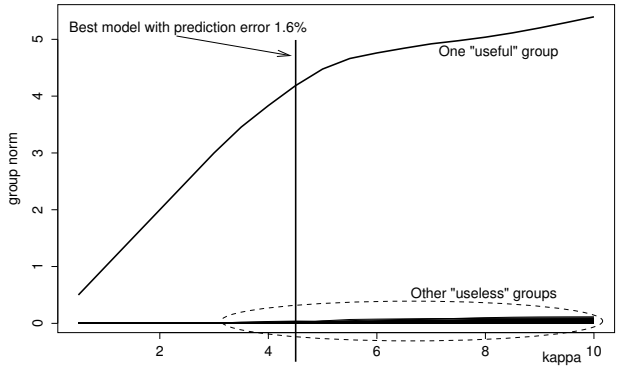


Figure 1. Solution path for the XOR problem with 3200 noise dimensions. The norm of the one "useful" group grows steeply when the constraint is relaxed. What appears as a horizontal "thickening" line is an overlay of 275 "useless" groups.

Without feature selection, prediction becomes very difficult: a SVM with RBF kernel achieves 42% test error (on the subset of the two "useful" features the error decreases to 1.5%). Moreover, simple feature selection methods like correlation-based scoring fail badly on these data.

We expand the dataset in a polynomial basis of degree 2, i.e. each pair of features (a, b) is mapped to a 5-dimensional vector $(a, b, a \cdot b, a^2, b^2)$. Given the 3202 features (2 + 3200 "useless"), this expansion yields $\approx 5 \cdot 10^6$ groups of size 5, each of which contains 5 quadratic interactions. We are, thus, working in a $\approx 2.5 \cdot 10^7$-dimensional space. Since the expanded feature set cannot be hold in the main memory, we only store the original dataset and recompute the expansions on demand. Despite this computational overhead, our active set algorithm allows us to optimize the Group-Lasso functional very efficiently, see also Figure 2. Since we are dealing with a classification problem, we choose the logistic model from the GLM family. Figure 1 shows the solution path for the logistic Group-Lasso when relaxing κ in 20 steps. Note that in the first iterations the algorithm was able to determine the one "useful" group of variables. The norm of the corresponding weight vector increases almost linearly until $\kappa \approx 4.5$, where the minimum error rate of 1.6% on the test set is obtained.

Testing both the completeness and the uniqueness of the active set gives a positive result, which guarantees that at this constraint value there are no alternative solutions. Further increasing κ leads to the selection of additional groups with spurious weights. The model obtained for $\kappa = 10$ uses 275 groups which include "useless" features and have norms < 0.2. Solutions for $\kappa > 5$ appear to be lacking completeness: our test identified a steeply increasing number of other groups that may also become active. Given that the "useless" variables are randomly drawn from a normal distribution, the observed lack of completeness might be caused by the limited numerical precision in the optimization routine: for models with $\kappa < 7$ we could indeed show by increasing the numerical precision that the solutions are complete, however at the price of drastically increasing computational costs. For larger models, however, we were not able to find complete solutions within any reasonable time limits. This result nicely shows that lacking completeness of Group-Lasso solutions is indeed a relevant issue in real-world applications which are necessarily computed with limited numerical precision. Besides the theoretical properties of our completeness test, this test might thus be also a valuable *practical* tool to detect possible ambiguities that are caused by numerical problems.

To compare the efficiency of our active set algorithm with related approaches, we measured the time needed to compute ten steps of the solution path ($\kappa = 1, 2, \ldots, 10$) for different numbers of "useless" groups. Figure 2 shows the observed computation times of three different methods: (1): the *blockwise sparse* method (Kim et al., 2006), (2): the *block coordinate* method by (Meier et al., 2008), (3): our algorithm. The comparison with method 1 was straight forward, since the same implementation was used (note that by dropping the active set selection mechanism, our method simply reduces to method (1)). In order to guarantee a fair comparison with method (2) for which we used the R-package `grplasso`, a few modifications were necessary: we first trained our method on the data and recorded the sequence of Lagrange parameters $\lambda_1, \ldots, \lambda_{10}$ corresponding to the sequence of constraints $\kappa = 1, 2, \ldots, 10$, since the `grplasso` package needs the Lagrange parameters on input. We also recorded the achieved log-likelihood at each step. We then trained method (2) on the dataset and adjusted its tolerance parameters as to (roughly) reproduce the recorded sequence of log-likelihoods. The double logarithmic scale in Figure 2 should make the interpretation of the plot rather insensitive against performance differences caused by using different implementations, since such differences are expected to produce additive shifts without changing the slopes.

For input instances that could be hold in the main memory, the log-log plot shows a relatively steep increase for the models (1) and (2), whereas method (3) increases linearly with a moderate slope. For the "out-of-core" models, we recomputed the groups whenever necessary (step **C/D** in our algorithm). We again see an almost linear increase of costs up to models including $\approx 10^6$ groups. Three observations seem to be important: (i) the slope of the curve for method (3) in the "out-of-core" regime does not even exceed the slope of the corresponding curve for method (2) at the end of the "cached" region; (ii) when fixing the costs at the level of method (2) at the end of the "cached" region, method (3) was able to solve instances which are larger by

at least $1 - 1.5$ orders of magnitude; (iii) comparison with method (1) shows that the active set formalism leads to a speed-up of several orders of magnitudes.

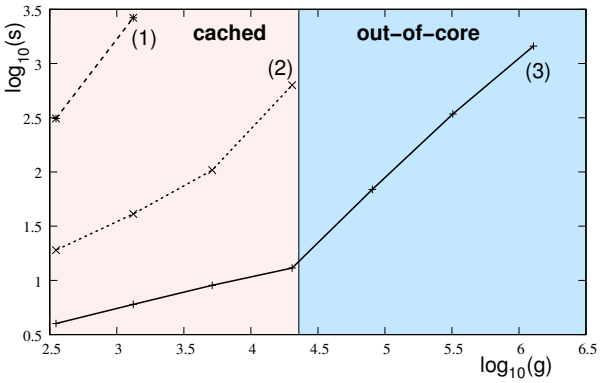


Figure 2. Log-log plot of computation time (y-axis, in seconds) for the XOR problem with logistic loss as a function of the number g of groups (x-axis). The three different methods are: 1:(Kim et al., 2006), 2:(Meier et al., 2008), 3: our algorithm.

Splice Site Detection. The prediction of splice sites has an important role in gene finding algorithms. Splice sites are the regions between coding (exons) and non-coding (introns) DNA segments. The $5'$ end of an intron is called a donor splice site and the $3'$ end an acceptor splice site. The *MEMset Donor* dataset (http://genes.mit.edu/burgelab/maxent/ssdata/) consists of a training set of 8415 true and 179438 false human donor sites. An additional test set contains 4208 true and 89717 "false" (or *decoy*) donor sites. A sequence of a real splice site is modeled within a window that consists of the last 3 bases of the exon and the first 6 bases of the intron. Decoy splice sites also match the consensus sequence at position zero and one. Removing this consensus "GT" results in sequences of length 7, i.e. sequences of 7 factors with 4 levels $\{A, C, G, T\}$, see (Yeo & Burge, 2004) for details. The goal of this experiment is to overcome the restriction to marginal probabilities (main effects) in the widely used *Sequence-Logo* approach (see Figure 4) by exploring all possible interactions up to order 4.

Following (Meier et al., 2008), the original training dataset is used to build a balanced training dataset and an unbalanced validation set which exhibits the same true/false ratio as the test set. The data are represented as a collection all factor interactions up to degree 4. Every interaction is encoded using dummy variables and treated as a group, leading to 120 groups of sizes varying between 4 (main effects) and 4^5 (4th order interactions). In total, we are working in a 33068-dimensional feature space. This dataset has also been analyzed in (Meier et al., 2008) with the Group-Lasso, but only up to 2nd order interactions.

To correct for the unbalancedness of the classes, the validation set is used to choose the best threshold τ on the classifier output. It is further used to select κ. The performance is measured in terms of the maximum correlation coefficient $\rho_{\max}$ between predicted and true labels.

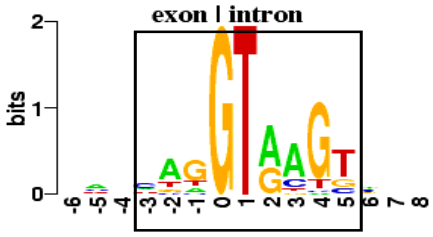


Figure 3. Sequence Logo representation of the human $5'$ splice site. The consensus "GT" appears at positions $0, 1$. The overall height of the stack of symbols at a certain position indicates the sequence conservation at that position, while the height of symbols within the stack indicates the relative frequency of each nucleic acid, see (Crooks et al., 2004). We model the splice sites in a window over positions $[-3, 5]$.

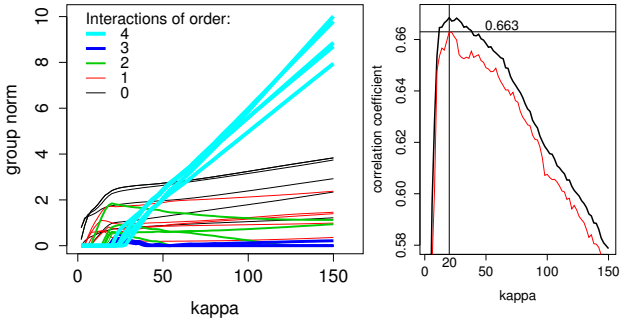


Figure 4. Left: solution path for donor splice site prediction. Color and thickness of curves indicate different orders of interactions. Right: Correlation coefficient as a function of κ. Bold curve: correlation on the validation set that is used for model selection (the thin vertical line indicates the chosen model). Thin curve: correlation on the separate test set.

From the correlation curve in Figure 4 we conclude that the inclusion of interactions of order three and greater does not improve the predictive performance and produces some pronounced overfitting effects. The model with the highest correlation coefficient ($\kappa = 20$) contains 36 groups: all 7 main effects, 21 1st-order interactions and 8 2nd-order interactions. Among the top-scoring groups we find the main effects at positions -1, 2 and 4, the interactions at positions $(4 : 5)$, $(-2 : -1)$ and $(2 : 3)$ and the triplet $(-3 : -2 : -1)$, which all share the property that they exclusively contain exon positions (or intron positions, respectively). One might conclude that long-range interactions between the preceding exon and the starting intron are of minor importance for splice site recognition. The completeness test reveals, however, that the solution with

36 groups is not complete, and that a complete model for $\kappa = 20$ additionally contains the four interactions $(-1:4)$, $(-2:5)$, $(-1:3:4)$ and $(-3:-1:2:5)$, all of which combine exon and intron positions. This is a nice example where the completeness test gives rise to query an initial hypothesis (about the weak exon-intron dependencies) which seems to be plausible from observing the Group-Lasso solution. It should be noticed that the obtained correlation coefficient of $\rho_{\max} = 0.663$ compares favorably with the result in the original paper (Yeo & Burge, 2004) ($\rho_{\max} = 0.659$), which has been viewed as among the best methods for short motive modeling.

The next experiment shows a situation where the completeness test indicates that the interpretability of the Group-Lasso might be generally complicated if relatively complex models are required. The problem is again the discrimination between true and "false" splice sites, this time, however, at the $3'$ end. Compared to the $5'$ situation, $3'$ (acceptor) splice site motives are less concentrated around the consensus nucleotide pair ("AG" at positions -2,-1 in Figure 5), which requires the use of larger windows. We trained the logistic Group-Lasso model on all interactions up to order 4 using windows of length 21. In total, we have 27896 groups which span a $22,458,100$-dimensional feature space. Despite this huge dimensionality, our active set algorithm was able to compute the solution path up to $\kappa = 150$ within roughly 20 hours. From the correlation curve in Figure 6 we conclude that in this example, the inclusion of 3rd- and 4th-order interactions does indeed increase the predictive performance. The optimal model at $\kappa = 66$ contains 386 groups. Among the 10 highest-scoring groups are the main effects at positions -3, -5 and 0, the 1st-order interactions $(-9:-8)$, $(-11:-10)$, $(-11:-9)$ and $(-12:-11)$, the triplet $(-6:-5:-3)$, the 3rd-order interaction $(-14:-9:0:1)$ and the 4th-order interaction $(-10:-8:-6:-3:2)$. The latter might be of particular interest, since it couples the position 2 which appears to be non-informative in the Sequence-Logo representation (Fig. 5) with positions at the end of the intron. This observation nicely emphasizes the strength of a model that is capable of exploring high-order dependencies among the positions.

A closer look at the results of the completeness tests in Figure 7 shows, however, that probably all solutions with $\kappa > 40$ are rather difficult to interpret, since a steeply increasing number of groups must be added to obtain complete models. This means that care should be taken when it comes to interpreting specific groups occurring in particular solutions (as we have done above). Since most of the models are not complete, it might well be that other groups not contained in a particular solution might be of high importance or even "substitute" identified groups. For the 4th-order interaction $(-10:-8:-6:-3:2)$ in the optimal solution with $\kappa = 66$ it might well be that there exist other groups that can take over the role of this interaction. Even though the high score of this group might indicate that a complete substitution is not very likely, the "discovery" of the coupling between position 2 and intron positions should not be accepted unquestioningly.

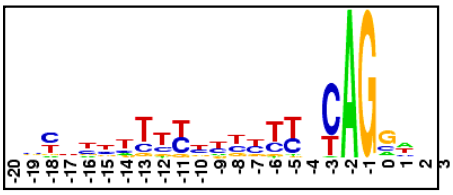


Figure 5. Sequence Logo representation of the human $3'$ splice site. The consensus "AG" appears at positions $-2, -1$. We use a window over positions $[-20, 2]$.

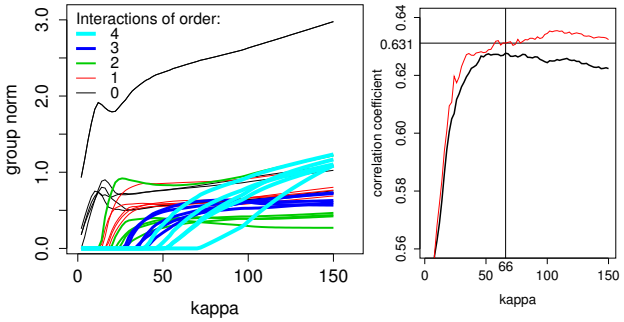


Figure 6. Left: solution path for $3'$ splice site prediction. The upper most curve represents the most important position -3 (last position of the intron). Right: Correlation coefficient as a function of κ. Bold curve: correlation on the validation set that is used for model selection. Thin curve: correlation on the separate test set.

5. Conclusion

The completeness- and uniqueness test presented here overcomes a severe problem of the Group-Lasso estimator for generalized linear models (GLM). Since in many practical applications the dimensionality exceeds the sample size, we cannot *a priori* assume that the active set of groups is unique, which somehow contradicts our goal of identifying important factors. Our testing procedure has the advantage that it identifies all groups that are potential candidates for the active set. Even if a solution is not complete, this latter property still allows us to explicitly list (and potentially investigate) the set of all *candidate* groups.

We have presented a highly efficient active-set algorithm that can handle extremely high-dimensional input spaces which typically arise when investigating high-order factor interactions or when using polynomial basis expansions. Our theoretical characterization of solutions is used to check both optimality and completeness/uniqueness.

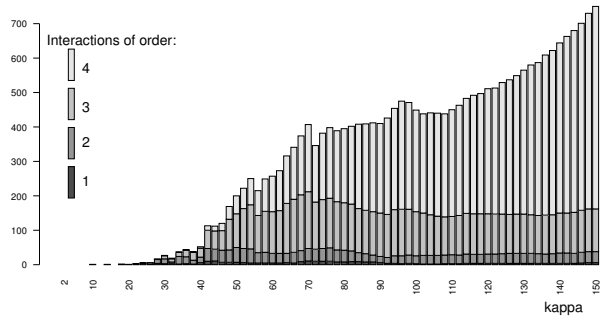


Figure 7. Acceptor splice site prediction: groups that must be included in the logistic Group-Lasso estimates to obtain complete models (gray values represent different orders of interactions).

The experiment on synthetic data in XOR configuration with additional noise features showed that the methods and concepts presented here can be successfully applied to problems with millions of groups. We demonstrated that non-completeness of solutions is indeed an important issue in real-world applications where round-off errors are unavoidable. Without any additional computational costs, the proposed completeness/uniqueness test easily detects such situations and additionally identifies all groups that must be included to achieve a complete model.

The splice-site prediction example confirmed these observations in a real-world context, where the inclusion of high-order factor interactions helps to increases the predictive performance but also leads to incomplete and, thus, potentially ambiguous solutions. The active set algorithm was able to approximate the solution path of the logistic Group-Lasso for feature-space dimensions up to $\approx 2 \cdot 10^7$ within a reasonable time, and the completeness test helped to avoid mis- or over-interpretations of identified interactions between the nucleotide positions. In particular for the $5'$ (donor-) splicing sites, we could show that the completeness test avoids a potentially severe misinterpretation regarding the independence of exon and intron positions.

While in the application examples we have focused on logistic classification problems, both the characterization of solutions and the algorithms proposed are valid for the much richer class of GLMs. Notable extensions include models for counting processes (e.g. Poisson or log-linear models). Details of such models for the analysis of sparse contingency tables in the spirit of the work in (Dahinden et al., 2007) will appear elsewhere. A C++ implementation of the active set algorithm with completeness test is available from the authors on request.

In the broad perspective – and in the light of recent theoretical results on the algorithmic complexity of feature selection (Nilsson et al., 2007) – one might conclude that feature selection can be simpler than previously thought.

References

Brown, L. D. (1986). *Fundamentals of statistical exponential families: with applications in statistical decision theory*. Hayworth, CA, USA: Institute of Mathematical Statistics.

Crooks, G., Hon, G., Chandonia, J., & Brenner, S. (2004). Weblogo: A sequence logo generator. *Genome Research*, *14*.

Dahinden, C., Parmigiani, G., Emerick, M., & Bühlmann, P. (2007). Penalized likelihood for sparse contingency tables with an application to full-length cDNA libraries. *BMC Bioinformatics*, *8*, 476.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Ann. Stat.*, *32*, 407–499.

Kim, Y., Kim, J., & Kim, Y. (2006). Blockwise sparse regression. *Statistica Sinica*, *16*, 375–390.

McCullaghand, P., & Nelder, J. (1983). *Generalized linear models*. Chapman & Hall.

Meier, L., van de Geer, S., & Bühlmann, P. (2008). The Group Lasso for Logistic Regression. *J. Roy. Stat. Soc. B*, *70*, 53–71.

Nilsson, R., Peña, J., Björkegren, J., & Tegnér, J. (2007). Consistent feature selection for pattern recognition in polynomial time. *JMLR*, *8*, 589–612.

Osborne, M., Presnell, B., & Turlach, B. (2000). On the LASSO and its dual. *J. Comp. and Graphical Statistics*, *9*, 319–337.

Shevade, K., & Keerthi, S. (2003). A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, *19*, 2246–2253.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *J. Roy. Stat. Soc. B*, *58*, 267–288.

Wainwright, M., Jaakkola, T., & Willsky, A. (2005). A new class of upper bounds on the log partition function. *IEEE Trans. Information Theory*, *51*.

Wedderburn, R. W. M. (1973). On the existence and uniqueness of the maximum likelihood estimates for certain generalized linear models. *Biometrika*, *63*, 27–32.

Yeo, G., & Burge, C. (2004). Maximum entropy modeling of short sequence motifs with applications to RNA splicing signals. *J. Comp. Biology*, *11*, 377–394.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *J. Roy. Stat. Soc. B*, 49–67.

Robust Matching and Recognition using Context-Dependent Kernels

Hichem Sahbi[1] SAHBI@TELECOM-PARISTECH.FR
Jean-Yves Audibert[2,3] AUDIBERT@CERTIS.ENPC.FR
Jaonary Rabarisoa[2] **and Renaud Keriven**[2] {RABARISO,KERIVEN}@CERTIS.ENPC.FR
[1] CNRS - LTCI, UMR 5141, Telecom ParisTech, 46 rue Barrault, 75013 Paris, France.
[2] Certis - Ecole des Ponts, 6 avenue Blaise Pascal, Cité Descartes, 77455 Marne-la-Vallée, France.
[3] Willow - ENS / INRIA, 45 rue d'Ulm, 75005 Paris, France.

Abstract

The success of kernel methods including support vector machines (SVMs) strongly depends on the design of appropriate kernels. While initially kernels were designed in order to handle fixed-length data, their extension to unordered, variable-length data became more than necessary for real pattern recognition problems such as object recognition and bioinformatics.

We focus in this paper on object recognition using a new type of kernel referred to as "context-dependent". Objects, seen as constellations of local features (interest points, regions, etc.), are matched by minimizing an energy function mixing (1) a fidelity term which measures the quality of feature matching, (2) a neighborhood criterion which captures the object geometry and (3) a regularization term. We will show that the fixed-point of this energy is a "context-dependent" kernel ("CDK") which also satisfies the Mercer condition. Experiments conducted on object recognition show that when plugging our kernel in SVMs, we clearly outperform SVMs with "context-free" kernels.

1. Introduction

Object recognition is one of the biggest challenges in vision and its interest is still growing (Everingham et al., 2007). Among existing methods, those based on machine learning (ML), show a particular interest as they are performant and theoretically well grounded (Bishop, 2007). ML approaches, such as the popular support vector networks (Boser et al., 1992), basically require the design of similarity measures, also referred to as *kernels*, which should provide high values when two objects share similar structures/appearances and should be invariant, as much as possible, to the linear and non-linear transformations. Kernel-based object recognition methods were initially *holistic*, i.e., each object is mapped into one or multiple fixed-length vectors and a similarity, based on color, texture or shape (Swain & Ballard, 1991; Chapelle et al., 1999), is then defined. *Local* kernels, i.e., those based on bags or local sets were introduced in order to represent data which cannot be represented by ordered and fixed-length feature vectors, such as graphs, trees, interest points, etc (Gartner, 2003). It is well known that both holistic and local kernels should satisfy certain properties among them the positive definiteness, low complexity for evaluation, flexibility in order to handle variable-length data and also invariance. Holistic kernels have the advantage of being simple to evaluate, discriminating but less flexible than local kernels in order to handle invariance[1]. While the design of kernels gathering flexibility, invariance and low complexity is a challenging task; the proof of their positive definiteness is sometimes harder (Cuturi, 2005). This property also known as the Mercer condition ensures, according to Vapnik's SVM theory (Vapnik, 1998), optimal generalization performance and also the uniqueness of the SVM solution.

Consider a database of objects (images), each one seen as a constellation of local features, for instance interest points (Schmid & Mohr, 1997; Lowe, 2004; Lazebnik et al., 2004), extracted using any suitable filter (Harris & Stephens, 1988). Again, original

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] In case of object recognition, invariance means robustness to occlusion, geometric transformations and illumination.

holistic kernels explicitly (or implicitly) map objects into fixed-length feature vectors and take the similarity as a decreasing function of any well-defined distance (Barla et al., 2002). In contrast to holistic kernels, local ones are designed in order to handle variable-length and unordered data. Two families of local kernels can be found in the literature; those based on statistical "length-insensitive" measures such as the Kullback Leibler divergence, and those which require a preliminary step of alignment. In the first family, the authors in (Kondor & Jebara, 2003; Moreno et al., 2003) estimate for each object (constellation of local features) a probability distribution and compute the similarity between two objects (two distributions) using the "Kullback Leibler divergence" in (Moreno et al., 2003) and the "Bhattacharyya affinity" in (Kondor & Jebara, 2003). Only the function in (Kondor & Jebara, 2003) satisfies the Mercer condition and both kernels were applied for image recognition tasks. In (Wolf & Shashua, 2003), the authors discuss a new type of kernel referred to as "principal angles" which is positive definite. Its definition is based on the computation of the principal angles between two linear subspaces under an orthogonality constraint. The authors demonstrate the validity of their method on visual recognition tasks including classification of motion trajectory and face recognition. An extension to subsets of varying cardinality is proposed in (Shashua & Hazan, 2004). In this first family of kernels, the main drawback, in some methods, resides is the strong assumption about the used probabilistic models in order to approximate the set of local features which may not hold true in practice.

In the second family, the "max" kernel (Wallraven et al., 2003) considers the similarity function, between two feature sets, as the sum of their matching scores and unlike discussed in (Wallraven et al., 2003) this kernel is actually not Mercer (Bahlmann et al., 2002). In (Lyu, 2005), the authors introduced the "circular-shift" kernel defined as a weighted combination of Mercer kernels using an exponent. The latter is chosen in order to give more prominence to the largest terms so the resulting similarity function approximates the "max" and also satisfies the Mercer condition. The authors combined local features and their relative angles in order to make their kernel rotation invariant and they show its performance for the particular task of object recognition. In (Boughorbel, 2005), the authors introduced the "intermediate" matching kernel, for object recognition, which uses virtual local features in order to approximate the "max" while satisfying the Mercer condition. Recently, (Grauman & Darrell, 2007) introduced the "pyramid-match" kernel, for object recognition and document analysis, which maps feature sets using a multi-resolution histogram representation and computes the similarity using a weighted histogram intersection. The authors showed that their function is positive definite and can be computed linearly with respect to the number of local features. Other matching kernels include the "dynamic programming" function which provides, in (Bahlmann et al., 2002), an effective matching strategy for handwritten character recognition, nevertheless the Mercer condition is not guaranteed.

Naive matching	**'H'**	**'i'**		**'S'**	**'i'**	**'r'**
'S'	0	0	-	1	0	0
'i'	0	1	-	0	1	0
'r'	0	0	-	0	0	1
Context-dependent	-	-	-	-	-	-
'S'	0	0	-	.38	0	0
'i'	0	.36	-	0	.39	0
'r'	0	0	-	0	0	.38

Table 1. This table shows a simple comparison between similarity measures when using naive matching (upper table) and context-dependent matching (lower table).

1.1. Motivation and Contribution

The success of the second family of local kernels strongly depends on the quality of alignments which are difficult to obtain mainly when images contain redundant and repeatable structures. Regardless the Mercer condition, a naive matching kernel (such as the "max"), which looks for all the possible alignments and sums the best ones, will certainly fail and results into many false matches (see Figure 1, left). The same argument is supported in (Schmid & Mohr, 1997), for the general problem of visual features matching, about the strong spatial correlation between interest points and the corresponding close local features in the image space. This limitation also appears in closely related areas such as text analysis, and particularly string alignment. A simple example, of aligning two strings ("Sir" and "Hi Sir") using a simple similarity measure $\mathbb{1}_{\{c_1=c_2\}}$ between any two characters c_1 and c_2, shows that without any extra information about the *context* (i.e., the sub-string) surrounding each character in ("Sir" and "Hi Sir"), the alignment process results into false matches (See Table 1). *Hence, it is necessary to consider the context as a part of the alignment process when designing kernels.*

In this paper, we introduce a new kernel, called "context-dependent" (or "CDK") and defined as the

fixed-point of an energy function which balances an "alignment quality" term and a "neighborhood" criterion. The alignment quality is inversely proportional to the expectation of the Euclidean distance between the most likely aligned features (see Section 2) while the neighborhood criterion measures the spatial coherence of the alignments; given a pair of features (f_p, f_q) with a high alignment quality, the neighborhood criterion is proportional to the alignment quality of all the pairs close to (f_p, f_q). *The general form of "CDK" captures the similarity between any two features by incorporating also their context, i.e., the similarity of the surrounding features.* Our proposed kernel can be viewed as a variant of "dynamic programming" kernel (Bahlmann et al., 2002) where instead of using the ordering assumption we consider a neighborhood assumption which states that two points match if they have similar features and if they satisfies a neighborhood criterion i.e., their neighbors match too. This also appears in other well studied kernels such as Fisher (Jaakkola et al., 1999), which implements the conditional dependency between data using the Markov assumption. "CDK" also implements such dependency with an extra advantage of being the fixed-point and the (sub)optimal solution of an energy function closely related to the goal of our application. This goal is to gather the properties of flexibility, invariance and mainly discrimination by allowing each local feature to consider its context in the matching process. Notice that the goal of this paper is not to extend local features to be global and doing so (as in (Mortensen et al., 2005; Amores et al., 2005)) makes local features less invariant, but rather to design a similarity kernel ("CDK") which captures the context while being invariant. Even though we investigate "CDK" in the particular task of object recognition, we can easily extend it to handle closely related areas in machine learning such as text alignment for document retrieval (Nie et al., 1999), machine translation (Sim et al., 2007) and bioinformatics (Scholkopf et al., 2004).

In the remainder of this paper we consider the following terminology and notation. A feature refers to a local interest point $x_i^p = (\ _g(x_i^p),\ _f(x_i^p), y_p)$, here i stands for the i^{th} sample of the subset $\mathcal{S}_p = \{x_1^p, \ldots, x_n^p\}$ and $y_p \in \mathbb{N}^+$ is a unique indicator which provides the class or the subset including x_i^p. $_g(x_i^p) \in \mathbb{R}^2$ stands for the $2D$ coordinates of the interest-point x_i^p while $_f(x_i^p) \in \mathbb{R}^s$ corresponds to the descriptor of x_i^p (for instance the 128 coefficients of the SIFT(Lowe, 2004)). We define $\mathcal{X}$ as the set of all possible features taken from all the possible images in the world and X is a random variable standing for a sample in $\mathcal{X}$. We also consider $k_t : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ as a symmetric function which, given two samples (x_i^p, x_j^q), provides a similarity measure. Other notations will be introduced as we go along through different sections of this paper which is organized as follows. We first introduce in Section 2, our energy function which makes it possible to design our context-dependent kernel and we show that this kernel satisfies the Mercer condition so we can use it for support vector machine training and other kernel methods. In Section 3 we show the application of this kernel in object recognition. We discuss in Section 4 the advantages and weaknesses of this kernel and the possible extensions in order to handle other tasks such as string matching and machine translation. We conclude in Section 5 and we provide some future research directions.

2. Kernel Design

Define $\mathcal{X} = \cup_{p \in \mathbb{N}^+} \mathcal{S}_p$ as the set of all possible interest points taken from all the possible objects in the world. We assume that all the objects are sampled with a given cardinality i.e., $|\mathcal{S}_p| = n$, $|\mathcal{S}_q| = m$, $\forall\ p, q \in \mathbb{N}^+$ (n and m might be different). Our goal is to design a kernel K which provides the similarity between any two objects (subsets) $\mathcal{S}_p$, $\mathcal{S}_q$ in $\mathcal{X}$.

Definition 1 (Subset Kernels) *let $\mathcal{X}$ be an input space, and consider $\mathcal{S}_p, \mathcal{S}_q \subseteq \mathcal{X}$ as two finite subsets of $\mathcal{X}$. We define the similarity function or kernel K between $\mathcal{S}_p = \{x_i^p\}$ and $\mathcal{S}_q = \{x_j^q\}$ as $K(\mathcal{S}_p, \mathcal{S}_q) = \sum_i^n \sum_j^m k\left(x_i^p, x_j^q\right)$.*

here k is symmetric and continuous on $\mathcal{X} \times \mathcal{X}$, so K will also be continuous and symmetric. Since K is defined as the cross-similarity k between all the possible sample pairs taken from $\mathcal{S}_p \times \mathcal{S}_q$, it is obvious that K has the big advantage of not requiring any (hard) alignment between the samples of $\mathcal{S}_p$ and $\mathcal{S}_q$. Nevertheless, for a given $\mathcal{S}_p$, $\mathcal{S}_q$, the value of $K(\mathcal{S}_p, \mathcal{S}_q)$ should be dominated by $\sum_i \max_j k\left(x_i^p, x_j^q\right)$, so k should be appropriately designed (see Section 2.1).

Let X be a random variable standing for samples taken from $\mathcal{S}_p$ and X' is defined in a similar way for the subset $\mathcal{S}_q$. We design our kernel $k(x_i^p, x_j^q) = \mathbb{P}(X' = x_j^q, X = x_i^p)$ as the *joint probability* that x_j^q matches x_i^p. Again, it is clear enough (see Figure 1 and Table 1) that when this joint probability is estimated using only the sample coordinates (without their contexts), this may result in many false matches and wrong estimate of $\left\{\mathbb{P}(X' = x_j^q, X = x_i^p)\right\}_{i,j}$.

Before describing the whole design of k, we start with our definition of context-dependent kernels.

Definition 2 (Context-Dependent Kernels) *we define a context-dependent kernel k as any symmetric, continuous and recursive function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ such that $k(x_i^p, x_j^q)$ is equal to*

$$c(x_i^p, x_j^q) \times h\left(\sum_{k,\ell} k(x_k^p, x_\ell^q) \ \mathbb{V}\left(x_i^p, x_k^p, x_j^q, x_\ell^q\right)\right),$$

here c is a positive (semi) definite and context-free (non-recursive) kernel, $\mathbb{V}(x, x', y, y')$ is a monotonic decreasing function of any (pseudo) distance involving (x, x', y, y') and $h(x)$ is monotonically increasing.

2.1. Approach

We consider the issue of designing k using a variational framework. Let $\mathcal{I}_p = \{1, \ldots, n\}$, $\mathcal{I}_q = \{1, \ldots, m\}$, $\mu = \{k(x_i^p, x_j^q)\}$, $d(x_i^p, x_j^q) = \| \ {}_f(x_i^p) - \ {}_f(x_j^q)\|_2$ and $\mathcal{N}_p(x_i^p) = \{x_k^p \in \mathcal{S}_p : k \neq i, \ \| \ {}_g(x_i^p) - \ {}_g(x_k^p)\|_2 \leq \epsilon_p\}$ (ϵ_p defines a neighborhood and $\mathcal{N}_q$ is defined in the same way for $\mathcal{S}_q$). Consider $\alpha, \beta \geq 0$, $(i, j) \in \mathcal{I}_p \times \mathcal{I}_q$, $\mu = \{k(x_i^p, x_j^q)\}$ is found by solving

$$\begin{aligned}
\min_{\mu} \quad & \sum_{i \in \mathcal{I}_p, j \in \mathcal{I}_q} k(x_i^p, x_j^q) \ d(x_i^p, x_j^q) + \\
& \beta \sum_{i \in \mathcal{I}_p, j \in \mathcal{I}_q} k(x_i^p, x_j^q) \ \log(k(x_i^p, x_j^q)) + \\
\alpha \quad & \sum_{i \in \mathcal{I}_p, j \in \mathcal{I}_q} k(x_i^p, x_j^q) \left(- \sum_{\substack{x_k^p \in \mathcal{N}_p(x_i^p), \\ x_\ell^q \in \mathcal{N}_q(x_j^q)}} k(x_k^p, x_\ell^q) \right) \\
\text{s.t.} \quad & k(x_i^p, x_j^q) \in [0, 1], \quad \sum_{i,j} k(x_i^p, x_j^q) = 1
\end{aligned} \tag{1}$$

The first term measures the quality of matching two descriptors ${}_f(x_i^p)$, ${}_f(x_j^q)$. In the case of SIFT, this is considered as the distance, $d(x_i^p, x_j^q)$, between the 128 SIFT coefficients of x_i^p and x_j^q. A high value of $d(x_i^p, x_j^q)$ should result into a small value of $k(x_i^p, x_j^q)$ and vice-versa.

The second term is a regularization criterion which considers that without any a priori knowledge about the aligned samples, the probability distribution $\{k(x_i^p, x_j^q)\}$ should be flat so the negative of the entropy is minimized. This term also helps defining a simple solution and solving the constrained minimization problem easily. The third term is a neighborhood criterion which considers that a high value of $k(x_i^p, x_j^q)$ should imply high kernel values in the neighborhoods $\mathcal{N}_p(x_i^p)$ and $\mathcal{N}_q(x_j^q)$. This criterion makes it possible to consider the context (spatial configuration) of each sample in the matching process.

We formulate the minimization problem by adding an equality constraint and bounds which ensure that $\{k(x_i^p, x_j^q)\}$ is a probability distribution.

Proposition 1 *(1) admits a solution in the form of a context-dependent kernel $k_t(x_i^p, x_j^q) = v_t(x_i^p, x_j^q)/Z_t$, with $t \in \mathbb{N}^+$, $Z_t = \sum_{i,j} v_t(x_i^p, x_j^q)$ and $v_t(x_i^p, x_j^q)$ defined as*

$$\exp\left(-\frac{d(x_i^p, x_j^q)}{\beta} - 1\right) \times \exp\left(\frac{2\alpha}{\beta} \sum_{k,\ell} \mathbb{V}(x_i^p, x_k^p, x_j^q, x_\ell^q) \ \ k_{t-1}(x_k^p, x_\ell^q)\right) \tag{2}$$

which is also a Gibbs distribution.

Proof. the proof, lengthy, is omitted and it is available in a research report (Sahbi et al., 2007).□

In (2), we set v_0 to any positive definite kernel (see proposition 3) and we define $\mathbb{V}(x_i^p, x_k^r, x_j^q, x_\ell^s)$ as $g(x_i^p, x_k^r) \times g(x_j^q, x_\ell^s)$ where g is a decreasing function of any (pseudo) distance involving (x_i^p, x_k^r), *not necessarily symmetric*. In practice, we consider $g(x_i^p, x_k^r) = \mathbb{1}_{\{r=p\}} \times \mathbb{1}_{\{x_k^r \in \mathcal{N}_p(x_i^p)\}}$.

It is easy to see that k_t is a P-kernel on any $\mathcal{S}_p \times \mathcal{S}_q$ (Haussler, 1999) (as the joint probability over sample pairs taken from any $\mathcal{S}_p$ and $\mathcal{S}_q$ sums to one), so the value of the subset kernel $K(\mathcal{S}_p, \mathcal{S}_q)$ defined in (1) is constant and *useless*. To make k_t (up to a factor) a P-kernel on $\mathcal{X} \times \mathcal{X}$ (and not on $\mathcal{S}_p \times \mathcal{S}_q$), we cancel the equality constraint in (1) and we can prove in a similar way that $k_t(x_i^p, x_j^q)$ is equal to $v_t(x_i^p, x_j^q)$ which is still a context-dependent kernel.

2.2. Mercer Condition

Let $\mathcal{X}$ be an input space and let $k_t : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ be symmetric and continuous. k_t is Mercer, i.e., positive (semi) definite, if and only if any Gram (kernel scalar product) matrix built by restricting k_t to any finite subset of $\mathcal{X}$ is positive (semi) definite. A Mercer kernel k_t guarantees the existence of a reproducing kernel Hilbert space $\mathcal{H}$ where k_t can be written as a dot product i.e., $\exists \Phi_t : \mathcal{X} \rightarrow \mathcal{H}$ such that $\forall \ x, x' \in \mathcal{X}$, $k_t(x, x') = \langle \Phi_t(x), \Phi_t(x') \rangle$.

Proposition 2 *ex. (S-Taylor & Cristianini, 2000) the sum and the product of any two Mercer kernels is a*

Mercer kernel. The exponential of any Mercer kernel is also a Mercer kernel.

Proof. see, for instance, (S-Taylor & Cristianini, 2000). □

Now, let us state *our result* about the positive definiteness of the "CDK" kernel.

Proposition 3 *consider* $g : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, *let* $\mathbb{V}(x_i^p, x_k^p, x_j^q, x_\ell^q) = g(x_i^p, x_k^p) g(x_j^q, x_\ell^q)$, *and* k_0 *positive definite. The kernel* k_t *is then positive definite.*

Proof. Initially ($t = 0$), k_0 is per definition a positive definite kernel. By induction, let us assume k_{t-1} a Mercer kernel i.e., $\exists \Phi_{t-1} : k_{t-1}(x, x') = \langle \Phi_{t-1}(x), \Phi_{t-1}(x') \rangle$, $\forall x, x' \in \mathcal{X}$. Now, the sufficient condition will be to show that $\left(\sum_{y,y'} \mathbb{V}(x, y, x', y')\ k_{t-1}(y, y') \right)$ is also a Mercer kernel. Then, by the closure of the exponential and the product (see proposition 2), k_t will then be Mercer. We need to show

$$\forall x_1, \ldots, x_d \in \mathcal{X}, \quad \forall c_1, \ldots, c_d \in \mathbb{R},$$
$$(*) = \sum_{i,j} c_i\ c_j \left(\sum_{y,y'} \mathbb{V}(x_i, y, x_j, y')\ k_{t-1}(y, y') \right) \geq 0$$

We have

$$\begin{aligned}
(*) &= \sum_{i,j} c_i\ c_j \sum_{y,y'} g(x_i, y)\ g(x_j, y')\ k_{t-1}(y, y') \\
&= \sum_{y,y'} \left(\sum_i c_i\ g(x_i, y) \right) \times \\
&\qquad \left(\sum_j c_j\ g(x_j, y') \right)\ k_{t-1}(y, y') \\
&= \sum_{y,y'} \gamma_y\ \gamma_{y'}\ k_{t-1}(y, y') \\
&= \left\| \sum_y \gamma_y \Phi_{t-1}(y) \right\|_{\mathcal{H}} \geq 0. \quad \square
\end{aligned}$$

Corollary 1 *K defined in (1) is also a Mercer kernel.*

Proof. the proof is straightforward for the particular case $n = m$. As $k_t(x_i^p, x_j^q) = \langle \Phi_t(x_i^p), \Phi_t(x_j^q) \rangle$, we can write $K(\mathcal{S}_p, \mathcal{S}_q) = \sum_{i,j} \langle \Phi_t(x_i^p), \Phi_t(x_j^q) \rangle = \langle \sum_i \Phi_t(x_i^p), \sum_j \Phi_t(x_j^q) \rangle$ and this corresponds to a dot product in some Hilbert space. The proof can be found in (S-Taylor & Cristianini, 2000) for the general case of finite subsets of any length. □

2.3. Algorithm and Setting

The factor β, in k_t, acts as a scale parameter and it is selected using

$$\beta \leftarrow \mathbb{E}_r \left[\mathbb{E}_{\{X_1^r, X_2^r : d(X_1^r, X_2^r) \leq \epsilon\}} \left[d(X_1^r, X_2^r) \right] \right] \quad (3)$$

here $\mathbb{E}$ denotes the expectation and X_1^r (also X_2^r) denotes a random variable standing for samples in $\mathcal{S}_r$. The coefficient α controls the tradeoff between the alignment quality and the neighborhood criteria. It is selected by cross-validation and it should guarantee $k_t(x_i^p, x_j^q) \in [0, 1]$. If $A = \sup_{i,j} \sum_{k,\ell} g(x_i^p, x_k^p) \times g(x_j^q, x_\ell^q)$, α should then be selected in $[0, \frac{\beta}{2\ A}]$.

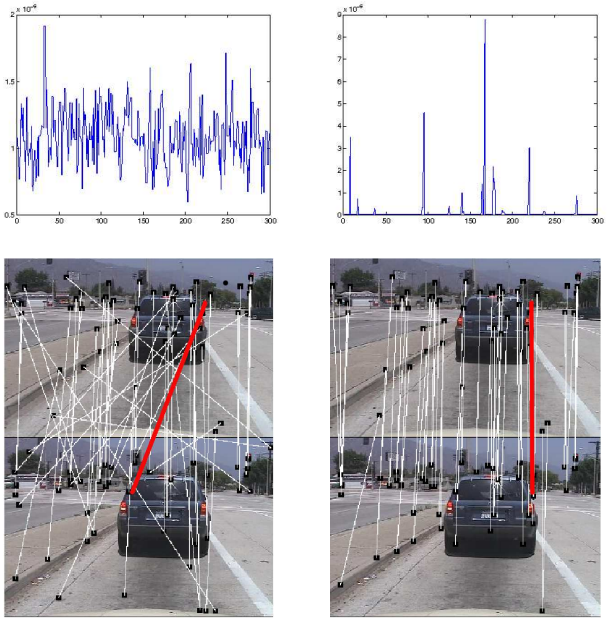


Figure 1. This figure shows a comparison of the matching results when using a naive matching strategy without geometry and our "context-dependent" kernel matching. (Top figures) show the distribution of the kernel values $k(x_i, x_j), j \in \mathcal{I}_q$ using a context-free kernel (left) and our "CDK" kernel (right). We can clearly see that the highest value changes its location so the matching results are now corrected (as shown for one particular and multiple matches in bottom figures).

Consider P, Q as the intrinsic adjacency matrices of $\mathcal{S}_p$ and $\mathcal{S}_q$ respectively defined as $P_{i,k} = g(x_i^p, x_k^p)$, $Q_{j,\ell} = g(x_j^q, x_\ell^q)$. Let U denote the unit matrix and consider $D_{i,j} = d(x_i^p, x_j^q)$, $\mu_{i,j}^{(t)} = k_t(x_i^p, x_j^q)$. Now, $\mu_{i,j}^{(t)}$ is iteratively found using Algorithm ("CDK") (see table 2) and converges to a fixed point (see. Section 2.4).

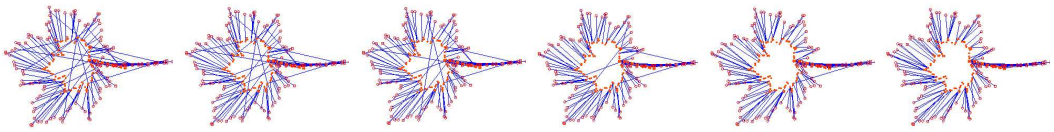


Figure 2. This figure shows the evolution of context-dependent silhouette matching on the Swedish set, for different and increasing values of α. We clearly see that when α increases the matching results are better. We set $\beta = 0.1$ and $t = 1$.

Algorithm (CDK)

Initialization:

Set β using (3) and $\alpha \in [0, \frac{\beta}{2\,A}]$
Set $\mu^{(0)} \leftarrow k_0$, $t \leftarrow 0$

Repeat until $t \to T_{max}$ or $\|\mu^{(t)} - \mu^{(t-1)}\|_2 \to 0$

$$\mu^{(t)} \leftarrow \exp\left(-D/\beta \;+\frac{2\alpha}{\beta} P\,\mu^{(t-1)}\,Q \;-\; U\right)$$

Table 2. The "CDK" kernel evaluation.

2.4. Convergence

Let us assume $0 \leq g \leq 1$, and remind $\mu^{(t)} \in \mathbb{R}^{n\times m}$ be the vector of components $\mu^{(t)}_{i,j} = k_t(x^p_i, x^q_j)$. Introduce the mapping $f : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ defined by its component $f_{i,j}(v)$ as

$$\exp\left(-1 - \frac{d(x^p_i, x^q_j)}{\beta} + \frac{2\alpha}{\beta}\sum_{k,\ell} g(x^p_i, x^p_k) g(x^q_j, x^q_\ell) v_{k,\ell}\right)$$

By construction of the kernel k_t, we have $\mu^{(t)} = f\big(\mu^{(t-1)}\big)$. Let A and B satisfy

$$\sup_{1\leq i\leq n,\ 1\leq j\leq m} \sum_{k,\ell} g(x^p_i, x^p_k) g(x^q_j, x^q_\ell) \leq A \qquad (4)$$

$$\sum_{i,j} \exp\left(-1 - \frac{d(x^p_i, x^q_j)}{\beta}\right) \leq B \qquad (5)$$

Consider $L = \frac{2B\alpha}{\beta}\exp\big(\frac{2\alpha A}{\beta}\big)$, and let $\mathcal{B} = \big\{v \in \mathbb{R}^{n\times m} : \forall\ 1 \leq i \leq n,\ 1 \leq j \leq m,\ |v_{i,j}| \leq 1\big\}$ be the $\|\cdot\|_\infty$-ball of radius 1. Finally, let $\|\cdot\|_1$ denote the 1-norm on $\mathbb{R}^{n\times m}$: $\|u\|_1 = \sum_{1\leq i\leq n,\ 1\leq j\leq n} |u_{i,j}|$.

Proposition 4 *If $\|\mu^{(0)}\|_\infty \leq 1$ and $2\alpha A \leq \beta$, then we have $f(\mathcal{B}) \subset \mathcal{B}$, and on $\mathcal{B}$, f is L-Lipschitz for the norm $\|\cdot\|_1$.*

In particular, if $L < 1$, then there exists a unique $\tilde{v} \in \mathcal{B}$ such that $f(\tilde{v}) = \tilde{v}$, and the sequence $(\mu^{(t)})$ satisfies

$$\|\mu^{(t)} - \tilde{v}\|_1 \leq L^t \|\mu^{(0)} - \tilde{v}\|_1 \underset{t\to+\infty}{\longrightarrow} 0. \qquad (6)$$

Proof. The first assertion is proved by induction by checking that for $\|v\|_\infty \leq 1$, we have

$$\begin{aligned} f_{i,j}(v) &\leq \exp\left(-1 + \tfrac{2\alpha}{\beta}\textstyle\sum_{k,\ell} g(x^p_i, x^p_k) g(x^q_j, x^q_\ell) v_{k,\ell}\right) \\ &\leq \exp\left(-1 + \tfrac{2\alpha}{\beta} A\right) \leq 1. \end{aligned}$$

For the second assertion, note that for any v in $\mathcal{B}$, we have $|\frac{\partial f_{i,j}}{\partial v_{k,\ell}}(v)| \leq \exp\big(-1 - \frac{d(x^p_i, x^q_j)}{\beta}\big)$. Let $C = \exp\big(\frac{2\alpha}{\beta}A\big)$, for any v, v' in $\mathcal{B}$, we have

$$\|f(v) - f(v')\|_1 \;=\; \sum_{i,j} |f_{i,j}(v) - f_{i,j}(v')| = (*)$$

$$\begin{aligned} (*) &\leq \sum_{i,j} \exp\big(-1 - \frac{d(x^p_i, x^q_j)}{\beta}\big)\frac{2\alpha}{\beta}\exp\big(\frac{2\alpha}{\beta}A\big) \\ &\quad \times \sum_{k,\ell} \big| g(x^p_i, x^p_k) g(x^q_j, x^q_\ell) v_{k,\ell} \\ &\qquad - g(x^p_i, x^p_k) g(x^q_j, x^q_\ell) v'_{k,\ell}\big| \\ &\leq \sum_{i,j} \exp\big(-1 - \frac{d(x^p_i, x^q_j)}{\beta}\big)\frac{2\alpha}{\beta} C\ \|v - v'\|_1 \\ &\leq L\|v - v'\|_1 \end{aligned}$$

which proves the second assertion. The last assertion directly comes from the fixed-point theorem. □

3. Performance

Experiments were conducted on the Swedish set (15 classes, 75 images per category) and a random subset of MNIST digit database (10 classes, 200 images per category). Each class in Swedish (resp. MNIST) is split into 50+25 (resp. 100+100) contours for training and testing. Interest points were sampled from each contour in MNIST (resp. Swedish) and encoded using the 60 (resp. 16) coefficients of the shape-context descriptor (Belongie et al., 2000).

3.1. Generalization and Comparison

We evaluate k_t, $t \in \mathbb{N}^+$ using two initializations: (i) linear $k_0(x, x') = k_l(x, x') = \langle x, x'\rangle$ (ii) and

polynomial $k_0(x,x') = k_p(x,x') = (\langle x,x'\rangle + 1)^2$. Our goal is to show the improvement brought when using $k_t,\ t \in \mathbb{N}^+$, so we compared it against the standard context-free kernels k_l and k_p (i.e., $k_t,\ t=0$). For this purpose, we trained a "one-versus-all" SVM classifier for each class in both MNIST and Swedish using the subset kernel $K(\mathcal{S}_p,\mathcal{S}_q) = \sum_{x\in\mathcal{S}_p, x'\in\mathcal{S}_q} k_t(x,x')$. The performance are measured, on different test sets, using n-fold cross-validation error ($n=5$).

We remind that β is set using (3) as the left-hand side of k_t corresponds to the Gaussian kernel with scale β. In practice, $\beta = 0.1$. The influence (and the performance) of the right-hand side of k_t increases as α increases (see. Figure 2), nevertheless the convergence of k_t to a fixed point is guaranteed only if $\alpha \in [0, \frac{\beta}{2\ A}]$. Therefore, it becomes obvious that α should be set to $\frac{\beta}{2\ A}$ where $A = \sup_{i,j} \sum_{k,\ell} g(x_i^p, x_k^p) \times g(x_j^q, x_\ell^q)$ (in practice, $0 \leq g \leq 1$ and $A = 1$).

Table 3 shows the 5-fold cross validation errors on MNIST and Swedish for different iterations; we clearly see the out-performance and the improvement of the "CDK" kernel ($k_t, t \in \mathbb{N}^+$) with respect to the context-free kernels used for initialization ($k_0 = k_l$ or k_p.)

Initialization	Linear	Polynomial
Iterations (MNIST)		
k_0	11.4± 4.42	9.15 ± 4.63
k_1	8.80± 4.77	5.6 ± 2.72
k_2	6.90± 3.55	5.8 ± 2.36
k_3	6.90± 3.41	5.2 ± 2.07
k_4	**6.90± 3.41**	**5.2 ± 2.07**
Iterations (Swedish)		
k_0	11.7± 2.88	6.53± 6.34
k_1	6.00± 2.30	3.33± 2.73
k_2	**3.06± 1.88**	**3.33± 2.73**

Table 3. This table shows the mean and the standard deviation of the 5-fold error on the MNIST (top) and Swedish (bottom) databases. We can see a clear and a consistent gain through different iterations and also the convergence of the errors.

4. Remarks and Discussion

The adjacency matrix P, in k_t, provides the intrinsic properties and also characterizes the geometry of an object $\mathcal{S}_p$. Let us remind $\mathcal{N}_p(x_i^p) = \{x_k^p \in \mathcal{S}_p : k \neq i,\ \| \ _g(x_i^p) - \ _g(x_k^p)\|_2 \leq \epsilon_p\}$ and $P_{i,j} = \mathbb{1}_{\{x_j^q \in \mathcal{N}_p(x_i^p)\}}$. It is easy to see that P is translation and rotation invariant and can also be made scale invariant when ϵ_p is adapted to the scale of $_g(x_i^p)$. It follows that the right-hand side of our kernel is invariant to any $2D$ similarity transformation. Notice, also, that the left-hand side of k_t involves similarity invariant descriptors $\ _f(x_i^p)$, $\ _f(x_j^q)$ so k_t (and K) is similarity invariant.

One current limitation of our kernel k_t resides in its evaluation complexity. Assuming k_{t-1} known, for a given pair x_i^p, x_j^q, this complexity is $O\left(\max(N^2, s)\right)$, where s is the dimension of $\ _f(x_i^p)$ and $N = \max_{i,p} \#\{\mathcal{N}_p(x_i^p)\}$. It is clear enough that when $N < \sqrt{s}$, the complexity of evaluating our kernel is strictly equivalent to that of usual kernels such as the linear. Nevertheless, the worst case ($N \gg \sqrt{s}$) makes our kernel evaluation prohibitive and this is mainly due to the right-hand side of $k_t(x_i^p, x_j^q)$ which requires the evaluation of kernel sums in a hypercube of dimension 4. A simple and straightforward generalization of the integral image (see for instance (Viola & Jones, 2001)) will reduce this complexity to $O(s)$.

Finally, the out-performance of our kernel comes essentially from the inclusion of the context. This strongly improves the precision and helps including the intrinsic properties (geometry) of objects. Even though tested only on visual object recognition, our kernel can be extended to many other pattern analysis problems such as bioinformatics, speech and text. For instance, in text analysis and particularity machine translation (Sim et al., 2007), the design of a similarity kernel between words in two different languages, can be achieved using any standard dictionary. Of course, the latter defines similarity between any two words (w_e, w_f) independently from their bilingual training text (or bitext), i.e., the phrases where (w_e, w_f) might appear and this results into bad translation performances. A better estimate of similarity between two words (w_e, w_f), can be achieved using their context i.e., the set of words which cooccur frequently with (w_e, w_f) (Koehn et al., 2003).

5. Conclusion

We introduced in this paper a new type of kernels referred to as context-dependent. Its strength resides in the improvement of the alignments between interest points and this is considered as a preliminary step in order to increase the robustness and the precision of object recognition.

We have also shown that our kernel is Mercer and applicable to SVM learning. The latter, achieved for shape recognition problems, has better performance than SVMs with context-free kernels. Future work in-

cludes the comparison of our kernel with other context-free kernels and its application in scene and object understanding using other standard databases.

Acknowledgements

This work was supported in part by the Agence Nationale de la Recherche, project "Modeles Graphiques et Applications" and the project "Infom@gic".

References

Amores, J., Sebe, N., & Radeva, P. (2005). Fast spatial pattern discovery integrating boosting with constellations of contextual descriptors. *In CVPR*.

Bahlmann, C., Haasdonk, B., & Burkhardt, H. (2002). Online handwriting recognition with support vector machines, a kernel approach. *IWFHR* (pp. 49–54).

Barla, A., Odone, F., & Verri, A. (2002). Hausdorff kernel for 3d object acquisition and detection. *ECCV, LNCS 2353* (pp. 20–33).

Belongie, S., Malik, J., & Puzicha, J. (2000). Shape context: A new descriptor for shape matching and object recognition. *In NIPS*.

Bishop, C. (2007). *Pattern recognition and machine learning*. Springer.

Boser, B., Guyon, I., & Vapnik, V. (1992). An training algorithm for optimal margin classifiers. *In Fifth Annual ACM Workshop on Computational Learning Theory, Pittsburgh*, 144–152.

Boughorbel, S. (2005). *Kernels for image classification with support vector machines*. Doctoral dissertation, PhD. Thesis, Faculte d'Orsay.

Chapelle, O., Haffner, P., & Vapnik, V. (1999). Svms for histogram-based image classification. *Transaction on Neural Networks*, *10*.

Cuturi, M. (2005). *Etude de noyaux de semigroupe pour objets structures dans le cadre de l'apprentissage statistique*. Doctoral dissertation, PhD thesis, ENSMP.

Everingham, M., Gool, L. V., Williams, C., Winn, J., & Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. *http://www.pascal-network.org/challenges/VOC/voc2007/workshop/*.

Gartner, T. (2003). A survey of kernels for structured data. *Multi Relational Data Mining*, *5*, 49–58.

Grauman, K., & Darrell, T. (2007). The pyramid match kernel: Efficient learning with sets of features. *In JMLR*, *8*, 725–760.

Harris, C., & Stephens, M. (1988). A combined corner and edge detector. *Alvey Vision Conference* (pp. 147–151).

Haussler, D. (1999). Convolution kernels on discrete structures. *Technical Report UCS-CRL-99-10, UC Santa Cruz*.

Jaakkola, T., Diekhans, M., & Haussler, D. (1999). Using the fisher kernel method to detect remote protein homologies. *ISMB* (pp. 149–158).

Koehn, P., Och, F. J., & Marcu, D. (2003). Statistical phrase-based translation. *In Proceedings of HLT/NAACL*.

Kondor, R., & Jebara, T. (2003). A kernel between sets of vectors. *In proceedings of ICML*.

Lazebnik, S., Schmid, C., & Ponce, J. (2004). Semi-local affine parts for object recognition. *In BMVC*.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *In IJCV*, *60*, 91–110.

Lyu, S. (2005). Mercer kernels for object recognition with local features. *In the proceedings of CVPR*.

Moreno, P., Ho, P., & Vasconcelos, N. (2003). A kullback-leibler divergence based kernel for svm classfication in multimedia applications. *In NIPS*.

Mortensen, E., Deng, H., & Shapiro, L. (2005). A sift descriptor with global context. *In CVPR* (pp. 184–190).

Nie, J.-Y., Simard, M., Isabelle, P., & Durand, R. (1999). Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the web. *Proceedings of the 22nd ACM SIGIR conference on Research and development in information retrieval*.

S-Taylor, J., & Cristianini, N. (2000). Support vector machines and other kernel-based learning methods. *Cambridge University Press*.

Sahbi, H., Audibert, J.-Y., Rabarisoa, J., & Keriven, R. (2007). Context-dependent kernel design for object matching and recognition. *Research Report N 2007D018, ENST Paris, ParisTech*.

Schmid, C., & Mohr, R. (1997). Local greyvalue invariants for image retrieval. *In PAMI*, *19*, 530–535.

Scholkopf, B., Tsuda, K., & Vert, J.-P. (2004). *Kernel methods in computational biology*. MIT Press.

Shashua, A., & Hazan, T. (2004). Algebraic set kernels with application to inference over local image representations. *In NIPS*.

Sim, K., Byrne, W., Gales, M., Sahbi, H., & Woodland, P. (2007). Consensus network decoding for statistical machine translation system combination. *In ICASSP*.

Swain, M., & Ballard, D. (1991). Color indexing. *International Journal of Computer Vision*, *7*, 11–32.

Vapnik, V. N. (1998). *Statistical learning theory*. A Wiley-Interscience Publication.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *In CVPR*.

Wallraven, C., Caputo, B., & Graf, A. (2003). Recognition with local features: the kernel recipe. *ICCV* (pp. 257–264).

Wolf, L., & Shashua, A. (2003). Learning over sets using kernel principal angles. *In JMLR*, *4*, 913–931.

Privacy-Preserving Reinforcement Learning

Jun Sakuma JUN@FE.DIS.TITECH.AC.JP
Shigenobu Kobayashi KOBAYASI@DIS.TITECH.AC.JP
Tokyo Institute of Technology, 4259 Nagatsuta-cho, Midoriku, Yokohama, 226-8502, Japan

Rebecca N. Wright REBECCA.WRIGHT@RUTGERS.EDU
Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854, USA

Abstract

We consider the problem of distributed reinforcement learning (DRL) from private perceptions. In our setting, agents' perceptions, such as states, rewards, and actions, are not only distributed but also should be kept private. Conventional DRL algorithms can handle multiple agents, but do not necessarily guarantee privacy preservation and may not guarantee optimality. In this work, we design cryptographic solutions that achieve optimal policies without requiring the agents to share their private information.

1. Introduction

With the rapid growth of computer networks and networked computing, a large amount of information is being sensed and gathered by distributed agents physically or virtually. Distributed reinforcement learning (DRL) has been studied as an approach to learn a control policy thorough interactions between distributed agents and environments—for example, sensor networks and mobile robots. DRL algorithms, such as the distributed value function approach (Schneider et al., 1999) and the policy gradient approach (Moallemi & Roy, 2004), typically seek to satisfy two types of physical constraints. One is constraints on communication, such as an unstable network environment or limited communication channels. The other is memory constraints to manage the huge state/action space. Therefore, the main emphasis of DRL has been to learn good, but sub-optimal, policies with minimal or limited sharing of agents' perceptions.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

In this paper, we consider the privacy of agents' perceptions in DRL. Specifically, we provide solutions for privacy-preserving reinforcement learning (PPRL), in which agents' perceptions, such as states, rewards, and actions, are not only distributed but are desired to be kept private. Consider two example scenarios:

Optimized Marketing (Abe et al., 2004): Consider the modeling of the customer's purchase behavior as a Markov Decision Process (MDP). The goal is to obtain the optimal catalog mailing strategy which maximizes the long-term profit. Timestamped histories of customer status and mailing records are used as state variables. Their purchase patterns are used as actions. Value functions are learned from these records to learn the optimal policy. If these histories are managed separately by two or more enterprises, they may not want to share their histories for privacy reasons (for example, in keeping with privacy promises made to their customers), but might still like to learn a value function from their joint data in order that they can all maximize their profits.

Load Balancing (Cogill et al., 2006): Consider a load balancing among competing factories. Each factory wants to accept customer jobs, but in order to maximize its own profit, may need to redirect jobs when heavily loaded. Each factory can observe its own backlog, but factories do not want to share their backlog information with each other for business reasons, but they would still like to make optimal decisions.

Privacy constraints prevent the data from being combined in a single location where centralized reinforcement algorithms (CRL) could be applied. Although DRL algorithms work in a distributed setting, they are designed to limit the total amount of data sent between agents, but do not necessarily do so in a way that guarantees privacy preservation. Additionally, DRL often sacrifices optimality in order to learn with low communication. In contrast, we propose solutions

that employ cryptographic techniques to achieve optimal policies (as would be learned if all the information were combined into a centralized reinforcement learning (CRL) problem) while also explicitly protecting the agents' private information. We describe solutions both for data that is "partitioned-by-time" (as in the optimized marketing example) and "partitioned-by-observation" (as in the load balancing example).

Related Work. Private distributed protocols have been considered extensively for data mining, pioneered by Lindell and Pinkas (Lindell & Pinkas, 2002), who presented a privacy-preserving data-mining algorithm for ID3 decision-tree learning. Private distributed protocols have also been proposed for other data mining and machine learning problems, including k-means clustering (Jagannathan & Wright, 2005; Sakuma & Kobayashi, 2008), support vector machines (Yu et al., 2006), boosting (Gambs et al., 2007), and belief propagation (Kearns et al., 2007).

Agent privacy in reinforcement learning has been previously considered by Zhang and Makedon (Zhang & Makedon, 2005). Their solution uses a form of average reward reinforcement learning that does not necessarily guarantee an optimal solution; further, their solution only applies partitioning by time. In contrast, our solutions guarantee optimality under appropriate conditions and we provide solutions both when the data is partitioned by time and by observation.

In principle, private distributed computations such as these can be carried out using secure function evaluation (SFE) (Yao, 1986; Goldreich, 2004), which is a general and well studied methodology for evaluating any function privately. However, although asymptotically polynomially bounded, these computations can be too inefficient for practical use, particular when the input size is large. For the reinforcement learning algorithms we address, we make use of existing SFE solutions for small portions of our computation in order as part of a more efficient overall solution.

Our Contribution. We introduce the concepts of partitioning by time and partitioning by observation in distributed reinforcement learning (Section 2). We show privacy-preserving solutions for SARSA learning algorithms with random action selection for both kinds of partitioning (Section 4). Additionally, these algorithms are expanded to Q-learning with greedy or ϵ-greedy action selection (Section 5). We provide experimental results in Section 6.

Table 1 provides a qualitative comparison of variants of reinforcement learning in terms of efficiency, learning accuracy, and privacy loss. We compare five approaches: CRL, DRL, independent distributed reinforcement learning (IDRL, explained below), SFE, and our privacy-preserving reinforcement learning solutions (PPRL). In CRL, all the agents send their perceptions to a designated agent, and then a centralized reinforcement is applied. In this case, the optimal convergence of value functions is theoretically guaranteed when the dynamics of environments follow a discrete MDP; however, privacy is not provided, as all the data must be shared.

	comp.	comm.	accuracy	privacy
CRL	good	good	good	none
DRL	good	good	medium	imperfect
IDRL	good	good	bad	perfect
PPRL	**medium**	**medium**	**good**	**perfect**
SFE	bad	bad	good	perfect

Table 1. Comparison of different approaches

On the opposite end of the spectrum, in IDRL (independent DRL), each agent independently applies CRL only using its own local information; no information is shared. In this case, privacy is completely preserved, but the learning results will be different and independent. In particular, accuracy will be unacceptable if the agents have incomplete but important perceptions about the environment. DRL can be viewed as an intermediate approach between CRL and IDRL, in that the parties share only some information and accordingly reap only some gains in accuracy.

The table also includes the direct use of general SFE and our approach of PPRL. Both PPRL and SFE obtain good privacy and good accuracy. Although our solution incurs a significant cost (as compared to CRL, IDRL, and DRL) in computation and communication to obtain this, it does so with significantly improved computational efficiency over SFE. We provide a more detailed comparison of the privacy, accuracy, and efficiency of our approach and other possible approaches along with our experimental results in Section 6.

2. Preliminaries

2.1. Reinforcement Learning and MDP

Let S be a finite *state set* and A be a finite *action set.* A *policy* π is a mapping from state/action pair (s, a) to the probability $\pi(s, a)$ with which action a is taken at state s. At time step t, we denote by s_t, a_t, and r_t, the state, action, and *reward* at time t, respectively.

A *Q-function* is the expected return $Q^\pi(s, a) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\}$, where γ is a discount factor ($0 \leq \gamma < 1$). The goal is to learn the optimal policy π maximizing the Q-function: $Q^*(s, a) =$

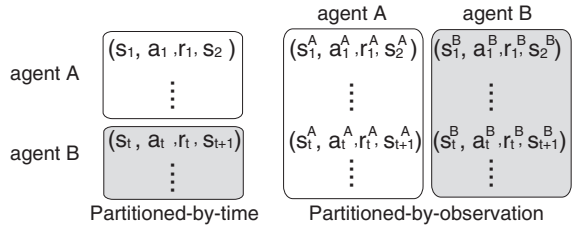


Figure 1. Partitioning model in the two-agent case

$\max_\pi Q(s,a)$ for all (s,a). In SARSA learning, Q-values are updated at each step as:

$$\begin{aligned} \Delta Q(s_t,a_t) &\leftarrow \alpha(r_t+\gamma Q(s_{t+1},a_{t+1})-Q(s_t,a_t)), \\ Q(s_t,a_t) &\leftarrow \Delta Q(s_t,a_t)+Q(s_t,a_t), \end{aligned} \quad (1)$$

where α is the learning rate. Q-learning is obtained by replacing the update of ΔQ by:

$$\Delta Q(s_t,a_t) \leftarrow \alpha(r_t+\gamma \max_a Q(s_{t+1},a)-Q(s_t,a_t)).$$

Iterating these updates under appropriate conditions, optimal convergence of Q-values is guaranteed with probability 1 in discrete MDPs (Sutton & Barto, 1998; Watkins, 1989); the resulting optimal policy can be readily obtained.

2.2. Modeling Private Information in DRL

Let $h_t=(s_t,a_t,r_t,s_{t+1},a_{t+1})$, let $H=\{h_t\}$, and suppose there are m agents. We consider two kinds of partitioning of H (see Fig. 1).

Partitioned-by-Time. This model assumes that only one agents interacts with the environment at any time step t. Let T^i be the set of time steps at which only ith agent has interactions with the environment. Then $T^i \cap T^j = \emptyset, (i \neq j)$ and the set $H^i=\{h_t \mid t \in T^i\}$ is considered the private information of the ith agent.

Partitioned-by-Observation. This model assumes that states and actions are represented as a collection of state and action variables. The state space and the action space are $S=\prod_i S^i$ and $A=\prod_i A^i$ where S^i and A^i are the space of the ith agent's state and action variables, respectively. Without loss of generality (and for notational simplicity), we consider each agent's local state and action spaces to consist of a single variable. If $s_t \in S$ is the joint state of the agents at time t, we denote by s_t^i the state that ith agent perceives and by a_t^i the action of ith agent. Let r_t^i be the *local reward* of ith agent obtained at time t. We define the *global reward* (or *reward* for short) as $r_t=\sum_i r_t^i$ in this model. Our Q-functions are evaluated based on this global reword. The perception of the ith agent at time t is denoted as $h_t^i=\{s_t^i,a_t^i,r_t^i,s_{t+1}^i,a_{t+1}^i\}$. The private information of the ith agent is $H^i=\{h_t^i\}$.

We note that partitioning by observation is more general than partitioning by time, in that one can always represent a sequence that is partitioned by time by one that is partitioned by observation. However, we provide more efficient solutions in simpler case of partitioning by time.

Let π^c be a policy learned by CRL. Then, informally, the objective of PPRL is stated as follows:

Statement 1. *The ith agent takes H^i as inputs. After the execution of PPRL, all agents learn a policy π which is equivalent to π^c. Furthermore, no agent can learn anything that cannot be inferred from π and its own private input.*

This problem statement can be formalized as in SFE (Goldreich, 2004). This is a strong privacy requirement which precludes consideration of solutions that reveal intermediate Q-values, actions taken, or states visited. We assume our agents behave *semi-honestly*, a common assumption in SFE—this assumes agents follows their specified protocol properly, but might also use their records of intermediate computations in order to attempt to learn other parties' private information.

3. Cryptographic Building Blocks

Our solutions make use of several existing cryptographic tools. Specifically, in our protocol, Q-values are encrypted by an additive homomorphic cryptosystem, which allows the addition of encrypted values without requiring their decryption, as described in Section 3.1. Using the homomorphic properties, this allows encrypted Q-values are updated in the regular RL manner, while unencrypted Q-values are not known to agents. For computations which cannot be treated by the homomorphic property, we use SFE as a primitive, as we describe in Section 3.2.

3.1. Homomorphic Public Key Cryptosystems

In a public key cryptosystem, encryption uses a *public key* that can be known to everyone, while decryption requires knowledge of the corresponding *private key*. Given a corresponding pair of $(\mathsf{sk},\mathsf{pk})$ of private and public keys and a message m, then $c=e_{\mathsf{pk}}(m;\ell)$ denotes a (random) encryption of m, and $m=d_{\mathsf{sk}}(c)$ denotes decryption. The encrypted value c uniformly distributes over $\mathbb{Z}_N$ if ℓ is taken from $\mathbb{Z}_N$ randomly. An *additive homomorphic cryptosystem* allows addition computations on encrypted values without knowl-

edge of the secret key. Specifically, there is some operation $\cdot$ (not requiring knowledge of sk) such that for any plaintexts m_1 and m_2, $e_{\mathsf{pk}}(m_1 + m_2; \ell) = e_{\mathsf{pk}}(m_1; \ell_1) \cdot e_{\mathsf{pk}}(m_2; \ell_2)$, where ℓ is uniformly random provided that at least one of ℓ_1 and ℓ_2 is. Based on this property, it also follows that given a constant k and the encryption $e_{\mathsf{pk}}(m_1; \ell)$, we can compute multiplications by k via repeated application of $\cdot$. This also enables a re-randomization property, which allows the computation of a new random encryption $c' = e_{\mathsf{pk}}(m; \ell')$ of m from an existing encryption $c = e_{\mathsf{pk}}(m; \ell)$ of m, again without knowledge of the private key or of m, as follows: $e_{\mathsf{pk}}(m; \ell) = \mathrm{Enc}_{\mathsf{pk}}(m; \ell_1) \cdot \mathrm{Enc}_{\mathsf{pk}}(0; \ell_2)$. In the rest of the paper, we omit the random number ℓ from our encryptions for simplicity.

In an (m, t)-*threshold cryptosystem*, m agents share a common public key pk while the agents hold different private keys $\mathsf{sk}^1, ..., \mathsf{sk}^n$. Each agent can encrypt any message with the common public key. Decryption cannot be performed by fewer than t agents, and can be performed by any group of at least t agents using a recovery algorithm based on the public key and their *decryption shares* $d_{\mathsf{sk}^1}(c), ..., d_{\mathsf{sk}^n}(c)$. We require a cryptosystem that provides semantic security (under appropriate computational hardness assumptions), re-randomization, the additive homomorphic property, and threshold decryption, such as the generalized Paillier cryptosystem (Dåmgard & Jurik, 2001).

3.2. Private Comparison and Division

As mentioned, secure function evaluation (SFE) is a cryptographic primitive which allows two or more parties to evaluate a specified function of their inputs without revealing (anything else about) their inputs to each other (Goldreich, 2004; Yao, 1986). Although our overall solution is more efficient than using SFE, we do make use of SFE for two kinds of computations.

One is the problem of *private comparison of random shares*. Let $\boldsymbol{x} = (x_1, ..., x_d) \in \mathbb{Z}_N^d$. For our purposes, A and B have *random shares* of $\boldsymbol{x}$ if A has $\boldsymbol{x}^A = (x_1^A, ..., x_d^A)$ and B has $\boldsymbol{x}^B = (x_1^B, ..., x_d^B)$ such that x_i^A and x_i^B are uniformly distributed in $\mathbb{Z}_N$ such that $x_i = (x_i^A + x_i^B) \mod N$ for all i. If A holds $\boldsymbol{x}^A$ and B holds $\boldsymbol{x}^B$, where $\boldsymbol{x}^A$ and $\boldsymbol{x}^B$ are random shares of $\boldsymbol{x}$, then private comparison of random shares computes the index i^* such that $i^* = \arg\max_i (x_i^A + x_i^B)$ in such a way that A learns only i^* and B learns nothing.

The other is a problem of *private division of random shares*. The input of A and B are random shares of x, $x^A \in \mathbb{Z}_N$ and $x^B \in \mathbb{Z}_N$, respectively. Let K be an integer known to both parties. Then, private division of random shares computes random shares Q^A and Q^B of quotient $Q \in \mathbb{Z}_N$ such that $x = (QK + R) \mod N$, where $R \in \mathbb{Z}_N\, (0 \leq R < K), Q = (Q^A + Q^B) \mod N$. After the protocol, A and B learn Q^A and Q^B, respectively, and nothing else.

We use private division of random shares in several places in our protocols to achieve *private division of encrypted values*. Suppose agent A has a key pair $(\mathsf{pk}, \mathsf{sk})$ and agent B knows pk and $e_{\mathsf{pk}}(x)$. The following protocol allows B to learn the encrypted quotient $e_{\mathsf{pk}}(Q)$ from $e_{\mathsf{pk}}(x)$ and K:

1. B computes $c \leftarrow e_{\mathsf{pk}}(x) \cdot e_{\mathsf{pk}}(-x^B), x^B \in_r \mathbb{Z}_N$ and send c to A.
2. A computes the decryption $x^A \leftarrow d_{\mathsf{sk}}(c) (\equiv x - x^B \mod N)$.
3. Using SFE for private division on A and B's inputs x^A and x^B, respectively, A and B obtain outputs Q^A and Q^B, respectively.
4. A sends $e_{\mathsf{pk}}(Q^A)$ to B.
5. B computes $e_{\mathsf{pk}}(Q) \leftarrow e_{\mathsf{pk}}(Q^A) \cdot e_{\mathsf{pk}}(Q^B)$.

4. Private Q-Value Update

In this section, we describe privacy-preserving SARSA update of Q-values under random action selection is described for our two partitioning models. We extend this to (ϵ-)greedy action selection in Section 5. We assume that reward r, learning rate α, and discount rate γ are non-negative rational numbers and that $\sum_{t=1}^{\infty}(\gamma^t L r_{\max}) < N$, where $r_{\max}$ is the largest reward that agents can obtain and $L \in \mathbb{Z}_N$ is a parameter defined in Section 4.1. In this paper, we describe protocols for two agents; these can be extended to m-agent case ($m \geq 3$) straightforwardly, as will be shown in an extended version of the paper.

4.1. Partitioned-by-Time Model

We first restrict our attention to the case where agent A has perceptions during $T^A = \{1, ..., t-1\}$ and B has perceptions during $T^B = \{t\}$. In this setting, A first computes can learn intermediate Q-values during the time period T^A, because they can be locally computed only from A's perception. At time t, the new Q-values must be computed based on the intermediate Q-values known to A and B's observation at time t. In brief, we do this by carrying out the update on encrypted Q-values using the homomorphic property to carry this out privately. However, the update includes the multiplication of rational numbers, such as α or γ, so the computation is not closed in $\mathbb{Z}_N$. Hence, we first scale these rational numbers by multiplying with large enough integers so that all computations are closed in $\mathbb{Z}_N$. We use private division of encrypted values to remove the scaling.

- Public input; L, K, learning rate α, discount rate γ
- A's input: $Q(s,a)$ (trained by A during T^A)
- B's input: (s_t, a_t)
- A's output: Nothing
- B's output: Encryption of updated Q-value $c(s_t, a_t)$

1. A: Compute $e^A(Q(s,a))$ for all (s,a) and send to B.
2. B: Take action a_t and get r_t, s_{t+1}.
3. B: Choose a_{t+1} randomly.
4. Update Q-value:
   (a) B: Compute $e^A(K\Delta Q(s_t,a_t))$ by eq. 3.
   (b) B: Do private division of $e^A(K\Delta Q(s_t,a_t))$ with A, then B learns $e^A(\Delta Q'(s_t,a_t))$.
   (c) B: Update $c(s_t,a_t)$ by eq. 4.

Figure 2. Private update of Q-values in partitioned-by-time model (SARSA/random action selection)

We now describe our protocol for private update, shown in Fig. 2, in more detail. Let pk_A be A's public key. At step 1, A computes $c(s,a) = e_{\mathsf{pk}_A}(Q(s,a))$ for all (s,a) and sends them to B. B takes action a_t, gets r_t, s_{t+1} (step 2), and chooses a_{t+1} randomly (step 3). A and B must now update the encrypted Q-value $c(s,a)$. By encrypting both sides of SARSA update (eq. 1), we obtain:

$$\begin{aligned} c(s_t,a_t) &\leftarrow e_{\mathsf{pk}_A}(\Delta Q(s_t,a_t) + Q(s_t,a_t)), \\ &= e_{\mathsf{pk}_A}(\Delta Q(s_t,a_t)) \cdot e_{\mathsf{pk}_A}(Q(s_t,a_t)) \\ &= \Delta c(s_t,a_t) \cdot c(s_t,a_t), \end{aligned} \quad (2)$$

where $\Delta c(s_t,a_t) = e_{\mathsf{pk}_A}(\Delta Q(s_t,a_t))$. If $\Delta c(s_t,a_t)$ is computed by B from what B observes, B can update $c(s_t,a_t)$ by eq. 2 locally. Therefore, step 4 is devoted to the computation of $\Delta c(s_t,a_t)$.

As mentioned, large integers K and L are used to treat the multiplication of rationals α and γ, where $\alpha\gamma K \in \mathbb{Z}_N$ and $Lr_t \in \mathbb{Z}_N$ for all r_t. Multiplying K to both sides of eq. 1 and multiplying L to r_t, we obtain

$$K\Delta Q(s_t,a_t) \leftarrow K\alpha(Lr_t + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)),$$

in which the computation is closed in $\mathbb{Z}_N$. Encrypting both sides by A's public key, we obtain

$$\begin{aligned} &e_{\mathsf{pk}_A}(K\Delta Q(s_t,a_t)) \\ &\quad = e_{\mathsf{pk}_A}(Lr_t)^{\alpha K} \cdot c(s_{t+1},a_{t+1})^{\alpha\gamma K} \cdot c(s_t,a_t)^{-\alpha K} \end{aligned} \quad (3)$$

Since K, L, α, γ are public and B has r_t, $c(s,a)$, B can compute $e_{\mathsf{pk}_A}(K\Delta Q(s_t,a_t))$ by eq. 3 (step 4(a)). B needs to divide $e_{\mathsf{pk}_A}(K\Delta Q(s_t,a_t))$ by K, however, division is again not allowable. Instead, a quotient $\Delta Q'(s_t,a_t)$ satisfying $\Delta Q(s_t,a_t) = K\Delta Q'(s_t,a_T) + R (0 \le R < K)$ is computed by private encrypted division and B obtains $e_{\mathsf{pk}_A}(\Delta Q'(s_t,a_t))$ (step 4(b)). Then, B finally computes

$$c(s_t,a_t) \leftarrow e_{\mathsf{pk}_A}(\Delta Q'(s_t,a_t)) \cdot c(s_t,a_t). \quad (4)$$

It follows that eq. 4 is equivalent to eq. 2 except for the truncation error included by the private encrypted division step (step 4(c)). This truncation is negligibly small if L is sufficiently large.

Lemma 1. *If A and B behave semi-honestly, then after the private update of Q-values for SARSA and random action selection in partitioned-by-time model, B correctly updates encrypted Q-values but learn nothing else. A learns nothing.*

The proof of this lemma (omitted for space) follows the standardized proof methodology of secure multi-party computation (Goldreich, 2004), showing that one can create the required algorithms, called *simulators*, for A and B. Intuitively, Step 4(b) is secure because it is implemented by SFE. Everything else that B receives except for messages received at steps for step 4(b) are encrypted by A's public key, so do not reveal anything. A does not receive anything except messages that are part of the SFE in step 4(b), so does not learn anything. Thus, the protocol is secure overall.

For the general setting of T^A and T^B, after time t, if B interacts with the environment at time $t+1$ again, the protocol can be started from step 2. When interaction switches back to A, an SFE step is used to change the encryption of the Q-values from A's private key to B's private key via an SFE step, and then the roles of A and B are switched.

4.2. Partitioned-by-Observation Model

In this model, we use a $(2,2)$-threshold cryptosystem. Both parties share a common public key pk: encryption of m by pk is denoted by $e(m)$ in this section. A and B hold different secret keys sk^A and sk^B for decryption shares, respectively. A and B cannot decrypt without both cooperating.

In this partitioning model, we write $a_t = (a_t^A, a_t^B)$, $s_t = (s_t^A, s_t^B)$, and $r_t = r_t^A + r_t^B$. A receives only (s_t^A, a_t^A, r_t^A) and B receives only (s_t^B, a_t^B, r_t^B). Private update of Q-values in this model is shown in Fig. 3. In this model, eq. 3 is rewritten as

$$\begin{aligned} &e(K\Delta Q(s_t,a_t)) = X^A \cdot X^B \cdot X \\ &X^A = e(Lr_t^A)^{\alpha KL}, \quad X^B = e(Lr_t^B)^{\alpha KL}, \\ &X = c(s_{t+1},a_{t+1})^{\alpha\gamma K} \cdot c(s_t,a_t)^{-\alpha K}. \end{aligned} \quad (5)$$

X^A and X^B can be computed by A and B. To obtain $c(s_{t+1},a_{t+1})$ and $c(s_t,a_t)$, let h,i,j,k be indices of Q-tables where $h \in S^A, i \in S^B, j \in A^A, k \in A^B$. At

step 4(a), A sends X^A and tables $\{c_{ik}\}, \{c'_{ik}\}$ with re-randomization such that

$$c_{ik} = c(s_t^A, i, a_t^A, k) \cdot e(0)\ (i \in S^B, k \in A^B), \quad (6)$$
$$c'_{ik} = c(s_{t+1}^A, i, a_{t+1}^A, k) \cdot e(0)\ (i \in S^B, k \in A^B), \quad (7)$$

to B. B determines $c(s_t, a_t) = c_{s_t^B, a_t^B}$, $c(s_{t+1}, a_{t+1}) = c'_{s_{t+1}^B, a_{t+1}^B}$ and obtains $e(K\Delta Q(s_t, a_t))$ by eq. 5 (step 4(b)). Then computes $e(\Delta Q'(s_t, a_t))$ by private division (step 4(c)). For all $(hijk)$, B sets

$$\Delta c_{hijk} \leftarrow \begin{cases} e(\Delta Q'(s_t, a_t)) & (i = s_t^B, k = a_t^B) \\ e(0) & \text{(o.w.)} \end{cases} \quad (8)$$

and sends $\{\Delta c_{hijk}\}$ to A (step 4(d)). Finally, for all (ik), Q-values are updated as

$$c(s_t^A, i, a_t^A, k) \leftarrow c(s_t^A, i, a_t^A, k) \cdot \Delta c_{s_t^A i a_t^A k}. \quad (9)$$

by A. With this update, $e(\Delta Q'(s_t, a_t))$ is added only when $(h, i, j, k) = (s_t^A, s_t^B, a_t^A, a_t^B)$. Otherwise, $e(0)$ is added. Note that A cannot tell which element is $e(\Delta Q'(s_t, a_t))$ in $\{\Delta c_{hijk}\}$ because of the re-randomization. Thus, eq. 9 is the desired update.

Lemma 2. *If A and B behave semi-honestly, then after the private update of Q-values for SARSA and random action selection in partitioned-by-observation model, A updates encrypted Q-values correctly but learns nothing. B learns nothing.*

By iterating private updates, encrypted Q-values trained by SARSA learning are obtained.

5. Private Greedy Action Selection

Private distributed algorithms for greedy action selection to compute $a^* = \arg\max_a Q(s, a)$ from encrypted Q-values in both partitioning models are described. These are used for: (1) (ϵ-)greedy action selection, (2) max operation in updates of Q-learning, and (3) extracting learned policies from final Q-values. In the partitioned-by-time model, this is readily solved by using private comparison, so is omitted.

5.1. Private Greedy Action Selection in Partitioned-by-observation Model

When A and B observe s_t^A and s_t^B, respectively, private greedy action selection requires that (1) A obtains a^{A*} and nothing else, (2) B obtains a^{B*} and nothing else, where $(a^{A*}, a^{B*}) = \arg\max_{(a^A, a^B)}(Q(s_t^A, a^A, s_t^B, a^B))$.

The protocol is described in Fig. 4. Threshold decryption is used here, too. First, A sends encrypted Q-values $c(s_t^A, i, j, k)$ with re-randomization for all

- Public input; L, K, learning rate α, discount rate γ
- A's input: (s_t^A, a_t^A), B's input: (s_t^B, a_t^B)
- A's output: Encryption of updated Q-value $c(s_t, a_t)$
- B's output: Nothing

1. A: Initialize $Q(s, a)$ arbitrarily and compute $c(s, a)(= e(Q(s, a)))$ for all (s, a).
2. Interaction with the environment:
   - A: Take action a_t^A and get r_t^A, s_{t+1}^A.
   - B: Take action a_t^B and get r_t^B, s_{t+1}^B.
3. Action selection:
   - A: Choose a_{t+1}^A randomly.
   - B: Choose a_{t+1}^B randomly.
4. Update Q-value:
   (a) A: Send $X^A, \{c_{ik}\}, \{c'_{ik}\}$ to B by eq. 6, 7.
   (b) B: Compute $e(K\Delta Q(s_t, a_t))$ by eq. 5
   (c) B: Do private division of $e(K\Delta Q(s_t, a_t))$ with A, then B learns $e(\Delta Q'(s_t, a_t))$.
   (d) B: Generate $\{\Delta c_{hijk}\}$ by eq. 8 and send it to A.
   (e) A: Update $c(s, a)$ with $\{\Delta c_{hijk}\}$ by eq. 9.

Figure 3. Private update of Q-values in partitioned-by-observation model (SARSA/random action selection)

(i, j, k). For all (i, k), B generates and sends a table $\{c_{ik}\}$ and $\{\sigma_{ik}\}$ whose values are set to

$$c_{ik} = c(s_t^A, i, s_t^B, \pi(k)) \cdot e(-Q_{i\pi(k)}^B), \quad (10)$$
$$\sigma_{ik}^B = d^B(c_{ik}), \quad (11)$$

where $\pi : S^B \mapsto S^B$ is a random permutation and $Q_{i\pi(k)}^B \in_r \mathbb{Z}_N$. At the third step, A recovers $Q_{ik}^A (= Q(s_t^A, i, s_t^B, k) - Q_{ik}^B)$. With these random shares of $Q(s_t^A, i, s_t^B, \pi(k))$, the values $(i^*, k^*) = \arg\max_{(i,k)}(Q_{ik}^A + Q_{ik}^B)$ are obtained by A using private comparison. Finally, B learns $a^{B*} = \pi^{-1}(k^*)$, where π^{-1} is the inverse of π.

Lemma 3. *If A and B behaves semi-honestly, then, after the execution of private greedy action selection, A learns a^{A*} and nothing else. B learns a^{B*} and nothing else.*

Note that a^{B*} is not learned by A because index k is obscured by the random permutation generated by B.

5.2. Security of PPRL

Privacy-preserving SARSA learning is constructed by alternate iterations of private update and random action selection. The policy π can be extracted by computing $\arg\max_a Q(s, a)$ for all (s, a) using private greedy action selection. The security follows from the earlier lemmas:

- A's input: $c(s,a)$ for all (s,a), s_t^A, B's input: s_t^B
- A's output: a^{A*}, B's output: a^{B*}

1. A: For all $i \in S^B, j \in A^A, k \in A^B$, send $c(s_t^A, i, j, k)$ to B.
2. B: For all $j \in A^A, k \in A^B$, compute c_{ik}(eq. 10), σ_{ik}^B(eq. 11) and send $\{c_{ik}\}, \{\sigma_{ik}\}$to A.
3. A: For all $i \in A^A, k \in A^B$, compute $\sigma_{ik}^B = d^B(c_{ik})$. Then, compute Q_{ik}^A by applying the threshold decryption recovery algorithm with public key pk and shares $\sigma_{ik}^A, \sigma_{ik}^B$.
4. A and B: Compute $(i^*, k^*) = \arg\max_{(i,k)}(Q_{ik}^A + Q_{ik}^B)$ by private comparison. (A learns $(i^*.k^*)$.)
5. A: Send k^* to B. Then output $a^{A*} = i^*$.
6. B: Output $a^{B*} = \pi^{-1}(k^*)$.

Figure 4. Private greedy action selection in partitioned-by-observation model

Theorem 1. *SARSA learning with private update of Q-values and random action selection is secure in the sense of Statement 1.*

Privacy-preserving SARSA learning and Q-learning with (ϵ-)greedy action selection can be constructed by combining private update and private greedy random action selection. However, these PPRLs do not follow Statement 1 because it does not allow agents to know greedy actions obtained in the middle of the learning. Therefore, the problem definition is relaxed as follows:

Statement 2. *The ith agent takes H^i as inputs. After the execution of PPRL, all agents learn a series of greedy actions during learning steps and a policy π which is equivalent to π^c. Furthermore, no agent learns anything else.*

Theorem 2. *SARSA and Q-learning with private update of Q-values and private greedy/ϵ-greedy action selection is secure in the sense of Statement 2.*

6. Experimental Results

We performed experiments to examine the efficiency of PPRL. Programs were written in Java 1.5.0. As the cryptosystem, (Dåmgard & Jurik, 2001) with 1024-bit keys was used. For SFE, Fairplay (Malkhi et al., 2004) was used. Experiments were carried out under Linux with 1.2 GHz CPU and 2GB RAM.

6.1. Random Walk Task

This random walk task is partitioned by time. The state space is $S = \{s_1, ..., s_n\}(n = 40)$ and the action space is $A = \{a_1, a_2\}$. The initial and goal states are s_1 and s_n, respectively. When a_1 is taken at $s_p(p \neq n)$, the agent moves to s_{p+1}. When a_2 is taken at $s_p(p \neq 1)$, the agent moves to s_{p-1}, but the agent does not move when $p = 1$. A reward $r = 1$ is given only when the agent takes a_1 at s_{n-1}; else, $r = 0$. The episode is terminated at s_n or after $1,000$ steps.

A learns $15,000$ steps and then B learns $15,000$ steps. CRL, IDRL, PPRL, and SFE were compared. SARSA learning with random or ϵ-greedy action selection was used for all settings. Table 2 shows the comparison results of computational cost, learning accuracy (number of steps to reach the goal state, averaged over 30 trials, and number of trials that successfully reach the goal state), and privacy preservation.

Learning accuracy of PPRL and SFE are the same as CRL because the policy learned by PPRL and SFE are guaranteed to be equal to the one learned by CRL. In contrast, the optimal policy is not obtained successfully by IDRL because learning steps for IDRL agents correspond to the half of others. Because most of the computation time is spent for private division and comparison, computation time with random selection is much smaller than with ϵ-greedy selection. These experiments demonstrate that PPRL obtains good learning accuracy, while IDRL does not, though computation time is larger than DRL and IDRL.

6.2. Load Balancing Task

In these experiments, we consider a load balancing problem (Cogill et al., 2006) in the partitioned-by-observation model with two factories A and B. Each factory can observe its own backlog $s^A, s^B \in \{0, ..., 5\}$. At each time step, each factory decides whether or not to pass a job to other factories; the action variable is $a^A, a^B \in \{0, 1\}$. Jobs arrive and are processed independently at each time step with probability 0.4 and 0.48, respectively. Agent A receives reward $r^A = 50 - (s^A)^2$. If A passes the job to B, then A's reward is reduced by 2 as a cost for redirection. If an overflow happens, the job is lost and $r^A = 0$ is given. Similarly, r^B is computed as well. Perceptions (s^A, a^A, r^A) and (s^B, a^B, r^B) are to be kept private. (In this task, actions cannot be kept private because the parties learn them from whether the job was passed or not.)

Distributed reward DRL (RDRL) (Schneider et al., 1999) is tested in addition to the four RLs tested earlier. RDRL is a variant of DRL, which is the same with IDRL except that global rewards are shared among distributed agents (Schneider et al., 1999). SARSA/ϵ-greedy action selection was used in all settings. Fig 5 shows the changes of sum of global rewards per episode. For avoiding overflows, cooperation be-

Table 2. Comparison of efficiency in random walk tasks

	comp. (sec)	accuracy avg.	accuracy #goal	privacy loss
CRL/rnd.	0.901	40.0	30/30	disclosed all
IDRL/rnd.	0.457	247	8/30	Stmt. 1
PPRL/rnd.	4.71×10^3	40.0	30/30	Stmt. 1
SFE/rnd.	$> 7.0 \times 10^6$	40.0	30/30	Stmt. 1
CRL/ϵ-grd.	0.946	40.0	30/30	disclosed all
IDRL/ϵ-grd.	0.481	—	0/30	Stmt. 2
PPRL/ϵ-grd.	3.36×10^4	40.0	30/30	Stmt. 2
SFE/ϵ-grd.	$> 7.0 \times 10^6$	40.0	30/30	Stmt. 2

Table 3. Comparison of efficiency in load balancing tasks.

	comp. (sec)	accuracy	privacy loss
CRL	5.11	90.0	disclosed all
RDRL	5.24	87.4	partially disclosed
IDRL	5.81	84.2	Stmt. 1
PPRL	8.85×10^5	90.0	Stmt. 2
SFE	$> 2.0 \times 10^7$	90.0	Stmt. 2

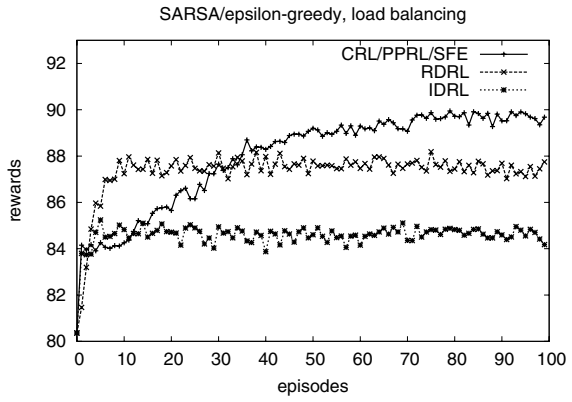


Figure 5. Performance evaluation (sum of global rewards in an episode, normalized by the number of steps in an episode) in load balancing tasks (average of 100 trials).

tween agents is essential in this task. The performance of IDRL agents is inferior to others because selfish behavior is learned. In contrast, CRL, PPRL and SFE agents successfully obtain cooperative behavior. The performance of RDRL is intermediate because perceptions of RDRL agents are limited. Efficiency is shown in Table 3. Since ϵ-greedy action selection was used, the privacy of IDRL, PPRL and SFE follow Statement 2. The privacy preservation of RDRL is between CRL and PPRL. As discussed in Section 1, PPRL achieves both the guarantee of privacy preservation and the optimality which is equivalent to that of CRL; SFE does the same, but at a much higher computational time.

Acknowledgments

This work was started at Tokyo Institute of Technology and carried out partly while the first author was a visitor of the DIMACS Center. The third author is partially supported by the National Science Foundation under grant number CNS-0822269.

References

Abe, N., Verma, N., Apte, C., & Schroko, R. (2004). Cross channel optimized marketing by reinforcement learning. *ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (pp. 767–772).

Cogill, R., Rotkowitz, M., Van Roy, B., & Lall, S. (2006). An Approximate Dynamic Programming Approach to Decentralized Control of Stochastic Systems. *LNCIS*, *329*, 243–256.

Dåmgard, I., & Jurik, M. (2001). A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. *Public Key Cryptography 2001*. Springer.

Gambs, S., Kégl, B., & Aïmeur, E. (2007). Privacy-preserving boosting. *Data Mining and Knowledge Discovery*, *14*, 131–170.

Goldreich, O. (2004). *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press.

Jagannathan, G., & Wright, R. N. (2005). Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. *ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (pp. 593–599).

Kearns, M., Tan, J., & Wortman, J. (2007). Privacy-Preserving Belief Propagation and Sampling. *NIPS 20*.

Lindell, Y., & Pinkas, B. (2002). Privacy Preserving Data Mining. *Journal of Cryptology*, *15*, 177–206.

Malkhi, D., Nisan, N., Pinkas, B., & Sella, Y. (2004). Fairplay: a secure two-party computation system. *USENIX Security Symposium* (pp. 287–302).

Moallemi, C. C., & Roy, B. V. (2004). Distributed optimization in adaptive networks. *NIPS 16*.

Sakuma, J., & Kobayashi, S. (2008). Large-scale k-means Clustering with User-Centric Privacy Preservation. *Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD) 2008, to appear*.

Schneider, J., Wong, W., Moore, A., & Riedmiller, M. (1999). Distributed value functions. *Int'l Conf. on Machine Learning* (pp. 371–378).

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Watkins, C. (1989). *Learning from Delayed Rewards*. Cambridge University.

Yao, A. C.-C. (1986). How to generate and exchange secrets. *IEEE Symposium on Foundations of Computer Science* (pp. 162–167).

Yu, H., Jiang, X., & Vaidya, J. (2006). Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data. *ACM SAC* (pp. 603–610).

Zhang, S., & Makedon, F. (2005). Privacy preserving learning in negotiation. *ACM SAC* (pp. 821–825).

On the Quantitative Analysis of Deep Belief Networks

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Iain Murray MURRAY@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4, Canada

Abstract

Deep Belief Networks (DBN's) are generative models that contain many layers of hidden variables. Efficient greedy algorithms for learning and approximate inference have allowed these models to be applied successfully in many application domains. The main building block of a DBN is a bipartite undirected graphical model called a restricted Boltzmann machine (RBM). Due to the presence of the partition function, model selection, complexity control, and exact maximum likelihood learning in RBM's are intractable. We show that Annealed Importance Sampling (AIS) can be used to efficiently estimate the partition function of an RBM, and we present a novel AIS scheme for comparing RBM's with different architectures. We further show how an AIS estimator, along with approximate inference, can be used to estimate a lower bound on the log-probability that a DBN model with multiple hidden layers assigns to the *test data*. This is, to our knowledge, the first step towards obtaining quantitative results that would allow us to directly assess the performance of Deep Belief Networks as generative models of data.

1. Introduction

Deep Belief Networks (DBN's), recently introduced by Hinton et al. (2006) are probabilistic generative models that contain many layers of hidden variables, in which each layer captures strong high-order correlations between the activities of hidden features in the layer below. The main breakthrough introduced by Hinton et al. was a greedy, layer-by-layer unsupervised learning algorithm that allows efficient training of these deep, hierarchical models. The learning procedure also provides an efficient way of performing approximate inference, which makes the values of the latent variables in the deepest layer easy to infer. These deep generative models have been successfully applied in many application domains (Hinton & Salakhutdinov, 2006; Bengio & LeCun, 2007).

The main building block of a DBN is a bipartite undirected graphical model called the Restricted Boltzmann Machine (RBM). RBM's, and their generalizations to exponential family models, have been successfully applied in collaborative filtering (Salakhutdinov et al., 2007), information and image retrieval (Gehler et al., 2006), and time series modeling (Taylor et al., 2006). A key feature of RBM's is that inference in these models is easy. An unfortunate limitation is that the probability of data under the model is known only up to a computationally intractable normalizing constant, known as the partition function. A good estimate of the partition function would be extremely helpful for model selection and for controlling model complexity, which are important for making RBM's generalize well.

There has been extensive research on obtaining deterministic approximations (Yedidia et al., 2005) or deterministic upper bounds (Wainwright et al., 2005) on the log-partition function of arbitrary discrete Markov random fields (MRF's). These variational methods rely critically on an ability to approximate the entropy of the undirected graphical model. However, for densely connected MRF's, such as RBM's, these methods are unlikely to perform well. There have also been many developments in the use of Monte Carlo methods for estimating the partition function, including Annealed Importance Sampling (AIS) (Neal, 2001), Nested Sampling (Skilling, 2004), and many others (see e.g. Neal (1993)). In this paper we show how one such method, AIS, by taking advantage of the bipartite structure of an RBM, can be used to efficiently estimate its partition function. We further show that this estimator, along with approximate inference, can be used to estimate a lower bound on the log-probability that a DBN model with multiple hidden layers assigns to training or test data. This result allows us to assess the performance of DBN's as generative models and to compare them to other probabilistic models, such as plain mixture models.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

2. Restricted Boltzmann Machines

A Restricted Boltzmann Machine is a particular type of MRF that has a two-layer architecture in which the visible, binary stochastic units $\mathbf{v} \in \{0,1\}^D$ are connected to hidden binary stochastic units $\mathbf{h} \in \{0,1\}^M$. The energy of the state $\{\mathbf{v}, \mathbf{h}\}$ is:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{D}\sum_{j=1}^{M} W_{ij} v_i h_j - \sum_{i=1}^{D} b_i v_i - \sum_{j=1}^{M} a_j h_j, \quad (1)$$

where $\theta = \{W, \mathbf{b}, \mathbf{a}\}$ are the model parameters: W_{ij} represents the symmetric interaction term between visible unit i and hidden unit j; b_i and a_j are bias terms. The probability that the model assigns to a visible vector $\mathbf{v}$ is:

$$p(\mathbf{v}; \theta) = \frac{p^*(\mathbf{v}; \theta)}{Z(\theta)} = \frac{1}{Z(\theta)} \sum_{h} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \quad (2)$$

$$Z(\theta) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \quad (3)$$

where p^* denotes unnormalized probability, and $Z(\theta)$ is the partition function or normalizing constant. The conditional distributions over hidden units $\mathbf{h}$ and visible vector $\mathbf{v}$ are given by logistic functions:

$$p(\mathbf{h}|\mathbf{v}) = \prod_j p(h_j|\mathbf{v}), \quad p(\mathbf{v}|\mathbf{h}) = \prod_i p(v_i|\mathbf{h}) \quad (4)$$

$$p(h_j = 1|\mathbf{v}) = \sigma(\sum_i W_{ij} v_i + a_j) \quad (5)$$

$$p(v_i = 1|\mathbf{h}) = \sigma(\sum_j W_{ij} h_j + b_i), \quad (6)$$

where $\sigma(x) = 1/(1+\exp(-x))$. The derivative of the log-likelihood with respect to the model parameter W can be obtained from Eq. 2:

$$\frac{\partial \ln p(\mathbf{v})}{\partial W_{ij}} = \mathrm{E}_{P_0}[v_i h_j] - \mathrm{E}_{P_{\mathrm{Model}}}[v_i h_j],$$

where $\mathrm{E}_{P_0}[\cdot]$ denotes an expectation with respect to the data distribution and $\mathrm{E}_{P_{\mathrm{Model}}}[\cdot]$ is an expectation with respect to the distribution defined by the model. The expectation $\mathrm{E}_{P_{\mathrm{Model}}}[\cdot]$ cannot be computed analytically. In practice learning is done by following an approximation to the gradient of a different objective function, called the "Contrastive Divergence" (CD) (Hinton, 2002):

$$\Delta W_{ij} = \epsilon\big(\mathrm{E}_{P_0}[v_i h_j] - \mathrm{E}_{P_T}[v_i h_j]\big). \quad (7)$$

The expectation $\mathrm{E}_{P_T}[\cdot]$ represents a distribution of samples from running the Gibbs sampler (Eqs. 5, 6), initialized at the data, for T full steps. Setting $T = \infty$ recovers maximum likelihood learning, although T is typically set to one.

Even though CD learning may work well in practice, the problem of model selection and complexity control still remains. Suppose we have two RBM's with parameter values θ_A and θ_B. Suppose that each RBM has different number of hidden units and was trained using different learning rates and different numbers of CD steps. On the validation set, we are interested in calculating the ratio:

$$\frac{p(\mathbf{v}; \theta_A)}{p(\mathbf{v}; \theta_B)} = \frac{p^*(\mathbf{v}; \theta_A)}{p^*(\mathbf{v}; \theta_B)} \frac{Z(\theta_B)}{Z(\theta_A)},$$

which requires knowing the ratio of partition functions.

3. Estimating Ratios of Partition Functions

Suppose we have two distributions defined on some space $\mathcal{V}$ with probability density functions: $p_A(\mathbf{v}) = p_A^*(\mathbf{v})/Z_A$ and $p_B(\mathbf{v}) = p_B^*(\mathbf{v})/Z_B$. One way to estimate the ratio of normalizing constants is to use a simple importance sampling (IS) method. Suppose that $p_A(\mathbf{v}) \neq 0$ whenever $p_B(\mathbf{v}) \neq 0$:

$$\frac{Z_B}{Z_A} = \frac{\int p_B^*(\mathbf{v}) \mathrm{d}\mathbf{v}}{Z_A} = \int \frac{p_B^*(\mathbf{v})}{p_A^*(\mathbf{v})} p_A(\mathbf{v}) \mathrm{d}\mathbf{v} = \mathrm{E}_{p_A}\left[\frac{p_B^*(\mathbf{v})}{p_A^*(\mathbf{v})}\right].$$

Assuming we can draw independent samples from p_A, the unbiased estimate of the ratio of partition functions can be obtained by using a simple Monte Carlo approximation:

$$\frac{Z_B}{Z_A} \approx \frac{1}{M} \sum_{i=1}^{M} \frac{p_B^*(\mathbf{v}^{(i)})}{p_A^*(\mathbf{v}^{(i)})} \equiv \frac{1}{M} \sum_{i=1}^{M} w^{(i)} = \hat{r}_{\mathrm{IS}}, \quad (8)$$

where $\mathbf{v}^{(i)} \sim p_A$. If p_A and p_B are not close enough, the estimator $\hat{r}_{\mathrm{IS}}$ will be very poor. In high-dimensional spaces, the variance of $\hat{r}_{\mathrm{IS}}$ will be very large (or possibly infinite), unless p_A is a near-perfect approximation to p_B.

3.1. Annealed Importance Sampling (AIS)

Suppose that we can define a sequence of intermediate probability distributions: $p_0, ..., p_K$, with $p_0 = p_A$ and $p_K = p_B$, which satisfy the following conditions:

- C1 $p_k(\mathbf{v}) \neq 0$ whenever $p_{k+1}(\mathbf{v}) \neq 0$.
- C2 We must be able to easily evaluate the unnormalized probability $p_k^*(\mathbf{v})$, $\forall \mathbf{v} \in \mathcal{V}$, $k = 0, ..., K$.
- C3 For each $k = 0, ..., K-1$, we must be able to draw a sample $\mathbf{v}'$ given $\mathbf{v}$ using a Markov chain transition operator $T_k(\mathbf{v}'; \mathbf{v})$ that leaves $p_k(\mathbf{v})$ invariant:

$$\int T_k(\mathbf{v}'; \mathbf{v}) p_k(\mathbf{v}) \mathrm{d}\mathbf{v} = p_k(\mathbf{v}'). \quad (9)$$

- C4 We must be able to draw (preferably independent) samples from p_A.

The transition operators $T_k(\mathbf{v}'; \mathbf{v})$ represent the probability density of transitioning from state $\mathbf{v}$ to $\mathbf{v}'$. Constructing a suitable sequence of intermediate probability distributions

will depend on the problem. One general way to define this sequence is to set:

$$p_k(\mathbf{v}) \propto p_A^*(\mathbf{v})^{1-\beta_k} p_B^*(\mathbf{v})^{\beta_k}, \quad (10)$$

with $0 = \beta_0 < \beta_1 < ... < \beta_K = 1$ chosen by the user. Once the sequence of intermediate distributions has been defined we have:

Annealed Importance Sampling (AIS) run:

1. Generate $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_K$ as follows:
   - Sample $\mathbf{v}_1$ from $p_A = p_0$
   - Sample $\mathbf{v}_2$ given $\mathbf{v}_1$ using T_1
   - ...
   - Sample $\mathbf{v}_K$ given $\mathbf{v}_{K-1}$ using T_{K-1}
2. Set

$$w^{(i)} = \frac{p_1^*(\mathbf{v}_1)}{p_0^*(\mathbf{v}_1)} \frac{p_2^*(\mathbf{v}_2)}{p_1^*(\mathbf{v}_2)} ... \frac{p_{K-1}^*(\mathbf{v}_{K-1})}{p_{K-2}^*(\mathbf{v}_{K-1})} \frac{p_K^*(\mathbf{v}_K)}{p_{K-1}^*(\mathbf{v}_K)}$$

Note that there is no need to compute the normalizing constants of any intermediate distributions. After performing M runs of AIS, the importance weights $w^{(i)}$ can be substituted into Eq. 8 to obtain an estimate of the ratio of partition functions:

$$\frac{Z_B}{Z_A} \approx \frac{1}{M} \sum_{i=1}^{M} w^{(i)} = \hat{r}_{\text{AIS}}. \quad (11)$$

Neal (2005) shows that for sufficiently large number of intermediate distributions K, the variance of $\hat{r}_{\text{AIS}}$ will be proportional to $1/MK$. Provided K is kept large, the total amount of computation can be split in any way between the number of intermediate distributions K and the number of annealing runs M without adversely affecting the accuracy of the estimator. If samples drawn from p_A are independent, the number of AIS runs can be used to control the variance in the estimate of $\hat{r}_{\text{AIS}}$:

$$\text{Var}(\hat{r}_{\text{AIS}}) = \frac{1}{M} \text{Var}(w^{(i)}) \approx \frac{\hat{s}^2}{M} = \hat{\sigma}^2, \quad (12)$$

where $\hat{s}^2$ is estimated simply from the sample variance of the importance weights.

3.2. Ratios of Partition Functions of two RBM's

Suppose we have two RBM's with parameter values $\theta_A = \{W^A, \mathbf{b}^A, \mathbf{a}^A\}$ and $\theta_B = \{W^B, \mathbf{b}^B, \mathbf{a}^B\}$ that define probability distributions p_A and p_B over $\mathcal{V} \in \{0,1\}^D$. Each RBM can have a different number of hidden units $\mathbf{h}^A \in \{0,1\}^{M_A}$ and $\mathbf{h}^B \in \{0,1\}^{M_B}$. The generic AIS intermediate distributions (Eq. 10) would be harder to sample from than an RBM. Instead we introduce the following sequence of distributions for $k = 0, ..., K$:

$$p_k(\mathbf{v}) = \frac{p_k^*(\mathbf{v})}{Z_k} = \frac{1}{Z_k} \sum_{\mathbf{h}} \exp\left(-E_k(\mathbf{v}, \mathbf{h})\right), \quad (13)$$

where the energy function is given by:

$$E_k(\mathbf{v}, \mathbf{h}) = (1 - \beta_k) E(\mathbf{v}, \mathbf{h}^A; \theta_A) + \beta_k E(\mathbf{v}, \mathbf{h}^B; \theta_B), \quad (14)$$

with $0 = \beta_0 < \beta_1 < ... < \beta_K = 1$. For $i = 0$, we have $\beta_0 = 0$ and so $p_0 = p_A$. Similarly, for $i = K$, we have $p_K = p_B$. For the intermediate values of k, we will have some interpolation between p_A and p_B.

Let us now define a Markov chain transition operator $T_k(\mathbf{v}'; \mathbf{v})$ that leaves $p_k(\mathbf{v})$ invariant. Using Eqs. 13, 14, it is straightforward to derive a block Gibbs sampler. The conditional distributions are given by logistic functions:

$$p(h_j^A = 1|\mathbf{v}) = \sigma\left((1 - \beta_k)(\sum_i W_{ij}^A v_i + a_j^A)\right) \quad (15)$$

$$p(h_j^B = 1|\mathbf{v}) = \sigma\left(\beta_k(\sum_i W_{ij}^B v_i + a_j^B)\right) \quad (16)$$

$$p(v_i' = 1|\mathbf{h}) = \sigma\Big((1 - \beta_k)(\sum_j W_{ij}^A h_j^A + b_i^A) + \beta_k(\sum_j W_{ij}^B h_j^B + b_i^B)\Big). \quad (17)$$

Given $\mathbf{v}$, Eqs. 15, 16 are used to stochastically activate hidden units $\mathbf{h}^A$ and $\mathbf{h}^B$. Eq. 17 is then used to draw a new sample $\mathbf{v}'$ as shown in Fig. 1 (left panel). Due to the special structure of RBM's, the cost of summing out $\mathbf{h}$ is linear in the number of hidden units. We can therefore easily evaluate:

$$\begin{aligned} p_k^*(\mathbf{v}) &= \sum_{\mathbf{h}^A, \mathbf{h}^B} e^{(1-\beta_k)E(\mathbf{v}, \mathbf{h}^A; \theta_A) + \beta_k E(\mathbf{v}, \mathbf{h}^B; \theta_B)} \\ &= e^{(1-\beta_k)\sum_i b_i^A v_i} \prod_{j=1}^{M_A} (1 + e^{(1-\beta_k)(\sum_i W_{ij}^A v_i + a_j^A)}) \\ &\times e^{\beta_k \sum_i b_i^B v_i} \prod_{j=1}^{M_B} (1 + e^{\beta_k(\sum_i W_{ij}^B v_i + a_j^B)}). \end{aligned}$$

We will assume that the parameter values of each RBM satisfy $|\theta| < \infty$, in which case $p(\mathbf{v}) > 0$ for all $\mathbf{v} \in \mathcal{V}$. This will ensure that condition C1 of the AIS procedure is always satisfied. We have already shown that conditions C2 and C3 are satisfied. For condition C4, we can run a blocked Gibbs sampler (Eqs. 5, 6) to generate samples from p_A. These sample points will not be independent, but the AIS estimator will still converge to the correct value, provided our Markov chain is ergodic (Neal, 2001). However, assessing the accuracy of this estimator can be difficult, as it depends on both the variance of the importance weights and on autocorrelations in the Gibbs sampler.

3.3. Estimating Partition Functions of RBM's

The partition function of an RBM can be found by finding the ratio to the normalizer for $\theta_A = \{0, \mathbf{b}^A, \mathbf{a}^A\}$, an RBM

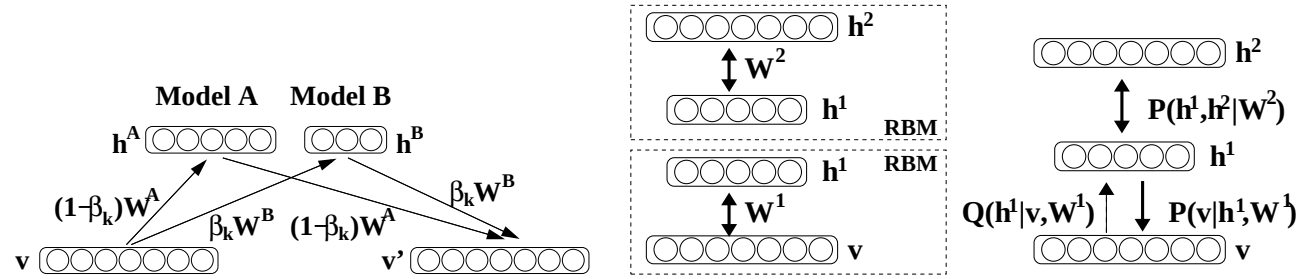


Figure 1. **Left:** The Gibbs transition operator $T_k(\mathbf{v}';\mathbf{v})$ leaves $p_k(\mathbf{v})$ invariant when estimating the ratio of partition functions Z_B/Z_A. **Middle:** Recursive greedy learning consists of learning a stack of RBMs. **Right:** Two-layer DBN as a generative model.

with a zero weight matrix. From Eq. 3, we know:

$$Z_A = 2^{M_A} \prod_i (1 + e^{b_i}). \tag{18}$$

Moreover,

$$p_A(\mathbf{v}) = \prod_i p_A(v_i) = \prod_i 1/(1 + e^{-b_i}),$$

so we can draw exact independent samples from this "base-rate" RBM. AIS in this case allows us to obtain an *unbiased* estimate of the partition function Z_B. This approach closely resembles simulated annealing, since the intermediate distributions of Eq. 13 take form:

$$p_k(\mathbf{v}) = \frac{\exp((1-\beta_k)\mathbf{v}^T\mathbf{b}^A)}{Z_k} \sum_{\mathbf{h}^B} \exp(-\beta_k E(\mathbf{v}, \mathbf{h}^B; \theta_B)).$$

We gradually change β_k (or inverse temperature) from 0 to 1, annealing from a simple "base-rate" model to the final complex model. The importance weights $w^{(i)}$ ensure that AIS produces correct estimates.

4. Deep Belief Networks (DBN's)

In this section we briefly review a greedy learning algorithm for training Deep Belief Networks. We then show how to obtain an estimate of the lower bound on the log-probability that the DBN assigns to the data.

4.1. Greedy Learning of DBN's

Consider learning a DBN with two layers of hidden features as shown in Fig. 1 (right panel). The greedy strategy developed by Hinton et al. (2006) uses a stack of RBM's (Fig. 1, middle panel). We first train the bottom RBM with parameters W^1, as described in section 2.

A key observation is that the RBM's joint distribution $p(\mathbf{v}, \mathbf{h}^1|W^1)$ is identical to that of a DBN with second-layer weights tied to $W^2 = W^{1\top}$. We now consider untying and refining W^2, improving the fit to the training data.

For any approximating distribution $Q(\mathbf{h}^1|\mathbf{v})$, the DBN's log-likelihood has the following variational lower bound:

$$\ln p(\mathbf{v}|W^1, W^2) \;\geq\; \sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{v}) \big[\ln p(\mathbf{h}^1|W^2) \;+\; \ln p(\mathbf{v}|\mathbf{h}^1, W^1) \big] + \mathcal{H}(Q(\mathbf{h}^1|\mathbf{v})), \tag{19}$$

where $\mathcal{H}(\cdot)$ is the entropy functional. We set $Q(\mathbf{h}^1|\mathbf{v}) = p(\mathbf{h}^1|\mathbf{v}, W^1)$ defined by the RBM (Eq. 5). Initially, when $W^2 = W^{1\top}$, Q is the DBN's true factorial posterior over $\mathbf{h}^1$, and the bound is tight. Therefore, any increase in the bound will lead to an increase in the true likelihood of the model. Maximizing the bound of Eq. 19 with frozen W^1 is equivalent to maximizing:

$$\sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{v}) \ln p(\mathbf{h}^1|W^2). \tag{20}$$

This is equivalent to training the second layer RBM with vectors drawn from $Q(\mathbf{h}^1|\mathbf{v})$ as data.

This scheme can be extended by training a third RBM on $\mathbf{h}^2$ vectors drawn from the second RBM. If we initialize $W^3 = W^{2\top}$, we are guaranteed to improve the lower bound on the log-likelihood, though the log-likelihood itself can fall (Hinton et al., 2006). Repeating this greedy, layer-by-layer training several times results in a deep, hierarchical model.

Recursive Greedy Learning Procedure for the DBN.

1. Fit parameters W^1 of a 1-layer RBM to data.
2. Freeze the parameter vector W^1 and use samples from $p(\mathbf{h}^1|\mathbf{v}, W^1)$ as the data for training the next layer of binary features with an RBM.
3. Proceed recursively for as many layers as desired.

In practice, when adding a new layer l, we typically do not initialize $W^l = W^{l-1\top}$, so the number of hidden units of the new RBM does not need to be the same as the number of the visible units of the lower-level RBM.

4.2. Estimating Lower Bounds for DBN's

Consider the same DBN model with two layers of hidden features shown in Fig. 1. The model's joint distribution is:

$$p(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2) = p(\mathbf{v}|\mathbf{h}^1)\, p(\mathbf{h}^2, \mathbf{h}^1), \tag{21}$$

where $p(\mathbf{v}|\mathbf{h}^1)$ is defined by Eq. 6), and $p(\mathbf{h}^1, \mathbf{h}^2)$ is the joint distribution defined by the second layer RBM. Note that $p(\mathbf{v}|\mathbf{h}^1)$ is normalized.

By explicitly summing out $\mathbf{h}^2$, we can easily evaluate an unnormalized probability $p^*(\mathbf{v},\mathbf{h}^1) = Zp(\mathbf{v},\mathbf{h}^1)$. Using the approximating factorial distribution Q, which we get as a byproduct of the greedy learning procedure, and the variational lower bound of Eq. 19, we obtain:

$$\ln \sum_{\mathbf{h}^1} p(\mathbf{v},\mathbf{h}^1) \geq \sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{v}) \ln p^*(\mathbf{v},\mathbf{h}^1) - \ln Z + \mathcal{H}(Q(\mathbf{h}^1|\mathbf{v})) = B(\mathbf{v}). \quad (22)$$

The entropy term $\mathcal{H}(\cdot)$ can be computed analytically, since Q is factorial. The partition function Z is estimated by running AIS on the top-level RBM. And the expectation term can be estimated by a simple Monte Carlo approximation:

$$\sum_{\mathbf{h}^1} Q(\mathbf{h}^1|\mathbf{v}) \ln p^*(\mathbf{v},\mathbf{h}^1) \approx \frac{1}{M} \sum_{i=1}^{M} \ln p^*(\mathbf{v},\mathbf{h}^{1(i)}), \quad (23)$$

where $\mathbf{h}^{1(i)} \sim Q(\mathbf{h}^1|\mathbf{v})$. The variance of this Monte Carlo estimator will be proportional to $1/M$ provided the variance of $\ln p^*(\mathbf{v},\mathbf{h}^{1(i)})$ is finite. In general, we will be interested in calculating the lower bound averaged over the test set containing N_t samples, so

$$\frac{1}{N_t} \sum_{n=1}^{N_t} B(\mathbf{v}^n) \approx \frac{1}{N_t} \sum_{n=1}^{N_t} \left[\frac{1}{M} \sum_{i=1}^{M} \ln p^*(\mathbf{v}^n,\mathbf{h}^{1(i)}) + \mathcal{H}(Q(\mathbf{h}^1|\mathbf{v}^n)) \right] - \ln \hat{Z} = \hat{r}_B - \ln \hat{Z} = \hat{r}_{\text{Bound}}. \quad (24)$$

In this case the variance of the estimator induced by the Monte Carlo approximation will asymptotically scale as $1/(N_t M)$. We will show in the experimental results section that the value of M can be small provided N_t is large.

The error of the overall estimator $\hat{r}_{\text{Bound}}$ in Eq. 24 will be mostly dominated by the error in the estimate of $\ln Z$. In our experiments, we obtained unbiased estimates of $\hat{Z}$ and its standard deviation $\hat{\sigma}$ using Eqs. 11, 12. We report $\ln \hat{Z}$ and $\ln(\hat{Z} \pm \hat{\sigma})$.

Estimating this lower bound for Deep Belief Networks with more layers is now straightforward. Consider a DBN with L hidden layers. The model's joint distribution and its approximate posterior distribution Q are given by:

$$p(\mathbf{v},\mathbf{h}^1,...,\mathbf{h}^L) = p(\mathbf{v}|\mathbf{h}^1)...p(\mathbf{h}^{L-2}|\mathbf{h}^{L-1})p(\mathbf{h}^{L-1},\mathbf{h}^L)$$
$$Q(\mathbf{h}^1,...,\mathbf{h}^L|\mathbf{v}) = Q(\mathbf{h}^1|\mathbf{v})Q(\mathbf{h}^2|\mathbf{h}^1)...Q(\mathbf{h}^L|\mathbf{h}^{L-1}).$$

The bound can now be obtained by using Eq. 22. Note that most of the computation resources will be spent on estimating the partition function Z of the top level RBM.

5. Experimental Results

In our experiments we used the MNIST digit dataset, which contains 60,000 training and 10,000 test images of ten handwritten digits (0 to 9), with 28×28 pixels. The dataset was binarized: each pixel value was stochastically set to 1 in proportion to its pixel intensity. Samples from the training set are shown in Fig. 2 (top left panel). Annealed importance sampling requires the β_k that define a sequence of intermediate distributions. In all of our experiments this sequence was chosen by quickly running a few preliminary experiments and picking the spacing of β_k so as to minimize the log variance of the final importance weights. The biases $\mathbf{b}^A$ of a base-rate model (see Eq. 18) were set by maximum likelihood, then smoothed to ensure that $p(\mathbf{v}) > 0, \forall\, \mathbf{v} \in \mathcal{V}$. Code that can be used to reproduce experimental results is available at www.cs.toronto.edu/∼rsalakhu.

5.1. Estimating partition functions of RBM's

In our first experiment we trained three RBM's on the MNIST digits. The first two RBM's had 25 hidden units and were learned using CD (section 2) with T=1 and T=3 respectively. We call these models CD1(25) and CD3(25). The third RBM had 20 hidden units and was learned using CD with T=1. For all three models we can calculate the exact value of the partition function simply by summing out the 784 visible units for each configuration of the hiddens. For all three models we used 500 β_k spaced uniformly from 0 to 0.5, 4,000 β_k spaced uniformly from 0.5 to 0.9, and 10,000 β_k spaced uniformly from 0.9 to 1.0, with a total of 14,500 intermediate distributions.

Table 1 shows that for all three models, using only 10 AIS runs, we were able to obtain good estimates of partition functions in just 20 seconds on a Pentium Xeon 3.00GHz machine. For model CD1(25), however, the variance of the estimator was high, even with 100 AIS runs. However, figure 3 (top row) reveals that as the number of annealing runs is increased, AIS can almost exactly recover the true value of the partition function across all three models.

We also estimated the ratio of normalizing constants of two RBM's that have different numbers of hidden units: CD1(20) and CD1(25). This estimator could be used to do complexity control. In detail, using 100 AIS runs with uniform spacing of 10,000 β_k, we obtained $\ln \hat{r}_{\text{AIS}} = \ln(Z_{\text{CD1(20)}}/Z_{\text{CD1(25)}}) = -24.49$ with an error estimate $\ln(\hat{r}_{\text{AIS}} \pm 3\hat{\sigma}) = (-24.19, -24.93)$. Each sample from CD1(25) was generated by starting a Markov chain at the previous sample and running it for 10,000 steps. Compared to the true value of -24.18, this result suggests that our estimates may have a small systematic error due to the Markov chain failing to visit some modes.

Our second experiment consisted of training two more realistic models: CD1(500) and CD3(500). We used exactly the same spacing of β_k as before and exactly the same base-rate model. Results are shown in table 1 (bottom row). For each model we were able to get what appears to be a rather

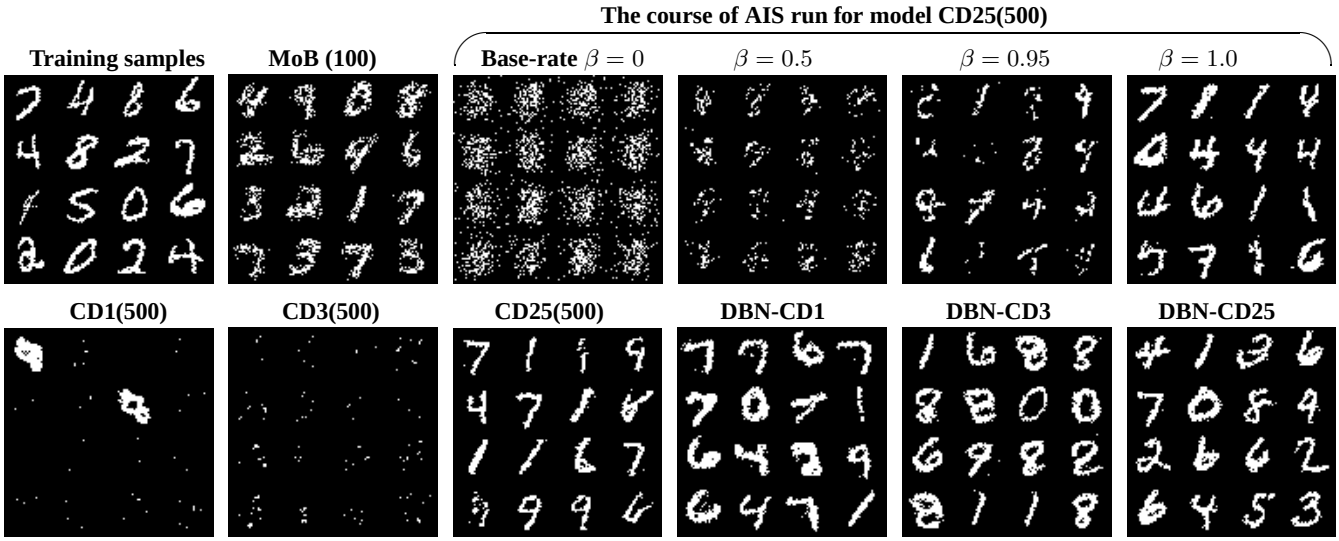


Figure 2. **Top row:** First two panels show random samples from the training set and a mixture of Bernoullis model with 100 components. The last 4 panels display the course of 16 AIS runs for CD25(500) model by starting from a simple base-rate model and annealing to the final complex model. **Bottom row:** Random samples generated from three RBM's and corresponding three DBN's models.

Table 1. Results of estimating partition functions of RBM's along with the estimates of the average training and test log-probabilities. For all models we used 14,500 intermediate distributions.

AIS Runs		True $\ln Z$	Estimates $\ln \hat{Z}$	Estimates $\ln(\hat{Z} \pm \hat{\sigma})$	Estimates $\ln(\hat{Z} \pm 3\hat{\sigma})$	Time (mins)	Avg. Test log-prob. true	Avg. Test log-prob. estimate	Avg. Train log-prob. true	Avg. Train log-prob. estimate
100	CD1(25)	255.41	256.52	255.00, 257.10	0.0000, 257.73	3.3	-151.57	-152.68	-152.35	-153.46
	CD3(25)	307.47	307.63	307.44, 307.79	306.91, 308.05	3.3	-143.03	-143.20	-143.94	-144.11
	CD1(20)	279.59	279.57	279.43, 279.68	279.12, 279.87	3.1	-164.52	-164.50	-164.89	-164.87
100	CD1(500)	—	350.15	350.04, 350.25	349.77, 350.42	10.4	—	-125.53	—	-122.86
	CD3(500)	—	280.09	279.99, 280.17	279.76, 280.33	10.4	—	-105.50	—	-102.81
	CD25(500)	—	451.28	451.19, 451.37	450.97, 451.52	10.4	—	-86.34	—	-83.10

accurate estimate of Z. Of course, we are relying on an empirical estimate of AIS's accuracy, which could potentially be misleading. Nonetheless, Fig. 3 (bottom row) shows that as we increase the number of annealing runs, the value of the estimator does not oscillate drastically.

While performing these tests, we observed that contrastive divergence learning with T=3 results in considerably better generative model than CD learning with T=1: the difference of 20 nats is striking! Clearly, the widely used practice of CD learning with T=1 is a rather poor "substitute" for maximum likelihood learning. Inspired by this result, we trained a model by starting with T=1, and gradually increasing T to 25 during the course of CD training, as suggested by (Carreira-Perpinan & Hinton, 2005). We call this model CD25(500). Training this model was computationally much more demanding. However, the estimate of the average test log-probability for this model was about -86, which is 39 and 19 nats better than the CD1(500) and CD3(500) models respectively. Fig. 2 (bottom row) shows samples generated from all three models by randomly initializing binary states of the visible units and running alternating Gibbs for 100,000 steps. Certainly, samples generated by CD25(500) look much more like the real handwritten digits, than either CD1(500) or CD3(500).

We also obtained an estimate of the log ratio of two partition functions $\hat{r}_{\text{AIS}} = \ln Z_{\text{CD25(500)}}/Z_{\text{CD3(500)}} = 169.96$, using 10,000 β_k and 100 annealing runs. The estimates of the individual log-partition functions were $\ln \hat{Z}_{\text{CD25(500)}} = 451.28$ and $\ln \hat{Z}_{\text{CD3(500)}} = 280.09$, in which case the log ratio is $451.28 - 280.09 = 171.19$. This is in agreement (to within three standard deviations) with the direct estimate of the ratio, $\hat{r}_{\text{AIS}} = 169.96$.

For a simple comparison we also trained several mixture of Bernoullis models (see Fig. 2, top left panel) with 10, 100, and 500 components. The corresponding average test log-probabilities were -168.95, -142.63, and -137.64. The data generated from the mixture model looks better than CD3(500), although our quantitive results reveal this is due to over-fitting. The RBM's make much better predictions.

5.2. Estimating lower bounds for DBN's

We greedily trained three DBN models with two hidden layers. The first model, called DBN-CD1, was greedily

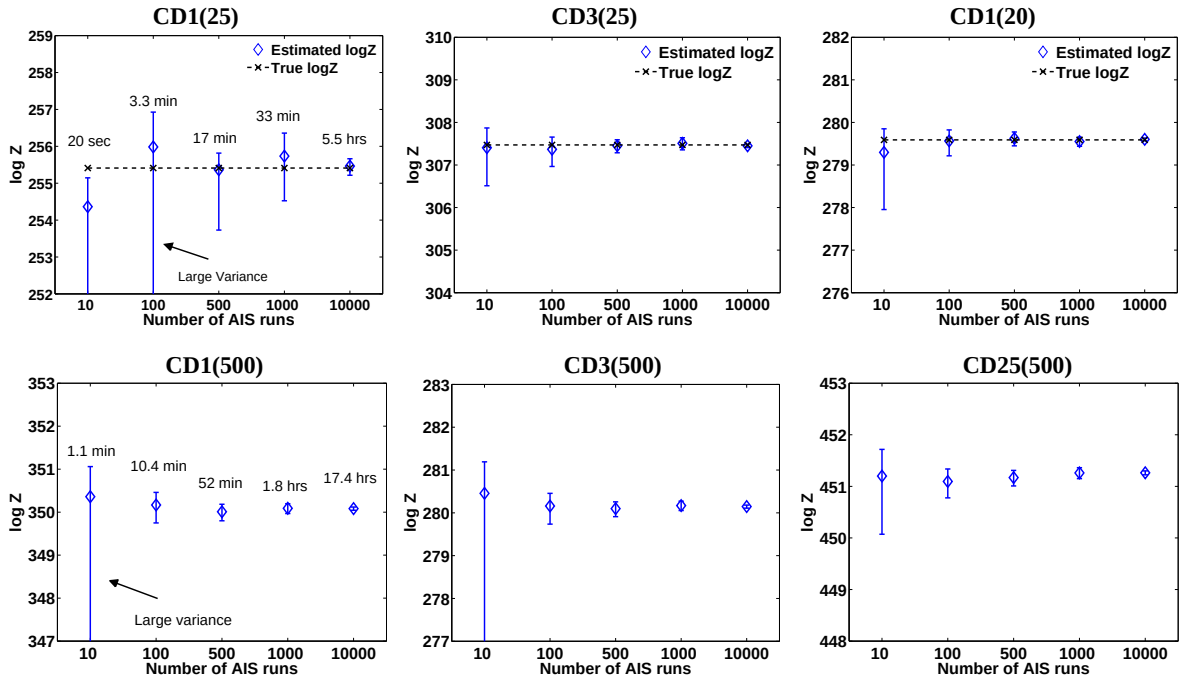


Figure 3. Estimates of the log-partition functions $\ln \hat{Z}$ as we increase the number of annealing runs. The error bars show $\ln(\hat{Z} \pm 3\hat{\sigma})$.

learned by freezing the parameter vector of the CD1(500) model and learning the 2^{nd} layer RBM with 2000 hidden units using CD with T=1. Similarly, the other two models, DBN-CD3 and DBN-CD25, added 2000 hidden units on top of CD3(500) and CD25(500), using CD with T=3 and T=25 respectively. Training the DBN's took roughly three times longer than the RBM's.

Table 2 shows the results. We used 15,000 intermediate distributions and 500 annealing runs to estimate the partition function of the 2^{nd} layer RBM. This took 2.3 hours. Further sampling was required for the simple Monte Carlo approximation of Eq. 23. We used M=5 samples from the approximating distribution $Q(\mathbf{h}|\mathbf{v})$ for each data vector $\mathbf{v}$. Setting M=100 did not make much difference. Table 2 also reports the empirical error in the estimate of the lower bound $\hat{r}_{\text{Bound}}$. From Eq. 24, we have $\text{Var}(\hat{r}_{\text{Bound}}) = \text{Var}(\hat{r}_B) + \text{Var}(\ln \hat{Z})$, both of which are shown in table 2. Note that models DBN-CD1 and DBN-CD3 significantly outperform their single layer counterparts: CD1(500) and CD3(500). Adding a second layer for those two models improves model performance by at least 25 and 7 nats. This corresponds to a dramatic improvement in the quality of samples generated from the models (Fig. 2, bottom row).

Observe that greedy learning of DBN's does not appear to suffer severely from overfitting. For single layer models, the difference between the estimates of training and test log-probabilities was about 3 nats. For DBN's, the corresponding difference in the estimates of the lower bounds was about 4 nats, even though adding a second layer introduced over twice as many (or one million) new parameters.

Table 2. Results of estimating lower bounds $\hat{r}_{\text{Bound}}$ (Eq. 24) on the average training and test log-probabilities for DBN's. On average, the total error of the estimator is about $\pm$ 2 nats.

	Model	Avg. bound log-prob	Error $\hat{r}_B$ ± 3 std	AIS error $\ln(\hat{Z} \pm 3\hat{\sigma}) - \ln \hat{Z}$
Test	DBN-CD1	-100.64	± 0.77	$-1.43, +0.57$
	DBN-CD3	-98.29	± 0.75	$-0.91, +0.31$
	DBN-CD25	-86.22	± 0.67	$-0.84, +0.65$
Train	DBN-CD1	-97.67	± 0.30	$-1.43, +0.57$
	DBN-CD3	-94.86	± 0.29	$-0.91, +0.31$
	DBN-CD25	-82.47	± 0.25	$-0.84, +0.65$

The result of our experiments for DBN-CD25, however, was very different. For this model, on the test data we obtained $\hat{r}_{\text{Bound}} = -86.22$. This is comparable to the estimate of -86.34 for the average test log-probability of the CD25(500) model. Clearly, we cannot confidently assert that DBN-CD25 is a better generative model compared to the carefully trained single layer RBM. This peculiar result also supports previous claims that if the first level RBM already models data well, adding extra layers will not help (LeRoux & Bengio, 2008; Hinton et al., 2006). As an additional test, instead of randomly initializing parameters of the 2^{nd} layer RBM, we initialized it by using the same parameters as the 1^{st} layer RBM but with hidden and visible units switched (see Fig. 1). This initialization ensures that the distribution over the visible units $\mathbf{v}$ defined by the two-layer DBN is *exactly the same* as the distribution over $\mathbf{v}$ defined by the 1^{st} layer RBM. Therefore, after learning parameters of the 2^{nd} layer RBM, the lower bound on the training data log-likelihood can only improve. After care-

fully training the second level RBM, our estimate of the lower bound on the test log-probability was only -85.97. Once again, we cannot confidently claim that adding an extra layer in this case yields better generalization.

6. Discussions

The original paper of Hinton et al. (2006) showed that for DBN's, each additional layer increases a lower bound (see Eq. 19) on the log-probability of the *training* data, provided the number of hidden units per layer does not decrease. However, assessing generalization performance of these generative models is quite difficult, since it requires enumeration over an exponential number of terms. In this paper we developed an annealed importance sampling procedure that takes advantage of the bipartite structure of the RBM. This can provide a good estimate of the partition function in a reasonable amount of computer time. Furthermore, we showed that this estimator, along with approximate inference, can be used to obtain an estimate of the lower bound on the log-probability of the *test* data, thus allowing us to obtain some quantitative evaluation of the generalization performance of these deep hierarchical models.

There are some disadvantages to using AIS. There is a need to specify the β_k that define a sequence of intermediate distributions. The number and the spacing of β_k will be problem dependent and will affect the variance of the estimator. We also have to rely on the empirical estimate of AIS accuracy, which could potentially be very misleading (Neal, 2001; Neal, 2005). Even though AIS provides an unbiased estimator of Z, it occasionally gives large overestimates and usually gives small underestimates, so in practice, it is more likely to underestimate of the true value of the partition function, which will result in an overestimate of the log-probability. But these drawbacks should not result in disfavoring the use of AIS for RBM's and DBN's: it is much better to have a slightly unreliable estimate than no estimate at all, or an extremely indirect estimate, such as discriminative performance (Hinton et al., 2006).

We find AIS and other stochastic methods attractive as they can just as easily be applied to undirected graphical models that generalize RBM's and DBN's to exponential family distributions. This will allow future application to models of real-valued data, such as image patches (Osindero & Hinton, 2008), or count data (Gehler et al., 2006).

Another alternative would be to employ deterministic approximations (Yedidia et al., 2005) or deterministic upper bounds (Wainwright et al., 2005) on the log-partition function. However, for densely connected MRF's, we would not expect these methods to work well. Indeed, preliminary results suggest that these methods provide quite inaccurate estimates of (or very loose upper bounds on) the partition function, even for small RBM's when *trained on real data*.

Acknowledgments

We thank Geoffrey Hinton and Radford Neal for many helpful suggestions. This research was supported by NSERC and CFI. Iain Murray is supported by the government of Canada.

References

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*. MIT Press.

Carreira-Perpinan, M., & Hinton, G. (2005). On contrastive divergence learning. *10th Int. Workshop on Artificial Intelligence and Statistics (AISTATS'2005)*.

Gehler, P., Holub, A., & Welling, M. (2006). The Rate Adapting Poisson (RAP) model for information retrieval and object recognition. *Proceedings of the 23rd International Conference on Machine Learning*.

Hinton, & Salakhutdinov (2006). Reducing the dimensionality of data with neural networks. *Science, 313*, 504 – 507.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation, 14*, 1711–1800.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*, 1527–1554.

LeRoux, N., & Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *To appear in Neural Computation*.

Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods* (Technical Report CRG-TR-93-1). Department of Computer Science, University of Toronto.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing, 11*, 125–139.

Neal, R. M. (2005). *Estimating ratios of normalizing constants using linked importance sampling* (Technical Report 0511). Department of Statistics, University of Toronto.

Osindero, S., & Hinton, G. (2008). Modeling image patches with a directed hierarchy of Markov random fields. *NIPS 20*. Cambridge, MA: MIT Press.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. *Proceedings of the Twenty-fourth International Conference (ICML 2004)*.

Skilling, J. (2004). Nested sampling. *Bayesian inference and maximum entropy methods in science and engineering, AIP Conference Proceeedings, 735*, 395–405.

Taylor, G. W., Hinton, G. E., & Roweis, S. T. (2006). Modeling human motion using binary latent variables. *Advances in Neural Information Processing Systems*. MIT Press.

Wainwright, M. J., Jaakkola, T., & Willsky, A. S. (2005). A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory, 51*, 2313–2335.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory, 51*, 2282–2312.

Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Andriy Mnih AMNIH@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4, Canada

Abstract

Low-rank matrix approximation methods provide one of the simplest and most effective approaches to collaborative filtering. Such models are usually fitted to data by finding a MAP estimate of the model parameters, a procedure that can be performed efficiently even on very large datasets. However, unless the regularization parameters are tuned carefully, this approach is prone to overfitting because it finds a single point estimate of the parameters. In this paper we present a fully Bayesian treatment of the Probabilistic Matrix Factorization (PMF) model in which model capacity is controlled automatically by integrating over all model parameters and hyperparameters. We show that Bayesian PMF models can be efficiently trained using Markov chain Monte Carlo methods by applying them to the Netflix dataset, which consists of over 100 million movie ratings. The resulting models achieve significantly higher prediction accuracy than PMF models trained using MAP estimation.

1. Introduction

Factor-based models have been used extensively in the domain of collaborative filtering for modelling user preferences. The idea behind such models is that preferences of a user are determined by a small number of unobserved factors. In a linear factor model, a user's rating of an item is modelled by the inner product of an item factor vector and a user factor vector. This means that the $N \times M$ preference matrix of ratings that N users assign to M movies is modeled by the product of an $D \times N$ user coefficient matrix U and a $D \times M$ factor matrix V (Rennie & Srebro, 2005; Srebro & Jaakkola, 2003). Training such a model amounts to finding the best rank-D approximation to the observed $N \times M$ target matrix R under the given loss function.

A variety of probabilistic factor-based models have been proposed (Hofmann, 1999; Marlin, 2004; Marlin & Zemel, 2004; Salakhutdinov & Mnih, 2008). In these models factor variables are assumed to be marginally independent while rating variables are assumed to be conditionally independent given the factor variables. The main drawback of such models is that inferring the posterior distribution over the factors given the ratings is intractable. Many of the existing methods resort to performing MAP estimation of the model parameters. Training such models amounts to maximizing the log-posterior over model parameters and can be done very efficiently even on very large datasets.

In practice, we are usually interested in predicting ratings for new user/movie pairs rather than in estimating model parameters. This view suggests taking a Bayesian approach to the problem which involves integrating out the model parameters. In this paper, we describe a fully Bayesian treatment of the Probabilistic Matrix Factorization (PMF) model which has been recently applied to collaborative filtering (Salakhutdinov & Mnih, 2008). The distinguishing feature of our work is the use of Markov chain Monte Carlo (MCMC) methods for approximate inference in this model. In practice, MCMC methods are rarely used on large-scale problems because they are perceived to be very slow by practitioners. In this paper we show that MCMC can be successfully applied to the large, sparse, and very imbalanced Netflix dataset, containing over 100 million user/movie ratings. We also show that it significantly increases the model's predictive accuracy, especially for the infrequent users, compared to the standard PMF models trained using MAP with regularization parameters that have been carefully tuned on the validation set.

Previous applications of Bayesian matrix factorization methods to collaborative filtering (Lim & Teh, 2007; Raiko et al., 2007) have used variational approxima-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

tions for performing inference. These methods attempt to approximate the true posterior distribution by a simpler, factorized distribution under which the user factor vectors are independent of the movie factor vectors. The consequence of this assumption is that that the variational posterior distributions over the factor vectors is a product of two multivariate Gaussians: one for the viewer factor vectors and one for the movie factor vectors. This assumption of independence between the viewer and movie factors seems unreasonable, and, as our experiments demonstrate, the distributions over factors in such models turn out to be non-Gaussian. This conclusion is supported by the fact that the Bayesian PMF models outperform their MAP trained counterparts by a much larger margin than the variationally trained models do.

2. Probabilistic Matrix Factorization

Probabilistic Matrix Factorization (PMF) is a probabilistic linear model with Gaussian observation noise (see Fig. 1, left panel). Suppose we have N users and M movies. Let R_{ij} be the rating value of user i for movie j, U_i and V_j represent D-dimensional user-specific and movie-specific latent feature vectors respectively. The conditional distribution over the observed ratings $R \in \mathbb{R}^{N\times M}$ (the likelihood term) and the prior distributions over $U \in \mathbb{R}^{D\times N}$ and $V \in \mathbb{R}^{D\times M}$ are given by:

$$p(R|U,V,\alpha) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^T V_j, \alpha^{-1})\right]^{I_{ij}} \quad (1)$$

$$p(U|\alpha_U) = \prod_{i=1}^{N}\mathcal{N}(U_i|0,\alpha_U^{-1}I) \quad (2)$$

$$p(V|\alpha_V) = \prod_{j=1}^{M}\mathcal{N}(V_j|0,\alpha_V^{-1}I), \quad (3)$$

where $\mathcal{N}(x|\mu,\alpha^{-1})$ denotes the Gaussian distribution with mean μ and precision α, and I_{ij} is the indicator variable that is equal to 1 if user i rated movie j and equal to 0 otherwise.

Learning in this model is performed by maximizing the log-posterior over the movie and user features with fixed hyperparameters (i.e. the observation noise variance and prior variances):

$$\ln p(U,V|R,\alpha,\alpha_V,\alpha_U) = \ln p(R|U,V,\alpha) + \\ + \ln p(U|\alpha_U) + \ln p(V|\alpha_V) + C,$$

where C is a constant that does not depend on the parameters. Maximizing this posterior distribution with respect to U and V is equivalent to minimizing the sum-of-squares error function with quadratic regularization terms:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(R_{ij} - U_i^T V_j\right)^2 \\ + \frac{\lambda_U}{2}\sum_{i=1}^{N} \| U_i \|_{\mathrm{Fro}}^2 + \frac{\lambda_V}{2}\sum_{j=1}^{M} \| V_j \|_{\mathrm{Fro}}^2, \quad (4)$$

where $\lambda_U = \alpha_U/\alpha$, $\lambda_V = \alpha_V/\alpha$, and $\| \cdot \|_{\mathrm{Fro}}^2$ denotes the Frobenius norm. A local minimum of the objective function given by Eq. 4 can be found by performing gradient descent in U and V.

The main drawback of this training procedure is the need for manual complexity control that is essential to making the model generalize well, particularly on sparse and imbalanced datasets. One way to control the model complexity is to search for appropriate values of regularization parameters λ_U and λ_V defined above. We could, for example, consider a set of reasonable parameter values, train a model for each setting of the parameters, and choose the model that performs best on the validation set. This approach however is computationally very expensive, since it requires training a multitude of models instead of training a single one.

Alternatively, we could introduce priors for the hyperparameters and maximize the log-posterior of the model over *both parameters and hyperparameters*, which allows model complexity to be controlled automatically based on the training data (Nowlan & Hinton, 1992; Salakhutdinov & Mnih, 2008). Though this approach has been shown to work in practice it is not well-grounded theoretically, and it is not difficult to construct a simple example for which such joint optimization would not produce the desired results.

In the next section we describe a fully Bayesian treatment of the PMF model with model parameters and hyperparameters integrated out using MCMC methods, which provides fully automatic complexity control.

3. Bayesian Probabilistic Matrix Factorization

3.1. The Model

The graphical model representing Bayesian PMF is shown in Fig. 1 (right panel). As in PMF, the likelihood of the observed ratings is given by Eq. 1. The prior distributions over the user and movie feature vec-

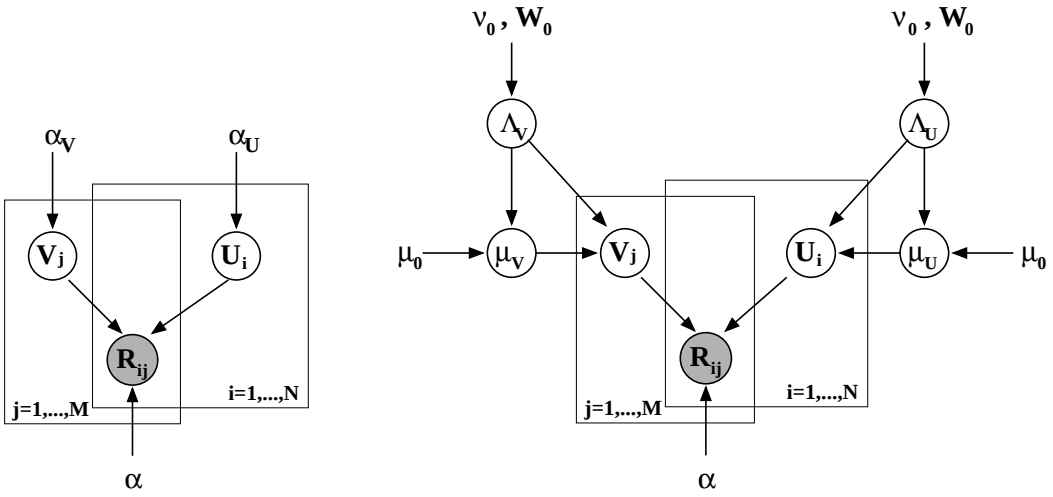


Figure 1. The left panel shows the graphical model for Probabilistic Matrix Factorization (PMF). The right panel shows the graphical model for Bayesian PMF.

tors are assumed to be Gaussian:

$$p(U|\mu_U, \Lambda_U) = \prod_{i=1}^{N} \mathcal{N}(U_i|\mu_U, \Lambda_U^{-1}), \tag{5}$$

$$p(V|\mu_V, \Lambda_V) = \prod_{i=1}^{M} \mathcal{N}(V_i|\mu_V, \Lambda_V^{-1}). \tag{6}$$

We further place Gaussian-Wishart priors on the user and movie hyperparameters $\Theta_U = \{\mu_U, \Lambda_U\}$ and $\Theta_V = \{\mu_V, \Lambda_V\}$:

$$\begin{aligned} p(\Theta_U|\Theta_0) &= p(\mu_U|\Lambda_U)p(\Lambda_U) \\ &= \mathcal{N}(\mu_U|\mu_0, (\beta_0\Lambda_U)^{-1})\mathcal{W}(\Lambda_U|W_0, \nu_0), \end{aligned} \tag{7}$$

$$\begin{aligned} p(\Theta_V|\Theta_0) &= p(\mu_V|\Lambda_V)p(\Lambda_V) \\ &= \mathcal{N}(\mu_V|\mu_0, (\beta_0\Lambda_V)^{-1})\mathcal{W}(\Lambda_V|W_0, \nu_0). \end{aligned} \tag{8}$$

Here $\mathcal{W}$ is the Wishart distribution with ν_0 degrees of freedom and a $D \times D$ scale matrix W_0:

$$\mathcal{W}(\Lambda|W_0, \nu_0) = \frac{1}{C}|\Lambda|^{(\nu_0 - D - 1)/2} \exp\left(-\frac{1}{2}\mathrm{Tr}(W_0^{-1}\Lambda)\right),$$

where C is the normalizing constant. For convenience we also define $\Theta_0 = \{\mu_0, \nu_0, W_0\}$. In our experiments we also set $\nu_0 = D$ and W_0 to the identity matrix for both user and movie hyperparameters and choose $\mu_0 = 0$ by symmetry.

3.2. Predictions

The predictive distribution of the rating value R_{ij}^* for user i and query movie j is obtained by marginalizing over model parameters and hyperparameters:

$$\begin{aligned} p(R_{ij}^*|R, \Theta_0) = \iint p(R_{ij}^*|U_i, V_j)p(U, V|R, \Theta_U, \Theta_V) \\ p(\Theta_U, \Theta_V|\Theta_0)\mathrm{d}\{U, V\}\mathrm{d}\{\Theta_U, \Theta_V\}. \end{aligned} \tag{9}$$

Since exact evaluation of this predictive distribution is analytically intractable due to the complexity of the posterior we need to resort to approximate inference.

One choice would be to use variational methods (Hinton & van Camp, 1993; Jordan et al., 1999) that provide deterministic approximation schemes for posteriors. In particular, we could approximate the true posterior $p(U, V, \Theta_U, \Theta_V|R)$ by a distribution that factors, with each factor having a specific parametric form such as a Gaussian distribution. This approximate posterior would allow us to approximate the integrals in Eq. 9. Variational methods have become the methodology of choice, since they typically scale well to large applications. However, they can produce inaccurate results because they tend to involve overly simple approximations to the posterior.

MCMC-based methods (Neal, 1993), on the other hand, use the Monte Carlo approximation to the predictive distribution of Eq. 9 given by:

$$p(R_{ij}^*|R, \Theta_0) \approx \frac{1}{K}\sum_{k=1}^{K} p(R_{ij}^*|U_i^{(k)}, V_j^{(k)}). \tag{10}$$

The samples $\{U_i^{(k)}, V_j^{(k)}\}$ are generated by running a Markov chain whose stationary distribution is the posterior distribution over the model parameters and hyperparameters $\{U, V, \Theta_U, \Theta_V\}$. The advantage of

the Monte Carlo-based methods is that asymptotically they produce exact results. In practice, however, MCMC methods are usually perceived to be so computationally demanding that their use is limited to small-scale problems.

3.3. Inference

One of the simplest MCMC algorithms is the Gibbs sampling algorithm, which cycles through the latent variables, sampling each one from its distribution conditional on the current values of all other variables. Gibbs sampling is typically used when these conditional distributions can be sampled from easily.

Due to the use of conjugate priors for the parameters and hyperparameters in the Bayesian PMF model, the conditional distributions derived from the posterior distribution are easy to sample from. In particular, the conditional distribution over the user feature vector U_i, conditioned on the movie features, observed user rating matrix R, and the values of the hyperparameters is Gaussian:

$$p(U_i|R,V,\Theta_U,\alpha) = \mathcal{N}\big(U_i|\mu_i^*,\left[\Lambda_i^*\right]^{-1}\big) \sim \prod_{j=1}^{M}\left[\mathcal{N}(R_{ij}|U_i^TV_j,\alpha^{-1})\right]^{I_{ij}} p(U_i|\mu_U,\Lambda_U), \tag{11}$$

where

$$\Lambda_i^* = \Lambda_U + \alpha\sum_{j=1}^{M}\left[V_jV_j^T\right]^{I_{ij}} \tag{12}$$

$$\mu_i^* = [\Lambda_i^*]^{-1}\left(\alpha\sum_{j=1}^{M}\left[V_jR_{ij}\right]^{I_{ij}} + \Lambda_U\mu_U\right). \tag{13}$$

Note that the conditional distribution over the user latent feature matrix U factorizes into the product of conditional distributions over the individual user feature vectors:

$$p(U|R,V,\Theta_U) = \prod_{i=1}^{N} p(U_i|R,V,\Theta_U).$$

Therefore we can easily speed up the sampler by sampling from these conditional distributions in parallel. The speedup could be substantial, particularly when the number of users is large.

The conditional distribution over the user hyperparameters conditioned on the user feature matrix U is given by the Gaussian-Wishart distribution:

$$p(\mu_U,\Lambda_U|U,\Theta_0) = \mathcal{N}(\mu_U|\mu_0^*,(\beta_0^*\Lambda_U)^{-1})\mathcal{W}(\Lambda_U|W_0^*,\nu_0^*), \tag{14}$$

where

$$\mu_0^* = \frac{\beta_0\mu_0 + N\bar{U}}{\beta_0+N}, \quad \beta_0^* = \beta_0+N, \quad \nu_0^* = \nu_0+N,$$

$$\left[W_0^*\right]^{-1} = W_0^{-1} + N\bar{S} + \frac{\beta_0 N}{\beta_0+N}(\mu_0-\bar{U})(\mu_0-\bar{U})^T$$

$$\bar{U} = \frac{1}{N}\sum_{i=1}^{N}U_i \quad \bar{S} = \frac{1}{N}\sum_{i=1}^{N}U_iU_i^T.$$

The conditional distributions over the movie feature vectors and the movie hyperparameters have exactly the same form. The Gibbs sampling algorithm then takes the following form:

Gibbs sampling for Bayesian PMF

1. Initialize model parameters $\{U^1, V^1\}$
2. For t=1,...,T
   - Sample the hyperparameters (Eq. 14):
     $$\Theta_U^t \sim p(\Theta_U|U^t,\Theta_0)$$
     $$\Theta_V^t \sim p(\Theta_V|V^t,\Theta_0)$$
   - For each $i = 1,...,N$ sample user features in parallel (Eq. 11):
     $$U_i^{t+1} \sim p(U_i|R,V^t,\Theta_U^t)$$
   - For each $i = 1,...,M$ sample movie features in parallel:
     $$V_i^{t+1} \sim p(V_i|R,U^{t+1},\Theta_V^t)$$

4. Experimental Results

4.1. Description of the dataset

The data, collected by Netflix, represent the distribution of all ratings Netflix obtained between October, 1998 and December, 2005. The training data set consists of 100,480,507 ratings from 480,189 randomly-chosen, anonymous users on 17,770 movie titles. As part of the training data, Netflix also provides validation data, containing 1,408,395 ratings. In addition, Netflix also provides a test set containing 2,817,131 user/movie pairs with the ratings withheld. The pairs were selected from the most recent ratings from a subset of the users in the training data set. Performance is assessed by submitting predicted ratings to Netflix which then posts the root mean squared error (RMSE) on an unknown half of the test set. As a baseline, Netflix provided the test score of its own system trained on the same data, which is 0.9514.

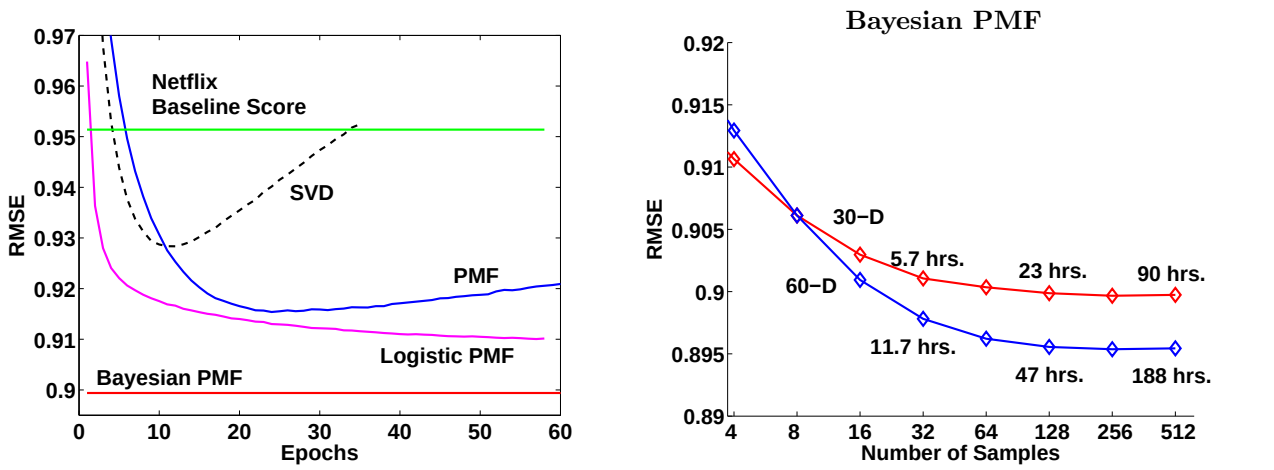


Figure 2. Left panel: Performance of SVD, PMF, logistic PMF, and Bayesian PMF using 30D feature vectors, on the Netflix validation data. The y-axis displays RMSE (root mean squared error), and the x-axis shows the number of epochs, or passes, through the entire training set. Right panel: RMSE for the Bayesian PMF models on the validation set as a function of the number of samples generated. The two curves are for the models with 30D and 60D feature vectors.

4.2. Training PMF models

For comparison, we have trained a variety of linear PMF models using MAP, choosing their regularization parameters using the validation set. In addition to linear PMF models, we also trained logistic PMF models, in which we pass the dot product between user- and movie-specific feature vectors through the logistic function $\sigma(x) = 1/(1 + \exp(-x))$ to bound the range of predictions:

$$p(R|U, V, \alpha) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|\sigma(U_i^T V_j), \alpha^{-1}) \right]^{I_{ij}}. \quad (15)$$

The ratings $1, ..., 5$ are mapped to the interval $[0, 1]$ using the function $t(x) = (x - 1)/4$, so that the range of valid rating values matches the range of predictions our model can make. Logistic PMF models can sometimes provide slightly better results than their linear counterparts.

To speed up training, instead of performing full batch learning, we subdivided the Netflix data into mini-batches of size 100,000 (user/movie/rating triples) and updated the feature vectors after each mini-batch. We used a learning rate of 0.005 and a momentum of 0.9 for training the linear as well as logistic PMF models.

4.3. Training Bayesian PMF models

We initialized the Gibbs sampler by setting the model parameters U and V to their MAP estimates obtained by training a linear PMF model. We also set $\mu_0 = 0$, $\nu_0 = D$, and W_0 to the identity matrix, for both user and movie hyperpriors. The observation noise precision α was fixed at 2. The predictive distribution was computed using Eq. 10 by running the Gibbs sampler with samples $\{U_i^{(k)}, V_j^{(k)}\}$ collected after each full Gibbs step.

4.4. Results

In our first experiment, we compared a Bayesian PMF model to an SVD model, a linear PMF model, and a logistic PMF model, all using 30D feature vectors. The SVD model was trained to minimize the sum-squared distance to the observed entries of the target matrix, with no regularization applied to the feature vectors. Note that this model can be seen as a PMF model trained using maximum likelihood (ML). For the PMF models, the regularization parameters λ_U and λ_V were set to 0.002. Predictive performance of these models on the validation set is shown in Fig. 2 (left panel). The mean of the predictive distribution of the Bayesian PMF model achieves an RMSE of 0.8994, compared to an RMSE of 0.9174 of a moderately regularized linear PMF model, an improvement of over 1.7%.

The logistic PMF model does slightly outperform its linear counterpart, achieving an RMSE of 0.9097. However, its performance is still considerably worse than that of the Bayesian PMF model. A simple SVD achieves an RMSE of about 0.9280 and after about 10 epochs begins to overfit heavily. This experiment clearly demonstrates that SVD and MAP-trained PMF models can overfit and that the predictive accuracy can be improved by integrating out model parameters and hyperparameters.

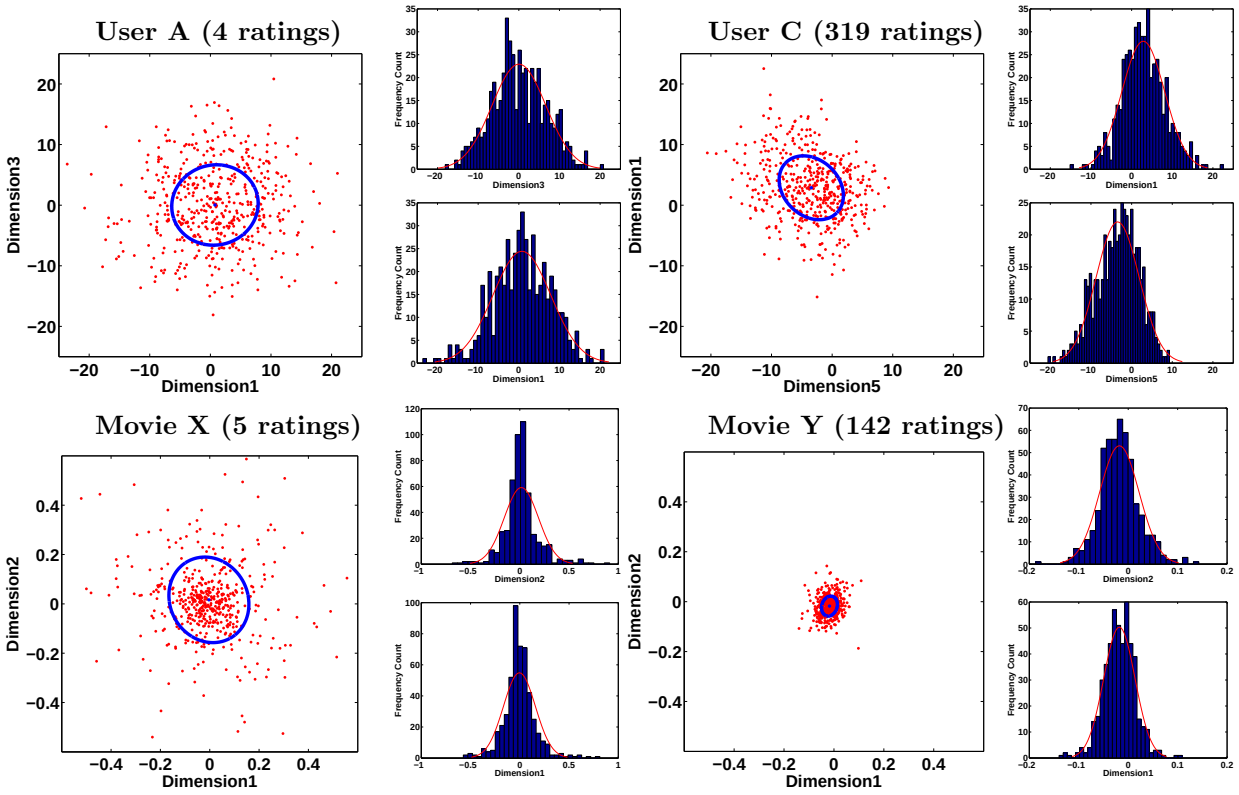


Figure 3. Samples from the posterior over the user and movie feature vectors generated by each step of the Gibbs sampler. The two dimensions with the highest variance are shown for two users and two movies. The first 800 samples were discarded as "burn-in".

D	Valid. RMSE		%	Test RMSE		%
	PMF	BPMF	Inc.	PMF	BPMF	Inc.
30	0.9154	0.8994	1.74	0.9188	0.9029	1.73
40	0.9135	0.8968	1.83	0.9170	0.9002	1.83
60	0.9150	0.8954	2.14	0.9185	0.8989	2.13
150	0.9178	0.8931	2.69	0.9211	0.8965	2.67
300	0.9231	0.8920	3.37	0.9265	0.8954	3.36

Table 1. Performance of Bayesian PMF (BPMF) and linear PMF on Netflix validation and test sets.

We than trained larger PMF models with $D = 40$ and $D = 60$. Capacity control for such models becomes a rather challenging task. For example, a PMF model with $D = 60$ has approximately 30 million parameters. Searching for appropriate values of the regularization coefficients becomes a very computationally expensive task. Table 1 further shows that for the 60-dimensional feature vectors, Bayesian PMF outperforms its MAP counterpart by over 2%. We should also point out that even the simplest possible Bayesian extension of the PMF model, where Gamma priors are placed over the precision hyperparameters α_U and α_V (see Fig. 1, left panel), significantly outperforms the MAP-trained PMF models, even though it does not perform as well as the Bayesian PMF models.

It is interesting to observe that as the feature dimensionality grows, the performance accuracy for the MAP-trained PMF models does not improve, and controlling overfitting becomes a critical issue. The predictive accuracy of the Bayesian PMF models, however, steadily improves as the model complexity grows. Inspired by this result, we experimented with Bayesian PMF models with $D = 150$ and $D = 300$ feature vectors. Note that these models have about 75 and 150 million parameters, and running the Gibbs sampler becomes computationally much more expensive. Nonetheless, the validation set RMSEs for the two models were 0.8931 and 0.8920. Table 1 shows that these models not only significantly outperform their MAP counterparts but also outperform Bayesian PMF models that have fewer parameters. These results clearly show that the Bayesian approach does not require limiting the complexity of the model based on the number of the training samples. In practice, however, we will be limited by the available computer resources.

For completeness, we also report the performance results on the Netflix *test set*. These numbers were ob-

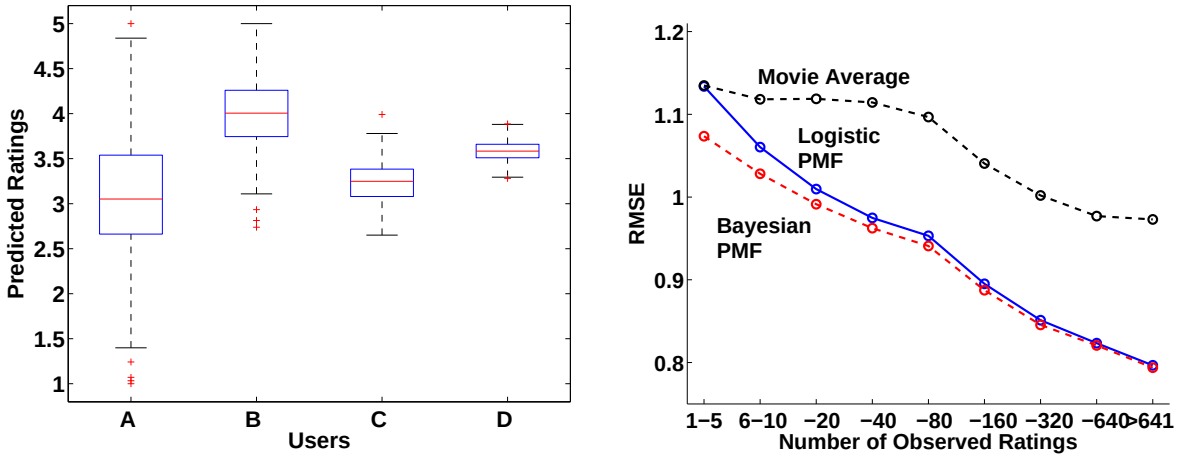


Figure 4. Left panel: Box plot of predictions, obtained after each full Gibbs step, for 4 users on a randomly chosen test movies. Users A,B,C, and D have 4, 23, 319 and 660 ratings respectively. Right panel: Performance of Bayesian PMF, logistic PMF, and the movie average algorithm that always predicts the average rating of each movie. The users were grouped by the number of observed ratings in the training data. The linear PMF model performed slightly worse than the logistic PMF model.

tained by submitting the predicted ratings to Netflix who then provided us with the test score on an unknown half of the test set. The test scores are slightly worse than the validation scores, but the relative behavior across all models remains the same.

To diagnose the convergence of the Gibbs sampler, we monitored the behaviour of the Frobenius norms of the model parameters and hyperparameters: U, V, Λ, and μ. Typically, after a few hundred samples these quantities stabilize. Fig. 2 (right panel) shows the RMSE error on the Netflix validation set for Bayesian PMF models as the number of samples increases. After obtaining a few hundred samples[1] the predictive accuracy does not significantly improve. Note that the initial predictive accuracy is already high because the Markov chain is initialized using the MAP values of the model parameters.

For the Bayesian PMF model with $D = 30$ we also collected samples over the user and movie feature vectors generated by each full step of the Gibbs sampler. The first 800 samples were discarded as "burn-in". Figure 3 shows these samples for two users and two movies projected onto the two dimensions of the highest variance. Users A and C were chosen by randomly picking among rare users who have fewer than 10 ratings and more frequent users who have more than 100 ratings in the training set. Movies X and Y were chosen in the same way. Note that the empirical distribution of the samples from the posterior appear to be non-Gaussian.

Using these samples from the posterior we also looked at the uncertainty of the predictions of four users on randomly chosen test movies. Figure 4 (left panel) shows results for users A,B,C, and D who have 4, 23, 319 and 660 ratings respectively. Note that there is much more uncertainty about the prediction of user A than about the prediction of user D, whose feature vector is well-determined. Figure 4 (right panel) shows that the Bayesian PMF model considerably outperforms the logistic PMF model on users with few ratings. As the number of ratings increases, both the logistic PMF and the Bayesian PMF exhibit similar performance.

The advantage of Bayesian PMF models is that by averaging over all settings of parameters that are compatible with the data as well as the prior they deal with uncertainty more effectively than the non-Bayesian PMF models, which commit to a single most probable setting.

Since the main concern when applying Bayesian methods to large datasets is their running time, we provide the times for our simple Matlab Bayesian PMF implementation. One full Gibbs step on a single core of a recent Pentium Xeon 3.00GHz machine for models with $D = 10, 30, 60, 300$ takes 6.6, 12.9 , 31.6, and 220 minutes respectively. Note that the most expensive aspect of training Bayesian PMF models is the inversion of a $D \times D$ matrix per feature vector[2] (see Eq. 13),

[1] We store the model parameters after each full Gibbs step as a sample. The fact that these samples are not independent does not matter for making predictions.

[2] In our implementation, we solve a system of D equa-

which is an $O(D^3)$ operation.

5. Conclusions

We have presented a fully Bayesian treatment of Probabilistic Matrix Factorization by placing hyperpriors over the hyperparameters and using MCMC methods to perform approximate inference. We have also demonstrated that Bayesian PMF models can be successfully applied to a large dataset containing over 100 million movie ratings, and achieve significantly higher predictive accuracy compared to the MAP-trained PMF models with carefully tuned regularization parameters. An additional advantage of using a Bayesian model is that it provides a predictive distribution instead of just a single number, allowing the confidence in the prediction to be quantified and taken into account when making recommendations using the model.

Using MCMC instead of variational methods for approximate inference in Bayesian matrix factorization models leads to much larger improvements over the MAP trained models, which suggests that the assumptions made by the variational methods about the structure of the posterior are not entirely reasonable. This conclusion is confirmed by inspecting the empirical distribution of the samples from the posterior, which appears to be significantly non-Gaussian.

A major problem of MCMC methods is that it is hard to determine when the Markov chain has converged to the desired distribution. In practice, we have to rely on rules of thumb to diagnose convergence, which means that there is a risk of using samples from a distribution that differs from the true posterior distribution, potentially leading to suboptimal predictions. Our results show that this problem is not a sufficient reason to reject MCMC methods.

For our models, the number of samples from the posterior that can be generated within a reasonable amount of time will typically be constrained by the available computer resources. However, as mentioned above, sampling the feature vectors for multiple users or movies in parallel provides an easy way to greatly speed up the process of generating samples using multiple cores.

Acknowledgments

We thank Geoffrey Hinton for many helpful discussions. This research was supported by NSERC.

tions instead of inverting a matrix. The computational cost of this operation is still $O(D^3)$.

References

Hinton, G. E., & van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. *COLT* (pp. 5–13).

Hofmann, T. (1999). Probabilistic latent semantic analysis. *Proceedings of the 15th Conference on Uncertainty in AI* (pp. 289–296). San Fransisco, California: Morgan Kaufmann.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, *37*, 183.

Lim, Y. J., & Teh, Y. W. (2007). Variational Bayesian approach to movie rating prediction. *Proceedings of KDD Cup and Workshop*.

Marlin, B. (2004). Modeling user rating profiles for collaborative filtering. In S. Thrun, L. Saul and B. Schölkopf (Eds.), *Advances in neural information processing systems 16*. Cambridge, MA: MIT Press.

Marlin, B., & Zemel, R. S. (2004). The multiple multiplicative factor model for collaborative filtering. *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada*. ACM.

Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods* (Technical Report CRG-TR-93-1). Department of Computer Science, University of Toronto.

Nowlan, S. J., & Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, *4*, 473–493.

Raiko, T., Ilin, A., & Karhunen, J. (2007). Principal component analysis for large scale problems with lots of missing values. *ECML* (pp. 691–698).

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany* (pp. 713–719). ACM.

Salakhutdinov, R., & Mnih, A. (2008). Probabilistic matrix factorization. *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press.

Srebro, N., & Jaakkola, T. (2003). Weighted low-rank approximations. *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), Washington, DC, USA* (pp. 720–727). AAAI Press.

Accurate Max-Margin Training for Structured Output Spaces

Sunita Sarawagi SUNITA@IITB.AC.IN
Rahul Gupta GRAHUL@CSE.IITB.AC.IN
IIT Bombay, India

Abstract

Tsochantaridis et al. (2005) proposed two formulations for maximum margin training of structured spaces: margin scaling and slack scaling. While margin scaling has been extensively used since it requires the same kind of MAP inference as normal structured prediction, slack scaling is believed to be more accurate and better-behaved. We present an efficient variational approximation to the slack scaling method that solves its inference bottleneck while retaining its accuracy advantage over margin scaling.

We further argue that existing scaling approaches do not separate the true labeling comprehensively while generating violating constraints. We propose a new max-margin trainer PosLearn that generates violators to ensure separation at each position of a decomposable loss function. Empirical results on real datasets illustrate that PosLearn can reduce test error by up to 25% over margin scaling and 10% over slack scaling. Further, PosLearn violators can be generated more efficiently than slack violators; for many structured tasks the time required is just twice that of MAP inference.

1. Introduction

The max-margin framework for training structured prediction models generalizes the benefits of support vector machines (SVMs) to predicting complex objects. A popular member of this framework is the margin scaling method (Tsochantaridis et al., 2005; Taskar, 2004; LeCun et al., 2006; Crammer & Singer, 2003) that tries to ensure that the score of the correct prediction is separated from the score of an incorrect prediction by a margin equal to the error of the prediction. This method has been used extensively in many applications, including sequence labeling (Taskar, 2004; Tsochantaridis et al., 2005), image segmentation (Taskar, 2004; Ratliff et al., 2007), grammar parsing (Taskar et al., 2004), dependency parsing (McDonald et al., 2005b), bipartite matching (Taskar, 2004) and text segmentation (McDonald et al., 2005a). A reason for its wide-spread use is that it can exploit the decomposability of the error function to find the most violating constraint using the maximum a-posteriori (MAP) inference algorithm used for prediction.

An alternative formulation (Tsochantaridis et al., 2005) is to ensure that all labelings are separated by a fixed margin of one but penalize violations in proportion to their errors. This method, called slack scaling, generally provides higher accuracy than margin scaling which gives too much importance to labelings with large errors even after they are well-separated, sometimes at the expense of instances that are not even separated. Another shortcoming of margin scaling is that it requires an error function that is linearly comparable with the feature values, whereas slack scaling is invariant to scaling of the error function. In spite of the advantages, slack scaling is not popular because it requires inferring the labeling which maximizes a non-decomposable metric – difference of score and error inverse.

In this paper we make two contributions in max-margin training of structured models.

First, we address the computational challenge of infering the labelings required when training via slack scaling. We propose a variational approximation of the slack loss so that the most violating labeling is found using the same loss augmented MAP inference as in margin scaling. We demonstrate that accuracy-wise our slack approximation is much better than margin scaling and close to the more expensive slack scaling.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Second, we propose a new max-margin framework for training models with decomposable error functions that, like the slack scaling method, is scale invariant and discounts labelings well-separated from the margin. The inference step it requires is much simpler than required by slack scaling. In particular, for Markov models we show that the inference of the most violating labelings is only a factor of two more expensive than in margin scaling. The basic idea of the new learner, that we call PosLearn, is to associate a different slack variable for each error position of a decomposable error function. We show that this leads to a better characterization of the loss than both the slack and margin scaling methods that define loss in terms of a *single* most violating labeling. Empirically, PosLearn reduces error by up to 25% over margin scaling and 10% over slack scaling in various tasks.

2. Existing Methods for Max-Margin Training

We consider structured prediction problems that associate a score $\mathrm{s}(\mathbf{x}, \mathbf{y})$ for each output $\mathbf{y} \in \mathcal{Y}$ of an input $\mathbf{x}$, and predict the output $\mathbf{y}^*$ with maximum score. The scoring function $\mathrm{s}(\mathbf{x}, \mathbf{y})$ is a dot product of a feature vector $\mathbf{f}(\mathbf{x}, \mathbf{y})$ defined jointly over the input $\mathbf{x}$ and output $\mathbf{y}$, and the corresponding parameter vector $\mathbf{w}$. The space of possible outputs $\mathcal{Y}$ can be exponentially large. Thus, efficient solutions for $\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \mathbf{f}(\mathbf{x}, \mathbf{y})$ crucially depend on the decomposability of the feature vector $\mathbf{f}$ over components of $\mathbf{y}$. During training the goal is to find a $\mathbf{w}$ using a set of labeled input-output pairs $(\mathbf{x}_i, \mathbf{y}_i) : i = 1 \ldots N$ so as to minimize prediction error. The error of predicting $\mathbf{y}$ for an instance $\mathbf{x}_i$ whose correct label is $\mathbf{y}_i$ is user-provided. We denote it by $L_i(\mathbf{y})$. In max-margin methods, the training goal is translated to finding a $\mathbf{w}$ that minimizes the sum of the loss on the labeled data while imposing a regularization penalty for overfitting. The loss is a computationally convenient combination of the user-provided error function and feature-derived scores so as to both minimize training error and maximize the margin between correct and incorrect outputs. There are two popular loss functions for structured learning tasks: margin scaler and slack scaler. We review them briefly.

2.1. Margin Scaling

In margin scaling, the goal is to find $\mathbf{w}$ such that the difference in score $\mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y}) = \mathbf{w}^T \mathbf{f}(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^T \mathbf{f}(\mathbf{x}_i, \mathbf{y})$ of the correct output $\mathbf{y}_i$ from an incorrect labeling $\mathbf{y}$ is at least $L_i(\mathbf{y})$. This is formulated as:

$$\begin{aligned} &\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} \xi_i \\ &\text{s.t. } \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y}) \geq L_i(\mathbf{y}) - \xi_i \quad \forall \mathbf{y} \neq \mathbf{y}_i, i : 1 \ldots N \\ &\xi_i \geq 0 \quad i : 1 \ldots N \end{aligned}$$

Two category of methods have been proposed to optimize the above QP. The first category is based on the cutting plane algorithm to avoid generating the exponentially many constraints. This involves incrementally finding the output $\mathbf{y}^M = \mathrm{argmax}_{\mathbf{y}}(\mathbf{w}^T \mathbf{f}(\mathbf{x}_i, \mathbf{y}) + L_i(\mathbf{y}))$ which most violates the constraint. $\mathbf{y}^M$ can be found using the same inference algorithm as MAP $\mathbf{y}^* = \mathrm{argmax}_{\mathbf{y}} \mathbf{w}^T \mathbf{f}(\mathbf{x}_i, \mathbf{y})$ when $L_i(\mathbf{y})$ decomposes over variable subsets no larger than the subsets in the decomposition of $\mathbf{f}(\mathbf{x}_i, \mathbf{y})$. This category includes exact gradient ascent methods (Tsochantaridis et al., 2005), stochastic gradient methods (Bordes et al., 2007) and online sub-gradient methods (Ratliff et al., 2007). The online structured learning methods of (Crammer & Singer, 2003) follow a perceptron based framework but their constraints are identical to the margin scaling method described here. The second category of methods (Taskar, 2004; Taskar et al., 2006) exploit the decomposability of the error function to create a combined program for the inference and parameter learning task.

2.2. Slack Scaling

Slack scaling demands a margin of one but scales the slacks of violating outputs in proportion to their errors. The corresponding optimization problem is:

$$\begin{aligned} &\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} \xi_i \\ &\text{s.t. } \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y}) \geq 1 - \frac{\xi_i}{L_i(\mathbf{y})} \quad \forall \mathbf{y} \neq \mathbf{y}_i, i : 1 \ldots N \\ &\xi_i \geq 0 \quad i : 1 \ldots N \end{aligned}$$

The optimization of the above QP via the cutting plane algorithm requires the inference of the labeling $\mathbf{y}^S = \mathrm{argmax}_{\mathbf{y}}(1 - \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y}) - \frac{\xi_i}{L_i(\mathbf{y})})$. Unlike for margin scaling, even with decomposable loss and scoring functions, it is not easy to find $\mathbf{y}^S$ efficiently. For this reason, the slack scaling approach is not popular.

However, the slack loss is in many ways better behaved than margin loss (Tsochantaridis et al., 2005). Margin scaling gives too much importance to instances which are already well-separated from the margin. This hurts because the loss ξ_i is determined by a single most violating labeling. If a labeling imposes a difficult margin

requirement because of its large error, the optimizer will appropriately increase ξ_i. After that, there is no incentive to improving separability of any other labeling of that instance. In contrast, the slack scaling loss will ignore instances that are separated by a margin of 1, and ξ_i is determined by labelings that matter because of their being close to the margin. Empirically, we found slack scaling to give better accuracy than margin scaling (Section 5). Slack scaling also makes it convenient for an end-user to define an error function and a feature vector and tune C because the error function can be arbitrarily scaled vis-a-vis the feature vector.

3. Approximate Slack Scaling

We present a variational approximation to the slack inference problem that is applicable for any structured model for which we can only solve for the MAP efficiently. The slack inference problem is to find

$$\begin{aligned}\mathbf{y}^S &= \text{argmax}_{\mathbf{y}\in\overline{\mathcal{Y}}}\left(1-\mathbf{w}^T\delta\mathbf{f}_i(\mathbf{y})-\frac{\xi_i}{L_i(\mathbf{y})}\right)\\ &= \text{argmax}_{\mathbf{y}\in\overline{\mathcal{Y}}}\left(\text{s}_i(\mathbf{y})-\frac{\xi_i}{L_i(\mathbf{y})}\right) \quad (1)\end{aligned}$$

where $\text{s}_i(\mathbf{y}) = \mathbf{w}^T\mathbf{f}(\mathbf{x}_i,\mathbf{y})$, $\overline{\mathcal{Y}} = \{\mathbf{y} : \mathbf{y}\neq\mathbf{y}_i,\ \text{s}_i(\mathbf{y}) - \frac{\xi_i}{L_i(\mathbf{y})} > \text{s}_i(\mathbf{y}_i)-1\}$ is the set of all violating labelings.

We approximate $\mathbf{y}^S$ with another labeling $\mathbf{y}^A$. Our approximation is based on the observation that $\text{s}_i(\mathbf{y}) - \frac{\xi_i}{L_i(\mathbf{y})}$ is concave in $L_i(\mathbf{y})$ and its variational approximation can be written as a linear function of $L_i(\mathbf{y})$ (Jordan et al., 1999). Here on, we drop the subscript i wherever possible.

Claim 3.1. $s(\mathbf{y})-\frac{\xi}{L(\mathbf{y})} = \min_{\lambda\geq 0} s(\mathbf{y})+\lambda L(\mathbf{y})-2\sqrt{\xi\lambda}$

Proof. Any concave function $f(z)$ can be expressed as $\min_{\lambda\geq 0}(z\lambda - f^*(\lambda))$ where $f^*(\lambda) = \min_z(z\lambda - f(z))$ is the conjugate function of $f(z)$. The result follows from the fact that the conjugate function of $\frac{-\xi}{z}$ is $2\sqrt{\xi\lambda}$. □

Let $F'(\mathbf{y};\lambda) \triangleq \text{s}(\mathbf{y}) + \lambda L(\mathbf{y}) - 2\sqrt{\xi\lambda}$ and $F(\lambda) \triangleq \max_{\mathbf{y}\neq\mathbf{y}_i} F'(\mathbf{y};\lambda)$

We now approximate the exact slack MAP objective with an upper bound as follows:

$$\begin{aligned}\max_{\mathbf{y}\in\overline{\mathcal{Y}}}\left(\text{s}(\mathbf{y})-\frac{\xi}{L(\mathbf{y})}\right) &= \max_{\mathbf{y}\in\overline{\mathcal{Y}}}\min_{\lambda\geq 0} F'(\mathbf{y};\lambda) \quad (2)\\ &\leq \min_{\lambda\geq 0}\max_{\mathbf{y}\in\overline{\mathcal{Y}}} F'(\mathbf{y};\lambda) \quad (3)\\ &\leq \min_{\lambda\geq 0} F(\lambda) \quad (4)\end{aligned}$$

For a fixed λ, we can compute $F(\lambda)$ using the loss augmented MAP algorithm employed in margin scaling to first find $\mathbf{y}_\lambda = \text{argmax}_{\mathbf{y}\neq\mathbf{y}_i} s(\mathbf{y}) + \lambda L(\mathbf{y})$ and then setting $F(\lambda) = F'(\mathbf{y}_\lambda;\lambda)$. The constraint $\mathbf{y}\neq\mathbf{y}_i$ can be met by asking the loss augmented MAP algorithm to return top two MAPs. The algorithmic extension to return top two MAPs is straight forward in many structured tasks.

We search for the λ for which the upper bound $F(\lambda)$ is minimized by exploiting the fact that $F(\lambda)$ is convex in λ.

Claim 3.2. *$F(\lambda)$ is convex in λ.*

Proof. It can be seen that $F'(\mathbf{y};\lambda)$ is convex in λ. Since $F(\lambda)$ is a max of finitely many convex functions, and max is also convex, $F(\lambda)$ is convex. □

We can compute $\min_{\lambda\geq 0} F(\lambda)$ using efficient line search algorithms such as Golden Search. During the search phase, for each λ that we encounter, we evaluate $F(\lambda)$ and thus get one labeling. Of all these labelings, we return the one with the highest $\text{s}(\mathbf{y}) - \frac{\xi}{L(\mathbf{y})}$ as $\mathbf{y}^A$. We show in the next section the range $[\lambda_l, \lambda_u]$ within which it is sufficient to perform the line search.

3.1. Upper and Lower Bounds for λ

Since $\lambda \geq 0$, we can use $\lambda_l = 0$ as the lower limit. However, with $\lambda = 0$, $F'(\mathbf{y};\lambda)$ is not able to distinguish between high and loss labelings with same scores commonly seen in early training iterations. It can be shown that with $\lambda_l = \frac{\epsilon}{L_{max}}$ where L_{max} is the maximum possible loss, $F(\lambda)$ for any $\lambda < \lambda_l$ will not return any violating labeling with slack score more than ϵ of the score of a labeling returned with $\lambda \geq \lambda_l$. By setting ϵ to the tolerance of the cutting-plane algorithm, we get a provably correct lower bound.

For the upper bound, it is sufficient to pick a λ_u such that for any $\lambda \geq \lambda_u$, either $F(\lambda)$ gets the same violator as $F(\lambda_u)$ or a non-violator that we are not interested in. It is sufficient to pick a λ_u such that $\text{argmax}_{\mathbf{y}\in\overline{\mathcal{Y}}} F'(\mathbf{y};\lambda_u)$ ($= \mathbf{y}'$ say) has the maximum loss among all violators in $\overline{\mathcal{Y}}$. Hence we need:

$$\text{s}(\mathbf{y}') + \lambda_u L(\mathbf{y}') \geq \max_{\mathbf{y}\in\overline{\mathcal{Y}},\ L(\mathbf{y})<L(\mathbf{y}')} \text{s}(\mathbf{y}) + \lambda_u L(\mathbf{y})$$

Let $\mathbf{y}_1 \triangleq \text{argmax}_{\mathbf{y}} s(\mathbf{y})$ and L_ε be the minimum difference between two distinct loss values (e.g. $L_\varepsilon = 1$ for Hamming loss). Then the right side can be atmost $\text{s}(\mathbf{y}_1) + \lambda_u(L(\mathbf{y}') - L_\varepsilon)$. So we require $\lambda_u \geq \frac{\text{s}(\mathbf{y}_1)-\text{s}(\mathbf{y}')}{L_\varepsilon}$.

Now, $\mathbf{y}' \in \overline{\mathcal{Y}} \Rightarrow \text{s}(\mathbf{y}') \geq \text{s}(\mathbf{y}_i) - 1 + \frac{\xi}{L(\mathbf{y}')} \geq$

$\mathrm{s}(\mathbf{y}_i) - 1 + \frac{\xi}{L_{max}}$, so we can conservatively set $\lambda_u = \frac{1}{L_\varepsilon}\left(\mathrm{s}(\mathbf{y}_1) - \mathrm{s}(\mathbf{y}_i) + 1 - \frac{\xi}{L_{max}}\right)$.

3.2. Limitation of Approximate Slack

In the worst case, it is possible that the exact slack MAP $\mathbf{y}^S$ violates the inequality but $\mathbf{y}^A$ does not, as we show next.

Claim 3.3. $s(\mathbf{y}^A) - \frac{\xi}{L(\mathbf{y}^A)} < s(\mathbf{y}_i) - 1 + \epsilon \not\Rightarrow s(\mathbf{y}^S) - \frac{\xi}{L(\mathbf{y}^S)} < s(\mathbf{y}_i) - 1 + \epsilon$

Proof. We prove the claim with a counter example. Let $\mathbf{y}_j, j = 1, 2, 3$ be three labelings with scores $s_j = -\frac{1}{2}, -\frac{13}{18}, -\frac{5}{6}$ and losses $L_j = 1, 2, 3$. Note that $s_1 > s_2 > s_3$ and $L_1 < L_2 < L_3$. Let the score of the true labeling be $s = 0$, the slack be $\xi = \frac{19}{36}$, and let $\epsilon \approx 0$. By computing $\mathrm{sgn}(s_j - \frac{\xi}{L_j} - s + 1 - \epsilon)$, we can see that labelings $\mathbf{y}_1$ and $\mathbf{y}_3$ are not violators but $\mathbf{y}_2$ is. In order to return $\mathbf{y}_2$ as the worst violator, there must exist λ such that $s_2 + \lambda L_2 \geq s_j + \lambda L_j,\ j = 1, 3$. This translates to the constraints $\lambda > \frac{2}{9}$ and $\lambda < \frac{1}{9}$, which are infeasible. □

The above counter example showed that it is impossible to approximate the slack scaled constraint by any method that depends on finding MAP with varying weights on error. This limitation though seemingly restrictive, only slightly hampers the performance in practice, as evident in our experimental results.

4. Position Learner

We next propose a new formulation for max-margin training that directly exposes the decomposability of the error function so as to require solving a considerably simpler inference problem. We show that this new formulation not only addresses the computational problem of slack scaling inference, but also provides a more accurate characterization of the loss of scoring functions.

The basic premise of the new learner, which we call PosLearn, is that when error is additive over a set of positions, the loss should also additively reflect margin violations at each possible error position. This is in contrast to both the margin and slack scaling where loss is in terms of a single most violating labeling.

Let $L_i(\mathbf{y}) = \sum_{c \in C} L_{i,c}(\mathbf{y}_c)$ denote a decomposition of the error function. Our goal during training is to ensure that at each possible error position c, the correct labeling has a margin over all labelings where c is incorrectly labeled. If not, we add a hinge loss on the difference in score between the correct labeling $\mathbf{y}_i$ and the best labeling $\mathrm{argmax}_{\mathbf{y}:\mathbf{y}_c \neq \mathbf{y}_{i,c}} \mathbf{w}^T \mathbf{f}(\mathbf{x}_i, \mathbf{y})$ incorrect at c. This yields the following constrained optimization

$$\begin{aligned} \min_{\mathbf{w},\xi} \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{N} \sum_{c} \xi_{i,c} \\ \text{s.t} \quad & \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y}) \geq 1 - \frac{\xi_{i,c}}{L_{i,c}(\mathbf{y}_c)} \quad \forall \mathbf{y} : \mathbf{y}_c \neq \mathbf{y}_{i,c} \\ & \xi_{i,c} \geq 0 \quad i : 1 \ldots N, \forall c \end{aligned}$$

In the above program, the number of slack variables is equal to the total number of error positions over all instances. Otherwise, the form of the QP is the same as in Section 2.2 and therefore can be solved via similar cutting plane algorithms. For a given position c of an instance i, the most violating constraint is the labeling

$$\mathbf{y}^{P:c} \triangleq \mathrm{argmax}_{\mathbf{y}:\mathbf{y}_c \neq \mathbf{y}_{i,c}} \left(\mathrm{s}_i(\mathbf{y}) - \frac{\xi_{i,c}}{L_{i,c}(\mathbf{y}_c)}\right) \qquad (5)$$

This inference problem can be solved efficiently by any structured learning task in which MAP can be found efficiently since

$$\max_{\mathbf{y}:\mathbf{y}_c \neq \mathbf{y}_{i,c}} \mathrm{s}_i(\mathbf{y}) - \frac{\xi_{i,c}}{L_{i,c}(\mathbf{y}_c)} = \max_{\mathbf{y}_c \neq \mathbf{y}_{i,c}} \left(\max_{\mathbf{y} \sim \mathbf{y}_c} \mathrm{s}_i(\mathbf{y}) - \frac{\xi_{i,c}}{L_{i,c}(\mathbf{y}_c)}\right)$$

where the outer max is over a small number of values as the size of c is typically small and the inner max is MAP inference with label of c constrained to $\mathbf{y}_c$. The MAPs $\mathbf{y}^{P:c}$ will typically be evaluated simultaneously for each c. In many structured learning tasks, all these MAPs can be found in just twice the amount of time it takes to compute a single unrestricted MAP, as we show in Section 4.3.1.

In addition to these computational advantages, PosLearn also provides better loss characterization than slack scaling.

4.1. Comparison with Slack Scaling

First, we claim that the PosLearn loss is an upper bound of the slack loss.

Claim 4.1. *The slack loss* $\sum_i \max_{\mathbf{y}} L_i(\mathbf{y})[1 - \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y})]_+$ *is upper bounded by the PosLearn loss* $\sum_i \sum_c \max_{\mathbf{y}:\mathbf{y}_c \neq \mathbf{y}_{i,c}} L_{i,c}(\mathbf{y}_c)[1 - \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y})]_+$

Proof. Let $\mathbf{y}^S = \mathrm{argmax}_{\mathbf{y} \neq \mathbf{y}_i} L_i(\mathbf{y})[1 - \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y})]_+$.

$$\begin{aligned} & L_i(\mathbf{y}^S)[1 - \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y}^S)]_+ \\ & \quad = \sum_c L_{i,c}(\mathbf{y}^S_c)[1 - \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y}^S)]_+ \\ & \quad \leq \sum_c \max_{\mathbf{y}:\mathbf{y}_c \neq \mathbf{y}_{i,c}} L_{i,c}(\mathbf{y}_c)[1 - \mathbf{w}^T \delta\mathbf{f}_i(\mathbf{y})]_+ \end{aligned}$$

□

Next, we show that slack scaling by defining the total loss in terms of a *single* most violating labeling, cannot discriminate amongst scoring functions as well as the PosLearn loss that involves different labelings at different error positions.

Consider one example where $\mathbf{w}$ is such that three labelings $\mathbf{y}_0 = [0\ 0\ 0\ 0], \mathbf{y}_1 = [1\ 1\ 0\ 0], \mathbf{y}_2 = [0\ 0\ 1\ 0]$, all have the same score of 1. Let $\mathbf{y}_0$ be the correct labeling, then $L(\mathbf{y}_1) = 2, L(\mathbf{y}_2) = 1$, assuming Hamming error. Let the score of all remaining labelings be 0. The total slack loss in this case is 2 whereas the PosLearn loss is 3. Now consider the case where $\mathbf{y}_2$ has score 0. The slack loss remains unchanged whereas PosLearn loss reduces to 2.

An important consequence of the reduced error coverage is that, when the cutting plane algorithm terminates in slack scaling, PosLearn could continue to find violating constraints. The reverse is not true.

4.2. Comparison with M^3N Training

The PosLearn program appears similar to the M^3N program of (Taskar, 2004) because both decompose the slack variable over multiple positions. However, the similarity is only superficial. The training objective of M^3N is Margin scaling and the position specific slack variables are for integrating training with inference for loss augmented MAP. In PosLearn the position specific slacks lead to a very different training objective.

4.3. Common Decomposable Error Functions

We show examples of decomposable error functions in several structured learning tasks and show how to efficiently find the most violating constraints over all error positions simultaneously.

4.3.1. Markov Models

Many structured prediction tasks can be modeled as Markov models. Popular examples are sequence labeling for information extraction (Lafferty et al., 2001), and grid models for image segmentation (Taskar, 2004; Boykov et al., 2001). A natural error function here is Hamming loss that decomposes over the nodes of the Markov network. Typical MAP inference algorithms based on belief propagation also give max-marginals at each node. The max-marginals gives us at each (node c, label y) pair, the best labeling $\mathbf{y}^{c:y}$ with node c labeled y. We can now find the most violating labeling at each position c via

$$\max_{y \neq \mathbf{y}_{i,c}} (1 - \mathbf{w}^T \delta \mathbf{f}_i(\mathbf{y}^{c:y})) L_{i,c}(y)$$

where $L_{i,c}(y) = 1$ when $y \neq \mathbf{y}_{i,c}$ for Hamming loss. In general $L_{i,c}(y)$ can be any arbitrary real-value, for example a mis-classification matrix $M(y', y)$ could give the cost of misclassifying a y' node as y.

4.3.2. Segmentation

The output space $\mathcal{Y}$ consists of all possible labeled segmentations of an input sequence $\mathbf{x}$. A segmentation $\mathbf{y}$ consists of a sequence of segments $s_1 \ldots s_p$ where each $s_j = (t_j, u_j, y_j)$ with t_j = segment start position, u_j = segment end position, and y_j= segment label. Segmentation models have been proposed as alternative models for information extraction that allows for more effective use of entity-level features (McDonald et al., 2005a; Sarawagi & Cohen, 2004).

The feature vector decomposes over segments and is a function of the segment and the label of the previous segment. Thus $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{p} \mathbf{f}(\mathbf{x}, s_j, y_{j-1})$. The error function also decomposes over segments as $L_i(\mathbf{y}) = \sum_{s \in \mathbf{y}} L_i(s)$ where for a segment $s = (t, u, y)$, $L_i((t, u, y))$ is defined as

$$L_i((t,u,y)) = \begin{cases} p_y + \sum_{(l',u',y') \in \mathbf{y}_i, \ t \leq u' \leq u} r_{y'} & (t,u) \notin \mathbf{y}_i \\ M(y', y) & (t,u,y') \in \mathbf{y}_i \end{cases}$$

where p_y is the precision penalty of labeling a segment as y and $r_{y'}$ is the recall penalty of missing a true segment of label y' and $M(y', y)$ is the misclassification cost matrix applicable when the same span appears in both segmentations.

The number of slack variables is the number of possible segment spans (t, u), which is $O(nm)$ for a sequence of length n and maximum segment size m.

The MAP segmentation can be found using an extension of the Viterbi algorithm (Sarawagi & Cohen, 2004). Viterbi also gives the highest scoring segmentation of the sequence from 1 to i with the last segment ending at i with label y for all possible i and y. Call this $\gamma(i, y)$. Similarly, we can use a backward Viterbi pass to get $\beta(i, y)$ the highest scoring segmentation from $i + 1$ to n with label y on the segment ending at i. These can be combined to find the most violating constraint for a slack variable corresponding to segment (t, u) as:

$$\max_{y', y:(t,u,y) \notin \mathbf{y}_i} L_i((t,u,y))[1 - \mathrm{s}(\mathbf{y}_i) + \gamma(t-1, y') + \mathbf{w}^T \mathbf{f}(\mathbf{x}, (t,u,y), y') + \beta(u, y)]_+$$

4.3.3. Unlabeled Dependency Parsing

In unlabeled dependency parsing, the goal is to assign each token to its 'head' token (or to a dummy token),

such that the head links form a directed spanning tree. The feature vector for a tree $\mathbf{y}$ over a sentence $\mathbf{x}$ is decomposable over the edges (McDonald et al., 2005b): $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_t \mathbf{f}(y_t, \mathbf{x}, t)$ where t is a token and y_t is its head. A natural error function for a dependency parse tree is then the number of words that are assigned an incorrect head word. In this case, the error and features decompose in exactly the same way, over individual words. The only coupling amongst the predictions of different words is that they need to form a tree.

We use the combinatorial non-projective parsing algorithm of (McDonald et al., 2005b), which cannot be easily extended to simultaneously return MAP for each position. For PosLearn we return the worst violator for each position by first finding the unrestricted MAP $\mathbf{y}^*$. Then, for each position where $\mathbf{y}^*$ is correct, we re-invoke MAP with the correct assignment disabled. In the worse case, this will lead to n MAP invocations.

5. Experiments

We present experimental results on three tasks — sequence labeling, text segmentation and dependency parsing, performed on the following datasets and settings:

CoNLL'03: We use the English benchmark from the CoNLL'03 shared task on named entity recognition. The corpus consists of train, development and test sets of $\approx$ 14000, 3200 and 3400 sentences respectively. We used exactly the same features as in the trained model from Stanford's Named Entity Recognizer [1].

Cora: This is a database of $\approx$ 500 citations (McCallum et al., 2000), containing entities such as Author, Journal, Title, Year and Volume. We used standard extraction features defined over the neighborhoods of each token and the label of the previous token (Peng & McCallum, 2004). For the segmentation task on this dataset, we also used the segment length feature.

Address: This is a collection of $\approx$ 400 non-US postal addresses. Unlike US addresses, these addresses are highly irregular and relatively difficult to segment. The features for sequence labeling and segmentation tasks are as defined in (Sarawagi & Cohen, 2004).

CoNLL-X: We use the freely available treebanks for Swedish, Dutch and Danish from the CoNLL X Shared Task for unlabeled dependency parsing. The training sets contain $\approx$ 11000, 13350, and 5200 sentences respectively. We use the first-order features, the online MIRA trainer (Crammer & Singer, 2003), and the non-projective parsing algorithm provided in the MSTParser package[2].

[1] http://nlp.stanford.edu/software/stanford-ner-2008-05-07.tar.gz

Table 1. Token mis-classifications (in %) of all approaches on all tasks. For sequence labeling and segmentation we also report span F1 (after '/').

	Margin	Slack	Approx	PosLearn
		Sequence Labeling		
Cora	12.3/74.9	10.0/82.9	9.9/83.0	**9.5/83.4**
Address	17.1/71.0	15.7/76.7	15.1/78.1	**14.2/78.4**
CoNLL	2.89/84.7	2.95/84.7	2.96/84.6	**2.82/85.1**
		Segmentation		
Cora	17.7/81.8	17.4/81.9	17.3/81.9	**16.2/83.1**
Address	15.4/77.6	15.4/77.5	15.4/77.6	**13.8/79.0**
		Dependency Parsing		
Danish	**12.4**	-	-	12.5
Dutch	**16.3**	-	-	16.9
Swedish	12.9	-	-	**12.8**

5.1. Results

Table 1 shows test errors (as defined in Section 4.3) and Span F1 (where ever applicable) of all four training approaches on all the tasks. For Cora and Address results are averaged over ten splits of 25% train — 75% test, the rest are with the standard training and test files as available in the benchmark. For sequence and segmentation tasks, we are able to solve the Slack inference problem exactly using a quadratic algorithm that finds the MAP for each possible error value. For dependency parsing, it was not easy to find MAP with a pre-specified error. Hence, numbers for Slack scaling methods are missing for this task.

Sequence labeling

We note that the errors go down in the order Margin > Slack > ApproxSlack > PosLearn, and PosLearn achieves $\approx$ 20% error reduction over Margin and 5-10% over Slack. The difference between PosLearn and Margin is statistically significant (p-value from paired t-test is < 0.001), while that between ApproxSlack and Slack is not. This confirms that the approximations done in ApproxSlack are empirically good.

We also reports the entity span F1 values in Table 1 (numbers after the "/"). PosLearn provides signifi-

[2] http://sourceforge.net/mstparser

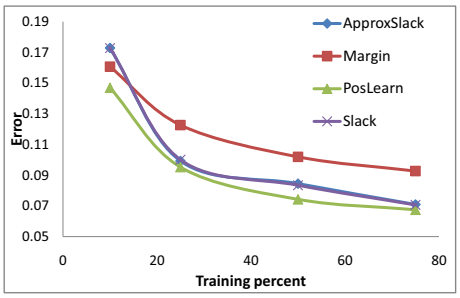


Figure 1. Sequence labeling error (in %) of all approaches on Cora as training size is increased.

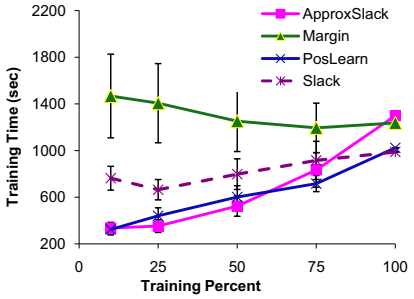


Figure 2. Comparison of training times on Cora over various training percentages. The error bars denote one standard deviation over ten random splits.

cant improvements over Margin for Cora and Address, going from 75 to 83 and 71 to 78 respectively. This shows that optimizing for the error directly translates to significantly better span F1 scores. For CoNLL'03 the gains are modest both for error and Span F1 for reasons we will highlight in Section 5.2. Figure 1 investigates the effect of increasing training size on the sequence labeling errors of all the approaches on Cora. PosLearn remains the best approach for all training sizes, with a 25% error reduction over Margin even for 75% training data. ApproxSlack and Slack are almost identical for all training sizes.

Figure 2 compares the training time of the four approaches on Cora over various training sizes. PosLearn and ApproxSlack turn out to be the cheapest of all the approaches. Two key observations here are (a) PosLearn is up to five times faster than Margin in spite of generating many more constraints, and (b) The training time of Margin reduces with an increase in data. These can be attributed to two reasons. First, PosLearn quickly generates a lot of relevant constraints and terminates in much fewer iterations, whereas Margin spends too much time in separating high loss labelings which are already far enough. Second, when data is scarce, Margin is not able to find good support vectors early on and takes many more iterations. This provides another empirical support for the recent observations in (Bottou & Bousquet, 2008) on the inverse dependence of training time on data sizes.

Segmentation

The results for segmentation are similar to sequence labeling. Again, PosLearn provides 7-10% decrease in error over Margin and Slack. ApproxSlack again turns out to be a close approximation to Slack.

Unlabeled dependency parsing

The difference between PosLearn and Margin turns out to be very insignificant in this case. We cannot evaluate Slack as its MAP inference algorithm is not feasible in this setting. Our discussion in the next section shows that we do not expect ApproxSlack to score over Margin either.

Note our baseline numbers are competitive with the state of the art for these tasks. For Swedish and Danish, the errors for Margin scaling are significantly lower than the average errors of the CoNLL X Shared Task participants — 15.8% and 15.5% respectively . For Dutch, Margin scaling model is better than the best model in the Shared Task (error 16.4%).

5.2. Discussion

We observed that Margin scaling was significantly worse than other loss functions for tasks like sequence labeling on the Address and Cora datasets, while being the highest performing on tasks like dependency parsing. We explain the reasons behind the varying gains of Margin relative to other loss functions, in particular PosLearn, based on the decomposition of the error function compared to the feature function.

We argue that margin scaling is a bad loss function only when the model comprises of features that strongly couple larger subset of variables than the error function. Consider the case when the feature function decomposes over each position of $\mathbf{y}$, exactly as in the error function. This is true for dependency parsing, and for sequence labeling models with no edge features. In such cases, a structured formulation adds little value, and a multi-class SVM with independent constraints over the local features and loss at each position, is just as adequate. The constraints of structured margin scaling turn out to be a summation of the constraints of multi-class SVM and the two solve equivalent objectives as shown in (Joachims, 2006). Interestingly, (McDonald et al., 2005b) indeed finds that such a model (which they call the factored model) is very close to the structured model using margin scal-

ing. We verify that for sequence labeling, if we disable all edge features, then for the Address dataset, span F1 drops from 71 to 62 and for Cora from 75 to 44 with Margin scaling. This indicates the strong importance of structured features for these datasets. In contrast, for CoNLL'03 where Margin is competitive with PosLearn, removal of edge features causes only a small drop in Span F1, from 84.7 to 81. Without edge features, PosLearn shows little or negative improvement over Margin scaling for all three datasets.

This indicates that in domains where the feature function does not induce strong coupling amongst variables, there is no reward in going beyond simple margin scaling, and possibly even multiclass SVMs. In truly structured problems where features strongly couple multiple variables, margin scaling gets adversely affected by the unnecessary margin requirements of high error labelings due to shared slack variables. PosLearn ignores labelings separated from the margin, and by defining per-position slacks instead of a single shared slack, handles such structured cases better.

6. Conclusion

We presented an efficient variational approximation to the slack scaling approach, which only requires a slightly modified loss augmented MAP algorithm, instead of the inefficient slack scaling inference algorithm. We demonstrated that in practice it performs much better than margin scaling and closely approximates slack scaling.

Next, we argued that all existing approaches that define loss in terms of a single most violating labeling achieve inadequate separation from the correct labeling. We proposed a new trainer, PosLearn that involves multiple labelings in trying to ensure max-margin separation at each possible error position in the structured output. The PosLearn constraints can be generated using only the MAP algorithm, and for many structured models the time required is no more than twice the time taken to find MAP. Empirically, this leads to significant error reduction over Margin scaling on structured models that induce strong coupling amongst output variables.

A compelling future direction is theoretically analyzing the generalizability of PosLearn vis-a-vis other loss scaling methods.

References

Bordes, A., Bottou, L., Gallinari, P., & Weston, J. (2007). Solving multiclass support vector machines with larank. *ICML* (pp. 89–96).

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. *NIPS*.

Boykov, Y., Veksler, O., & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, *23*, 1222–1239.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, *3*, 951–991.

Joachims, T. (2006). Training linear SVMs in linear time. *KDD*.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. (1999). An introduction to variational methods for graphical models. In M. I. Jordan (Ed.), *Learning in graphical models*. MIT Press.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the International Conference on Machine Learning (ICML-2001)*. Williams, MA.

LeCun, Y., Chopra, S., Hadsell, R., Marc'Aurelio, R., & Huang, F. (2006). A tutorial on energy-based learning. *Predicting Structured Data*. MIT Press.

McCallum, A., Nigam, K., Reed, J., Rennie, J., & Seymore, K. (2000). Cora: Computer science research paper search engine. http://cora.whizbang.com/.

McDonald, R., Crammer, K., & Pereira, F. (2005a). Flexible text segmentation with structured multilabel classification. *HLT/EMNLP*.

McDonald, R., Crammer, K., & Pereira, F. (2005b). Online large-margin training of dependency parsers. *ACL* (pp. 91–98).

Peng, F., & McCallum, A. (2004). Accurate information extraction from research papers using conditional random fields. *HLT-NAACL* (pp. 329–336).

Ratliff, N., Bagnell, J., & Zinkevich, M. (2007). (online) subgradient methods for structured prediction. *AIStats*.

Sarawagi, S., & Cohen, W. W. (2004). Semi-markov conditional random fields for information extraction. *NIPS*.

Taskar, B. (2004). *Learning structured prediction models: A large margin approach*. Doctoral dissertation, Stanford University.

Taskar, B., Klein, D., Collins, M., Koller, D., & Manning, C. (2004). Max-margin parsing. *EMNLP*.

Taskar, B., Lacoste-Julien, S., & Jordan, M. I. (2006). Structured prediction, dual extragradient and bregman projections. *J. Mach. Learn. Res.*, *7*, 1627–1653.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, *6(Sep)*, 1453–1484.

Fast Incremental Proximity Search in Large Graphs

Purnamrita Sarkar PSARKAR@CS.CMU.EDU
Machine Learning Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213

Andrew W. Moore AWM@GOOGLE.COM
Amit Prakash AMITPRAKASH@GOOGLE.COM
Google Inc. Pittsburgh, PA 15213

Abstract

In this paper we investigate two aspects of ranking problems on large graphs. First, we augment the deterministic pruning algorithm in Sarkar and Moore (2007) with sampling techniques to compute approximately correct rankings with high probability under random walk based proximity measures *at query time*. Second, we prove some surprising locality properties of these proximity measures by examining the short term behavior of random walks. The proposed algorithm can answer queries *on the fly without caching any information about the entire graph*. We present empirical results on a 600,000 node author-word-citation graph from the Citeseer domain on a single CPU machine where the average query processing time is around 4 seconds. We present quantifiable link prediction tasks. On most of them our techniques outperform Personalized Pagerank, a well-known diffusion based proximity measure.

1. Introduction

Link prediction in social networks, personalized graph search techniques, fraud detection and collaborative filtering in recommender networks are important practical problems that greatly rely on graph theoretic measures of similarity. Given a node in a graph we would like to ask which other nodes are most similar to this node. Ideally we would like this similarity measure to capture the graph structure such as having many common neighbors or having several short paths between two nodes. This kind of structural information can be easily quantified using random walks on graphs: diffusion of information from one node to another. Most random-walk based ranking algorithms can be categorized into two broad categories.

Probability of reaching a node: This is the basis of measures like *personalized page rank*. Personalized page-rank vectors (PPV) have been used for keyword search in databases (Balmin et al., 2004) and entity-relation graphs (Chakrabarti, 2007). These approaches focus on computing approximate PPV at query time (details in section 6), and quantify the performance in terms of the deviation of the approximation from the exact. However, it is not clear if PPV itself has good predictive power.

Expected number of hops to reach a node: This is also called the hitting time (Aldous & Fill, 2001). The symmetric version of this is the *commute time* between two nodes. These metrics have been shown to be empirically effective for ranking in recommender networks (Brand, 2005) and link prediction problems (Liben-Nowell & Kleinberg, 2003). These measures usually require $O(n^3)$ computation. Recently Spielman and Srivastava (2008) have come up with a novel approximation algorithm for efficiently computing commute times by random projections. However it is only applicable to undirected graphs.

Sarkar and Moore (2007) introduced the notion of truncated commute times and demonstrated that it had good predictive power for link prediction tasks. However their algorithm (GRANCH) required storing potential nearest neighbors of all nodes in the graph in order to answer nearest neighbor queries. The key contribution in this paper are: 1) we combine sampling with deterministic pruning to design an algorithm which retrieves top k neighbors of a query in truncated commute time incrementally without caching information about all nodes in the graph. 2) We investigate locality properties of truncated hitting and commute times. 3) We show that on several link prediction tasks these measures outperform PPV in terms of predictive

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

power, while on others they do comparably. 4) Our algorithm can process queries at around 4 seconds on average on graphs of the order of 600,000 nodes on a single CPU machine.

The rest of the paper is organized as follows: in section 2 we provide relevant background. In section 3 we introduce our hybrid algorithm, and provide sample complexity results for random sampling. The locality properties of truncated hitting and commute times are investigated in section 4. We present empirical results in section 5, and conclude with related work in section 6.

2. Background

A graph $G = (V, E)$ is defined as a set of vertices V edges E. The ij^{th} entry of the adjacency matrix W denotes the weight on edge (i, j), and is zero if the edge does not exist. $P = p_{ij}, i, j \in V$ denotes the transition probability matrix of this Markov chain, so that $p_{ij} = w_{ij}/\sum_j W_{ij}$ if $(i, j) \in E$ and zero otherwise.

Hitting time h_{ij}: The hitting time from node i to node j is defined as the expected number of steps in a random walk starting from i before node j is visited for the first time. Recursively h_{ij} can be written as $h_{ij} = 1 + \sum_k p_{ik}h_{kj}$, if $i \neq j$ and zero otherwise.

Commute time c_{ij}: Commute time between a pair of nodes is defined as $c_{ij} = h_{ij} + h_{ji}$.

2.1. Truncated Hitting Time

The hitting and commute times are sensitive to long range paths (Liben-Nowell & Kleinberg, 2003) which result in non-local nature. They are also prone to be small if one of the nodes is of large degree (Brand, 2005). This renders them ineffective for personalization purposes. In order to overcome these shortcomings, Sarkar and Moore (2007) define a *T-truncated hitting time*, where only paths of length less than T are considered. We shall use h, h^T interchangeably to denote truncated hitting time. h^T_{ij} can be defined recursively as

$$h^T_{ij} = 1 + \sum_k p_{ik} h^{T-1}_{kj} \quad (1)$$

where h^T is defined to be zero if $i = j$ or if $T = 0$. The above equation expresses h^T in a one step look-ahead fashion. The expected time to reach a destination within T timesteps is equivalent to one step plus the average over the hitting times of it's neighbors to the destination in $T - 1$ hops. If there is no path of length smaller than T from i to j, this automatically sets $h^T(i, j)$ to T.

2.2. GRANCH (Sarkar & Moore, 2007)

The truncated hitting time from all nodes to a destination node can be computed in $O(ET)$ time using dynamic programming. However in order to compute the hitting time from a query node to a destination, one has to compute the hitting time of all nodes to the destination, thus computing the entire matrix which takes $O(NET)$ time (N and E are the number of nodes and edges respectively).

In order to get around the above problem the graph is decomposed into N overlapping neighborhoods for each node. Each neighborhood is computed in a way to include potential nearest neighbors and prune away the rest. The authors provide bounds on the hitting time from all nodes within the neighborhood of i to i. The hitting time from any node outside the boundary to the destination is quantified by only two numbers: a lower and an upper bound. As the neighborhoods are expanded more the bounds become tighter. This way each column of the truncated hitting time (H^T) matrix is filled up **partially**. After iterating over all nodes it is possible to look at one row and obtain ranking from the bounds on hitting time from a node.

GRANCH computes all pairs of nearest neighbors by caching information for all nodes in the graph. This does not work when the graph is changing continuously. We introduce a hybrid algorithm which essentially combines the above branch and bound trick with sampling techniques to obtain nearest neighbors of a query node in commute time with high probability.

3. Hybrid Algorithm

We present an algorithm to compute approximate nearest neighbors in commute times, without iterating over the entire graph. We combine random sampling with the branch and bound pruning scheme mentioned before, in order to obtain upper and lower bounds on *commute times* from a node. This lets us compute the k nearest neighbors from a query node *on the fly*.

For any query node we compute hitting time from it using sampling. We maintain a bounded neighborhood for the query node at a given time-step. We compute estimated bounds on the commute time from the nodes within the neighborhood to the query. Commute time of nodes outside the neighborhood to the query are characterized by a single upper and lower bound. We expand this neighborhood until this lower bound exceeds $2T'$, which guarantees that with high probability we are excluding nodes which are more than $2T'$ commute distance away. These bounds are then used for ranking the nodes inside the neighborhood.

We will first describe a simple sampling scheme to obtain ϵ-approximate truncated hitting times **from** a query node with high probability.

3.1. Sampling Scheme

We propose a sampling scheme to estimate the truncated hitting time from a given query node i in a graph. We run M independent T-length random walks from i. Lets say out of these M runs m random walks hit j for the first time at $\{t_{k_1}, ...t_{k_m}\}$ time-steps. From these we can estimate the following

1. The probability of hitting any node j *for the first time* from the given source node within T steps can be estimated by $\frac{m}{M}$.
2. The first hitting time can be estimated by

$$\hat{h}^T(i,j) = \frac{\sum_r t_{k_r}}{M} + (1 - \frac{m}{M})T$$

We provide bounds (details in Appendix) similar to Fogaras et al. (2004)

1. The number of samples M required in order to give an ϵ- correct answer with probability $1 -$.
2. The number of samples M required in order to get the top k neighbors correct.

Theorem 3.1 *For a given node i, in order to obtain $P(\exists u \in \{1,\ldots,n\}, |\hat{h}^T(i,u) - h^T(i,u)| \geq \epsilon T) \leq$, number of samples M should be at least $\frac{1}{2\epsilon^2}\log(\frac{2n}{\delta})$.*

Theorem 3.2 *Let $v_j, j = 1:k$ be the top k neighbors of i in exact T-truncated hitting time. Let $\alpha = h^T(i,v_{k+1}) - h^T(i,v_k)$. Then number of samples M should be at least $\frac{2T^2}{\alpha^2}\log(nk/\ )$ in order to have $Pr(\exists j \leq k, q > k, \hat{h}^T(i,v_j) > \hat{h}^T(i,v_q)) \leq$.*

The details are provided in the appendix. The above theorem says nothing about the order of the top k neighbors, only that if the gap between the hitting times from i to the k^{th} and $k+1^{th}$ nearest neighbors is large, then it is easy to retrieve the top k nearest neighbors. We could change the statement slightly to obtain a sample complexity bound to guarantee the exact order of the top k neighbors with high probability. The main difference will be that it will depend on $\min_{j \leq k} h^T(i,v_{j+1}) - h^T(i,v_j)$.

3.2. Lower and Upper Bounds on c_{ij}^T

Let us denote the neighborhood of node j by $NBS(j)$. The boundary of this is denoted by (j).In Eqn (1) $h^t(i,j)$ is computed using the hitting time from its direct neighbors to j, which are computed in the $t-1^{th}$ iteration. Since only the hitting times of nodes within $NBS(j)$ are stored, a boundary node would not have access to the hitting time of at least one of its neighbors. Those values can be upper and lower bounded as follows. The fastest possible way to reach node j from any node outside $NBS(j)$ would be by jumping to the node on the boundary (j) which has the closest optimistic hitting time to j. This gives us a lower bound on the hitting time of all nodes outside $NBS(j)$ to j.

$$lb(j) = 1 + \min_{p \in \delta(j)} ho_{pj}^{T-1} \tag{2}$$

The pessimistic bound is T. Plugging in these bounds in equation (1) whenever the neighbors are outside the neighborhood of the destination gives the expressions for optimistic (ho_{ij}^T) and pessimistic (hp_{ij}^T) bounds on hitting times (details in Sarkar and Moore (2007)).

Now we have the expressions for the lower and upper bounds for the hitting times of the nodes in $NBS(j)$ to j (ho and hp values). The hitting time from j to nodes within $NBS(j)$ can be estimated using the sampling scheme described in section 3.1. Combining the two leads to the following.

Theorem 3.3 *The truncated commute time between nodes $i \in NBS(j)$ and j will be lower and upper bounded by co_{ij}^T and cp_{ij}^T with probability $1 -$ if the number of samples for estimating $\hat{h}_{ij}^T$ exceeds the lower bound in theorem 3.1, where*

$$co_{ij}^T = \hat{h}_{ji}^T + ho_{ij}^T - \epsilon T \tag{3}$$

$$cp_{ij}^T = \hat{h}_{ji}^T + hp_{ij}^T + \epsilon T \tag{4}$$

We would use $\widehat{co}_{ij} = \hat{h}_{ji}^T + ho_{ij}^T$ and similarly $\widehat{cp}_{ij}$ to denote estimates of these bounds. In order to prune away nodes which are not potential nearest neighbors we also need to obtain a lower bound on the commute time between j and any node outside $NBS(j)$. The incoming lower bound is given by equation 2. Now note that for the outgoing lower bound we need the minimum of $h_{jk}^T, \forall k \notin NBS(j)$.

Lemma 3.4 *The number of samples M should be at least $\frac{1}{2\epsilon^2}\log(\frac{2n}{\delta})$ in order to obtain $Pr(|\min_{k \notin NBS(j)} \hat{h}_{jk}^T - \min_{k \notin NBS(j)} h_{jk}^T| \geq \epsilon T) \leq 2$.*

Thus an estimate of the outgoing lower bound can be computed from the hitting times obtained from sampling. Combining the two we obtain an estimate on the lower bound on $2T$-truncated commute time $\widehat{lb\text{-}ct}(j)$ from j to any node outside $NBS(j)$.

$$\widehat{lb\text{-}ct}(j) = 1 + \min_{p \in \delta(j)} ho_{pj}^{T-1} + \min_{k \notin NBS(j)} \hat{h}_{jk}^T \tag{5}$$

For our implementation, we always used estimated, not the exact bounds. This introduces an additive error in our results (proof excluded for lack of space).

3.3. Expanding Neighborhood

Now we need to find a heuristic to expand the neighborhood such that both the outgoing and incoming

components of the lower bound increase quickly so that the threshold of $2T'$ is reached soon.

For the incoming lower bound we just find the x closest nodes on the boundary which have small optimistic hitting time to the query. We add the neighbors of these nodes to the neighborhood of j. For the outgoing lower bound, we compute the nodes outside the boundary which a random walk is most probable to hit. We do this by maintaining a set of paths from j which stop at the boundary. These paths are augmented one step at a time. This enables one step look-ahead in order to figure out which nodes outside the boundary are the most probable nodes to be hit. We add y of these nodes to the current boundary.

3.3.1. Ranking

The ranking scheme is similar to GRANCH and is rather intuitive. So far we only have lower and upper bounds for commute times from node j to the nodes in $NBS(j)$. Lets denote this set as S. The commute time from j to any node outside S is guaranteed to be bigger than $2T'$. The true k^{th} nearest neighbor will have commute time larger than the k^{th} smallest lower bound i.e. *co* value. Lets denote the k^{th} smallest *co* value by X. Now consider the nodes which have upper bounds (*cp* values) smaller than X. These are guaranteed to have commute time smaller than the true k^{th} nearest neighbor. Adding a multiplicative slack of α to X allows one to return the α-approximate k nearest neighbors which have commute time within $2T'$. Note that the fact that no node outside set S has hitting time smaller than $2T'$ is crucial for ranking, since that guarantees the *true k^{th} nearest neighbor* within $2T'$ commute distance to be within S. Since all our bounds are probabilistic, i.e. are true with high probability (because of the sampling), we return α-approximate k nearest neighbors with high probability. Also the use of estimated bounds ($\widehat{co}$,$\widehat{cp}$) will introduce an additive error of $2\epsilon T$ (ignoring a small factor of $\epsilon\alpha T$).

3.4. The Algorithm at a Glance

In this section we describe how to use the results in the last subsections to compute nearest neighbors in truncated commute time from a node. *Given T, α, k our goal is to return the top k α-approximate nearest neighbors (within $2\epsilon T$ additive error) w.h.p.*

First we compute the outgoing hitting times from a node using sampling. We initialize the neighborhood with the direct neighbors of the query node (We have set up our graph so that there are links in both directions of an edge, only the weights are different). At any stage of the algorithm we maintain a bounded neighborhood for the query node. For each node inside the neighborhood the hitting times *to* the query can be bounded using dynamic programming. Combining these with the sampled hitting times gives us the estimated $\widehat{co}$, and $\widehat{cp}$ values. We also keep track of the lower bound $\widehat{lb\text{-}ct}$ of the commute time from any node outside the neighborhood to the query node. At each step we expand the neighborhood using the heuristic in section 3.3. Similar to GRANCH we recompute the bounds again, and keep expanding until $\widehat{lb\text{-}ct}$ exceeds $2T'$. W.h.p this guarantees that all nodes outside the neighborhood have commute time larger than $2T'-\epsilon T$.

Then we use the ranking as in section 3.3.1 to obtain k α-approximate nearest neighbors (with an additive slack of $2\epsilon T$) in commute time. We start with a small value of T' and increase it until all k neighbors can be returned. As in Sarkar and Moore (2007) it is easy to observe that the lower bound can only increase, and hence at some point it will exceed $2T'$ and the algorithm will stop. The question is how many nodes can be within $2T'$ commute distance from the query node. In section 4 we prove that this quantity is not too large for most query nodes.

4. Locality Properties of h^T

In this section we analyze the locality properties of truncated hitting times. We show that *most* nodes in a graph will have only a *small* number of neighbors within $2T'$ T-truncated commute time. We would do this in three steps. First we show that number of nodes within hitting time T' **from** a node i is small. Then we would make a similar argument that the number of nodes within T'- hitting distance **to** i is also small. Finally we would make an argument about the neighbors of i in commute time.

Theorem 4.1 *For any graph G and constants T and T', the number of nodes within a truncated hitting distance of T' from any node is at most $T^2/(T-T')$.*

Let $P_{ij}^{<T}$ denote the probability of hitting node j starting at i within T steps and $\tilde{P}_{ij}^{t}$ the probability of hitting j in exactly t steps for the first time from i.

$$T' \geq h_{ij} \geq T(1-P_{ij}^{<T}) \implies P_{ij}^{<T} \geq \frac{T-T'}{T} \qquad (6)$$

Define S_i as the neighborhood of i which consists of only the nodes within hitting time T' **from** i.

$$\sum_{j\in S_i} P_{ij}^{<T} = \sum_{j\in S_i}\sum_{t=1}^{T-1} \tilde{P}_{ij}^{t} \leq \sum_{t=1}^{T-1}\sum_{j\in S_i} P_{ij}^{t} \leq T-1$$

However the left hand side is lower bounded by $|S_i|\frac{T-T'}{T}$ using (6). Which leads us to the upper bound

$|S_i| \leq \frac{T^2}{T-T'}$. So all total there are only $N\frac{T^2}{T-T'}$ pairs within T' hitting distance. If T and T' are constant w.r.t n, then using the above bound and counting arguments we can also show that there can be at most $O(\sqrt{n})$ nodes with more than $\sqrt{n}$ nodes within T' hitting time *to them*.

We have shown that not more than $O(\sqrt{n})$ nodes can have more than $O(\sqrt{n})$ nodes with hitting time smaller than T' **to** them. We already have a bound of $T^2/(T-T')$ on the number of nodes with hitting time smaller than T' **from** a node. We want to bound the number of nodes within commute time $2T'$.

In other words we have proven that $\{j|h_{ij}^T \leq T'\}$ and $\{j|h_{ji}^T \leq T'\}$ are small. Now we need to prove that $\{j|h_{ij}^T + h_{ji}^T \leq 2T'\}$ is also small. Note that the above set consists of

1. $S_1 = \{j|h_{ij}^T \leq T' \bigcap h_{ij}^T + h_{ji}^T \leq 2T'\}$
2. $S_2 = \{j|h_{ij}^T > T' \bigcap h_{ij}^T + h_{ji}^T \leq 2T'\}$

S_1 can have at most $T^2/(T-T')$ nodes. Now consider S_2. S_2 will have size smaller than $|\{j|h_{ji}^T \leq T'\}|$. Using our result from before we can say the following

Lemma 4.2 *Let T be constant w.r.t n. If T' is bounded away from T by a* ***constant*** *w.r.t n, i.e. $T^2/(T-T')$ is constant w.r.t n, then not more than $O(\sqrt{n})$ nodes will have more than $O(\sqrt{n})$ neighbors with truncated commute time smaller than $2T'$.*

The impact of lemma 4.2 is that in a sequence of $O(\sqrt{n})$ nearest neighbor queries (each selected at random), for each node, there would be at most $O(\sqrt{n})$ other nodes within $2T'$ commute distance on average.

5. Empirical Results

We have examined our algorithm on Entity Relation (ER) datasets extracted from the Citeseer corpus, as in Chakrabarti (2007). This is a graph of authors, papers and title-words extracted from Citeseer.

5.1. Dataset and Link Structure

The link structure is obvious:

1. Between a paper and a word appearing in its title.
2. From a paper to the paper it cites, and one with one-tenth the strength the other way.
3. Between a paper and each of its authors.

As observed by Chakrabarti (2007), the weights on these links are of crucial importance. Unlike some other approaches (Balmin et al., 2004; Chakrabarti, 2007) we also put links *from* the paper layer *to* the word layer. This allows flow of information from one paper to another via the common words they use. The links between an author and a paper are undirected. The links within the paper layer are directed. We use the convention in Chakrabarti (2007) to put a directed edge from the cited paper to the citing paper with one-tenth the strength.

For any paper we assign a total weight of W to the words in its title, a total weight of P to the papers it cites and A to the authors on it. We use an inverse frequency scheme for the paper-to-word link weight, i.e. the weight on link from paper p to word w is $W \times 1/f_w/(\sum_{p\to u} 1/f_u)$, where f_w is the number of times word w has appeared in the dataset. We set $W = 1, A = 10, P = 10$ so that the word layer to paper layer connection is almost directed. We add a self loop to the leaf nodes, with the same weight as its single edge, so that the hitting times from these leaf nodes are not very small.

We use two subgraphs of Citeseer. The small one has around $75,000$ nodes and $260,000$ edges: $16,445$ words, $28,719$ papers and $29,713$ authors. The big one has around $600,000$ nodes with 3 million edges : $81,664$ words, $372,495$ papers and $173,971$ authors.

5.2. Preprocessing

We remove the stopwords and all words which appear in more than 1000 papers from both the datasets. The number of such words was around 30 in the smaller dataset and 360 in the larger one. We will make the exact dataset used available on the web.

5.3. Experiments

The tasks we consider are as follows,

1. Paper prediction for words (Word task): We pick a paper X at random, remove the links between it and its title words. Given a query of exactly those words we rank the papers in the training graph. For different values of y the algorithm has a score of 1 if X appears in the closest y papers. *For any search engine, it is most desirable that the paper appears in the top k results, $k \leq 10$.*
2. Paper prediction for authors (Author task): Exactly the above, only the link between the paper and its authors are removed.

The hybrid algorithm is compared with: 1) Exact truncated hitting time from the query, 2) Sampled truncated hitting time from the query, 3) Exact truncated commute time from the query, 4) Exact truncated hitting time to the query, 5) Personalized Pagerank Vector and 6) Random predictor. Note that we can compute a high accuracy estimate of the *exact* hitting time *from* a query node by using a huge number of

samples. We can also compute the exact hitting time *to* a node by using dynamic programming by iterating over all nodes. Both of these will be slower than the sampling or the hybrid algorithm as in Table 1.

Distance from a set of nodes Hitting and commute times are classic measures of proximity from a single node. We extend these definitions in a very simple fashion in order to find near-neighbors of a set of nodes. The necessity of this is clear, since a query often consists of more than one word. We define the hitting time from a set of nodes as an weighted average of the hitting times from the single nodes. For hitting time to a set, we can change the stopping condition of a random walk to "stop when it hits any of the nodes in the set". We achieve this via a simple scheme: for any query q, we *merge* the nodes in the query in a new mega node Q as follows. For any node $v \notin Q$ $P(Q, v) = \sum_{q \in Q} w(q)P(q, v)$, where $P(q, v)$ is the probability of transitioning to node v from node q in the original graph. $\sum_{q \in Q} w(q) = 1$. We use a uniform weighing function, i.e. $w(q) = 1/|Q|$. The hitting/commute time is computed on this modified graph from Q. These modifications to the graph are local and can be done at query time, and then we undo the changes for the next query.

Our **average query size** is the average number of words (authors) per paper for the word (author) task. These numbers are around 3 (2) for the big subgraph, and 5 (2) for the small subgraph.

Figure 1 has the performance of all the algorithms for the author task on the (A) smaller, (B) larger dataset and the word task on the (C) smaller and (D) larger dataset, and Table 1 has the average runtime. As mentioned before for any paper in the testset, we remove all the edges between it and the words (authors) and then use the different algorithms to rank all papers within 3 hops of the words (5 for authors, since authors have smaller degree and we want to have a large enough candidate set) in the new graph and the removed paper. For any algorithm the percentage of papers in the testset which get ranked within the top k neighbors of the query nodes is plotted on the y axis vs. k on the x axis. We plot the performances for six values of k: $1, 3, 5, 10, 20$ and 40.

The results are extremely interesting. Before going into much detail let us examine the performance of the exact algorithms. Note that for the author task the *exact hitting time* ***to a node*** and the *exact commute time from a node* consistently beats the *exact hitting time* ***from a node***, and PPV. However for the word task the outcome of our experiments are **the opposite**. This can be explained in terms of the inherent directionality of a task. The distance *from* the word-layer to the paper-layer gives more information than the distance from the paper layer to the word layer, whereas both directions are important for the author task, which is why commute times, and hitting time to a node outperform all other tasks.

We only compare the predictive power of PPV with our measures, not the *runtime*. Hence we use the exact version of it. We used $c = 0.1$, so that the average path-length for PPV is around 10, since we use $T = 10$ for all our algorithms (however, much longer paths can be used for the exact version of PPV). PPV and hitting time from a node essentially relies on the probability of reaching the destination from the source. Even though hitting time uses information only from a truncated path, in all of our experiments it performs better than PPV, save one, where it behaves comparably.

Word task: The sampling based hitting time beats PPV consistently by a small margin on the bigger dataset, whereas it performs comparably on the smaller one. Hitting times and PPV beat the hitting time to nodes for the word task. In fact for $k = 1, 3, 5$, for the smaller dataset the hitting time *to* a query node isn't much of an improvement over the random predictor (which is zero). This emphasizes our claim that the hitting time *to* the word layer does not provide significant information for the word task. As a result the performance of the exact and hybrid commute times deteriorates.

Author task: The hitting time *to the query* and the exact commute time *from a query* have the best performance by a large margin. The hybrid algorithm has almost similar performance. Hitting time from the query is beaten by these. PPV does worse than all the algorithms except of course the random predictor.

Number of samples: For the small graph, we use $100,000$ samples for computing the high accuracy approximation of hitting time from a node; 5000 samples for the word task and 1000 samples for the author task. We use 1.5 times each for the larger graph. We will like to point out that our derived sample complexity bounds are interesting for their asymptotic behavior. In practice we expect much fewer samples to achieve low probability of error. In Figure 1 sometimes the exact hitting (commute) times does worse than the sampled hitting time (hybrid algorithm). This might happen by chance, with a small probability.

6. Related Work

In this section we briefly examine algorithms which have been developed using random walks on graphs, and their applications. Brand (2005) uses different random walk based measures to compute the top k rec-

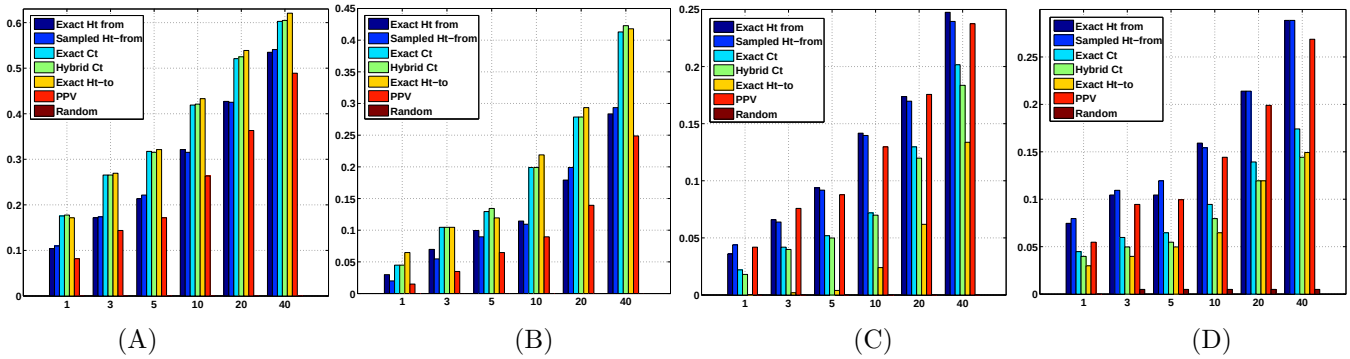


Figure 1. Author task for (A) Small and (B) Large datasets. Word task for (C) Small and (D) Large datasets. x-axis denotes the number of neighbors and y-axis denotes accuracy.

Table 1. Run-time in seconds for Exact hitting time from query, Sampled hitting time from query, Exact commute time, Hybrid commute time, Exact hitting time to query, PPV

# nodes	# edges	Task	Exact Ht-from	Sampled Ht-from	Exact Ct	Hybrid Ct	Exact Ht-to	PPV
74,877	259,320	Author	1.8	**.02**	9.2	**.28**	6.7	18
		Word	3.1	**0.3**	10.4	**1.2**	6.56	50
628, 130	2, 865, 660	Author	6.9	**.07**	79.07	**1.8**	67.2	337.5
		Word	12.3	**0.7**	88.0	**4.3**	70	486

ommendations for a particular customer in a customer-movie graph from the movielens dataset. The submatrices of the hitting and commute times matrices are computed by iterative sparse matrix multiplications (details in Sarkar and Moore (2007)). However it is only tractable to compute these measures on graphs with a few thousand nodes for most purposes.

Liben-Nowell and Kleinberg (2003) showed that the hitting and commute times perform poorly for link prediction tasks, because of their sensitivity to long paths. The most effective measure was shown to be the Katz measure (Katz, 1953) which directly sums over the collection of paths between two nodes with exponentially decaying weights. However, ranking under the Katz score would require solving for a row of the matrix $(I-\gamma A)^{-1}-I$, where I and A are the identity and adjacency matrices of the graph and γ is the decay factor. Even if A is sparse the fast linear solvers will take at least $O(E)$ time.

Tong et al. (2007) uses escape probability from node i to node j to compute *direction aware proximity* in graphs. A fast matrix solver is used to compute this between *one pair of nodes*, in $O(E)$ time. Multiple pairs of proximity require computation and storage of the inverse of the matrix $I-\gamma P$, which would be intractable for large graphs ($10K$ nodes). Jeh and Widom (2002b) use the notion of *expected f-hitting distance*, which is the hitting time (in a random walk with restart) between a set of nodes in a product graph with N^2 nodes. The quadratic time complexity is reduced by limiting the computation between source and destination pairs within distance of r.

The main idea of *personalized pagerank* is to bias the probability distribution towards a set of webpages particular to a certain user, resulting in a user-specific view of the web. It has been proven (Jeh & Widom, 2002a) that the PPV for a set of webpages can be computed by linearly combining those for each individual webpage. However it is hard to store all possible personalization vectors or compute the personalized pagerank vector at query time because of the sheer size of the internet graph. There have been many novel algorithms for efficiently computing PPV (Jeh & Widom, 2002a; Haveliwala, 2002; Fogaras et al., 2004). Most of these algorithms compute partial PPVs offline and combine them at query time.

The *ObjectRank* algorithm (Balmin et al., 2004) computes keyword-specific ranking in a publication database of authors and papers, where papers are connected via citations and co-authors. The personalized pagerank for each word is computed and stored offline, and at query time combined linearly. Chakrabarti

(2007) et al. show how to compute approximate personalized pagerank vectors using clever neighborhood expansion schemes which would drastically reduce the amount of offline storage and computation.

7. Conclusion

Many graph-based learning algorithms rely on computing proximity measures in graphs. These graphs can be very large and undergoing continuous change, hence fast *incremental* algorithms are needed. In this paper we have combined sampling techniques with branch and bound pruning to compute near neighbors of a query node with high probability. Our proximity measures have been empirically shown to often outperform a popular alternative, namely personalized pagerank on link-prediction tasks.

Acknowledgements

Thanks to Soumen Chakrabarti for sharing his code and data, Tamás Sarlós and Martin Zinkevich for helpful discussions and Geoffrey Gordon and Anupam Gupta for valuable suggestions.

Appendix

Proof of Theorem 3.1: We provide a bound on the number of samples M required in order to give an ϵ-correct answer with probability $1-\delta$. We denote the estimate of a random variable x by $\hat{x}$ from now on. Lets denote by $X^r(i,u)$ the first arrival time at node u from node i on the r^{th} trial. Define $X^r(i,u)=T$ if the path does not hit u on trial r. Note that $\hat{h}^T(i,u)=\sum_r X^r(i,u)/M$, and $E[\hat{h}^T(i,u)]=h^T(i,u)$. $\{X^r(i,u)\in[1,T], r=1:M\}$ are i.i.d. random variables. The Hoeffding bound gives

$$P(|\hat{h}^T(i,u)-h^T(i,u)|\geq\epsilon T)\leq 2\exp(-\tfrac{2M(\epsilon T)^2}{T^2})$$
$$=2\exp(-2M\epsilon^2)$$

Now we want the probability of a bad estimate for any u to be low. We upper bound this error probability using union and Hoeffding bounds and set the upper bound to be less than a small value δ. Hence we have $P(\exists u\in\{1,\ldots,n\}, |\hat{h}^T(i,u)-h^T(i,u)|\geq\epsilon T)\leq 2n\exp(-2M\epsilon^2)\leq\delta$. This gives the lower bound of $\frac{1}{2\epsilon^2}\log(\frac{2n}{\delta})$.

Proof of Theorem 3.2: Consider a sampled path of length T starting from i. We define $X^r(i,u)$ as before. For two arbitrary nodes u and v, WLOG let $h^T(i,u)>h^T(i,v)$. The idea is to define a random variable whose expected value will equal $h^T(i,u)-h^T(i,v)$. We define a random variable $Z^r=X^r(i,u)-X^r(i,v)$. $\{Z^r\in[-(T-1),T-1], r=1:M\}$ are i.i.d. random variables. Note that $E(Z^r)=h^T(i,u)-h^T(i,v)$.

The probability that the ranking of u and v will be exchanged in the estimated h^T values from M samples equals $P(\hat{h}^T(i,u)<\hat{h}^T(i,v))$. This probability equals $P(\sum_{r=1}^M Z_r/M<0)$ which using the Hoeffding bound is smaller than $\exp(-2M(h^T(i,u)-h^T(i,v))^2/(2T)^2)=\exp(-M(h^T(i,u)-h^T(i,v))^2/2T^2)$. Let $v_1,v_2,\ldots,v_k$ be the top k neighbors of i in exact truncated hitting time.

$$Pr(\exists j\leq k, q>k, \hat{h}^T(i,v_j)>\hat{h}^T(i,v_q))$$
$$\leq\sum_{j\leq k}\sum_{q>k}Pr(\hat{h}^T(i,v_j)>\hat{h}^T(i,v_q))$$
$$\leq\sum_{j\leq k}\sum_{q>k}\exp(-\tfrac{M(h^T(i,v_q)-h^T(i,v_j))^2}{2T^2})$$
$$\leq nk\exp(-\tfrac{M(h^T(i,v_{k+1})-h^T(i,v_k))^2}{2T^2})$$

Let $\alpha=h^T(i,v_{k+1})-h^T(i,v_k)$. Setting the above probability to be less than δ gives us the desired lower bound of $\frac{2T^2}{\alpha^2}\log(nk/\delta)$ on M.

Proof of lemma 3.4: Let S be a set of nodes. Let $q=\arg\min_{k\in S}h(j,k)$. Let $m=\arg\min_{k\in S}\hat{h}(j,k)$. We know that $h(j,q)\leq h(j,m)$, since q is the true minimum, and $\hat{h}(j,m)\leq\hat{h}(j,q)$, since m is the node which has the minimum estimated h-value. Using the sample complexity bounds from theorem 3.1, we have $\hat{h}(j,m)\leq\hat{h}(j,q)\leq^{w.h.p}h(j,q)+\epsilon T$. For the other part of the inequality we have $\hat{h}(j,m)\geq^{w.h.p}h(j,m)-\epsilon T\geq h(j,q)-\epsilon T$. Using both sides we get $h(j,q)-\epsilon T\leq^{w.h.p}\hat{h}(j,m)\leq^{w.h.p}h(j,q)+\epsilon T$. Using S to be the set of nodes outside the neighborhood of j yields lemma 3.4.

References

Aldous, D., & Fill, J. A. (2001). *Reversible markov chains.*

Balmin, A., Hristidis, V., & Papakonstantinou, Y. (2004). ObjectRank: Authority-based keyword search in databases. *VLDB, 2004.*

Brand, M. (2005). A Random Walks Perspective on Maximizing Satisfaction and Profit. *SIAM '05.*

Chakrabarti, S. (2007). Dynamic personalized pagerank in entity-relation graphs. *WWW '07* (pp. 571–580). New York, NY, USA: ACM Press.

Fogaras, D., Rácz, B., Csalogány, K., & Sarlós, T. (2004). Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments. .

Haveliwala, T. (2002). Topic-sensitive pagerank. *WWW.*

Jeh, G., & Widom, J. (2002a). Scaling personalized web search. *Stanford University Technical Report.*

Jeh, G., & Widom, J. (2002b). Simrank: A measure if structural-context similarity. *KDD.*

Katz, L. (1953). A new status index derived from sociometric analysis. *Psychometrika.*

Liben-Nowell, D., & Kleinberg, J. (2003). The link prediction problem for social networks. *CIKM '03.*

Sarkar, P., & Moore, A. (2007). A tractable approach to finding closest truncated-commute-time neighbors in large graphs. *Proc. UAI.*

Spielman, D., & Srivastava, N. (2008). Graph sparsification by effective resistances. *Proceedings of the STOC'08.*

Tong, H., Koren, Y., & Faloutsos, C. (2007). Fast direction-aware proximity for graph mining. *Proc. KDD.*

Inverting the Viterbi Algorithm: An Abstract Framework for Structure Design

Michael Schnall-Levin MSCHNALL@MATH.MIT.EDU
Leonid Chindelevitch LEONIDUS@MATH.MIT.EDU
Bonnie Berger[1] BAB@MIT.EDU
Department of Mathematics and Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge MA

Abstract

Probabilistic grammatical formalisms such as hidden Markov models (HMMs) and stochastic context-free grammars (SCFGs) have been extensively studied and widely applied in a number of fields. Here, we introduce a new algorithmic problem on HMMs and SCFGs that arises naturally from protein and RNA design, and which has not been previously studied. The problem can be viewed as an inverse to the one solved by the Viterbi algorithm on HMMs or by the CKY algorithm on SCFGs. We study this problem theoretically and obtain the first algorithmic results. We prove that the problem is NP-complete, even for a 3-letter emission alphabet, via a reduction from 3-SAT, a result that has implications for the hardness of RNA secondary structure design. We then develop a number of approaches for making the problem tractable. In particular, for HMMs we develop a branch-and-bound algorithm, which can be shown to have fixed-parameter tractable worst-case running time, exponential in the number of states of the HMM but linear in the length of the structure. We also show how to cast the problem as a Mixed Integer Linear Program.

1. Introduction

Probabilistic grammatical formalisms such as hidden Markov models (HMMs) and stochastic context-free grammars (SCFGs) have found many applications in areas such as computational biology and natural language processing. Because of their intuitive representation, their power to capture some of the essential relationships present in data, and the existence of polynomial-time algorithms (such as the Viterbi algorithm) and practical training procedures (such as the Baum-Welch algorithm), these formalisms have enjoyed tremendous popularity in the past decades.

Three natural problems for a model have been described: the decoding problem (given a model and a sequence, find the most likely derivation), the evaluation problem (given a model and a sequence, find the likelihood of the sequence being generated), and the learning problem (given a set of sequences, learn the parameters of the underlying model). In this paper, we identify another natural problem on HMMs and SCFGs, which is the inverse of the decoding problem: given a derivation and a model, find a sequence for which this derivation is the most likely one. Because the decoding problem is solved by the Viterbi algorithm in HMMs and by the CKY algorithm in SCFGs, we refer to our problem for these two models as the Inverse-Viterbi and the Inverse-CKY problem, respectively.

The motivation for our problem comes from protein and RNA design. The design of biological molecules with a desired structure is a long sought-after goal in computational biology. While a number of achievements have been made in protein structure design, the problem remains difficult (Butterfoss & Kuhlman, 2005; Park et al., 2004; Pokala & M., 2001). For RNA, there has been recent interest in secondary structure design (Breaker, 1996), and a number of fairly successful heuristics have been developed to solve this problem (Hofacker, 1994; Andronescu, 2004; Busch & Backofen, 2006). Generally, structure design can be divided into two goals: the positive-design aspect of finding a sequence that has low energy in the desired structure, and the negative-design aspect of blocking the sequence from having low energy in other struc-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1]To whom correspondence should be addressed

tures. While some work has explored the negative-design aspect in protein structure design (Butterfoss & Kuhlman, 2005), most work has focused solely on the positive-design aspect. In RNA secondary structure design, the positive-design aspect is largely trivial (desired paired positions in the secondary structure can simply be chosen to be complementary bases) and the negative-design aspect, which involves attempting to block erroneous base pairings in other structures, is central to solving the problem.

Our goal in formulating the Inverse-Viterbi and Inverse-CKY problems is to simultaneously capture the positive-design and negative-design aspects of the design problem. Our framework for viewing design within the context of HMMs or SCFGs is the following. When HMMs (SCFGs) are used for structure prediction, the emitted string represents the biological sequence and the goal is to find the hidden state-path (derivation tree for SCFGs) that represents the structure this sequence will adopt. A state-path of high probability for that sequence is the analogue of a structure with low energy. By inverting this problem, we can use the same HMM (SCFG) for design. Now, a state-path (derivation tree) representing the desired structure is known and the goal is to find a sequence which will adopt this structure (i.e. a string for which this state-path is optimal).

Our Contribution. We have defined a novel problem (the Inverse-Viterbi problem) on HMMs and its analogue on SCFGs, that as far as we know has never been studied before. We show that the problem is NP-hard for HMMs (and as a result for SCFGs). We then give approaches for making the problem tractable. In particular, for HMMs we give a branch-and-bound algorithm. This algorithm can be shown to have fixed-parameter tractable running time: if there are K states, the emission alphabet is Σ, the path length is n, and all of the log-probabilities in the model are greater than $-B$ (so that there are no 0 transition probabilities and all the probabilities in the model are greater than e^{-B}) and are defined to a precision δ, then the branch-and-bound algorithm has worst-case running time $O((2B/\delta)^{K-2}nK^2|\Sigma|)$, which is exponential in the number of states but linear in the path length. We also show how to cast the problem as a simple Mixed Integer Linear Program.

Our hardness proof shows that the RNA secondary structure design problem is hard in a certain sense: a polynomial-time algorithm that only depends on the energy model for RNA secondary structure being SCFG-like, as is the case for the Zuker energy model (the most successful model curently available for RNA secondary structure prediction (Zuker & Stiegler, 1981)), without making additional assumptions on the particular form of the energy model, is not possible unless $P = NP$.

In presenting an abstract formulation of the design problem and giving the theoretical results derived in this paper, our goal is not to provide methods that will necessarily be immediately applied to the protein or RNA structure design problems. Instead, we believe that the abstract framework given in this paper may prove to be useful in understanding the design problem and facilitating the development of new methods for design. The Inverse-Viterbi and Inverse-CKY problems are novel and natural problems on HMMs and SCFGs, and so we believe a theoretical exploration is interesting in its own right.

2. Problem Description and Hardness Results

2.1. Definition of the Models

An HMM consists of a set $\mathcal{N}$ of K states and an alphabet Σ, with $\mathcal{N} \cap \Sigma = \emptyset$. The symbols in Σ are emitted on transitions between the states. The probability of emitting the symbol a when transitioning from the state s_k to the state s_l is specified by the value of the parameter $p^a_{s_k,s_l}$. These parameters determine the HMM. We assume (without loss of generality) that there is a unique initial state S.

The normalization condition requires that

$$\sum_{s_l \in \mathcal{N}} \sum_{a \in \Sigma} p^a_{s_k,s_l} = 1 \text{ for } k = 1, \ldots, K$$

Similarly, an SCFG consists of a set $\mathcal{N}$ of K non-terminal symbols, and a set Σ of terminal symbols, with $\mathcal{N} \cap \Sigma = \emptyset$. The non-terminals are rewritten according to a set $\mathcal{R}$ of rewriting rules. The probability of applying each rewriting rule α is specified by the value of the parameter p_α. These parameters determine the SCFG. We assume (without loss of generality) that there is a unique starting non-terminal symbol S.

Every rule α replaces a single non-terminal with a string γ of non-terminals and terminals:

$$\alpha = N_k \rightarrow \gamma$$

Here N_k (the terminal symbol being rewritten) is referred to as the left-hand side of the rule, abbreviated as $l(\alpha)$.

The normalization condition requires that

$$\sum_{\{\alpha \in \mathcal{R} | l(\alpha) = N_k\}} p_\alpha = 1, \text{ for } k = 1 \ldots, , K$$

We do not insist that the SCFG be in Chomsky Normal Form (CNF) because in some applications (such as RNA secondary structure design), the correspondence between the design and inverse problem defined in this paper may only be natural if the SCFG is not converted to CNF.

We use boldface letters to indicate sequences of symbols. A state-path of length n in the HMM is written as $\boldsymbol{\pi} = \pi_1 \ldots \pi_n$, where each π_i is a state in the HMM. Such a path emits a sequence of $n-1$ emission symbols, $\boldsymbol{\omega} = \omega_1 \ldots \omega_{n-1}$ where each ω_i is a symbol from Σ. The joint probability of a state-path $\boldsymbol{\pi}$ and an emission sequence $\boldsymbol{\omega}$ is given by $\Pr(\boldsymbol{\pi}, \boldsymbol{\omega}) = \prod_{i=1}^{n-1} p^{\omega_i}_{\pi_i, \pi_{i+1}}$. It is frequently more convenient to deal with sums rather than products, and so we work in log-space, taking $q^a_{s_1,s_2} := \log(p^a_{s_1,s_2})$ and therefore $\log(\Pr(\boldsymbol{\pi}, \boldsymbol{\omega})) = \sum_{i=1}^{n-1} q^{\omega_i}_{\pi_i, \pi_{i+1}}$.

A derivation of length n in the SCFG is the successive application of rewriting rules, beginning with the starting symbol S, which generates a yield $\boldsymbol{\omega} = \omega_1 \ldots \omega_n$ where each ω_i is a symbol from Σ. The derivation can be summarized in the form of a tree $\mathcal{T}$. The joint probability of a derivation tree $\mathcal{T}$ and a yield $\boldsymbol{\omega}$ is given by $\Pr(\mathcal{T}, \boldsymbol{\omega}) = \prod_{\alpha \in \mathcal{R}(\mathcal{T})} p_\alpha$, where $\mathcal{R}(\mathcal{T})$ denotes the multiset of rewriting rules used to derive $\mathcal{T}$. As with HMMs, it is convenient to work instead with the log-probabilities, $q_\alpha := \log p_\alpha$, which gives $\log(\Pr(\mathcal{T}, \boldsymbol{\omega})) = \sum_{\alpha \in \mathcal{R}(\mathcal{T})} q_\alpha$.

2.2. Definition of the Direct Problem

In the original Viterbi problem, one is given an emission sequence $\boldsymbol{\omega_0}$ from an HMM and the goal is to find the most likely state-path to have generated $\boldsymbol{\omega_0}$: the $\boldsymbol{\pi}$ that maximizes the conditional probability given the emission $\Pr(\boldsymbol{\pi} | \boldsymbol{\omega_0})$. Since $\Pr(\boldsymbol{\pi} | \boldsymbol{\omega_0}) = \frac{\Pr(\boldsymbol{\pi}, \boldsymbol{\omega_0})}{\Pr(\boldsymbol{\omega_0})}$, and $\boldsymbol{\omega_0}$ is fixed, it is equivalent to simply maximize the joint probability $\Pr(\boldsymbol{\pi}, \boldsymbol{\omega_0})$. The Viterbi problem can therefore be expressed as: given $\boldsymbol{\omega_0}$, find an element of $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi}, \boldsymbol{\omega_0})$ (we consider arg max as the *set* of all arguments maximizing the function). For an HMM with K states and an emission of length n, the Viterbi algorithm finds the best state-path using dynamic programming in time $O(nK^2|\Sigma|)$ (Viterbi, 1967).

Similarly, the direct problem for an SCFG is formulated as follows: given a yield $\boldsymbol{\omega}$, find the derivation tree $\mathcal{T}$ which maximizes the joint probability $\Pr(\mathcal{T}, \boldsymbol{\omega})$. In other words, given $\boldsymbol{\omega}$, we find an element of $\arg\max_{\mathcal{T}} \Pr(\mathcal{T}, \boldsymbol{\omega})$. The optimal derivation is referred to as the Viterbi parse of $\boldsymbol{\omega}$. For a derivation of length n in an SCFG with rewriting rules $\mathcal{R}$ in Chomsky Normal Form, the CKY algorithm finds the Viterbi parse in time $O(n^3|\mathcal{R}|)$ (Durbin et al., 1999). Modified versions of the CKY algorithm can also handle SCFGs in similar forms, such as those used in RNA structure prediction, with the same time complexity (for example see (Dowell & Eddy, 2004)).

2.3. Definition of the Inverse Problem

In the Inverse-Viterbi problem, a desired output of the Viterbi algorithm is known and the goal is to design an input to the Viterbi algorithm that will return this output. In mathematical terms the problem is: given a state-path $\boldsymbol{\pi_0}$, find an $\boldsymbol{\omega}$ so that $\boldsymbol{\pi_0}$ is in $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi}, \boldsymbol{\omega})$, or determine that none exists.

In an HMM used for structure prediction, the above definition of the inverse problem captures what it means to do structure design: one knows the structure (state-path) and tries to find a sequence that has a higher score with that structure than with any other structure. It is important to emphasize that for many $\boldsymbol{\pi}$ there will be no such $\boldsymbol{\omega}$. In fact, it can be shown that only polynomially many paths are designable (Elizalde & Woods, 2006). This captures the notion that many structures are not designable: there is no sequence that will lead to these structures.

To illustrate this distinction, consider the 2-state HMM shown in Figure 1. Say that the desired state-path to design is $B^n = B \ldots B$. The most likely emission given this state-path is $a^{n-1} = a \ldots a$, but when run on such a path the Viterbi algorithm will not return B^n. In fact, the only sequence that the Viterbi algorithm will return B^n on is b^{n-1}. This simple case illustrates that to design a path of all B's it is important not just to pick emissions likely given this path, but to simulatenously block other possible paths, in this case those paths containing A's. Note further that the probability of b^{n-1} being emitted from B^n at random is $(0.2)^{n-1}$. Therefore, neither picking the most likely emission sequence nor randomly generating sequences from the state-path will in general solve the Inverse-Viterbi problem with probability greater than exponentially small in the length of the state-path.

We incorporate one generalization into our definition of the problem of inverting the Viterbi algorithm, because it seems natural to the design problem. We allow constraints on the emissions that can be chosen in any position (given as the Σ_i below). The algorithms we develop in this paper handle this generalization without any added complexity.

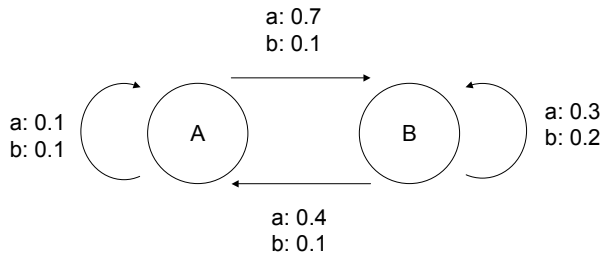


Figure 1. A 2-state HMM illustrating the distinction between the Inverse-Viterbi problem and the trivial problem of finding the most likely emission from a given state-path. The 2 states are A and B, while the 2 possible emissions are a and b. Each transition is marked with the possible emissions followed by their corresponding probabilities. In order to design B^n the only possible sequence is b^{n-1}, which is the least likely sequence to be produced by B^n.

INVERSE-VITERBI:
Input: An HMM, a state-path $\boldsymbol{\pi_0}$ of length n and for every position i in $1, \ldots, n$ a set $\Sigma_i \subseteq \Sigma$ giving allowed emissions at position i.
Output: An $\boldsymbol{\omega}$ where each $\omega_i \in \Sigma_i$ so that $\boldsymbol{\pi_0}$ is in $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi}, \boldsymbol{\omega})$, or $\emptyset$ if no such $\boldsymbol{\omega}$ exists.

Similarly, the inverse problem for an SCFG requires one to find an input that corresponds to a given output. In other words, given a derivation $\mathcal{T}_0$, we would like to find an $\boldsymbol{\omega}$ such that $\mathcal{T}_0$ is in $\arg\max_{\mathcal{T}} \Pr(\mathcal{T}, \boldsymbol{\omega})$, or determine that none exists. Note that this problem only makes sense if the tree $\mathcal{T}_0$ has had all of its leaves removed (we will call such a tree "naked"); in other words, the tree includes the specification of non-terminals but not the terminal symbols produced.

INVERSE-CKY:
Input: An SCFG, a naked derivation tree $\mathcal{T}_0$ that corresponds to an emitted string of n terminals and for every position i in $1, \ldots, n$ a set $\Sigma_i \subseteq \Sigma$ giving the allowed emissions at position i.
Output: An $\boldsymbol{\omega}$ where each $\omega_i \in \Sigma_i$ so that $\mathcal{T}_0$ is in $\arg\max_{\mathcal{T}} \Pr(\mathcal{T}, \boldsymbol{\omega})$, or $\emptyset$ if no such $\boldsymbol{\omega}$ exists.

2.4. NP-hardness of the Inverse Problem

We now prove that the Inverse-Viterbi problem is NP-hard. To do so, we introduce the decision problem corresponding to Inverse-Viterbi:

DESIGNABLE:
Input: An HMM and a state-path $\boldsymbol{\pi_0}$
Output: YES if there is an $\boldsymbol{\omega}$ so that $\boldsymbol{\pi_0}$ is in $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi}, \boldsymbol{\omega})$, otherwise NO.

An algorithm that solves Inverse-Viterbi would also solve Designable and so by proving Designable is NP-complete, we show that Inverse-Viterbi is NP-hard.

Theorem 1. *Designable is NP-Complete.*

Proof. Clearly Designable is in NP so we just need to show Designable is NP-hard. We do so by presenting a polynomial-time reduction from 3-SAT to Designable.

In outline, the construction is achieved by creating an HMM with one component that can emit all possible non-satisfying assignments for the 3-SAT problem along with a special state outside of this component that can emit all binary strings, but that does so with smaller probability. Because this probability is small, the path consisting of repeatedly being in the special state is only designable if a specific sequence of 0's and 1's could not possibly be emitted by the component corresponding to the 3-SAT formula. And such a sequence is, by the construction, a satisfying assignment of the 3-SAT formula.

In full detail, the construction is as follows (see Figure 2 for an illustration). Assume the 3-SAT formula consists of m variables and r clauses. The HMM consists of a begin state B, two special states S and T and $r(m+1)$ states labelled $X_{i,j}$ where $1 \leq i \leq r$ and $1 \leq j \leq m+1$. The emission alphabet consists of 0, 1, and the special symbol #. The state B transitions to either S or any of $X_{i,1}$ with equal probability, $\frac{1}{r+1}$, while emitting #. The state S transitions to itself while emitting 0 or 1, each with probability $\frac{1}{2}$. The state T transitions to itself with probability 1 while emitting #. The r sets of states $X_{i,1}, \ldots, X_{i,m+1}$ for $1 \leq i \leq r$ are arranged in independent chains, each corresponding to the ith clause, that emit all strings $\{0,1\}^m$ that do not satisfy the ith clause. Such a chain is constructed by the following: if the ith clause contains the jth variable un-negated then $X_{i,j}$ transitions to $X_{i,j+1}$ while emitting 0 with probabilty 1, if the ith clause contains the jth variable negated then $X_{i,j}$ transitions to $X_{i,j+1}$ while emitting 1 with probabilty 1, and if the ith clause doesn't contain the jth variable then $X_{i,j}$ transitions to $X_{i,j+1}$ while emitting 0 or 1 each with probability $\frac{1}{2}$. Finally, $X_{i,m+1}$ transitions to T while emitting # with probability 1.

The state-path to design is BS^{m+1}. We observe that the joint probability of this state-path and an emission sequence of the form $\#\{0,1\}^m$ is $(\frac{1}{r+1})(\frac{1}{2})^m$, and that only emissions of this form have non-zero probability for this state-path. We further observe that the only other state-path that could emit such a sequence must be of the form $BX_{i,1} \ldots X_{i,m+1}$, and the joint

probability of such a sequence and such a state-path is $(\frac{1}{r+1})(\frac{1}{2})^{m-3}$ if the emission sequence contains a # followed by a non-satisfying assignment to the 3-SAT formula, but the joint probability is zero if the emission sequence contains a # followed by a satisfying assignment. Since $(\frac{1}{r+1})(\frac{1}{2})^{m-3} > (\frac{1}{r+1})(\frac{1}{2})^{m}$, the only sequence that could design BS^{m+1} is a # followed by a satisfying assignment and therefore BS^{m+1} is designable if and only if there is a satisfying assignment to the 3-SAT formula.

The above construction is done in polynomial time, and therefore we have successfully given a polynomial reduction from 3-SAT to Designable. □

Corollary 1. *Inverse-CKY is NP-hard.*

Proof. An HMM can be thought of as an SCFG with a non-terminal corresponding to each state and a terminal to each letter in the emission alphabet. Every branching rule rewrites a state as a letter and another state, so that all derivation trees are right-branching. Since the problem is hard on HMMs it is also hard on the extended class of SCFGs. □

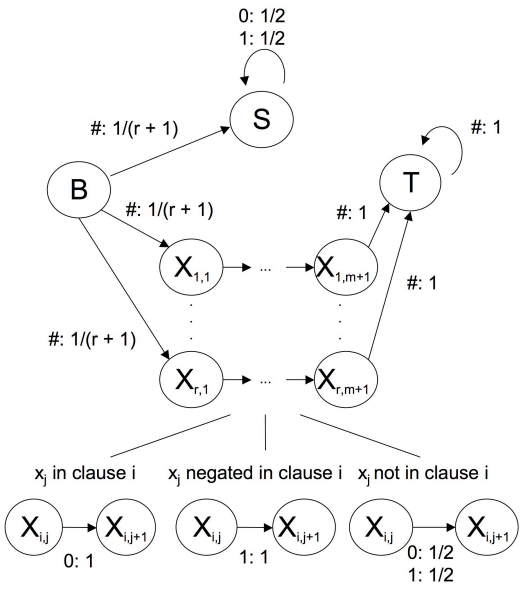


Figure 2. The reduction from 3-SAT to DESIGNABLE. Each transition is marked with all non-zero probability emissions followed by their corresponding probabilities.

3. Algorithmic Results

In this section, we give two approaches for finding a solution to the inverse problem, a branch-and-bound algorithm and a formulation of the problem as a Mixed Integer Linear Program. Both of these are derived from the same basic approach, based on a set of constraints we develop that are satisfied by an $\boldsymbol{\omega}$ if and only if it is a solution to the inverse problem. Below we first develop these constraints. Similar constraints and a Mixed Integer Linear Program can be developed for SCFGs. For reasons of space and simplicity of presentation, we only give the details for HMMs in this section. We illustrate the formulation of constraints and a Mixed Integer Linear Program for an SCFG used for RNA secondary structure prediction in a supplement.[2]

3.1. Constraint Formulation

Conceptually, the set of inequalities for HMMs is derived by looking at how the Viterbi algorithm works and enforcing constraints on $\boldsymbol{\omega}$ so that the Viterbi algorithm is forced to return the desired state-path $\boldsymbol{\pi_0}$.

The Viterbi algorithm calculates an n by K table of values $M_{i,s}$ of the best log-probability scores for the state-path from positions 1 to i with final state s. Because of the special form of the HMM score, this table can be filled in iteratively:
(1) $M_{1,S} = 0$ and $M_{1,s} = -\infty$ for all $s \neq S$
(2) $M_{i,s} = \max_{s'}(M_{i-1,s'} + q_{s',s}^{\omega_{i-1}})$ for $2 \leq i \leq n$ and all s

The best state in the nth position is then read off as $\pi_n \in \arg\max_s(M_{n,s})$, and the earlier ones are read off by a traceback routine: the best state in position $n-1$ is an s' that maximized $(M_{n-1,s'} + q_{s',\pi_n}^{\omega_{n-1}})$, and so on.

From the above we can directly read off the constraints on the emission symbol ω_i in position i for $1 \leq i \leq n-1$, that need to be satisfied in order to design a state-path with states π_i. For the Viterbi algorithm to return the desired path, we need for every state in this path to traceback to the previous state in the desired path and for the last state in this path to have the best log-probability score:
(3) $M_{i,\pi_i} + q_{\pi_i,\pi_{i+1}}^{\omega_i} \geq \max_{s \neq \pi_i}(M_{i,s} + q_{s,\pi_{i+1}}^{\omega_i})$ for $1 \leq i \leq n-1$
(4) $M_{n,\pi_n} \geq \max_{s \neq \pi_n}(M_{n,s})$

3.2. Branch-and-Bound Algorithm

What is particularly nice about inequalities (1)-(4) is that they allow for an inductive method for choosing possible ω_i in an emission sequence based only on the choices of ω_j for $1 \leq j \leq i-1$. This is because the inequality constraining the choice of ω_i (inequality 3 above) only depends on the values for $M_{i,s}$. And the

[2]See http://groups.csail.mit.edu/cb/inv_viterbi/scfg.pdf

values for $M_{i,s}$ only depend on the choices made for ω_1 through ω_{i-1}. This naturally leads to a branch-and-bound algorithm. Branch-and-bound algorithms are frequently useful in solving computationally hard problems. A branch-and-bound algorithm is complete (it always finds the correct answer) and frequently efficient on many problem instances.

The branch-and-bound algorithm steps through position i from 1 to $n-1$, at each step maintaining a list of emission sequences of length i that could be extended to possible length $n-1$ sequences the algorithm will ultimately return. At each step i, the algorithm forms emission sequences of length i from the emission sequences of length $i-1$ stored in the previous stage by appending possible emission symbols onto the sequences from the previous stage. In order to avoid performing an exhaustive search, at every stage the algorithm prunes the search space by applying two elimination rules. The first elimination rule ensures that for a given length $i-1$ sequence from the previous stage, an ω_i is only appended onto this sequence to form a length i sequence if the traceback constraint (constraint 3) is satisfied by the choice ω_i. The second elimination rule examines pairs $\boldsymbol{\omega}$ and $\tilde{\boldsymbol{\omega}}$ of partial strings of length i that remain after the application of the first elimination rule. It eliminates $\boldsymbol{\omega}$ due to $\tilde{\boldsymbol{\omega}}$, if given that $\boldsymbol{\omega}$ can be extended to a solution to the design problem, then $\tilde{\boldsymbol{\omega}}$ must also be able to be extended to a solution.

Specifically, the second elimination rule is based on the following observation. If for all states s, $M_{i+1,\pi_{i+1}} - M_{i+1,s}$ is at least as large under $\tilde{\boldsymbol{\omega}}$ as it is under $\boldsymbol{\omega}$ (i.e. if for all states s, the relative preference of $\tilde{\boldsymbol{\omega}}$ for π_i to state s is at least as large as that of $\boldsymbol{\omega}$), then the traceback constraints (inequality 3 above) on all positions j for $j > i$ and the ending constraints (inequality 4 above) can only be easier to satisfy when extending $\tilde{\boldsymbol{\omega}}$ than when extending $\boldsymbol{\omega}$.

It is important to note that for the case of a 2-state HMM the branch-and-bound is an exact polynomial-time algorithm. This is because there is only one $M_{i+1,\pi_{i+1}} - M_{i+1,s}$ value to compare the choices for ω_i on (there is only one state s other than π_i at every position since there are only 2 states to choose from), and so there is always a *best* choice for ω_i at every position based on the past choices.

The above branch-and-bound algorithm is exact for all HMMs, but has no guaranteed worst-case running time. If we make additional assumptions about our HMM, however, we can show that the algorithm also has fixed-parameter tractable running time. Specifically, we assume that all q values (the log-probabilities)

Algorithm 1 Branch-and-Bound Algorithm

Input: An HMM, a desired state-path $\boldsymbol{\pi_0}$ of length n, and for every position i in $1, \ldots, n$ a set $\Sigma_i \subseteq \Sigma$ giving the allowed emissions at position i
Output: A sequence $\boldsymbol{\omega}$ such that $\boldsymbol{\pi_0}$ is in $\arg\max_{\boldsymbol{\pi}} \Pr(\boldsymbol{\pi}, \boldsymbol{\omega})$ or $\emptyset$ if no such sequence exists.
Variables: A list L_i of all partial sequences of length i considered at the ith iteration each together with its corresponding K-vector of values $M_{i,s}$.
Initialize: $L_0 = \{(\epsilon, \mathbf{0})\}$
for $i = 1$ **to** $n-1$ **do**
  Set $L_i = \emptyset$
  for all $(\boldsymbol{\omega^{i-1}}, \boldsymbol{v^{i-1}}) \in L_{i-1}$ and all $\omega_i \in \Sigma_i$ **do**
    Form $\boldsymbol{\omega^i} = \boldsymbol{\omega^{i-1}}\omega_i$ by concatenation
    Compute the K-vector $\boldsymbol{v^i}$ of values $M_{i+1,s}$
    Add $(\boldsymbol{\omega^i}, \boldsymbol{v^i})$ to L_i iff Elim Rule 1 doesn't apply
  end for
  for all $(\boldsymbol{\omega^i}, \boldsymbol{v^i}) \in L_i$ **do**
    From $\boldsymbol{v^i}$ compute and store the $(K-1)$-vector $\boldsymbol{u}$ of values $M_{i+1,\pi_{i+1}} - M_{i+1,s}$ for $s \neq \pi_{i+1}$
  end for
  Apply Elim Rule 2 to all pairs of entries of L_i
end for
for all $(\boldsymbol{\omega^{n-1}}, \boldsymbol{v^{n-1}}) \in L_{n-1}$ **do**
  if $M_{n,\pi_n} < \max_{s \neq \pi_n}(M_{n,s})$ **then**
    Remove $(\boldsymbol{\omega^{n-1}}, \boldsymbol{v^{n-1}})$ from L_{n-1}
  end if
end for
Return: An element of L_{n-1} or $\emptyset$ if L_{n-1} is empty.
Elim Rule 1: Eliminate $\boldsymbol{\omega^i}$ if $M_{i,\pi_i} + q^{\omega_i}_{\pi_i,\pi_{i+1}} < \max_{s \neq \pi_i}(M_{i,s} + q^{\omega_i}_{s,\pi_{i+1}})$
Elim Rule 2: Eliminate $\boldsymbol{\omega^i}$ due to $\tilde{\boldsymbol{\omega}}^{\boldsymbol{i}}$ if $\tilde{\boldsymbol{\omega}}^{\boldsymbol{i}} \in L_i$ has $(K-1)$-vector $\boldsymbol{u}$ componentwise $\geq$ that of $\boldsymbol{w^i}$

satisfy $q \geq -B$ and that there are no zero probabilities in the model. Furthermore, we assume that these q values have been rounded off to precision δ.

Under these assumptions, we can see that any two values $M_{i,s}$ and $M_{i,s'}$ satisfy $|M_{i,s} - M_{i,s'}| \leq B$. This follows from the definitions:
$M_{i,s} = \max_{s'}(M_{i-1,s'} + q^{\omega_{i-1}}_{s',s})$ and
$M_{i,s'} = \max_{s}(M_{i-1,s} + q^{\omega_{i-1}}_{s,s'})$.
Let the maximum in the expression for $M_{i,s}$ be attained with s_0. Then

$$M_{i,s'} \geq M_{i-1,s_0} + q^{\omega_{i-1}}_{s_0,s'}$$

$$= M_{i-1,s_0} + q^{\omega_{i-1}}_{s_0,s} + (q^{\omega_{i-1}}_{s_0,s'} - q^{\omega_{i-1}}_{s_0,s})$$

$$= M_{i,s} + (q^{\omega_{i-1}}_{s_0,s'} - q^{\omega_{i-1}}_{s_0,s}),$$

so that, upon rearranging,

$$M_{i,s} - M_{i,s'} \leq q^{\omega_{i-1}}_{s_0,s} - q^{\omega_{i-1}}_{s_0,s'} \leq 0 - (-B) = B,$$

and by symmetry, we also get $M_{i,s'} - M_{i,s} \leq B$, so finally, $|M_{i,s} - M_{i,s'}| \leq B$.

In particular, only $2B/\delta$ distinct values are possible for each of the $(K-1)$ possible $M_{i,\pi_i} - M_{i,s}$ values. In the branch-and-bound algorithm, it is only impossible to remove either $\boldsymbol{\omega}$ or $\tilde{\boldsymbol{\omega}}$ (both of length i) due to the other if they are incomparable: the values one gives for $M_{i,\pi_i} - M_{i,s}$ are larger for some s and smaller for some other s. But there are only $(2B/\delta)^{K-2}$ incomparable values: for two sequences that share the first $(K-2)$ $M_{i,\pi_i} - M_{i,s}$ values, any values for the last $M_{i,\pi_i} - M_{i,s}$ will make them comparable.

Therefore, in the branch-and-bound algorithm there are at most $(2B/\delta)^{K-2}$ sequence possibilities that must be retained at any stage, and so with a careful implementation the running time of the algorithm is $O((2B/\delta)^{K-2} n K^2 |\Sigma|)$. This bound is exponential in the number of states, but linear in the length of the structure to be designed. (This bound is independent of the base used to get the q values (log-probabilities), because changing the base introduces a factor into both B and δ that cancels.)

For SCFGs in CNF, a similar idea allows one to obtain an exact algorithm that runs in polynomial time if there are only 2 non-terminal symbols. However, the idea used above for candidate string elimination does not immediately generalize to SCFGs because of their non-linear nature; an HMM outputs one symbol per state, but a non-terminal in an SCFG can generally end up producing any substring of the output string.

3.3. Casting the Problem as a Mixed Integer Linear Program

We can also start with the inequalities that must be satisfied for $\boldsymbol{\omega}$ and cast the inverse problem as the problem of finding a feasible solution to a Mixed Integer Linear Program. We provide this simple formulation because it allows both practical and theoretical tools developed for integer programming to be applied directly to our problem.

The formulation as a Mixed Integer Linear Program is done by defining 0-1 variables $\epsilon_{i,j}$, where $\epsilon_{i,j} = 1$ indicates that the jth emission symbol is chosen for ω_i. Enforcing that there is only one emission choice made at every position is equivalent to requiring $\sum_j \epsilon_{i,j} = 1$ for $i = 1$ to $n-1$. Each maximum in the constraints is replaced by $\geq$, while the traceback constraints are enforced by additional equalities.

Integer Linear Program For HMMs:
Objective: Feasible Solution

Variables:
$\epsilon_{i,j}$, 0-1 valued, for $1 \leq i \leq n-1$ and $1 \leq j \leq |\Sigma|$
$M_{i,s}$, for $1 \leq i \leq n$ and $1 \leq s \leq K$

Constraints:
$\sum_j \epsilon_{i,j} = 1$ for all $1 \leq i \leq n-1$
$\epsilon_{i,j} = 0$ if $j \notin \Sigma_i$ for all $1 \leq i \leq n-1$
$M_{1,S} = 0$ and $M_{1,s} = -\infty$ for all $s \neq S$
$M_{i,s} \geq \sum_j \epsilon_{i-1,j}(M_{i-1,s'} + q^j_{s',s})$ for all s, s' and all $i \geq 2$
$M_{i,\pi_i} = \sum_j \epsilon_{i-1,j}(M_{i-1,\pi_{i-1}} + q^j_{\pi_{i-1},\pi_i})$ for all $i \geq 2$
$M_{n,s} \leq M_{n,\pi_n}$ for all $s \neq \pi_n$

4. Simulations

We implemented our branch-and-bound algorithm and examined its running time on synthetic data in order to demonstrate that in practice the algorithm frequently runs fast when an exhaustive search would be infeasible. In order to do this, we randomly generated HMMs by drawing each-transition-emission pair probability from the uniform distribution and then normalizing the values, rounding off to precision $\delta = 0.01$. We then separately generated both arbitrary state-paths and designable state-paths at random from this HMM (the latter by randomly sampling emission sequences and running the Viterbi algorithm on these sequences) and timed our branch-and-bound algorithm on these instances. We found that our algorithm ran significantly faster on arbitrary paths, the majority of which are not designable, than on arbitrary designable paths (taking milliseconds rather than seconds per run).

Figure 3 shows running times of simulations on random designable state-paths for different numbers of states K and path lengths n, with fixed emission alphabet of size $|\Sigma| = 20$. For each pair of K and n values, 10 HMMs were generated at random and for each of these HMMs, 10 designable paths were generated at random, as described above. The branch-and-bound algorithm was then run and the average time to design a sequence over these 100 runs was recorded. On these problem instances, the running time of the algorithm scales roughly linearly with path length n. Interestingly, while the running times initially increased with increasing K values, the running times were lower for $K = 50$ and $K = 100$ than for $K = 20$, an observation that was repeated for multiple experiments. The longest run of the algorithm took 80 seconds. A solution by exhaustive search would require examining $|\Sigma|^n$ possible sequences, which for that run would have

been 20^{400} sequences. All code was implemented in Matlab and run on a 3.06 GHz Intel Xeon PC.

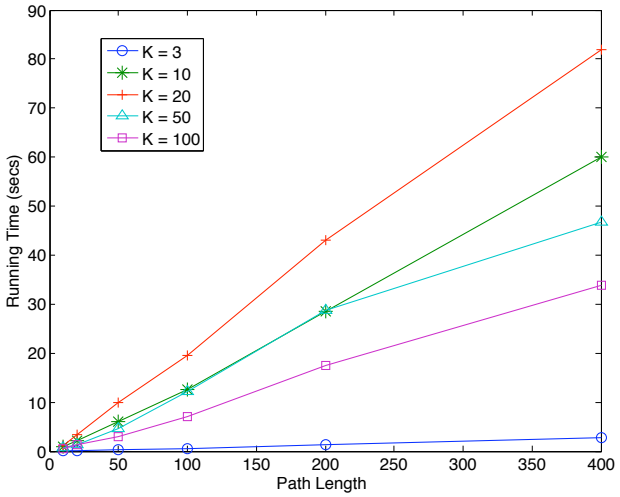


Figure 3. Running times of the branch-and-bound algorithm on designable paths. Simulations shown for number of states $K = 3$, 10, 20, 50, 100, path lengths $n = 10$, 20, 50, 100, 200, 400 and emission alphabet size $|\Sigma| = 20$.

5. Conclusions

We have introduced a novel problem on HMMs and SCFGs, the Inverse-Viterbi problem, inspired by protein and RNA structure design, and have given a number of theoretical results for the problem. In particular, our hardness result demonstrates that a polynomial time algorithm for RNA secondary structure design that only exploits the general form of the Zuker energy or similar SCFG models (and not the particulars of a specific model) is not possible unless $P = NP$.

There are a number of possible extensions to this work. Developing more efficient algorithms on both HMMs and SCFGs may be possible and in particular, extending our branch-and-bound algorithm to SCFGs would be useful. It is also possible to explore extensions of the problem to more general probabilistic models such as Markov Random Fields. The framework given here may also be useful for developing new algorithms for design in specific applications. Areas where the negative-design aspect plays a large role, such as RNA secondary structure design, are the most likely candidates to benefit from such an approach. Given the widespread use of grammars, the inverse problem we have defined here may find applications to other fields.

6. Acknowledgements

The authors thank Andreas Schulz, Michael Baym, Jérôme Waldispühl, Mathieu Blanchette, and particularly Michael Collins for advice and helpful discussions. M. Schnall-Levin is supported by NDSEG and Hertz Foundation fellowships and L. Chindelevitch is supported in part by an NSERC PGS-D Scholarship.

References

Andronescu, M. e. a. (2004). A new algorithm for RNA secondary structure design. *Journal of Molecular Biology*, *336*, 607–624.

Breaker, R. R. (1996). Are engineered proteins getting competition from RNA? *Current Opinion in Biotechnology*, *7*, 442–448.

Busch, A., & Backofen, R. (2006). INFO-RNA- a fast approach to inverse RNA folding. *Bioinformatics*, *22*, 1823–1831.

Butterfoss, G., & Kuhlman, B. (2005). Computer-based design of novel protein structures. *Annual Review of Biophysics and Biomolecular Structure*, *35*, 49–65.

Dowell, R., & Eddy, S. (2004). Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics*, *5*.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1999). *Biological sequence analysis: Probablistic models of proteins and nucleic acids*. Cambridge University Press.

Elizalde, S., & Woods, K. (2006). Bounds on the number of inference functions of a graphical model. *ArXiv Mathematics e-prints*, *math/0610233*.

Hofacker, I. L. e. a. (1994). Fast folding and comparison of RNA secondary structures. *Monatshefte f. Chemie*, *125*, 167–188.

Park, S., Yang, X., & Saven, J. G. (2004). Advances in computational protein design. *Current Opinion in Structural Biology*, *14*, 487–494.

Pokala, N., & M., H. T. (2001). Review: Protein design- where we were, where we are, where we're going. *Journal of Structural Biology*, *134*, 269–281.

Viterbi, A. J. (1967). Error bounds for convolution codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *13*, 260–269.

Zuker, M., & Stiegler, P. (1981). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, *9*, 133–148.

Compressed Sensing and Bayesian Experimental Design

Matthias W. Seeger SEEGER@TUEBINGEN.MPG.DE
Hannes Nickisch HN@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, Spemannstr. 38, Tübingen, Germany

Abstract

We relate compressed sensing (CS) with Bayesian experimental design and provide a novel efficient approximate method for the latter, based on expectation propagation. In a large comparative study about linearly measuring natural images, we show that the simple standard heuristic of measuring wavelet coefficients top-down systematically outperforms CS methods using random measurements; the sequential projection optimisation approach of (Ji & Carin, 2007) performs even worse. We also show that our own approximate Bayesian method is able to learn measurement filters on full images efficiently which outperform the wavelet heuristic. To our knowledge, ours is the first successful attempt at "learning compressed sensing" for images of realistic size. In contrast to common CS methods, our framework is not restricted to sparse signals, but can readily be applied to other notions of signal complexity or noise models. We give concrete ideas how our method can be scaled up to large signal representations.

1. Introduction

There has been a lot of recent interest in the area of *compressed sensing* (CS) (Candès et al., 2006; Donoho, 2006), where it is argued that if signals can be expected to be compressible due to sparseness after some linear transform, then they can be reconstructed from a number of measurements significantly below the Nyquist/Shannon limit, if the measurement design is not too regular. In this paper, we relate CS to the more general notion of statistical (Bayesian) experimental design.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Through this view, characteristics of signals and algorithms, defined in an abstract mathematical way in the CS literature so far, become understandable and workable. The experimental design approach applies to signals of low complexity in general, not only to sparse ones. It has the potential to clearly outperform the randomised designs, favoured by theoretical CS arguments, in cases where signals are not well-described by common CS assumptions. For example, CS has been viewed with some scepticism so far by researchers in computer vision and image statistics (Weiss et al., 2007). While images exhibit transform sparsity to some degree, purely random measurement designs can be suboptimal for them. The reason is that there is more to low-level image statistics than sparsity. Much of this knowledge can be modeled tractably (Simoncelli, 1999) and could therefore be incorporated into a Bayesian experimental design architecture. To our knowledge, the current CS reconstruction schemes are purely estimation-based and lack proper representations of uncertainty (which is what fundamentally drives experimental design), and the theory deals exclusively with signals which are *unstructured except for random sparsity*. We present experimental results sheding more light on the relationship between CS and images. Similar to (Weiss et al., 2007), we find that standard approaches to linear image measurement (wavelet coefficients) give significantly better reconstruction results than using random measurements favoured by CS, even if modern CS reconstruction algorithms are applied. Yet, our experimental evidence is more substantial than theirs. Beyond that, we show that our efficient approximation to sequential Bayesian design can be used to learn measurements which indeed outperform measuring wavelet coefficients top-down. Our method provides a practically efficient solution to the problem posed in (Weiss et al., 2007), namely how to learn measurement filters automatically from data (using very little concrete knowledge about the signal class) which perform close to or even better than "standard" ones obtained through decades of research and experience. In contrast, the uncertain

components analysis algorithm suggested by them requires a large database of image patches to be run, and could hardly be scaled up to the realistic dimensions treated here[1].

An approximate Bayesian approach to compressed sensing has been presented in (Ji & Carin, 2007), making use of sparse Bayesian learning (SBL) (Tipping, 2001). Our method is based on a different, more general inference approximation, expectation propagation (Minka, 2001), and outperforms theirs very significantly, for prediction based on the same design and, even more so, for sequential design optimisation, as we show in comparative experiments below. Moreover, strongly underdetermined problems (many more variables than observations) are dealt with more efficiently in our framework. In addition, our framework is generalised easily to non-Gaussian observation likelihoods, skew prior terms, and generalised linear models (Gerwinn et al., 2008), and our methodology, our comparisons, as well as our discussion here have a broader scope. Our method is an extension of the scheme in (Seeger et al., 2007). However, the applications to images considered here are orders of magnitude larger than theirs, and several novel ideas are proposed here in order to increase computational efficiency substantially. While much work has been done in statistics on experimental design for the classical Gaussian-linear model, Gaussian priors are entirely inappropriate for images[2], and designs optimized for them are suboptimal (see also (Seeger et al., 2007)). We are not aware of existing methods for the model used here, which scale comparable to ours, with the exception of (Ji & Carin, 2007).

A different approach for optimising measurement design is given in (Elad, 2007), where $\boldsymbol{X}$ is designed *a priori* with the aim of making its rows maximally decoherent. In our setup, $\boldsymbol{X}$ is designed sequentially, using Bayesian information criteria.

The structure of the paper is as follows: The experimental design view on CS is detailed in Section 2. Our framework for approximate inference is described in Section 3, where we also show how to apply it to large problems, especially in sequential experimental design. Our approach is validated through a series of experiments, comparing it to (Ji & Carin, 2007) and common CS methods on artificial data (Section 4.1), and analysing the suitability of CS and Bayesian experimental design on natural images (Section 4.2).

[1]Their experiments are on 4×4 image patches, while ours run efficiently on 64×64 images.

[2]Reconstruction under the Gaussian-linear model is simply the method of least squares, often referred to as "linear reconstruction". Much of the improved performance through CS is due to the use of non-linear sparse reconstruction techniques.

2. Compressed Sensing and Experimental Design

Compressed sensing (CS) (Candès et al., 2006; Donoho, 2006) can be motivated as follows. Suppose a signal, such as an image or a sound waveform, is measured and then transferred over some channel or stored. Traditionally, the measurement obeys the Nyquist/Shannon theorem, allowing for an exact reconstruction of the (band-limited) signal if there is no measurement noise. However, what follows is usually some form of lossy compression, exploiting redundancies and non-perceptibility of losses. Given that, can the information needed for a satisfactory reconstruction not be measured below the Nyquist frequency (this is called undersampling)? In many key applications today, the measurement itself is the main bottleneck for cost reductions or higher temporal/spatial resolution. Recent theoretical results indicate that undersampling should work well if randomized designs are used, and if the signal reconstruction method specifically takes the "compressibility" into account.

Bayesian experimental design encompasses the CS problem. Here, the "compressibility" of signals is encoded in a *prior distribution*, under which signals of low complexity in general, or high (transform) sparsity in particular, have most mass. While an undersampling violates the Nyquist theorem, signals can often still be reconstructed if they are sufficiently likely under the prior. But not every way of undersampling will do. Experimental design is concerned with optimising the measurement structure (called design), so as to obtain the desired information at the lowest possible cost. This is easily explained by considering the model of interest here. Let $\boldsymbol{u} \in \mathbb{R}^n$ be latent variables (pixels of an image), and let $\boldsymbol{y} \in \mathbb{R}^m$ be noisy measurements thereof. The model class of interest is

$$P(\boldsymbol{u}|\boldsymbol{y}) \propto N(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{u}, \sigma^2\boldsymbol{I}) \prod_{i=1}^{q} t_i(s_i), \quad \boldsymbol{s} = \boldsymbol{B}\boldsymbol{u}. \quad (1)$$

The likelihood $P(\boldsymbol{y}|\boldsymbol{u})$ is Gaussian and underdetermined ($n > m$). The prior[3] is a product of univariate non-Gaussian potentials $t_i(s_i)$. It is computationally advantageous, yet not essential, that the $\log t_i$ be concave (Seeger, 2008), and in this paper we use Laplacian

[3]We do not require that the prior potential is actually a normalisable distribution over $\boldsymbol{u}$, the models of interest here are of the undirected Markov random field (or "energy-based") type.

potentials

$$t_i(s_i) = \frac{\tau}{2} e^{-\tau|s_i|}, \qquad (2)$$

which are of this sort. If number of image pixels n is large, it is important for computational efficiency that matrix-vector multiplications (MVMs) with $\boldsymbol{B}$ and $\boldsymbol{B}^T$ (less important: with $\boldsymbol{X}$, $\boldsymbol{X}^T$) can be done efficiently, and that $\boldsymbol{B}$ does not have to be stored explicitly.

The unknown signal $\boldsymbol{u}$ (an image for now) should be "compressible", *i.e.* it should exhibit *transform sparsity*[4]: after some fixed linear mapping $\boldsymbol{B}$, such as a wavelet transform, $\boldsymbol{s} = \boldsymbol{B}\boldsymbol{u}$ has many coefficients s_i close to zero. An image coder would set these to exactly zero, thereby compressing the image. "Expected transform sparsity" is encoded in a sparsity prior, in our case the product of Laplacians (2). As opposed to a Gaussian, a Laplace distribution concentrates more mass close to zero, forcing coefficients to be very small. On the other hand, the Laplacian also has more mass in the tails, which allows for occasional large values. These points are explained further in (Seeger, 2008; Tipping, 2001).

Next, the design is $\boldsymbol{X}$, the measurement matrix. In our example, each row of $\boldsymbol{X}$ is a linear filter specifying a single image measurement. In this paper, we assume that all rows of $\boldsymbol{X}$ have unit norm[5]. The problem of experimental design is how to choose $\boldsymbol{X}$ among many candidates of the same cost, so that subsequent measurements allow for the best reconstruction of $\boldsymbol{u}$. This decision has to be taken *without* doing real measurements for most candidates. In a Bayesian variant, the posterior distribution $P(\boldsymbol{u}|\boldsymbol{y})$ encodes all present knowledge. To score a candidate $\boldsymbol{X}_*$ (new rows of $\boldsymbol{X}$), assume for the moment that the outcome $\boldsymbol{y}_*$ is known. We can measure the decrease in uncertainty from $P(\boldsymbol{u}|\boldsymbol{y})$ to $P(\boldsymbol{u}|\boldsymbol{y}, \boldsymbol{y}_*)$ by the *entropy difference* $\mathrm{H}[P(\boldsymbol{u}|\boldsymbol{y})] - \mathrm{H}[P(\boldsymbol{u}|\boldsymbol{y}, \boldsymbol{y}_*)]$. Not knowing $\boldsymbol{y}_*$, we integrate it out using $P(\boldsymbol{y}_*|\boldsymbol{y}) = \int P(\boldsymbol{y}_*|\boldsymbol{u}) P(\boldsymbol{u}|\boldsymbol{y})\, d\boldsymbol{u}$. This expected information score drives the optimisation of the design. It is clear that such scores are fundamentally based on the posterior as representation of uncertainty, so that algorithms which merely estimate good solutions from given data cannot be used directly in order to compute them[6]. With such methods, either rough rules of thumb have to be followed to obtain a design ("make it random" in CS), or many measurements have to be taken in a trial-and-error fashion. In Bayesian experimental design, a permanently refined uncertainty representation is used to avoid uninformative data sampling, so often many fewer real measurements are required.

3. Approximate Inference

Bayesian inference is in general not analytically tractable for models of the form (1), and has to be approximated. Moreover, the applications of interest here demand a high efficiency in many dimensions ($n = 4096$ in the natural image experiments here). Importantly, Bayesian experimental design does not only require inference just once, but many times in a sequential fashion. We make use of the *expectation propagation* (EP) method (Minka, 2001), together with a robust and efficient representation for $Q(\boldsymbol{u}) \approx P(\boldsymbol{u}|\boldsymbol{y})$. Our framework has previously been used in a different context (Seeger, 2008), where details can be found which are omitted here. As a novelty, we will show here how the framework can be run efficiently for large n, and how sequential design optimisation can be done orders of magnitude faster.

In EP, the posterior $P(\boldsymbol{u}|\boldsymbol{y})$ is approximated by a Gaussian $Q(\boldsymbol{u})$ with free (variational) parameters $\boldsymbol{b}$, $\boldsymbol{\pi}$, which are formally introduced by replacing $t_i(s_i)$ by $\tilde{t}_i(s_i) = e^{b_i s_i - \pi_i s_i^2/2}$ in (1). The distribution $Q(\boldsymbol{u})$ is represented by lower triangular $\boldsymbol{L}$ and $\boldsymbol{\gamma}$,

$$\begin{aligned} \boldsymbol{L}\boldsymbol{L}^T &= \sigma^{-2}\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{B}^T\boldsymbol{\Pi}\boldsymbol{B} = \mathrm{Cov}_Q[\boldsymbol{u}]^{-1}, \\ \boldsymbol{\gamma} &= \boldsymbol{L}^{-1}(\sigma^{-2}\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{B}^T\boldsymbol{b}),\ \boldsymbol{\Pi} = \mathrm{diag}\,\boldsymbol{\pi}, \end{aligned}$$

so that $\mathrm{E}_Q[\boldsymbol{u}] = \boldsymbol{L}^{-T}\boldsymbol{\gamma}$. The (b_i, π_i) are then updated sequentially by matching the Gaussian moments of the *tilted distributions*

$$\hat{P}_i(\boldsymbol{u}) \propto N(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{u}, \sigma^2\boldsymbol{I}) \prod_{j \neq i} \tilde{t}_j(s_j) \tilde{t}_i(s_i)^{1-\eta} t_i(s_i)^{\eta}$$

with the new $Q'(\boldsymbol{u})$. Here, $\eta \in (0, 1]$ is a fractional parameter[7]. In each local update, we need to compute the non-Gaussian moments of the marginal $\hat{P}_i(s_i)$, and to update the $Q(\boldsymbol{u})$ representation, which is done by an $O(n^2)$ Cholesky update of $\boldsymbol{L}$. Note that (Ji & Carin, 2007) employ the variational mean field approximation of (Tipping, 2001), which is specific to sparse linear models (more precisely, all t_i have to be Gaussian

[4] In our experiments, we use an extended notion of sparsity, see Section 3.2.

[5] When designing $\boldsymbol{X}$, it is important to keep its rows of the same scale. Otherwise, a measurement can always be improved (at fixed noise level σ^2) simply by increasing its norm. Put differently, we place a prior on $\boldsymbol{X}$ which is uniform over all matrices with rows of unit norm.

[6] It is one thing to learn to predict well, yet a different issue to estimate its own uncertainty well, and methods employing "premature sparsification" often perform badly w.r.t. the latter (see Section 4.1).

[7] $\eta = 1$ gives standard EP, but choosing $\eta < 1$ can increase the robustness of the algorithm on the sparse linear model significantly (Seeger, 2008). We use $\eta = 0.9$ in all our experiments.

scale mixtures, thus even functions), while EP can be applied with little modification to models with skew priors or non-Gaussian skew likelihoods as well (Gerwinn et al., 2008).

In our applications of sequential design, we need to score the informativeness of new candidates $\boldsymbol{x}_*$ (as row of $\boldsymbol{X}$), which we do by the entropy difference (see Section 2). If Q' is the approximate posterior after including $\boldsymbol{x}_*$, then $2\mathrm{H}[Q'] = \log|\mathrm{Cov}_{Q'}[\boldsymbol{u}]| + C$, where Q' differs from Q in that $(\boldsymbol{X}')^T\boldsymbol{X}' = \boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{x}_*\boldsymbol{x}_*^T$, and $\boldsymbol{\pi} \to \boldsymbol{\pi}'$. We approximate the entropy difference by assuming that $\boldsymbol{\pi}' = \boldsymbol{\pi}$, whence

$$\mathrm{H}[Q] - \mathrm{H}[Q'] = \frac{1}{2}\log\left(1 + \sigma^{-2}\boldsymbol{x}_*^T\mathrm{Cov}_Q[\boldsymbol{u}]\boldsymbol{x}_*\right).$$

Since $\|\boldsymbol{x}_*\| = 1$ by assumption, this score is maximized by choosing $\boldsymbol{x}_*$ along the principal (leading) eigendirection[8] of $\mathrm{Cov}_Q[\boldsymbol{u}]$. The same score is used by (Ji & Carin, 2007).

3.1. Large-Scale Applications

There are two major issues with trying to apply our method for large sizes n. First, the EP site updates are done in random sweeps over n sites, because it is not clear which particular site ordering leads to fastest convergence. This problem is severe in our sequential design application to natural images, since there are many small changes to $\boldsymbol{X}$, $\boldsymbol{y}$ (individual new measurements), after each of which EP convergence has to be regained. We approach it by forward scoring many site candidates before each EP update, thereby always updating the one which gives the largest posterior change. This is detailed just below. Second, the robust Q representation of (Seeger, 2008) is of size $O(n^2)$, and each update costs $O(n^2)$. We sketch a different representation of size $O(m^2)$ below, which can be used to drive our framework as well. In contrast, (Ji & Carin, 2007) use a heuristic of setting many of the π_i to ∞ early in the iteration, which leads to much worse results than we obtain (see Section 4.1, Section 4.2).

Our selective updating scheme for EP hinges on the fact that we can maintain all site marginals $\boldsymbol{h}$, $\boldsymbol{\rho}$, $Q(s_i) = N(h_i, \rho_i)$, up to date at all times. For a site i, we can quantify the change of Q through an update there by $\mathrm{D}[Q'(s_i)\,\|\,Q(s_i)]$ (Q' the posterior after the update at i), which can be computed in $O(1)$. Importantly, $\mathrm{D}[Q'(\boldsymbol{u})\,\|\,Q(\boldsymbol{u})] = \mathrm{D}[Q'(s_i)\,\|\,Q(s_i)]$ (because $Q(\boldsymbol{u}|s_i) = Q'(\boldsymbol{u}|s_i)$), so the score precisely measures the global amount of change $Q \to Q'$. We maintain a list of candidate sites, which are scored before each EP update, and the update is done for the winner only. The list is then evolved by replacing the lower half of worst-scoring sites by others randomly drawn from $\{1, \ldots, q\}$. Importantly, the marginals $\boldsymbol{h}$, $\boldsymbol{\rho}$ can be updated along with the representation, at the expense of only *one* additional $\boldsymbol{L}$ backsubstitution and MVM with $\boldsymbol{B}$. Namely, if $\pi_i' = \pi_i + \Delta\pi_i$, $b_i' = b_i + \Delta b_i$, and $\boldsymbol{w} := \boldsymbol{B}\boldsymbol{L}^{-T}(\boldsymbol{L}^{-1}\boldsymbol{B}_{i,\cdot}^T)$, then

$$\boldsymbol{\rho}' = \boldsymbol{\rho} - \frac{\Delta\pi_i}{1 + \rho_i\Delta\pi_i}\boldsymbol{w} \circ \boldsymbol{w}, \quad \boldsymbol{h}' = \boldsymbol{h} + \frac{\Delta b_i - h_i\Delta\pi_i}{1 + \rho_i\Delta\pi_i}\boldsymbol{w}.$$

Here, $\boldsymbol{L}^{-1}\boldsymbol{B}_{i,\cdot}^T$ has to be computed for the $\boldsymbol{L}$ update anyway. This idea is used in the experiments described in Section 4.2.

For large n, storing an $n \times n$ matrix in memory becomes prohibitive. In a less costly representation, we exploit $m \ll n$. We require[9] that $\boldsymbol{B} = \boldsymbol{I}$. The Woodbury formula gives

$$\mathrm{Cov}_Q[\boldsymbol{u}] = \boldsymbol{\Pi}^{-1} - \boldsymbol{\Pi}^{-1}\boldsymbol{X}^T\boldsymbol{L}^{-T}\boldsymbol{L}^{-1}\boldsymbol{X}\boldsymbol{\Pi}^{-1},$$

where $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Pi}^{-1}\boldsymbol{X}^T$, so $\boldsymbol{L}$ (different from above) is of size m^2 only. An EP update requires $O(m^2)$ and two MVMs with $\boldsymbol{X}$, rather than $O(n^2)$ above. While this representation is exact, it is numerically less robust to update than the $O(n^2)$ one.

3.2. Image Model. Other Methods

In this section, we provide further details about the concrete model we use in our experiments with natural images. Our prior encourages two different notions of sparsity in an image. First, a multi-scale wavelet transform of $\boldsymbol{u}$ should be sparse, modeling the observation that natural images can be compressed well in a wavelet domain. Second, the finite differences in the horizontal and vertical direction should exhibit sparsity, accounting for spatial smoothness often found in images[10]. A frequently used penalty term for the latter is the L_1 norm of the image gradient, also known as *total variation*.

Our model is an instance of (1), where all t_i are Laplacian (2). $\boldsymbol{s}$, and therefore $\boldsymbol{B}$, decompose into two different parts: $\boldsymbol{B}^T = (\boldsymbol{B}^{(sp)T}\,\boldsymbol{B}^{(tv)T})$. Equivalently, the prior is the product of two potentials. The *transform sparsity* potential is a sparsity prior on the wavelet coefficients of $\boldsymbol{u}$. Note that the Laplace distribution is a sensible candidate to fit wavelet coefficient histograms from natural images (Simoncelli, 1999). Thus,

[8] We compute $\boldsymbol{x}_*$ by the Lanczos algorithm.

[9] More generally, $\boldsymbol{B}^T\boldsymbol{\Pi}\boldsymbol{B}$ must be easy to invert. If $\boldsymbol{B}$ is invertible and $\boldsymbol{B}^{-1}$-MVM feasible, we represent $Q(\boldsymbol{s})$ rather than $Q(\boldsymbol{u})$.

[10] Recall what we mean by sparsity from Section 2: *most* coefficients are forced to be small, by allowing *some* to be large. Occasional large components in the gradient correspond to edges in the image.

$\boldsymbol{B}^{(sp)} \in \mathbb{R}^{n\times n}$ is a multi-scale orthonormal wavelet transform, and the potential is $\exp(-\tau_{sp}\|\boldsymbol{B}^{(sp)}\boldsymbol{u}\|_1)$. The *total variation* potential is a Laplace prior on the image gradient, *i.e.* the differences between horizontal and vertical pixel neighbours[11]. $\boldsymbol{B}^{(tv)} \in \mathbb{R}^{2(n-\sqrt{n})\times n}$ is a sparse structured matrix, mapping the image $\boldsymbol{u}$ to its gradient. Here, we assume that $n = 2^{2k}$ for simplicity. The total variation potential is $\exp(-\tau_{tv}\|\boldsymbol{B}^{(tv)}\boldsymbol{u}\|_1)$.

Therefore, we have $q \approx 3n$ for the size of $\boldsymbol{s}$. Also, the potentials come with different scale parameters τ_{sp}, τ_{tv}. Importantly, neither of $\boldsymbol{B}^{(sp)}$, $\boldsymbol{B}^{(tv)}$ has to be stored in memory, and MVM with $\boldsymbol{B}$ or $\boldsymbol{B}^T$ can be done in $O(n)$.

We also briefly describe the methods we compare against. Most of them come with a transform sparsity potential only, so that $\boldsymbol{s} = \boldsymbol{B}^{(sp)}\boldsymbol{u}$. The method of (Ji & Carin, 2007) is called SBL here. In L_p reconstruction, $\hat{\boldsymbol{s}} = \mathrm{argmin}\{\|\boldsymbol{s}\|_p \,|\, \boldsymbol{X}\boldsymbol{B}^{(sp)T}\boldsymbol{s} = \boldsymbol{y}\}$, $\hat{\boldsymbol{u}} = \boldsymbol{B}^{(sp)T}\hat{\boldsymbol{s}}$. For L_2 we just solve the normal equations, while for L_1 this is a linear program. Note that the latter is used in many CS publications (Candès et al., 2006; Donoho, 2006). A method with transform sparsity *and* total variation potential, called $L_1 + TV$ here, is given by the following quadratic program: $\hat{\boldsymbol{u}} = \mathrm{argmin}\, \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{u}\|_2^2 + \tau_{sp}\sigma^2\|\boldsymbol{B}^{(sp)}\boldsymbol{u}\|_1 + \tau_{tv}\sigma^2\|\boldsymbol{B}^{(tv)}\boldsymbol{u}\|_1$ (Candès & Romberg, 2004). We used the following code in our experiments:

SBL: `www.ece.duke.edu/~shji/BCS.html`
L_1: `www.acm.caltech.edu/l1magic/`
$L_1 + TV$: `www.stanford.edu/~mlustig/`

4. Experiments

In this section, we provide experimental results for different instances of our framework, comparing to CS and approximate Bayesian methods on synthetic data (Section 4.1), and on the task of measuring natural images (Section 4.2).

4.1. Artificial Setups

It is customary in the CS literature to test methods on synthetic data, generated following the "truly sparse and otherwise unstructured" assumptions under which asymptotic CS theorems are proven. We do the same here, explicitly using the "(non-)uniform spikes" (Ji & Carin, 2007), but cover some other heavy-tailed distributions as well. It seems that not many signals of real-world interest are strictly and randomly sparse, so that studies looking at the robustness of CS theoretical claims are highly important. In this section, signals are sparse as such, so that $\boldsymbol{B} = \boldsymbol{I}$ and $\boldsymbol{u} = \boldsymbol{s}$ here. We compare methods described in Section 3.2. It is important to stress that all methods compared here (except for L_2) are based on exactly the same underlying model (1) with $\boldsymbol{B} = \boldsymbol{I}$, and differences arise only in the nature of computations (approximate Bayesian versus maximum a-posteriori optimisation) and in whether $\boldsymbol{X}$ is sequentially designed (EP, SBL) or chosen at random (L_p reconstruction; we follow CS theory (Candès et al., 2006; Donoho, 2006) and sample rows of $\boldsymbol{X}$ uniformly of unit norm). Results are shown in Figure 1.

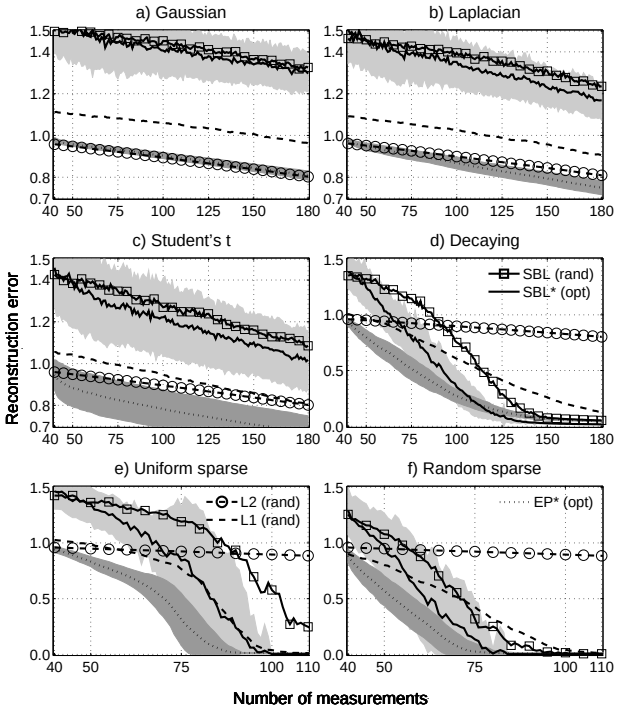


Figure 1. Comparison on 6 random synthetic signals $\boldsymbol{u} \in \mathbb{R}^{512}$. Shown are L_2-reconstruction errors (mean±std.dev. over 100 runs). All methods start with same random initial $\boldsymbol{X}$ ($m = 40$), then "(rand)" add random rows, "(opt)" optimise new rows sequentially. Noise variance $\sigma^2 = 0.005$, prior scale $\tau = 5$. SBL: (Ji & Carin, 2007), Lp: L_p reconstruction, EP: our method. (a-c): i.i.d. zero mean, unit variance Gaussian, Laplacian (Eq. 2), Student's t (3 d.o.f.). (d): $\frac{n}{2}$ of $u_i = 0$, $\frac{n}{4}$ exponential decay $1, \ldots, 0$, $\frac{n}{4}$ minus that, randomly permuted. (e-f): 20 $u_i \neq 0$ at random; (e) uniform spikes, $u_i \in \{\pm 1\}$; (f): non-uniform spikes, $u_i \sim \frac{1}{4} + |t|$, $t \sim N(0, 1)$; as in (Ji & Carin, 2007). Distributions in (d-f) normalised to unit variance.

The "sparsity" (or super-Gaussianity) of the signal distributions increases from (1a) to (1e-f). For Gaussian signals (1a), L_2 reconstruction based on random measurements is optimal. While all CS methods and SBL

[11] This potential on its own is not normalisable as distribution over $\boldsymbol{u}$, being invariant against adding a constant to all pixels.

(random and designed) lead to large errors, EP with design matches the L_2 results, thus shows robust behaviour. For Laplacian and Student's t signals (1b-c), designed EP outperforms L_2 reconstruction significantly, while even the CS L_1 method still does worse than simple least squares. SBL performs poorly in all three cases with signals not truly sparse, thus is not robust against rather modest violations of the strict CS assumptions. Its non-robustness is also witnessed by large variations across trials.

On the other hand, L_2 performs badly on truly sparse signals. In all cases (1d-f), EP with design significantly outperforms all other methods, including designed SBL, with special benefits at rather small numbers of measurements. SBL does better now with truly sparse signals, and is able to outperform L_1.

From the superior performance of EP with design on all signal classes, we conclude that experimental design can sequentially find measurements that are significantly better than random ones, even if signals are truly sparse. Moreover, the superior performance is robust against large deviations away from the underlying model, more so even than classical L_1 or L_2 estimation. The poor performance of SBL (Ji & Carin, 2007) seems to come from their desire for "premature sparsification". During their iterations, many π_i are clamped to $+\infty$ early for efficiency reasons. This does not hurt mean predictions from current observations much, but affects their covariance approximation drastically: most directions not supported by the data right now are somewhat ruled out for further measurements, since posterior variance along them (which should be large!) is shrunk in their method. In contrast, in our EP method, none of the π_i become very large with modest m, and our covariance approximation seems good enough to successfully drive experimental design. Without premature sparsification, our scheme is still efficient, since the most relevant site updates are found actively, and the need to eliminate variables does not arise.

4.2. Natural Images

In this section, we are concerned about finding linear filters which allow for good reconstruction of natural images from noisy measurements thereof. Since natural images exhibit sparsity in wavelet or Fourier domains, CS theory seems to suggest that random measurements should be well-suited for this purpose, and there have been considerable efforts to develop hardware which can perform such random measurements cost-efficiently (Duarte et al., 2008). On the other hand, much is known about low level natural image statistics, and powerful linear measurement transforms have emerged there, such as multi-scale wavelet transforms, based on which natural image reconstruction should be substantial better than for random measurements (Weiss et al., 2007).

The sparsity of images in a wavelet domain is highly structured, there is a clear ordering among the coefficients from coarse to fine scales: natural images typically have much more energy in the coarse-scale coefficients, and coefficients with very small values are predominantly found in the fine scales. In our experiments, we employ a simple heuristic for linearly measuring images, called *wavelet heuristic* in the sequel: every measurement computes a single wavelet coefficient, and the sequential ordering of the measurements is deterministic top-down, from coarse to fine scales[12]. This ordering is a pragmatic strategy: if mainly the coarse-scale coefficients are far from zero, they should be measured first[13]. Do state-of-the-art CS reconstruction algorithms, based on random linear image measurements, perform better than simple L_2 reconstruction based on the wavelet heuristic? And how does Bayesian sequential design perform on this task, if the model described in Section 3.2 is used? Note that no prior knowledge about typical ordering or dependence among wavelet coefficients in encoded in this model either. Results of our study are given in Figure 2.

In fact, we started our exploration with what is shown in (2a), where 100 initial filters are drawn at random (except for L_2(heur)). Intrigued by the fact that the wavelet heuristic method L_2(heur) outperformed all CS variants significantly, we tried to give them a head-start, supplying $m = 100, 200, 400$ wavelet heuristic measurements initially (2b-d). However, the systematic under-performance of methods which have sparsity regularizers built in, yet do *random* rather than wavelet measurements, remains consistently present. From these results we conclude, much as (Weiss et al., 2007) argued on theoretical grounds, that if natural images are to be measured successively by unit norm, but otherwise unconstrained linear filters, then *drawing these filters at random leads to significantly worse*

[12]This ordering follows the recursive definition of such transforms: downsampling by factor two (coarse), horizontal differences, vertical differences, diagonal corrections at each stage. Our ordering is coarse → horizontal → vertical → diagonal, descending just as the transform does.

[13]Note that another problem with common CS assumptions applied to images is that the typical scale of coefficients along a coarse-to-fine ordering follows a smooth power law, it does not exhibit the abrupt drop from "significantly above noise level" to "exactly zero" often required by CS theory.

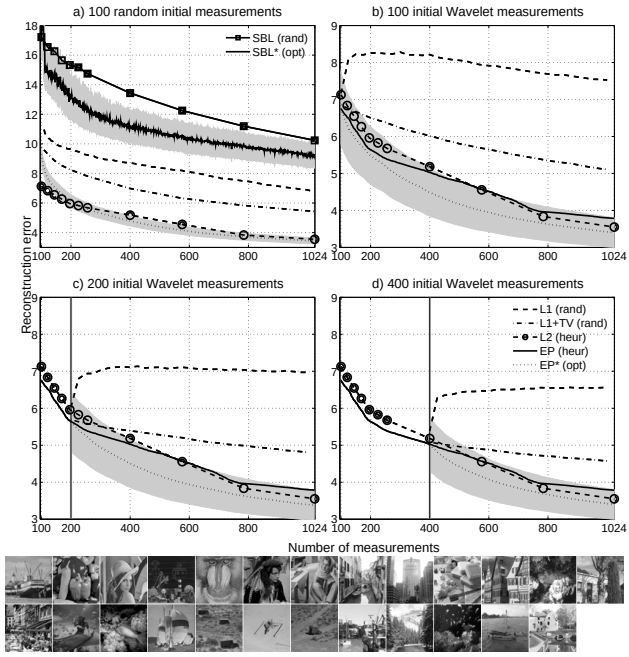


Figure 2. Experiments for measuring natural images ($64 \times 64 = 4096$ pixels). Shown are L_2-reconstruction errors averaged over 25 grayscale images typically used in computer vision research (from `decsai.ugr.es/cvg/dbimagenes/`) ($\pm\frac{1}{4}$std.dev. for "$*$"). Noise level $\sigma^2 = 0.005$. SBL: (Ji & Carin, 2007), Lp: L_p reconstruction, $L_1 + \mathrm{TV}$: Lasso with TV/wavelet penalties, EP: our method. True σ^2 supplied, τ parameters chosen optimally for each method individually: $\tau_{sp} = \tau_{tv} = 0.075$ ($L_1 + \mathrm{TV}$), $\tau_{sp} = 0.075$, $\tau_{tv} = 0.5$ (EP). New rows of $\boldsymbol{X}$ random unit norm (rand), actively designed (opt), acc. to wavelet heuristic (heur).
(a): Start with $m = 100$, $\boldsymbol{X}$ random unit norm. (b-d): Start with $m = 100, 200, 400$, $\boldsymbol{X}$ acc. to wavelet heuristic.

reconstructions than using standard wavelet coefficient filters top-down. While CS theorems are mathematically intriguing, and while there certainly are important applications that benefit from these results[14], linear image measurement is probably not among them.

On the other hand, the wavelet heuristic method is significantly outperformed by our EP method, where $\boldsymbol{X}$ is designed sequentially. In (2a), EP quickly recovers from the suboptimal initial random $\boldsymbol{X}$. Moreover, even when started from the same point as the wavelet heuristic (2b-d), the designed measurements lead to improvements over the heuristic immediately. EP(heur) is doing EP reconstruction, but based on the same measurements as L_2(heur). While it slightly outperforms L_2 reconstruction, the significant difference is due to the choice of the measurements. Our method therefore provides an efficient solution to the problem posed in (Weiss et al., 2007), namely how to learn measurements automatically from data, starting from little concrete domain knowledge. On the particular problem of measuring images linearly, our findings should be put into perspective, by noting that the L_2 wavelet heuristic is vastly faster to compute[15]. Moreover, $\boldsymbol{X}$ is optimised sequentially, particular to the image $\boldsymbol{u}$ (but without knowing the underlying $\boldsymbol{u}$), while the wavelet heuristic filters are always the same. Finally, the final $\boldsymbol{X}$ is is dense and unstructured. However, our method can be used in the same way to address applications where strong structural constraints on allowable $\boldsymbol{X}$ are present, and where wavelet (or purely random) measurements are not an option.

In this setting, SBL (Ji & Carin, 2007) performs much worse than all other methods tried, whether using random, wavelet or designed measurements. Results for SBL in cases (b-d) were even worse and are not included to facilitate comparison among the others. This is most probably an extreme instance of the problem noted in Section 4.1. Premature sparsification, in light of not strictly sparse signals, leads to poor results even with random $\boldsymbol{X}$. Their covariance estimates seem too bad to steer sequential design in a useful direction[16].

Finally, the deterioration of L_1, when adding random to initial wavelet measurements, is somewhat puzzling, especially since it does not happen for $L_1 + \mathrm{TV}$. These additional measurements provide novel information about the true $\boldsymbol{u}$, so a valid inference method should rather improve.

5. Discussion

We have shown how to address the compressive sensing problem with Bayesian experimental design, where designs are optimised to rapidly decrease uncertainty and do not have to be chosen at random. In a large study

[14]The theoretical CS setting is more extreme than what is really required here, in that there is no prior knowledge about where the non-zeros will lie. We speculate that more suitable applications could lie in steganography, spam or intrusion detection, where a signal has to be detected which has been hidden by an adversary.

[15]EP sequential design is still very efficient. A typical run on one image took 53 min (on 64bit 2.33GHz AMD), for $n = 4096$ and $q = 12160$ sites: 16785 initial EP updates, then 308 increments of $\boldsymbol{X}$ by 3 rows each, with on average only 8.8 site updates needed to regain EP convergence (up to 85 updates after *some* increments).

[16]In cases (b-d), top wavelet coefficients are measured initially, so their method confidently starts with a highly over-sparse solution and fails. Note that, as opposed to EP, we restarted the SBL code after each new measurement, so that poor current solutions are not inherited when new data is obtained.

about linearly measuring natural images, we show that CS reconstruction methods based on randomly drawn filters are outperformed significantly by standard least squares reconstruction measuring coarse-scale wavelet coefficients. Our findings suggest that the applicability of CS results (with their insistence on strict and unstructured signal sparsity) to natural image applications should be reconsidered. We also show that our Bayesian sequential design method, starting from a model with little domain knowledge built in, is able to find filters with significantly better reconstruction properties than top-down wavelet coefficients. Our findings indicate that efficient Bayesian experimental design techniques are highly promising for CS applications of different kinds just as well.

Why do random measurement filters enjoy good properties in CS theory, but are not useful in the case of natural images? We think that this seeming contradiction really comes from an erroneous "extrapolation" of what CS theorems really mean. Any structure apart from a randomly distributed sparsity pattern is ignored there. Also, they are *minimax* results, in that the reconstruction error for the worst sparsity pattern is bounded. But undersampled image reconstruction is not a worst-case problem, and much is known about the sparsity structure of natural images. It may be that L_1 or $L_1 + TV$ are minimax methods (for known $\boldsymbol{B}$), but that does not imply much about their *typical* performance. We suspect that our doubts about CS with random measurements extend beyond natural images to other signals of common interest in normal non-adversarial situations, since interest in a signal class implies that statistical knowledge about them beyond random sparsity has been obtained.

Our experience with the method of (Ji & Carin, 2007), which we compare against in our study, raises another more speculative, yet interesting point. Several methods very frequently used in machine learning today can loosely be summarised as trying to detect very sparse solutions early on, mainly with the aim of high computational efficiency. For example, SBL (Tipping, 2001) is much more aggressive in this respect than our EP method here. Early sparsification does not seem to hurt mean prediction performance much, and thus is embraced for efficiency. However, our experiences here indicate that it is the *covariance* (or uncertainty) estimates that can be badly hurt by such sparsity-by-elimination processes, and that in contexts such as experimental design, where covariances are more important than predictive means, their application should probably be avoided. The challenge is then to develop methods that run efficiently *without* eliminating many variables early on, and our selective site updating method for EP is a step in that direction.

References

Candès, E., & Romberg, J. (2004). Practical signal recovery from random projections. *Proceedings of SPIE*.

Candès, E., Romberg, J., & Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theo.*, *52*, 489–509.

Donoho, D. (2006). Compressed sensing. *IEEE Trans. Inf. Theo.*, *52*, 1289–1306.

Duarte, M., Davenport, M., Takhar, D., Laska, J., Sun, T., Kelly, K., & Baraniuk, R. (2008). Single pixel imaging via compressive sampling. To appear in IEEE Signal Processing Magazine.

Elad, M. (2007). Optimized projections for compressed sensing. *IEEE Transactions on Signal Processing*.

Gerwinn, S., Macke, J., Seeger, M., & Bethge, M. (2008). Bayesian inference for spiking neuron models with a sparsity prior. *Advances in NIPS 20*.

Ji, S., & Carin, L. (2007). Bayesian compressive sensing and projection optimization. *Proceedings of ICML 24*.

Minka, T. (2001). Expectation propagation for approximate Bayesian inference. *Uncertainty in AI 17*.

Seeger, M. (2008). Bayesian inference and optimal design in the sparse linear model. To appear in Journal of Machine Learning Research.

Seeger, M., Steinke, F., & Tsuda, K. (2007). Bayesian inference and optimal design in the sparse linear model. *AI and Statistics 11*.

Simoncelli, E. (1999). Modeling the joint statistics of images in the Wavelet domain. *Proceedings 44th SPIE* (pp. 188–195).

Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *J. M. Learn. Res.*, *1*, 211–244.

Weiss, Y., Chang, H., & Freeman, W. (2007). Learning compressed sensing. Snowbird Learning Workshop, Allerton, CA.

Multi-Classification by Categorical Features via Clustering

Yevgeny Seldin[†] SELDIN@CS.HUJI.AC.IL
Naftali Tishby[†,‡] TISHBY@CS.HUJI.AC.IL
[†]School of Computer Science and Engineering, [‡]Center for Neural Computation,
The Hebrew University of Jerusalem, Israel

Abstract

We derive a generalization bound for multi-classification schemes based on grid clustering in categorical parameter product spaces. Grid clustering partitions the parameter space in the form of a Cartesian product of partitions for each of the parameters. The derived bound provides a means to evaluate clustering solutions in terms of the generalization power of a built-on classifier. For classification based on a single feature the bound serves to find a globally optimal classification rule. Comparison of the generalization power of individual features can then be used for feature ranking. Our experiments show that in this role the bound is much more precise than mutual information or normalized correlation indices.

1. Introduction

Clustering is one of the basic tools for dimensionality reduction in categorical spaces. In this paper we study classifiers based on a soft grid clustering of categorical parameter product spaces. The grid clustering is defined by a set of stochastic mappings $\{q_i : \mathcal{X}_i \mapsto \{1, .., m_i\}\}$, one for each parameter i, which map the possible values $\mathcal{X}_i$ of the parameter X_i to a reduced set C_i of size m_i. A classifier based on the grid clustering then assigns a separate prediction strategy to each partition cell. For example, in collaborative filtering we can cluster a thousand by thousand space of viewers by movies into a five by five space of viewer clusters by movie clusters (here X_1 are the viewers and X_2 are the movies). Then we can predict a missing entry within some partition cell with an average of ratings in that cell.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Grid clustering or some other form of dimensionality reduction can be helpful and even essential when the sample size is limited. However, an appropriate choice of clustering resolution is crucial for good results. A coarse clustering may be highly imprecise - think of the extreme of putting all the data into one big cluster. On the other hand, a fine clustering may be statistically unreliable - at the opposite extreme, if we put every parameter value into a separate cluster, some parameter combinations may not occur in the training set at all. Thus, unification of parameter values amplifies the statistical reliability, but reduces the precision. In this paper we relate this tradeoff to generalization properties of a classifier based on the clustering.

Applications of grid clustering to data with intrinsically categorical features are abundant. Seldin, Slonim and Tishby (2007) consider grid clustering from an MDL perspective and demonstrate its success in predicting missing values in the context of collaborative filtering. The same work achieves state of the art performance in terms of coherence of obtained clusters with manual annotation in the context of gene expression and stock data analysis. Here we also suggest a new application of grid clustering for feature ranking.

We are not aware of any previous work on generalization properties of models based on grid clustering. A somewhat related work is (Srebro, 2004), which derives a generalization bound for matrix approximation with bounded norm factorization. However, matrix factorization is a different model and the proof is based on a different technique (Rademacher complexities).

The key point of this paper is a derivation of a generalization bound for classification based on grid clustering. The bound is derived by using the PAC-Bayesian technique (McAllester, 1999). The power of the PAC-Bayesian technique lies in its ability to handle heterogeneous hypothesis classes so that the generalization bound for a specific hypothesis depends on the complexity of that hypothesis rather than on the complexity of the whole class. A classical example of an appli-

cation of the PAC-Bayesian bound are SVMs (Langford, 2005). It is well known that the VC-dimension of separating hyperplanes in $\mathbb{R}^n$ is $n+1$. As well, the VC-dimension of separating hyperplanes with a margin γ, assuming all points are bounded in a unit sphere, is $min\{\frac{1}{\gamma^2}, n\} + 1$. The ability to slice the hypothesis space into infinitely many subspaces characterized by a finer notion of complexity (the size of the margin) rather than the coarse VC-dimension of the whole class makes it possible to derive a better bound that remains meaningful even for infinite dimensional spaces.

In this paper we propose a fine measure of complexity of a grid partition of a cardinal space. The proposed measure of complexity is related to the entropy of a partition along each dimension i. The bound enables us to consider all possible partitions of the product space and to choose one with better generalization properties. In the case of a single parameter it is easy to find a global optimum of the bound. The mapping rule achieving the optimum is shown to be the optimal classification rule from a generalization point of view. Although the bound is not perfectly tight, its shape follows an error on a validation set extremely well. In the experimental section we apply the bound to feature ranking and it is shown to be much more precise than standard mutual information or normalized correlation rankings.

2. A Brief Review of the PAC-Bayesian Generalization Bound

To set the stage, we start with a simplified version of the PAC-Bayesian bound, called Occam's razor. Let $\mathcal{H}$ be a countable hypothesis space. For a hypothesis $h \in \mathcal{H}$ denote by $L(h)$ an expected and by $\hat{L}(h)$ an empirical loss of h. We assume the loss is bounded by b.

Theorem 1 (Occam's razor)**.** *For any data generating distribution and for any "prior distribution" $P(h)$ over $\mathcal{H}$ with a probability greater than $1-\delta$ over drawing an i.i.d. sample of size N, for all $h \in \mathcal{H}$:*

$$L(h) \leq \hat{L}(h) + b\sqrt{\frac{-\ln P(h) - \ln \delta}{2N}}. \qquad (1)$$

Proof. The proof is fairly simple and provides a good illustration of what the "prior distribution" $P(h)$ is. By Hoeffding's inequality $P\{L(h) - \hat{L}(h) \geq \varepsilon(h)\} \leq e^{-2N\varepsilon(h)^2/b^2}$ for any given $h \in \mathcal{H}$. We require that $e^{-2N\varepsilon(h)^2/b^2} \leq P(h)\delta$ for some prior $P(h)$ that satisfies $\sum_{h\in\mathcal{H}} P(h) = 1$. Then, by the union bound $L(h) \leq \hat{L}(h) + \varepsilon(h)$ for all $h \in \mathcal{H}$ with a probability of $1-\delta$. The minimal value of ε that satisfies the requirement is $\varepsilon(h) = b\sqrt{\frac{-\ln P(h) - \ln \delta}{2N}}$, which completes the proof.

We now introduce the notion of a randomized classifier. Let Q be any (posterior) distribution over $\mathcal{H}$. A randomized classifier associated with Q works by choosing a new classifier h from $\mathcal{H}$ according to Q every time a classification is made. We denote the loss of a strategy Q by $L(Q) = \mathbb{E}_{h\sim Q} L(h)$ and similarly $\hat{L}(Q) = \mathbb{E}_{h\sim Q}\hat{L}(h)$. By taking an expectation of (1) over the choice of h and exploiting the concavity of the square root we obtain that with a probability greater than $1-\delta$:

$$L(Q) \leq \hat{L}(Q) + b\sqrt{\frac{-\mathbb{E}_{h\sim Q}\ln P(h) - \ln \delta}{2N}}. \qquad (2)$$

The PAC-Bayesian bound (McAllester, 1999) was derived to allow uncountably infinite hypothesis spaces, though in our case the hypothesis space is finite. We cite a slightly tighter version of the bound proved in (Maurer, 2004).

Theorem 2 (PAC-Bayesian Bound)**.** *For any data distribution and for any "prior" P over $\mathcal{H}$ fixed ahead of training with a probability greater than $1-\delta$ for all distributions Q over $\mathcal{H}$:*

$$L(Q) \leq \hat{L}(Q) + b\sqrt{\frac{D(Q\|P) + \frac{1}{2}\ln(4N) - \ln \delta}{2N}}, \qquad (3)$$

where $D(Q\|P) = \mathbb{E}_{h\sim Q} \ln \frac{Q(h)}{P(h)}$ is the Kullback-Leibler (KL) divergence between the distributions Q and P.

If the loss function is bounded by one (1), (2), and (3) may be written in the form of a bound on the KL-divergence between $\hat{L}(Q)$ and $L(Q)$. For example, (1) may be obtained as: $D(\hat{L}(h)\|L(h)) \leq \frac{-\ln P(h) - \ln \delta}{N}$ if we start from the $P\{L(h) - \hat{L}(h) \geq \varepsilon\} \leq e^{-ND(L(h)+\varepsilon\|L(h))}$ form of Hoeffding's inequality. This provides a better bound in cases when $\hat{L}(Q)$ is sufficiently small (less than $\frac{1}{8}$). The choice of the square root form of the bounds is based on their easier analytical tractability for subsequent minimization.

By writing $D(Q\|P) = -H(Q) - \mathbb{E}_{h\sim Q}\ln P(h)$ it is easy to see that (3) is an improvement over (2) when $H(Q) > \frac{1}{2}\ln(4N)$. In our experiments (2) is usually tighter. It is an open question whether $\frac{1}{2}\ln(4N)$ can be removed from (3), at least in the case of a countable $\mathcal{H}$. For example, (Blanchard & Fleuret, 2007) suggest a parameterized tradeoff $\frac{k+1}{k}D(Q\|P) + \ln(k+1) + 3.5 + \frac{1}{2k}$ instead of $D(Q\|P) + \frac{1}{2}\ln(4N)$. For our data the tradeoff does not improve the results and therefore we omit its discussion.

3. A Formal Definition of Grid Clustering

Before proceeding to the results we provide a formal definition of grid clustering as used in this paper.

Definition 1. Grid Clustering *of the parameter space* $\mathcal{X}_1 \times .. \times \mathcal{X}_d$ *is a set of distributions* $q_i(C_i|X_i)$ *defining the probability of mapping* $X_i \in \mathcal{X}_i$ *to* $C_i \in \{1, .., m_i\}$. *If each of* $q_i(C_i|X_i)$ *is deterministic, we call the clustering a* deterministic grid clustering. *Otherwise it is a* stochastic grid clustering.

In the following sections we assume some unknown joint probability distribution $p(Y, X_1, .., X_d)$ of the parameters and the label exists. The set of all possible labels is denoted by $\mathcal{Y}$ and its size is denoted by n_y. The size of $\mathcal{X}_i$ is denoted by n_i. The cardinality of C_i is m_i. The value of each m_i can vary in the range of $1 \leq m_i \leq n_i$ for different partitions. A hypothesis h in a *deterministic* grid clustering is comprised of a set of deterministic mappings $q_i(C_i|X_i)$, for simplicity denoted by $q_i(X_i) : \mathcal{X}_i \to \{1, .., m\}$, and a set of $\prod_i m_i$ labels, one for each partition cell. We denote the hypothesis space by $\mathcal{H}$ and decompose it as $\mathcal{H} = \mathcal{H}|_1 \times .. \times \mathcal{H}|_d \times \mathcal{H}|_{\mathcal{Y}|\bar{m}}$. Here $\mathcal{H}|_i$ is a space of all possible partitions of $\mathcal{X}_i$, or, in other words, a projection of $\mathcal{H}$ onto dimension i. $\bar{m} = (m_1, .., m_d)$ is a vector of cardinalities of the partitions along each dimension and $\mathcal{H}|_{\mathcal{Y}|\bar{m}}$ is a space of all possible labelings of $\prod_i m_i$ partition cells. Similarly, $h \in \mathcal{H}$ is decomposed as $h = h|_1 \times .. \times h|_d \times h|_{y|\bar{m}}$. It is assumed that a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is given. The loss of h, denoted by $L(h)$, is defined as an expectation over p of L: $L(h) = \mathbb{E}_p l(h(X_1, .., X_d), Y(X_1, .., X_d))$. The empirical loss of h on a sample S of size N is denoted by $\hat{L}(h)$ and equals the average loss on the sample.

For stochastic mappings $q_i(C_i|X_i)$ it is assumed that a random realization of the mapping is done prior to the prediction. In other words, we choose a hypothesis h at random by determining the values of $q_i(X_i)$ according to $q_i(C_i|X_i)$ before we make a prediction. $Q = \left\{\{q_i(C_i|X_i)\}_{i=1}^d, q(Y|C_1, .., C_d)\right\}$ collectively denotes a distribution over $\mathcal{H}$ associated with a randomized classifier called Q. The loss of Q is denoted by $L(Q)$ and equals $L(Q) = \mathbb{E}_{h\sim Q} L(h)$. The empirical loss of Q is denoted by $\hat{L}(Q)$.

We define $q_i(C_i) = \frac{1}{n_i} \sum_{x_i} q_i(C_i|x_i)$ to be a marginal distribution over C_i corresponding to a *uniform* distribution over $\mathcal{X}_i$ and the conditional distribution $q_i(C_i|X_i)$ of our choice. The entropy of a partition along a dimension i with respect to a uniform distribution over $\mathcal{X}_i$ is then $H_U(q_i) \equiv H_U(C_i) = -\sum_{c_i} q_i(c_i) \ln q_i(c_i)$. The mutual information between X_i and C_i with respect to a uniform distribution over $\mathcal{X}_i$ is $I_U(X_i; C_i) = \frac{1}{n_i} \sum_{x_i, c_i} q_i(c_i|x_i) \ln[q_i(c_i|x_i)/q_i(c_i)]$.

4. Generalization Bound for Multi-Classification with Grid Clustering

In this section we state and prove a generalization bound for multi-classification with stochastic grid clustering:

Theorem 3. *For any probability measure* p *over instances and for any loss function* l *bounded by* b, *with a probability of at least* $1-\delta$ *over a selection of an i.i.d. sample* S *of size* N *according to* p, *for all randomized classifiers* $Q = \left\{\{q_i(C_i|X_i)\}_{i=1}^d, q(Y|C_1, .., C_d)\right\}$:

$$L(Q) \leq \hat{L}(Q) + b\sqrt{\frac{\sum_i n_i H_U(C_i) + K}{2N}}, \qquad (4)$$

$$K = \sum_i (m_i \ln n_i + \frac{(\ln(n_i) + 1)^2}{4}) + (\prod_i m_i) \ln n_y - \ln \delta. \qquad (5)$$

It is also possible to replace (4) in the theorem with:

$$L(Q) \leq \hat{L}(Q) + b\sqrt{\frac{\sum_i n_i I_U(X_i; C_i) + \frac{1}{2}\ln(4N) + K}{2N}}. \qquad (6)$$

Proof. The bounds (4) and (6) are direct consequences of (2) and (3) respectively for an appropriate choice of a prior P over $\mathcal{H}$. The main part of the proof is to define a prior P that will provide a meaningful complexity-related slicing of $\mathcal{H}$ and then to calculate $-\mathbb{E}_Q \ln P(h)$ for (4) and $D(Q\|P)$ for (6).

To define the prior P over $\mathcal{H}$ we count the hypotheses in $\mathcal{H}$. For a fixed partition there are $n_y^{\prod_i m_i}$ possibilities to assign the labels to the partition cells. There are n_i possibilities to choose the number of clusters along a dimension i. There are at most $\binom{n_i+m_i-1}{m_i-1} \leq n_i^{m_i-1}$ possibilities to choose a cluster cardinality profile along a dimension i. (This is the number of possibilities to place $m_i - 1$ ones in a sequence of $n_i + m_i - 1$ ones and zeros, where ones symbolize a partition of zeros ("balls") into m_i bins.) We take the $n_i^{m_i-1}$ bound for simplicity. For a fixed cardinality profile $|c_{i1}|, .., |c_{im_i}|$ (over a single dimension) there are $\binom{n_i}{|c_{i1}|, .., |c_{im_i}|}$ possibilities to assign X_i-s to the clusters. This multinomial coefficient can be bounded from above by $e^{n_i H_U(C_i)}$ (see (Cover & Thomas, 1991, page 284) for an elegant proof). Putting all the combinatorial calculations together it is possible to define a distribution $P(h)$ over

$\mathcal{H}$ that satisfies:

$$P(h) \geq \frac{1}{\exp\left[\sum_i \left(n_i H_U(C_i) + m_i \ln n_i\right) + \left(\prod_i m_i\right) \ln n_y\right]}. \quad (7)$$

We pause to stress that unlike in most applications of the PAC-Bayesian bound, in our case the prior P and the posterior Q are defined over slightly different hypothesis spaces. The posterior Q is defined for *named* clusterings - we explicitly specify for each X_i the "name" of C_i it is mapped to. Whereas the prior P is defined over *unnamed* partitions - we only check the cardinality profile of C_i, but we cannot recover which X_i-s are mapped to a given C_i. Nevertheless, the "named" distribution Q induces a distribution over the "unnamed" space by summing up over all possible name permutations. This enables us to compute $-\mathbb{E}_Q \ln P(h)$ we need for the bound.

We now turn to bound $-\mathbb{E}_Q \ln P(h)$. This is done by showing that Q is concentrated around the hypotheses (hard partitions) h for which the entropies of the partitions are close to the entropies $H_U(q_i)$. By the decomposition property we can write: $P(h) = P(h|_1)..P(h|_d)P(h|_{y|\bar{m}})$, and similarly for Q. Then $-\mathbb{E}_Q \ln P(h) = -\sum_i \mathbb{E}_Q \ln P(h|_i) - \mathbb{E}_Q \ln P(h|_{y|\bar{m}})$, and similarly for $D(Q\|P)$. The last term is easy to compute since P is uniform over $\mathcal{H}_{\mathcal{Y}|\bar{m}}$ and Q is defined for a fixed $\bar{m}$. Therefore, $-\mathbb{E}_Q \ln P(h|_{y|\bar{m}}) = \left(\prod_i m_i\right) \ln n_y$. For the first d terms we need to compute or at least to bound $Q(h|_i)$.

Recall that $h|_i$ is obtained from Q by drawing a cluster C_i for each $X_i \in \mathcal{X}_i$ independently according to the distribution $q_i(C_i|X_i)$. Let $\hat{q}_i = \{\frac{|c_{i1}|}{n_i}, .., \frac{|c_{im_i}|}{n_i}\}$ denote an empirical cluster cardinality profile along a dimension i obtained by such assignment. Then:

$$\mathbb{E}_{q_i} H(\hat{q}_i) = H_U(q_i) - \mathbb{E}_{q_i} D(\hat{q}_i \| q_i) \leq H_U(q_i), \quad (8)$$

where $H(\hat{q}_i) = -\sum_{c_i} \hat{q}(c_i) \ln \hat{q}(c_i)$ and $D(\hat{q}_i\|q_i) = \sum_{c_i} \hat{q}_i(c_i) \ln \frac{\hat{q}_i(c_i)}{q_i(c_i)}$. And also:

$$P_{q_i}\left\{H(\hat{q}_i) - \mathbb{E}H(\hat{q}_i) \geq \varepsilon\right\} \leq e^{-2n_i\varepsilon^2/(\ln(n_i)+1)^2}. \quad (9)$$

The latter inequality follows from the fact that the empirical entropy $H(\hat{q}_i)$ satisfies a bounded differences property with a constant equal to $\frac{1}{\ln(n_i)+1}$. See (Paninski, 2003) for a more detailed proof of (8) and (9).

Now, if $\hat{q}_i$ is the cardinality profile of $h|_i$, then $Q(h|_i) = Q(\hat{q}_i) \equiv P_{q_i}\{\hat{q}_i\}$. Let $\varepsilon(\hat{q}_i) = \max\{0, H(\hat{q}_i) - H_U(q_i)\}$. Since $H(\hat{q}_i) - H_U(q_i) \leq H(\hat{q}_i) - \mathbb{E}H(\hat{q}_i)$ by (8), from (9) we have: $Q(\hat{q}_i) \leq e^{-2n_i\varepsilon(\hat{q}_i)^2/(\ln(n_i)+1)^2}$. Thus:

$$-\mathbb{E}_Q \ln P(h|_i) = -\sum_{h|_i \in \mathcal{H}|_i} Q(h|_i) \ln P(h|_i)$$

$$= \sum_{h|_i \in \mathcal{H}|_i} Q(\hat{q}_i)(n_i H(\hat{q}_i) + m_i \ln n_i)$$

$$= \sum_{h|_i \in \mathcal{H}|_i} Q(\hat{q}_i)[n_i H_U(q_i) + m_i \ln n_i + n_i(H(\hat{q}_i) - H_U(q_i))]$$

$$\leq n_i H_U(q_i) + m_i \ln n_i + \sum_{h|_i \in \mathcal{H}|_i} Q(\hat{q}_i) n_i \varepsilon(\hat{q}_i)$$

$$\leq n_i H_U(q_i) + m_i \ln n_i + \sum_{h|_i \in \mathcal{H}|_i} n_i \varepsilon(\hat{q}_i) e^{\frac{-2n_i\varepsilon(\hat{q}_i)^2}{(\ln(n_i)+1)^2}}$$

$$\leq n_i H_U(q_i) + m_i \ln n_i + \int_0^\infty n_i \varepsilon e^{-2n_i\varepsilon^2/(\ln(n_i)+1)^2} d\varepsilon$$

$$= n_i H_U(C_i) + m_i \ln n_i + \frac{1}{4}(\ln(n_i)+1)^2.$$

This completes the proof of (4).

For (6) what remains is to compute $\mathbb{E}_Q \ln Q(h|_i)$. To do so we bound $\ln Q(\hat{q}_i)$ from above. The bound follows from the fact that if we draw n_i values of C_i according to $q_i(C_i|X_i)$ the probability of the resulting type $\hat{q}_i$ is bounded from above by $e^{-n_i H_U(C_i|X_i)}$, where $H_U(C_i|X_i) = -\frac{1}{n_i}\sum_{x_i,c_i} q_i(c_i|x_i) \ln q_i(c_i|x_i)$ (see Theorem 12.1.2 in (Cover & Thomas, 1991)). Thus $\mathbb{E}_Q \ln Q(h|_i) \leq -n_i H_U(C_i|X_i)$, which together with the identity $I_U(X_i;C_i) = H_U(C_i) - H_U(C_i|X_i)$ completes the proof of (6).

5. An Optimal Solution for a Single Feature and Feature Ranking

In this section we show that if there is only one parameter X (i.e., $d = 1$) a globally optimal (from a generalization point of view) classification rule may be efficiently found by examining the "direct" mappings $q(Y|X)$. In other words, for a single parameter there is no need for intermediate clustering. The obtained result is used in the applications section for feature ranking. It is shown there that the bound follows extremely well the shape of the true error of a classifier based on a single feature and is much more precise than mutual information or normalized correlation indices.

To prove the optimality of direct mappings we start with the observation that for any clustering C a classification rule $q(Y|X)$ defined as

$$q(y|x) = \sum_c q(c|x) q(y|c) \quad (10)$$

achieves the same loss as the loss of a hypothesis h based on the clustering C. Therefore, the space of all direct mappings $q(Y|X)$ incorporates all possible solutions that may be achieved via intermediate clustering. It remains to show that the generalization power of the

direct mappings is not worse than the generalization power of clustering-based solutions and that the global optimum may be efficiently found.

To analyze the generalization power of a direct mapping we define n_y clusters c_y, one for each label $y \in \mathcal{Y}$, i.e., $C_y = \{c_y : y \in \mathcal{Y}\}$. All instances x mapped to a cluster c_y obtain the label y. Thus the clustering C_y is identified with the labeling Y, in particular $q(C_y|X) = q(Y|X)$, and we can replace $H_U(C_y)$ in (4) with $H_U(Y)$ and $I_U(X;C_y)$ in (6) with $I_U(X;Y)$. Moreover, in our construction there are only $(n_y!)$ possibilities to assign the labels to the clusters and not $n_y^{n_y}$ as in the case of general clustering. In addition, the cardinality of C_y is fixed at n_y and does not change from 1 to n, where n is the cardinality of X. This further reduces a $\ln(n)$ factor from the bound. Thus, the definition of K in (5) is improved to:

$$K_y = \ln[\binom{n+n_y-1}{n_y-1}] + \frac{(\ln n + 1)^2}{4} + \ln(n_y!) - \ln\delta. \tag{11}$$

(We used the tighter bound $\binom{n+n_y-1}{n_y-1}$ instead of n^{n_y-1} on the number of partitions.) And we get:

$$L(Q) \le \hat{L}(Q) + b\sqrt{\frac{nH_U(Y) + K_y}{2N}} \tag{12}$$

instead of (4) and

$$L(Q) \le \hat{L}(Q) + b\sqrt{\frac{nI_U(X;Y) + K'_y}{2N}} \tag{13}$$

instead of (6) for $K'_y = K_y + \frac{1}{2}\ln(4N)$.

For any other clustering C the direct mapping $q(Y|X)$ defined by (10) satisfies $I_U(X;C) \ge I_U(X;Y)$ by the information processing inequality (Cover & Thomas, 1991). Furthermore, since in $\mathcal{H}$ every partition cell gets a single label, $H_U(Y|C) = 0$. Therefore, $H_U(Y) \le H_U(C)$ because $H_U(Y) = H_U(Y) - H_U(Y|C) = I_U(C;Y) = H_U(C) - H_U(C|Y) \le H_U(C)$. Adding the fact that the empirical losses are equal for the clustering-based classification and the associated direct mapping we obtain that both (12) and (13) for the direct mapping are tighter than (4) and (6) for the corresponding clustering solution.

We can further optimize (13) by looking for an optimal classification rule $q^*(Y|X)$ that minimizes it. The minimum is achieved by iteration of the following self-consistent equations, where $\hat{p}(x,y)$ is the empirical joint distribution of X and Y (the derivation is done by taking a derivative of the bound with respect to $q(Y|X)$ and is omitted due to lack of space):

$$q(y|x) = \frac{q(y)}{Z(x)} e^{-\frac{2}{b}\left(\sum_{y'} \hat{p}(x,y')l(y',y)\right)\sqrt{2N(nI_U(X;Y)+K'_y)}}, \tag{14}$$

$q(y) = \frac{1}{n}\sum_x q(y|x)$, $Z(x) = \sum_y q(y|x)$, and $I_U(X;Y) = \frac{1}{n}\sum_{x,y} q(y|x)\ln[q(y|x)/q(y)]$. Although $\sqrt{I_U(X;Y)}$ is not necessarily convex, in our experiments the iterations always converged to a global optimum. It is also possible to optimize a parameterized tradeoff $\hat{L}(Q) + \beta I_U(X;Y)$, which is convex since both mutual information $I_U(X;Y)$ and the empirical loss $\hat{L}(Q)$ are convex with respect to $q(Y|X)$. A linear search over β then leads to a global optimum of (13).

Note that the direct mapping is no longer optimal when there is more than one parameter. For example, for two parameters X_1, X_2, each with a cardinality n, the conditional distribution $p(Y|X_1,X_2)$ is defined over the product space of size $n^2 n_y$. This requires at least an order of $n^2 n_y$ samples - a number quadratic in n - for the direct inference to be possible. However, from (4) and (6) it follows that with grid clustering for relatively small cluster cardinalities m_i it may be possible to achieve reliable estimations when the sample size N is linear in n. This is further discussed in the next section.

A related bound for generalization in prediction by a single feature is suggested in (Sabato & Shalev-Shwartz, 2007). Sabato and Shalev-Shwartz designed an estimator for the loss of a prediction rule based on the empirical frequencies $q_{emp}(y|x) = \hat{p}(y|x)$. They prove that their estimate is at most $O(\frac{\ln(N/\delta)\sqrt{\ln(1/\delta)}}{\sqrt{N}})$ far from the generalization error of q_{emp}. Compared to their work, a strong advantage of bounds (12) and (13) is that they hold for any prediction rule $q(Y|X)$. In particular, they hold for the maximum likelihood prediction $q_{ml}(x) = \arg\max_y \hat{p}(y|x)$ that performs much better than q_{emp} in practice.

6. A Bound for Estimation of a Joint Probability Distribution in Grid Clustering

For a fixed set of mappings $\{q_i(C_i|X_i)\}$ denote by $p(Y,\bar{C})$ the joint probability distribution of Y and $\bar{C}$, where $\bar{C}$ stays for $\langle C_1,..,C_d\rangle$ for brevity. Denote by $\hat{p}(Y,\bar{C})$ its empirical counterpart. Clearly, $p(Y,\bar{C})$ is determined by $p(Y,X_1,..,X_d)$ and the set $\{q_i(C_i|X_i)\}$. In this section we bound the deviation between $p(Y,\bar{C})$ and its empirical estimation.

Theorem 4. *For any probability measure p over instances and an i.i.d. sample S of size N according to p, with a probability of at least $1-\delta$ for all grid clusterings $Q = \{q_i(C_i|X_i)\}_{i=1}^d$ the following holds:*

$$D(\hat{p}(Y,\bar{C})\|p(Y,\bar{C})) \le \frac{\sum_i n_i H_U(C_i) + K_2}{N} \tag{15}$$

$$K_2 = \sum_i \left(m_i \ln n_i + \frac{(\ln(n_i)+1)^2}{4} \right) + n_y \left(\prod_i m_i \right) \ln(N+1) - \ln \delta. \quad (16)$$

Proof. The proof is based on the law of large numbers cited below (Cover & Thomas, 1991).

Theorem 5 (The Law of Large Numbers)**.** *Let $Z_1, .., Z_N$ be i.i.d. distributed by $p(Z)$. Then:*

$$P\{D(\hat{p}(Z)\|p(Z)) > \varepsilon\} \leq e^{-N\varepsilon + |Z| \ln(N+1)}, \quad (17)$$

where $|Z|$ stays for the cardinality of Z.

Note that the cardinality of the random variable $\langle Y, \bar{C} \rangle$ is $n_y \prod_i m_i$. For the proof of theorem 4 we require that the right hand side of (17) be smaller than $P(h)\delta$. Application of a union bound and reversion of the requirement on ε bounds $D(\hat{p}(Y,\bar{C})\|p(Y,\bar{C}))$ for the case of hard partitions by $\frac{n_y(\prod_i m_i)\ln(N+1) - \ln(P(h)\delta)}{N}$ for all h. Since $D(p\|q)$ is convex in the pair (p,q) (Cover & Thomas, 1991, Theorem 2.7.2), for soft partitions $D(\hat{p}(Y,\bar{C})\|p(Y,\bar{C}))$ is bounded from above by $\mathbb{E}_Q \frac{n_y(\prod_i m_i)\ln(N+1) - \ln(P(h)\delta)}{N}$. The calculation of $-\mathbb{E}_Q \ln P(h)$ done earlier completes the proof.

Applying the inequality relating the L_1 norm and the KL divergence $\|P_1 - P_2\|_1 \leq \sqrt{2D(P_1\|P_2)}$ (see (Cover & Thomas, 1991)) we obtain a bound on the variational distance.

Corollary 1. *Under the conditions of theorem 4:*

$$\|p(Y,\bar{C}) - \hat{p}(Y,\bar{C})\|_1 \leq \sqrt{\frac{2\left(\sum_i n_i H_U(C_i) + K_2\right)}{N}} \quad (18)$$

7. Generalization Bound for the Logarithmic Loss in Grid Clustering

The goal of this section is to provide a bound on the logarithmic loss $-\mathbb{E}\ln\hat{p}(Y|\bar{C}) = -\sum_{y,\bar{c}} p(y,\bar{c}) \ln \hat{p}(y|\bar{c})$. This loss corresponds to the prediction (and compression) power of the hypothesis. Since *ln* is an unbounded function and $\hat{p}(y|c)$ is not bounded from zero, we define a smoothed distribution:

$$p^*(y|\bar{c}) = \frac{\hat{p}(y|\bar{c}) + \gamma}{n_y\gamma + 1},$$

where $\gamma > 0$ is the smoothing parameter. To complete the definition: $p^*(\bar{c}) = \hat{p}(\bar{c})$ and $p^*(y,\bar{c}) = p^*(\bar{c})p^*(y|\bar{c})$. Instead of proving the bound for $\hat{p}$ it will be proved for p^*:

$$-\mathbb{E}\ln p^*(Y|\bar{C}) = -\sum_{y,\bar{c}} p(y,\bar{c}) \ln p^*(y|\bar{c})$$

$$= \sum_{y,\bar{c}} (\hat{p}(y,\bar{c}) - p(y,\bar{c})) \ln p^*(y|\bar{c}) - \sum_{y,\bar{c}} \hat{p}(y,\bar{c}) \ln p^*(y|\bar{c})$$

$$\leq \frac{1}{2}\|p(Y,\bar{C}) - \hat{p}(Y,\bar{C})\|_1 \ln \frac{n_y\gamma + 1}{\gamma} - \sum_{y,\bar{c}} \hat{p}(y,\bar{c}) \ln(\hat{p}(y|\bar{c}) + \gamma) + \ln(n_y\gamma + 1) \quad (19)$$

$$\leq \varepsilon \ln \frac{n_y\gamma + 1}{\gamma} - \sum_{y,\bar{c}} \hat{p}(y,\bar{c}) \ln \hat{p}(y|\bar{c}) + \ln(n_y\gamma + 1)$$

$$= \hat{H}(Y|\bar{C}) + \varepsilon \ln \frac{1}{\gamma} + (\varepsilon + 1)\ln(n_y\gamma + 1), \quad (20)$$

where inequality (19) is justified by (18) and ε is defined as half of its right hand side. $\hat{H}(Y|\bar{C})$ stays for the empirical estimation of the entropy of Y given $\bar{C}$. Equation (20) is minimized for $\gamma = \frac{\varepsilon}{n_y}$, when we get:

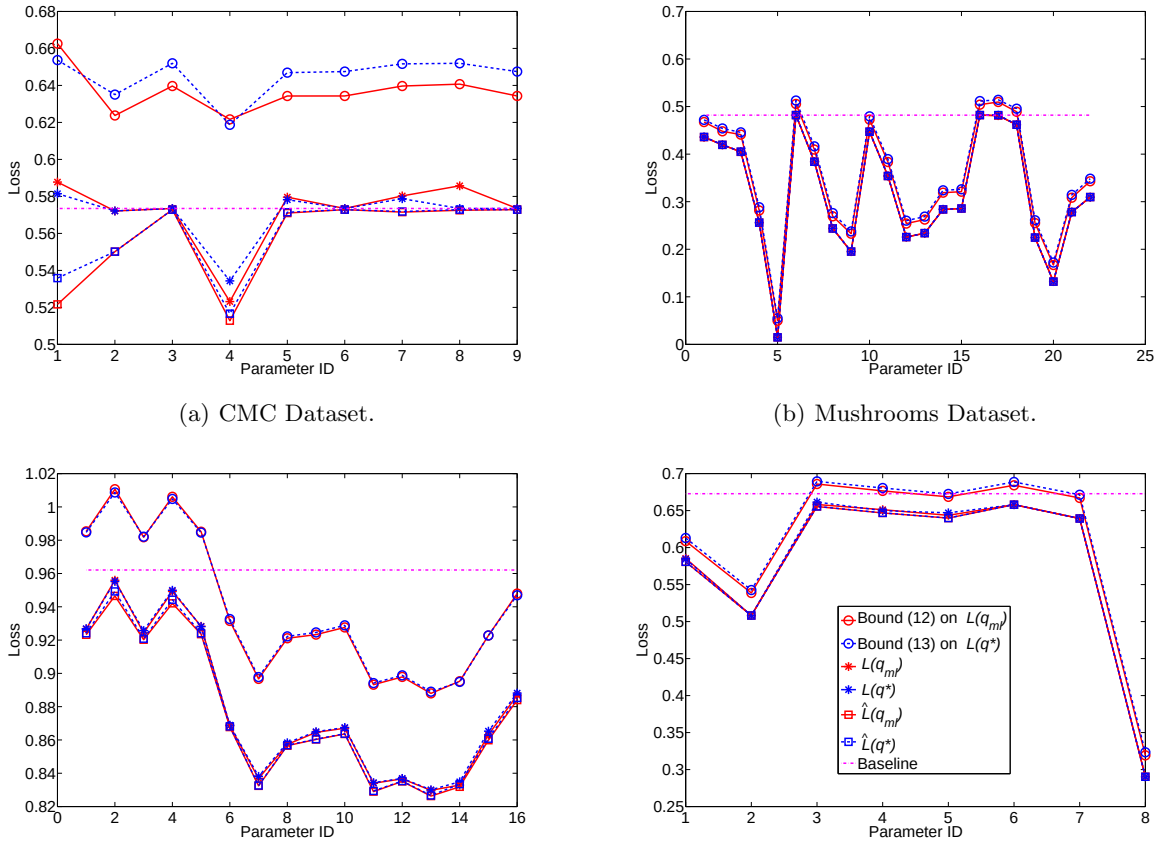
$$-\mathbb{E}\ln p^*(Y|\bar{C}) \leq \hat{H}(Y|\bar{C}) + \varepsilon \ln \frac{n_y}{\varepsilon} + (\varepsilon + 1)\ln(\varepsilon + 1). \quad (21)$$

One natural application of the bound (21) to be studied in future work is to the broadly used "bag-of-words" models, where a decision is made based on multiple observations with the conditional independence assumption on the observations given the label. For example, in the bag of words model for document classification by topic we assume that the words are independent given a topic (the label Y). There is a single parameter X coming from the space of all words, but the classification is based on multiple observations of this parameter - all words in the document. Since we have a single parameter we can resort to the direct mappings $q(Y|X)$, as in section 5. Usually, a topic that maximizes the log likelihood of all the words in a document is assigned. After simple algebraic manipulations this can be translated to maximization of a sum of $\ln[q(y|x)]$ over the document (up to correct normalization by $\ln[q(y)]$), which is directly related to the expectation bounded in (21).

A related work in this context (Shamir, Sabato & Tishby, 2008) uses some different techniques to derive a bound for $|H(Y|C) - \hat{H}(Y|C)|$. We note that $H(Y|C) = -\mathbb{E}\ln p(Y|C)$ is the minimal logarithmic loss that could be achieved if we knew the true joint distribution $p(Y,C)$. Thus, (Shamir et al., 2008) give a lower bound on the performance of any prediction model based on grid clustering, whereas (21) is an upper bound on the performance of the prediction strategy $p^*(Y|\bar{C})$.

8. Applications

In this section we provide a series of applications of the bounds (12) and (13) to prediction by a single feature



(a) CMC Dataset.

(b) Mushrooms Dataset.

(c) Letters Dataset.

(d) Nursery Dataset

Figure 1. **Application of bounds (12) and (13).** This figure displays an application of bounds (12) and (13) to the four datasets discussed in text. The legend in subfigure (d) corresponds to all the graphs. The graphs contain the training loss $\hat{L}(q_{ml})$, the test loss $L(q_{ml})$ and the value of the bound (12) for the maximum likelihood prediction rule $q_{ml}(x) = \arg\max_y \hat{p}(y|x)$. A second triplet on the graphs corresponds to $\hat{L}(q^*)$, $L(q^*)$, and the value of the bound (13) for the prediction rule $q^*(Y|X)$ that minimizes (13). Baseline corresponds to the performance level that can be achieved by predicting the test labels using a marginal distribution of Y on the train set. All the calculations are done per parameter. For better visibility of the points they have been connected with lines, but the lines have no meaning.

and feature ranking, as suggested in section 5. We use (12) to bound the generalization error of the maximum likelihood classification rule. For zero-one loss the maximum likelihood rule $q_{ml}(X)$ returns for each value of x the most frequent value of Y that appeared with that x in the sample: $q_{ml}(x) = \arg\max_y \hat{p}(y|x)$. We also use the iterations (14) to find a classification rule $q^*(Y|X)$ that minimizes (13).

The experiments were conducted on four datasets obtained at the UCI Machine Learning Repository: Contraceptive Method Choice (CMC), Mushrooms, Letters and Nursery. In all the experiments we use 5 random partitions of the data into 80% train and 20% test subsets. Table 1 provides a short summary of the main parameters of the datasets. See (Asuncion & Newman, 2007) for a full description.

Figure 1 shows the training loss and the test loss of the maximum likelihood classification rule $q_{ml}(Y|X)$ for the four datasets considered. We stress that the maximum likelihood rule is calculated per parameter; actually there are d maximum likelihood rules $q_{ml}(Y|X_i)$, one for each parameter i of a given problem. Along with the test loss we draw the value of the bound (12). Note that the bound is quite tight and follows the shape of the test loss remarkably well in all the cases. The gap between the bound and the test loss is less than 0.1.

The same figure includes an additional triplet of lines - training loss, test loss, and the bound (13) value - corresponding to the $q^*(Y|X)$ classification rule that minimizes (13). The performance of q^* is very close to the performance of q_{ml} and the value of (13) is very

Table 1. **Description of the datasets:** for every set we give the number of features, d, a list of cardinalities of the features, n_i, the number of labels, n_y, and a train set size, N, which is 80% of a dataset size.

DATA SET	d	n_i-S	n_y	N
CMC	9	34, 4, 4, 15, 2, 2, 4, 4, 2	3	1,178
MUSHROOMS	22	6, 4, 10, 2, 9, 2, 2, 2, 12, 2, 5, 4, 4, 9, 9, 1, 4, 3, 5, 9, 6, 7, 2	2	6,499
LETTERS	16	16 FOR ALL n_i-S	26	16,000
NURSERY	8	3, 5, 4, 4, 3, 2, 3, 3	5	10,368

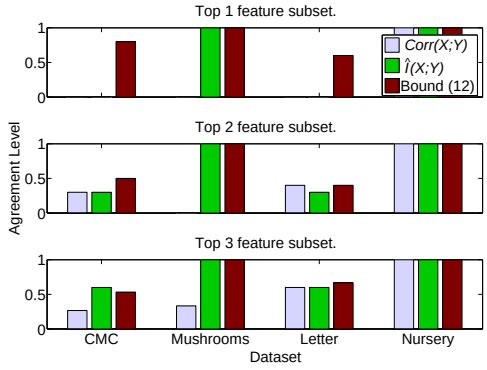


Figure 2. **Feature Ranking.** Agreement of $Corr(X;Y)$, $\hat{I}(X;Y)$, and the bound (12) with the test set on the top-1, top-2, and top-3 feature subsets.

close to the value of (12) with a small advantage to q_{ml} and (12) on average.

We conclude this section by comparing the bound (12) applied to feature ranking with the standard empirical mutual information $\hat{I}(X;Y) = \sum_{x,y} \hat{p}(x)\hat{p}(y|x) \ln \frac{\hat{p}(y|x)}{\hat{p}(y)}$ and the normalized correlation coefficient $Corr(X;Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}}$ indices. We compare agreement between the top-1, top-2, and top-3 parameter subsets suggested by the indices with the corresponding test-based sets - Figure 2. For the top-1 choice (the best single parameter) our bound is clearly superior - it provides a significant level of success in two cases where the other two indices completely fail. For the top-2 choice there is a slight advantage over the mutual information and a clear advantage over the normalized correlation. In top-3 the bound performs similarly to the mutual information and is still superior to the normalized correlation.

9. Discussion

This paper derives generalization bounds for multi-classification based on grid clustering. The bounds enable evaluation of clustering solutions based on generalization properties of a built-on classifier. We acknowledge that the $(\prod_i m_i) \ln n_y$ term in the bounds limits their applicability to relatively few dimensional problems. Nevertheless, this domain contains enough challenges such as feature ranking, where our bounds are especially tight, collaborative filtering and many more. An interesting direction for future work would be to extend the applicability of the approach to higher dimensions by utilizing dependencies between the parameters.

Acknowledgements: We thank Ohad Shamir for useful discussions. This work was partially supported by Leibnitz Center for Research in Computer Science and the NATO SfP Programme.

References

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Blanchard, G., & Fleuret, F. (2007). Occam's hammer. *COLT*.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. John Wiley & Sons.

Langford, J. (2005). Tutorial on practical prediction theory for classification. *JMLR*, *6*.

Maurer, A. (2004). A note on the PAC-Bayesian theorem. www.arxiv.org.

McAllester (1999). Some PAC-Bayesian theorems. *Machine Learning*, *37*.

Paninski, L. (2003). Estimation of entropy and mutual information. *Neural Computation*, *15*.

Sabato, S., & Shalev-Shwartz, S. (2007). Prediction by categorical features: Generalization properties and application to feature ranking. *COLT*.

Seldin, Y., Slonim, N., & Tishby, N. (2007). Information bottleneck for non co-occurrence data. *NIPS*.

Shamir, O., Sabato, S., & Tishby, N. (2008). Learning and generalization with the information bottleneck method. Preprint.

Srebro, N. (2004). *Learning with matrix factorizations*. Doctoral dissertation, MIT.

SVM Optimization: Inverse Dependence on Training Set Size

Shai Shalev-Shwartz SHAI@TTI-C.ORG
Nathan Srebro NATI@UCHICAGO.EDU
Toyota Technological Institute at Chicago, 1427 East 60th Street, Chicago IL 60637, USA

Abstract

We discuss how the runtime of SVM optimization should **decrease** as the size of the training data increases. We present theoretical and empirical results demonstrating how a simple subgradient descent approach indeed displays such behavior, at least for linear kernels.

1. Introduction

The traditional runtime analysis of training Support Vector Machines (SVMs), and indeed most runtime analysis of training learning methods, shows how the training runtime *increases* as the training set size increases. This is because the analysis views SVM training as an optimization problem, whose size increases as the training size increases, and asks "what is the runtime of finding a very accurate solution to the SVM training optimization problem?". However, this analysis ignores the underlying goal of SVM training, which is to find a classifier with low generalization error. When our goal is to obtain a good predictor, having more training data at our disposal should not increase the runtime required to get some desired generalization error: If we can get a predictor with a generalization error of 5% in an hour using a thousand examples, then given ten thousand examples we can always ignore nine thousand of them and do exactly what we did before, using the same runtime. But, can we use the extra nine thousand examples to get a predictor with a generalization error of 5% in *less* time?

In this paper we begin answering the above question. But first we analyze the runtime of various SVM optimization approaches in the data-laden regime, i.e. given unlimited amounts of data. This serves as a basis to our investigation and helps us compare different optimization approaches when working with very large data sets. A similar type of analysis for unregularized linear learning was recently presented by Bottou and Bousquet (2008)—here we handle the more practically relevant case of SVMs, although we focus on linear kernels.

We then return to the finite-data scenario and ask our original question: How does the runtime required in order to get some desired generalization error change with the amount of available data? In Section 5, we present both a theoretical analysis and a thorough empirical study demonstrating that, at least for linear kernels, the runtime of the subgradient descent optimizer PEGASOS (Shalev-Shwartz et al., 2007) does indeed decrease as more data is made available.

2. Background

We briefly introduce the SVM setting and the notation used in this paper, and survey the standard runtime analysis of several optimization approaches. The goal of SVM training is to find a linear predictor $\mathbf{w}$ that predicts the label $y \in \pm 1$ associated with a feature vector $\mathbf{x}$ as $\mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$. This is done by seeking a predictor with small empirical (hinge) loss relative to a large classification "margin". We assume that instance-label pairs come from some source distribution $P(\mathbf{X}, Y)$, and that we are given access to labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$ sampled i.i.d. from P. Training a SVM then amounts to minimizing, for some regularization parameter λ, the regularized empirical hinge loss:

$$\hat{f}_\lambda(\mathbf{w}) = \hat{\ell}(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \qquad (1)$$

where $\hat{\ell}(\mathbf{w}) = \frac{1}{m} \sum_i \ell(\mathbf{w}; (\mathbf{x}_i, y_i))$ and $\ell(\mathbf{w}; (\mathbf{x}, y)) = \max\{0, 1 - y \langle \mathbf{w}, \mathbf{x} \rangle\}$ is the hinge loss. For simplicity, we do not allow a bias term. We say that an optimization method finds an ϵ-accurate solution $\tilde{\mathbf{w}}$ if $\hat{f}_\lambda(\tilde{\mathbf{w}}) \leq \min_{\mathbf{w}} \hat{f}_\lambda(\mathbf{w}) + \epsilon$.

Instead of being provided with the feature vectors directly, we are often only provided with their inner products through a kernel function. Our focus here is on "linear kernels", i.e. we assume we are indeed provided with the feature vectors themselves. This scenario is natural in several applications, including document analysis where the bag-of-words vectors provide a sparse high dimensional representation that does not necessarily benefit from the kernel trick. We use d to denote the dimensionality of the feature

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

vectors. Or, if the feature vectors are sparse, we use d to denote the average number of non-zero elements in each feature vector (e.g. when input vectors are bag-of-words, d is the average number of words in a document).

The runtime of SVM training is usually analyzed as the required runtime to obtain an ϵ-accurate solution to the optimization problem $\min_{\mathbf{w}} \hat{f}_\lambda(\mathbf{w})$.

Traditional optimization approaches converge linearly, or even quadratically, to the optimal solution. That is, their runtime has a logarithmic, or double logarithmic, dependence on the optimization accuracy ϵ. However, they scale poorly with the size of the training set. For example, a naïve implementation of interior point search on the dual of the SVM problem would require a runtime of $\Omega(m^3)$ per iteration, with the number of iterations also theoretically increasing with m. To avoid a cubic dependence on m, many modern SVM solvers use "decomposition techniques": Only a subset of the dual variables is updated at each iteration (Platt, 1998; Joachims, 1998). It is possible to establish linear convergence for specific decomposition methods (e.g. Lin, 2002). However, a careful examination of this analysis reveals that the number of iterations before the linearly convergent stage can grow as m^2. In fact, Bottou and Lin (2007) argue that any method that solves the dual problem very accurately might in general require runtime $\Omega(dm^2)$, and also provide empirical evidence suggesting that modern dual-decomposition methods come close to a runtime of $\Omega(dm^2 \log(1/\epsilon))$. Therefore, for the purpose of comparison, we take the runtime of dual-decomposition methods as $O(dm^2 \log 1/\epsilon)$.

With the growing importance of handling very large data sets, optimization methods with a more moderate scaling on the data set size were presented. The flip side is that these approaches typically have much worse dependence on the optimization accuracy. A recent example is SVM-Perf (Joachims, 2006), an optimization method that uses a cutting planes approach for training linear SVMs. Smola et al. (2008) showed that SVM-Perf can find a solution with accuracy ϵ in time $O(md/(\lambda\epsilon))$.

Although SVM-Perf does have a much more favorable dependence on the data set size, and runs much faster on large data sets, its runtime still increases (linearly) with m. More recently, Shalev-Shwartz et al. (2007) presented PEGASOS, a simple stochastic subgradient optimizer for training linear SVMs, whose runtime does not at all increase with the sample size. PEGASOS is guaranteed to find, with high probability, an ϵ-accurate solution in time[1] $\tilde{O}(d/(\lambda\epsilon))$. Empirical comparisons show that PEGASOS is considerably faster than both SVM-Perf and dual decomposition methods on large data sets with sparse, linear, kernels (Shalev-Shwartz et al., 2007; Bottou, Web Page).

These runtime guarantees of SVM-Perf and PEGASOS are not comparable with those of traditional approaches: the runtimes scale better with m, but worse with ϵ, and also depend on λ. We will return to this issue in Section 4.

3. Error Decomposition

The goal of supervised learning, in the context we consider it, is to use the available training data in order to obtain a predictor with low generalization error (expected error over future predictions). However, since we cannot directly observe the generalization error of a predictor, the training error is used as a surrogate. But in order for the training error to be a good surrogate for the generalization error, we must restrict the space of allowed predictors. This can be done by restricting ourselves to a certain hypothesis class, or in the SVM formulation studied here, minimizing a combination of the training error and some regularization term.

In studying the generalization error of the predictor minimizing the training error on a limited hypothesis class, it is standard to decompose this error into:

- The **approximation error**— the minimum generalization error achievable by a predictor in the hypothesis class. The approximation error does not depend on the sample size, and is determined by the hypothesis class allowed.
- The **estimation error**—the difference between the approximation error and the error achieved by the predictor in the hypothesis class minimizing the training error. The estimation error of a predictor is a result of the training error being only an estimate of the generalization error, and so the predictor minimizing the training error being only an estimate of the predictor minimizing the generalization error. The quality of this estimation depends on the training set size and the size, or complexity, of the hypothesis class.

A similar decomposition is also possible for the somewhat more subtle case of regularized training error minimization, as in SVMs. We are now interested in the generalization error $\ell(\hat{\mathbf{w}}) = \mathbf{E}_{(\mathbf{X},Y)\sim P}\left[\ell(w;\mathbf{X},Y)\right]$ of the predictor $\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \hat{f}_\lambda(\mathbf{w})$ minimizing the training objective (1). Note that for the time being we are only concerned with the (hinge) loss, and not with the misclassification error, and even measure the generalization error in terms of the hinge loss. We will return to this issue in Section 5.2.

- The approximation error is now the generalization error $\ell(\mathbf{w}^*)$ achieved by the predictor $\mathbf{w}^* = \arg\min_{\mathbf{w}} f_\lambda(\mathbf{w})$ that minimizes the *regularized* gen-

[1] The $\tilde{O}(\cdot)$ notation hides logarithmic factors.

eralization error:

$$f_\lambda(\mathbf{w}) = \ell(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 .$$

As before, the approximation error is independent of the training set or its size, and depends on the regularization parameter λ. This parameter plays a role similar to that of the complexity of the hypothesis class: Decreasing λ can decrease the approximation error.

- The estimation error is now the difference between the generalization error of $\mathbf{w}^*$ and the generalization error $\ell(\hat{\mathbf{w}})$ of the predictor minimizing the training objective $\hat{f}_\lambda(\mathbf{w})$. Again, this error is a result of the training error being only an estimate of the generalization error, and so the training objective $\hat{f}_\lambda(\mathbf{w})$ being only an estimate of the regularized loss $f_\lambda(\mathbf{w})$.

The error decompositions discussed so far are well understood, as is the trade-off between the approximation and estimation errors controlled by the complexity of the hypothesis class. In practice, however, we do not minimize the training objective exactly and so do not use the mathematically defined $\hat{\mathbf{w}}$. Rather, we use some optimization algorithm that runs for some finite time and yields a predictor $\tilde{\mathbf{w}}$ that only minimizes the training objective $\hat{f}_\lambda(\mathbf{w})$ to within some accuracy ϵ_{acc}. We should therefore consider the decomposition of the generalization error $\ell(\tilde{\mathbf{w}})$ of this predictor. In addition to the two error terms discussed above, a third error term now enters the picture:

- The **optimization error** is the difference in generalization error between the actual minimizer of the training objective and the output $\tilde{\mathbf{w}}$ of the optimization algorithm. The optimization *error* is controlled by the optimization *accuracy* ϵ_{acc}: The optimization accuracy is the difference in the training objective $\hat{f}_\lambda(\mathbf{w})$ while the optimization error is the resulting difference in generalization error $\ell(\tilde{\mathbf{w}}) - \ell(\hat{\mathbf{w}})$.

This more complete error decomposition, also depicted in Figure 1, was recently discussed by Bottou and Bousquet (2008). Since the end goal of optimizing the training error is to obtain a predictor $\tilde{\mathbf{w}}$ with low generalization error $\ell(\tilde{\mathbf{w}})$, it is useful to consider the entire error decomposition, and the interplay of its different components.

Before investigating the balance between the data set size and runtime required to obtain a desired generalization error, we first consider two extreme regimes: one in which only a limited training set is available, but computational resources are not a concern, and the other in which the training data available is virtually unlimited, but computational resources are bounded.

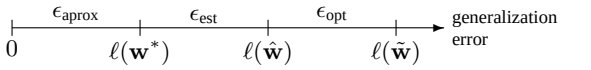


Figure 1. Decomposition of the generalization error of the output $\tilde{\mathbf{w}}$ of the optimization algorithm: $\ell(\tilde{\mathbf{w}}) = \epsilon_{\text{aprx}} + \epsilon_{\text{est}} + \epsilon_{\text{opt}}$.

Table 1. Summary of Notation

error (hinge loss)	$\ell(\mathbf{w};(\mathbf{x},y)) = \max\{0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\}$
empirical error	$\hat{\ell}(\mathbf{w}) = \frac{1}{m}\sum_{(\mathbf{x},y)\in S}\ell(\mathbf{w};(\mathbf{x},y))$
generalization error	$\ell(\mathbf{w}) = \mathbf{E}\left[\ell(\mathbf{w};\mathbf{X},Y)\right]$
SVM objective	$\hat{f}_\lambda(\mathbf{w}) = \hat{\ell}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$
Expected SVM obj.	$f_\lambda(\mathbf{w}) = \ell(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$
Reference predictor	$\mathbf{w}_0$
Population optimum	$\mathbf{w}^* = \arg\min_{\mathbf{w}} f_\lambda(\mathbf{w})$
Empirical optimum	$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \hat{f}_\lambda(\mathbf{w})$
ϵ_{acc}-optimal predictor	$\tilde{\mathbf{w}}$ s.t. $\hat{f}_\lambda(\tilde{\mathbf{w}}) \le \hat{f}_\lambda(\hat{\mathbf{w}}) + \epsilon_{\text{acc}}$

3.1. The Data-Bounded Regime

The standard analysis of statistical learning theory can be viewed as an analysis of an extreme regime in which training data is scarce, and computational resources are plentiful. In this regime, the optimization error diminishes, as we can spend the time required to optimize the training objective very accurately. We need only consider the approximation and estimation errors. Such an analysis provides an understanding of the sample complexity as a function of the target error: how many samples are necessary to guarantee some desired error level.

For low-norm (large-margin) linear predictors, the estimation error can be bounded by $O\left(\frac{\|\mathbf{w}^*\|}{\sqrt{m}}\right)$ (Bartlett & Mendelson, 2003), yielding a sample complexity of $m = O\left(\frac{\|\mathbf{w}^*\|^2}{\epsilon^2}\right)$ to get a desired generalization error of $\ell(\mathbf{w}^*) + \epsilon$ (tighter bounds are possible under certain conditions, but for simplicity and more general applicability, here we stick with this simpler analysis).

3.2. The Data-Laden Regime

Another extreme regime is the regime in which we have virtually unlimited data (we can obtain samples on-demand), but computational resources are limited. This is captured by the PAC framework (Valiant, 1984), in which we are given unlimited, on-demand, access to samples, and consider computationally tractable methods for obtaining a predictor with low generalization error. Most work in the PAC framework focuses on the distinction between polynomial and super-polynomial computation. Here, we are interested in understating the details of this polynomial dependence—how does the runtime scale with the parameters of interest? Discussing runtime as a function of data set size is inappropriate here, since the data set size is unlimited. Rather, we are interested in understanding the runtime as a function of the target error: How much runtime is required to guarantee some desired error level.

As the data-laden regime does capture many large data set situations, in which data is virtually unlimited, such an analysis can be helpful in comparing different optimization approaches. We saw how traditional runtime guarantees of different approaches are sometimes seemingly incomparable: One guarantee might scale poorly with the sample size, while another scales poorly with the desired optimization accuracy. The analysis we perform here allows us to compare such guarantees and helps us understand which methods are appropriate for large data sets.

Recently, Bottou and Bousquet (2008) carried out such a "data-laden" analysis for unregularized learning of linear separators in low dimensions. Here, we perform a similar type of analysis for SVMs, i.e. regularized learning of a linear separator in high dimensions.

4. Data-Laden Analysis of SVM Solvers

To gain insight into SVM learning in the data-laden regime we perform the following "oracle" analysis: We assume there is some good low-norm predictor $\mathbf{w}_0$, which achieves a generalization error (expected hinge loss) of $\ell(\mathbf{w}_0)$ and has norm $\|\mathbf{w}_0\|$. We train a SVM by minimizing the training objective $\hat{f}_\lambda(\mathbf{w})$ to within optimization accuracy ϵ_{acc}. Since we have access to an unrestricted amount of data, we can choose what data set size to work with in order to achieve the lowest possible runtime.

We will decompose the generalization error of the output predictor $\tilde{\mathbf{w}}$ as follows:

$$\begin{aligned}\ell(\tilde{\mathbf{w}}) = \ell(\mathbf{w}_0) \\ + (f_\lambda(\tilde{\mathbf{w}}) - f_\lambda(\mathbf{w}^*)) \\ + (f_\lambda(\mathbf{w}^*) - f_\lambda(\mathbf{w}_0)) \\ + \frac{\lambda}{2}\|\mathbf{w}_0\|^2 - \frac{\lambda}{2}\|\tilde{\mathbf{w}}\|^2 \end{aligned} \quad (2)$$

The degradation in the regularized generalization error, $f_\lambda(\tilde{\mathbf{w}}) - f_\lambda(\mathbf{w}^*)$, which appears in the second term, can be bounded by the empirical degradation: For all $\mathbf{w}$ with $\|\mathbf{w}\|^2 \leq 2/\lambda$ (a larger norm would yield a worse SVM objective than $\mathbf{w}=0$, and so can be disqualified), with probability at least $1-\delta$ over the training set (Sridharan, 2008):

$$f_\lambda(\mathbf{w}) - f_\lambda(\mathbf{w}^*) \leq 2\left[\hat{f}_\lambda(\mathbf{w}) - \hat{f}_\lambda(\mathbf{w}^*)\right]_+ + O\left(\frac{\log\frac{1}{\delta}}{\lambda m}\right)$$

where $[z]_+ = max(z, 0)$. Recalling that $\tilde{\mathbf{w}}$ is an ϵ_{acc}-accurate minimizer of $\hat{f}_\lambda(\mathbf{w})$, we have:

$$f_\lambda(\tilde{\mathbf{w}}) - f_\lambda(\mathbf{w}^*) \leq 2\epsilon_{\text{acc}} + O\left(\frac{\log\frac{1}{\delta}}{\lambda m}\right) \quad (3)$$

Returning to the decomposition (2), the third term is non-positive due to the optimality of $\mathbf{w}^*$, and regarding δ as a constant we obtain that with arbitrary fixed probability:

$$\ell(\tilde{\mathbf{w}}) \leq \ell(\mathbf{w}_0) + 2\epsilon_{\text{acc}} + \frac{\lambda}{2}\|\mathbf{w}_0\|^2 + O\left(\frac{1}{\lambda m}\right) \quad (4)$$

In order to obtain an upper bound of $\ell(\mathbf{w}_0) + O(\epsilon)$ on the generalization error $\ell(\tilde{\mathbf{w}})$, each of the three remaining terms on the right hand side of (4) must be bounded from above by $O(\epsilon)$, yielding:

$$\epsilon_{\text{acc}} \leq O(\epsilon) \quad (5)$$

$$\lambda \leq O\left(\frac{\epsilon}{\|\mathbf{w}_0\|^2}\right) \quad (6)$$

$$m \geq \Omega\left(\frac{1}{\lambda\epsilon}\right) \geq \Omega\left(\frac{\|\mathbf{w}_0\|^2}{\epsilon^2}\right) \quad (7)$$

Using the above requirements on the optimization accuracy ϵ_{acc}, the regularization parameter λ and the working sample size m, we can revisit the runtime of the various SVM optimization approaches.

As discussed in Section 2, dual decomposition approaches require runtime $\Omega(m^2 d)$, with a very weak dependence on the optimization accuracy. Substituting in the sample size required for obtaining the target generalization error of $\ell(\mathbf{w}_0) + \epsilon$, we get a runtime of $\Omega\left(\frac{d\|\mathbf{w}_0\|^4}{\epsilon^4}\right)$.

We can perform a similar analysis for SVM-Perf by substituting the requirements on ϵ_{acc}, λ and m into its guaranteed runtime of $O\left(\frac{dm}{\lambda\epsilon_{\text{acc}}}\right)$. We obtain a runtime of $O\left(\frac{d\|\mathbf{w}_0\|^4}{\epsilon^4}\right)$, matching that in the analysis of dual decomposition methods above. It should be noted that SVM-Perf's runtime has been reported to have only a logarithmic dependence on $1/\epsilon_{\text{acc}}$ in practice (Smola et al., 2008). If that were the case, the runtime guarantee would drop to $\tilde{O}\left(\frac{d\|\mathbf{w}_0\|^4}{\epsilon^3}\right)$, perhaps explaining the faster runtime of SVM-Perf on large data sets in practice.

As for the stochastic gradient optimizer PEGASOS, substituting in the requirements on ϵ_{acc} and λ into its $\tilde{O}(d/(\lambda\epsilon_{\text{acc}}))$ runtime guarantee yields a data-laden runtime of $\tilde{O}\left(\frac{d\|\mathbf{w}_0\|^2}{\epsilon^2}\right)$. We see, then, that in the data-laden regime, where we can choose a data set of arbitrary size in order to obtain some target generalization error, the runtime guarantee of PEGASOS dominates those of other methods, including those with a much more favorable dependence on the optimization accuracy.

The traditional and data-laden runtimes, ignoring logarithmic factors, are summarized in the following table:

Method	ϵ_{acc}-accurate	$\ell(\tilde{\mathbf{w}}) \leq \ell(\mathbf{w}_0) + \epsilon$
Dual decompositoin	dm^2	$\frac{d\|\mathbf{w}_0\|^4}{\epsilon^4}$
SVM-Perf	$\frac{dm}{\lambda\epsilon_{\text{acc}}}$	$\frac{d\|\mathbf{w}_0\|^4}{\epsilon^4}$
PEGASOS	$\frac{d}{\lambda\epsilon_{\text{acc}}}$	$\frac{d\|\mathbf{w}_0\|^2}{\epsilon^2}$

5. The Intermediate Regime

We have so far considered two extreme regimes: one in which learning is bounded only by available data, but not by computational resources, and another where it is bounded only by computational resources, but unlimited data is available. These two analyzes tell us how many samples are needed in order to guarantee some target error rate (regardless of computational resources), and how much computation is needed to guarantee this target error rate (regardless of available data). However, if we have just enough samples to allow a certain error guarantee, the runtime needed in order to obtain such an error rate might be much higher than the runtime given unlimited samples. In terms of the error decomposition, the approximation and estimation errors together would already account for the target error rate, requiring the optimization error to be extremely small. Only when more and more samples are available might the required runtime decrease down to that obtained in the data-laden regime.

Accordingly, we study the runtime of a training method as a decreasing function of the available training set size. As argued earlier, studied this way, the required runtime should never increase as more data is available. We would like to understand how the excess data can be used to decrease the runtime.

In many optimization methods, including dual decomposition methods and SVM-Perf discussed earlier, the computational cost of each basic step increases, sometimes sharply, with the size of the data set considered. In such algorithms, increasing the working data set size in the hope of being able to optimize to within a lower optimization accuracy is a double-edged sword. Although we can reduce the required optimization accuracy, and doing so reduces the required runtime, we also increase the computational cost of each basic step, which sharply increases the runtime.

However, in the case of a stochastic gradient descent approach, the runtime to get some desired optimization accuracy does not increase as the sample size increases. In this case, increasing the sample size is a pure win: The desired optimization accuracy decreases, with no counter effect, yielding a net decrease in the runtime.

In the following sections, we present a detailed theoretical analysis based on performance guarantees, as well as an empirical investigation, demonstrating a decrease in PEGASOS runtime as more data is available.

5.1. Theoretical Analysis

Returning to the "oracle" analysis of Section 4 and substituting into equation (4) our bound on the optimization accuracy of PEGASOS after running for time T, we obtain:

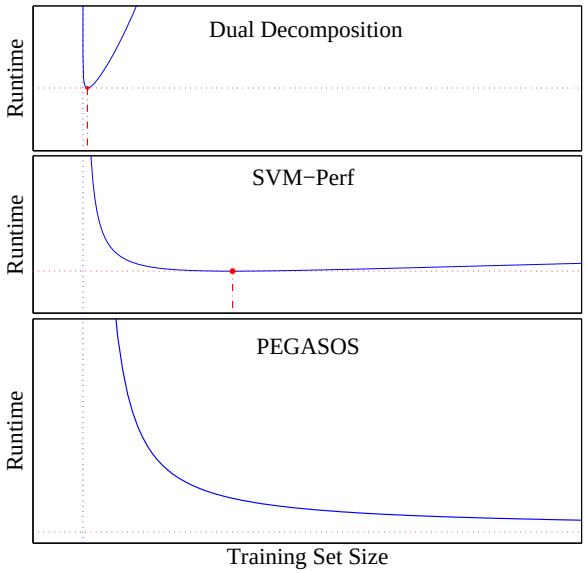


Figure 2. Descriptive behavior of the runtime needed to achieve some fixed error guarantee based on upper bounds for different optimization approaches (solid curves). The dotted lines are the sample-size requirement in the data-bounded regime (vertical) and the runtime requirement in the data-laden regime (horizontal). In the top two panels (dual decomposition and SVM-Perf), the minimum runtime is achieved for some finite training set size, indicated by a dash-dotted line.

$$\ell(\tilde{\mathbf{w}}) \leq \ell(\mathbf{w}_0) + \tilde{O}(\frac{d}{\lambda T}) + \frac{\lambda}{2}\left\|\mathbf{w}_0\right\|^2 + O(\frac{\left\|\mathbf{w}_0\right\|}{\sqrt{m}}) \quad (8)$$

The above bound is minimized when $\lambda = \tilde{\Theta}(\sqrt{\frac{d}{\|\mathbf{w}_0\|^2 T}})$, yielding $\ell(\tilde{\mathbf{w}}) \leq \ell(\mathbf{w}_0) + \epsilon(T, m)$ with

$$\epsilon(T, m) = \tilde{O}\left(\left\|\mathbf{w}_0\right\| \sqrt{\frac{d}{T}}\right) + O\left(\frac{\left\|\mathbf{w}_0\right\|}{\sqrt{m}}\right). \quad (9)$$

Inverting the above expression, we get the following bound on the runtime required to attain generalization error $\ell(\tilde{\mathbf{w}}) \leq \ell(\mathbf{w}_0) + \epsilon$ using a training set of size m:

$$T(m;\epsilon) = \tilde{O}\left(\frac{d}{\left(\frac{\epsilon}{\left\|\mathbf{w}_0\right\|} - O(\frac{1}{\sqrt{m}})\right)^2}\right). \quad (10)$$

This runtime analysis, which monotonically decreases with the available data set size, is depicted in the bottom panel of Figure 2. The data-bounded (statistical learning theory) analysis describes the vertical asymptote of $T(\cdot;\epsilon)$—at what sample size is it at all possible to achieve the desired error. The analysis of the data-laden regime of Section 4 described the minimal runtime using any amount of data, and thus specifies the horizontal asymptote $\inf T(m;\epsilon) =$

$\lim_{m\to\infty} T(m;\epsilon)$. The more detailed analysis carried out here bridges between these two extreme regimes.

Before moving on to empirically observing this behavior, let us contrast this behavior with that displayed by learning methods whose runtime required for obtaining a fixed optimization accuracy does increase with data set size. We can repeat the analysis above, replacing the first term on the right hand side of (8) with the guarantee on the optimization accuracy at runtime of T, for different algorithms.

For SVM-Perf, we have $\epsilon_{\text{acc}} \leq O\left(dm/(\lambda T)\right)$. The optimal choice of λ is then $\lambda = \Theta\left(\sqrt{\frac{dm}{T\|\mathbf{w}_0\|^2}}\right)$ and the runtime needed to guarantee generalization error $\ell(\mathbf{w}_0)+\epsilon$ when running SVM-Perf on m samples is $T(m;\epsilon) = O\left(dm \middle/ \left(\frac{\epsilon}{\|\mathbf{w}_0\|} - O(\frac{1}{\sqrt{m}})\right)^2\right)$. The behavior of this guarantee is depicted in the middle panel of Figure 2. As the sample size increases beyond the statistical limit $m_0 = \Theta(\|\mathbf{w}_0\|^2/\epsilon^2)$, the runtime indeed decreases sharply, until it reaches a minimum, corresponding to the data laden bound, precisely at $4m_0$, i.e. when the sample size is four times larger than the minimum required to be able to reach the desired target generalization error. Beyond this point, the other edge of the sword comes into play, and the runtime (according to the performance guarantees) increases as more samples are included.

The behavior of a dual decomposition method with runtime $\Theta(m^2 d \log\frac{1}{\epsilon_{\text{acc}}})$ is given by $T(m;\epsilon) = m^2 d \log(1/(\epsilon - \Theta\frac{\|\mathbf{w}_0\|}{\sqrt{m}}))$ and depicted in the top panel of Figure 2. Here, the optimal sample size is extremely close to the statistical limit, and increasing the sample size beyond the minimum increases the runtime quadratically.

5.2. Empirical Analysis

The above analysis is based on upper bounds, and is only descriptive, in that it ignores various constants and even certain logarithmic factors. We now show that this type of behavior can be observed empirically for the stochastic subgradient optimizer PEGASOS.

We trained PEGASOS[2] on training sets of increasing size taken from two large data sets, the Reuters CCAT and the CoverType datasets[3]. We measured the average hinge loss of the learned predictor on a (fixed) held-out test set. For each training set size, we found the median number of iterations (over multiple runs with multiple training sets) for achieving some target average hinge loss, which was very slightly above the best "test" hinge loss that could be reliably obtained by training on the entire available training set. For each training set size we used the optimal λ for achieving the desired target hinge loss[4]. The (median) required number of iterations is displayed in Figure 3. For easier interpretability and reproducibility, we report the number of iterations. Since each PEGASOS iteration takes constant time, the actual runtime is proportional to the number of iterations.

So far we have measured the generalization error only in terms of the average hinge loss $\ell(\tilde{\mathbf{w}})$. However, our true goal is usually to attain low misclassification error, $P(Y \neq \operatorname{sign}\langle\tilde{\mathbf{w}}, \mathbf{X}\rangle)$. The dashed lines in Figure 3 indicate the (median) number of iterations required to achieve a target misclassification error, which again is very slightly above the best that can be hoped for with the entire data set.

These empirical results demonstrate that the runtime of SVM training using PEGASOS indeed *decreases* as the size of the training set increases. It is important to note that PEGASOS is the fastest published method for these datasets (Shalev-Shwartz et al., 2007; Bottou, Web Page), and so we are indeed investigating the best possible runtimes. To gain an appreciation of this, as well as to observe the runtime dependence on the training set size for other methods, we repeated a limited version of the experiments using SVM-Perf and the dual decomposition method SVM-Light (Joachims, 1998). Figure 4 and its caption report the runtimes required by SVM-Perf and SVM-Light to achieve the same fixed misclassification error using varying data set sizes. We can indeed verify that PEGASOS's

[2] We used a variant of the method described by Shalev-Shwartz et al. (2007), with a single example used in each update: Following Bottou (Web Page), instead of sampling an example independently at each iteration, a random permutation over the training set is used. When the permutation is exhausted, a new, independent, random permutation is drawn. Although this variation does not match the theoretical analysis, it performs slightly better in practice. Additionally, the PEGASOS projection step is skipped, as it can be shown that even without it, $\|\mathbf{w}\|^2 \leq 4/\lambda$ is maintained.

[3] The binary text classification task CCAT from the Reuters RCV1 collection and Class 1 in the CoverType dataset of Blackard, Jock & Dean. CCAT consists of 804,414 examples with 47,236 features of which 0.16% are non-zero. CoverType has 581,012 examples with 54 features of which 22% are non-zero. We used 23,149 CCAT examples and 58,101 CoverType examples as test sets and sampled training sets from the remainder.

[4] Selecting λ based on results on the test set seems like cheating, and is indeed slightly cheating. However, the same λ was chosen for multiple random training sets of the same size, and represents the optimal λ for the *learning problem*, not for a specific training set (i.e. we are not gaining here from random fluctuations in learning). The setup in which the optimal λ is "known" is common in evaluation of SVM runtime. Choosing λ by proper validation involves many implementation choices that affect runtime, such as the size of the holdout and/or number of rounds of cross-validation, the range of λs considered, and the search strategy over λs. We therefore preferred a "known λ" setup, where we could obtain results that are cleaner, more interpretable, and less affected by implementation details. The behavior displayed by our results is still indicative of a realistic situation where λ must be selected.

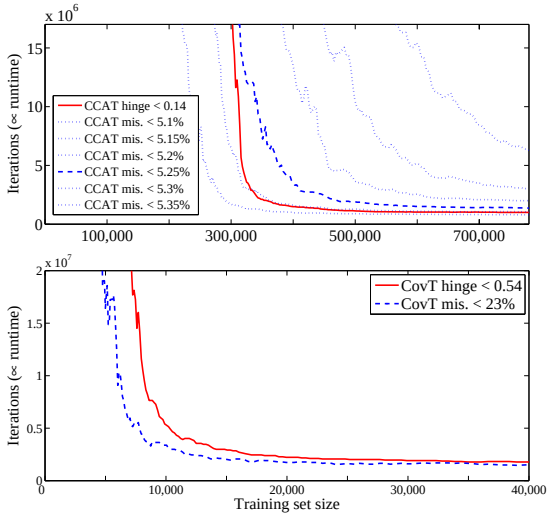


Figure 3. Number of PEGASOS iterations required to achieve the desired hinge loss (solid lines) or misclassification error (dashed and dotted lines) on the test set. Top: CCAT. The minimum achievable hinge loss and misclassification error are 0.132 and 5.05%. Bottom: CoverType. The minimum achievable hinge loss and misclassification error are 0.536 and 22.3%.

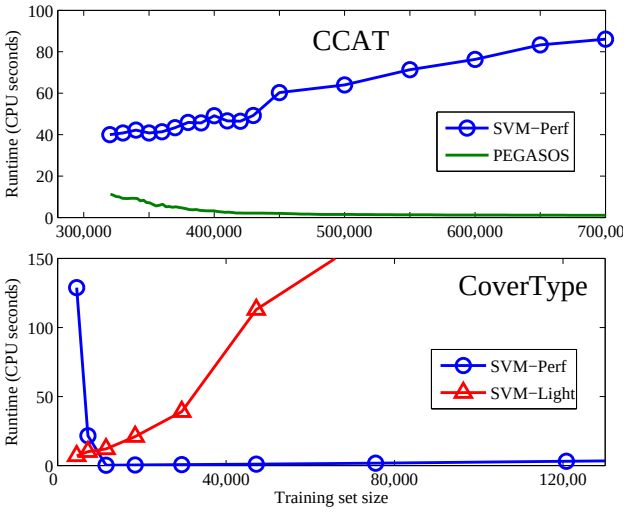


Figure 4. Runtime required to achieve average misclassification error of 5.25% on CCAT (top) and 23% on CoverType (bottom) on a 2.4 GHz Intel Core2, using optimal λ settings. SVM-Light runtimes for CCAT increased from 1371 seconds using 330k examples to 4.4 hours using 700k examples. SVM-Light runtimes for CoverType increased to 552 seconds using 120k examples.

runtime is significantly lower than the optimal SVM-Perf and SVM-Light runtimes on the CCAT dataset. On the CoverType data set, PEGASOS and SVM-Perf have similar optimal runtimes (both optimal runtimes were under a second, and depending on the machine used, each method was up to 50% faster or slower than the other), while SVM-Light's runtime is significantly higher (about 7 seconds). We also clearly see the increase in runtime for large training set sizes for both SVM-Light and SVM-Perf. On the CoverType dataset, we were able to experimentally observe the initial decrease in SVM-Perf runtime, when we are just past the statistical limit, and up to some optimal training set size. On CCAT, and on both data sets for SVM-Light, the optimal data set size is the minimal size statistically required and any increase in data set size increases runtime (since the theoretical analysis is just an upper bound, it is possible that there is no initial decrease, or that it is very narrow and hard to detect experimentally).

In order to gain a better understanding of the reduction in PEGASOS's runtime, we show in Figure 5 the average (over multiple training sets) generalization error achieved by PEGASOS over time, for various data set sizes. It should not be surprising that the generalization error decreases with the number of iterations, nor that it is lower for larger data sets. The important observation is that for smaller data sets the error decreases more slowly, even before the statistical limit for that data set is reached, as opposed to the hypothetical behavior depicted in the insert of Figure 5. This can also be seen in the dotted plots of Figure 3, which are essentially contour lines of the generalization error as a function of runtime and training set size—the error decreases when *either* runtime or training set size increase. And so, fixing the error, we can trade off between the runtime and data set size, decreasing one of them when the other is increased.

The hypothetical situation depicted in the insert occurs when runtime and dataset size each limit the attainable error independently. This corresponds to "L"-shaped contours: both a minimum runtime and a minimum dataset size are required to attain each error level, and once both requirements are met, the error is attainable. In such a situation, the runtime does *not* decrease as data set size increases, but rather, as in the "L"-shaped graph, remains constant once the statistical limit is passed. This happens, e.g., if the optimization can be carried out with a single pass over the data (or at least, if one pass is enough for getting very close to $\ell(\hat{\mathbf{w}})$). Although behavior such as this has been reported using *second-order* stochastic gradient de-

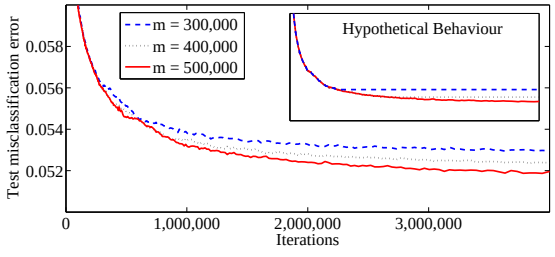


Figure 5. Average misclassification error achieved by PEGASOS on the CCAT test set as a function of runtime (#iterations), for various training set sizes. The insert is a cartoon depicting a hypothetical situation discussed in the text.

scent for *unregularized* linear learning (Bottou & LeCun, 2004), this is not the case here. Unfortunately we are not aware of an efficient one-pass optimizer for SVMs.

6. Discussion

We suggest here a new way of studying and understanding the runtime of training: Instead of viewing additional training data as a computational burden, we view it as an asset that can be used to our benefit. We already have a fairly good understanding, backed by substantial theory, on how additional training data can be used to lower the generalization error of a learned predictor. Here, we consider the situation in which we are satisfied with the error, and study how additional data can be used to decrease training runtime. To do so, we study runtime as an explicit function of the acceptable predictive performance.

Specifically, we show that a state-of-the-art stochastic gradient descent optimizer, PEGASOS, indeed requires training runtime that monotonically decreases as a function of the sample size. We show this both theoretically, by analyzing the behavior of upper bounds on the runtime, and empirically on two standard datasets where PEGASOS is the fastest known SVM optimizer. To the best of our knowledge, this is the first demonstration of a SVM optimizer that displays this natural behavior.

The reason PEGASOS's runtime decreases with increased data is that its runtime to get a fixed optimization accuracy does not depend on the training set size. This enables us to leverage a decreased estimation error, without paying a computational penalty for working with more data.

The theoretical analysis presented in Section 5.1, and we believe also the empirical reduction in PEGASOS's runtime, indeed relies on this decrease in estimation error. This decrease is significant close to the statistical limit on the sample size, as is evident in the results of Figure 3—a roughly 10–20% increase in sample size reduces the runtime by about a factor of five. However, the decrease diminishes for larger sample sizes. This can also be seen from the theoretical analysis—having a sample size which is greater than the statistical limit by a constant factor enables us to achieve a runtime which is greater than the theoretical (data-laden) limit by a constant factor (in fact, as the careful reader probably noticed, since our data-laden theoretical analysis ignores constant factors on ϵ and m, it seems that the training set size needed to be within the data-laden regime, as specified in equation (7), is the same as the minimum data set size required statistically). Such "constant factor" effects should not be discounted—having four times as much data (as is roughly the factor for CoverType) is often quite desirable, as is reducing the runtime by a factor of ten (as this four-fold increase achieves).

We are looking forward to seeing methods that more explicitly leverage large data sets in order to reduce runtime, achieving stronger decreases in practice, and being able to better leverage very large data sets. Although it seems that not much better can be done theoretically given only the simple oracle assumption of Section 4, a better theoretical analysis of such methods might be possible using richer assumptions. We would also like to see practical methods for non-linear (kernelized) SVMs that display similar behavior. Beyond SVMs, we believe that many other problems in machine learning, usually studied computationally as optimization problems, can and should be studied using the type of analysis presented here.

References

Bartlett, P. L., & Mendelson, S. (2003). Rademacher and gaussian complexities: risk bounds and structural results. *J. Mach. Learn. Res., 3*, 463–482.

Bottou, L. (Web Page). Stochastic gradient descent examples. `http://leon.bottou.org/projects/sgd`.

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. *Advances in Neural Information Processing Systems 20*.

Bottou, L., & LeCun, Y. (2004). Large scale online learning. *Advances in Neural Information Processing Systems 16*.

Bottou, L., & Lin, C.-J. (2007). Support vector machine solvers. In L. Bottou, O. Chapelle, D. DeCoste and J. Weston (Eds.), *Large scale kernel machines*. MIT Press.

Joachims, T. (1998). Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods—Support Vector learning*. MIT Press.

Joachims, T. (2006). Training linear svms in linear time. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Lin, C.-J. (2002). A formal analysis of stopping criteria of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks, 13*, 1045–1052.

Platt, J. C. (1998). Fast training of Support Vector Machines using sequential minimal optimization. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods–Support Vector learning*. MIT Press.

Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. *Proceedings of the 24th International Conference on Machine Learning*.

Smola, A., Vishwanathan, S., & Le, Q. (2008). Bundle methods for machine learning. *Advances in Neural Information Processing Systems 20*.

Sridharan, K. (2008). Fast convergence rates for excess regularized risk with application to SVM. `http://ttic.uchicago.edu/~karthik/con.pdf`.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM, 27*, 1134–1142.

Data Spectroscopy: Learning Mixture Models using Eigenspaces of Convolution Operators

Tao Shi TAOSHI@STAT.OSU.EDU
Department of Statistics, Ohio State University

Mikhail Belkin MBELKIN@CSE.OSU.EDU
Department of Computer Science and Engineering, Ohio State University

Bin Yu BINYU@STAT.BERKELEY.EDU
Department of Statistics, University of California Berkeley

Abstract

In this paper we develop a spectral framework for estimating mixture distributions, specifically Gaussian mixture models. In physics, spectroscopy is often used for the identification of substances through their spectrum. Treating a kernel function $K(x, y)$ as "light" and the sampled data as "substance", the spectrum of their interaction (eigenvalues and eigenvectors of the kernel matrix K) unveils certain aspects of the underlying parametric distribution p, such as the parameters of a Gaussian mixture. Our approach extends the intuitions and analyses underlying the existing spectral techniques, such as spectral clustering and Kernel Principal Components Analysis (KPCA).

We construct algorithms to estimate parameters of Gaussian mixture models, including the number of mixture components, their means and covariance matrices, which are important in many practical applications. We provide a theoretical framework and show encouraging experimental results.

1. Introduction

Gaussian mixture models are a powerful tool for various tasks of data analysis, modeling and exploration. The basic problem is to estimate the parameters of a Gaussian mixture distribution $p(x) = \sum_{g=1}^{G} \pi^g p^g(x)$, from sampled data $x_1, \ldots, x_n \in \mathbb{R}^d$, where the mixture component $p^g = N(\mu^g, \Sigma^g)$ has the mean μ^g and the covariance matrix Σ^g, $g = 1, \ldots, G$. Gaussian mixture models are used in a broad range of scientific and engineering applications, including computer vision, speech recognition, and many other areas.

However, effectiveness of modeling hinges on choosing the right parameters for the mixture distribution. The problem of parameter selection for mixture models has a long history, going back to the work of (Pearson, 1894, [9]), who introduced the Method of Moments and applied it to the study of a population of Naples crabs, deducing the existence of two subspecies within the population.

The most commonly used method for parameter estimation is *Maximum Likelihood Estimation* (MLE), which suggests choosing the parameters in a way that maximizes the likelihood of the observed data, given a model. In modern practice this is most commonly done through the iterative optimization technique known as Expectation Maximization (EM) algorithm ([3]), which is typically initialized using k-means clustering. Recently significant progress on understanding theoretical issues surrounding learning mixture distributions and EM has been made in theoretical computer science, e.g., [2, 4].

Another set of methods for inferring mixture distribution is based on the Bayesian inference, which is done using a prior distribution on the parameters of the model. In recent literature ([7]) the Dirichlet process mixture models were used to produce posterior distribution for parameters of a mixture model. The inference procedure involves applying Markov Chain Monte-Carlo to draw samples from the posterior distribution.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

In this paper we propose a new method for estimating parameters of a mixture distribution, which is closely related to non-parametric spectral methods, such as spectral clustering (e.g., [8]) and Kernel Principal Components Analysis [11]. Those methods, as well as certain methods in manifold learning (e.g., [1]), construct a kernel matrix or a graph Laplacian matrix associated to a data set. The eigenvectors and eigenvalues of that matrix can then be used to study the structure of the data set. For example, in spectral clustering the presence of a small non-zero eigenvalue indicates the presence of clusters, while the corresponding eigenvector shows how the data set should be split. In particular, we note the work [12] where the authors analyze dependence of spectra on the input density distribution in the context of classification and argue that lower eigenfunctions can be truncated without sacrificing classification accuracy. We will develop the intuitions and analyses underlying these methods and take them a step further by offering a framework, which can be applied to analyzing parametric families, in particular a mixture of Gaussian distributions.

We would like to study mixture distributions by building explicit connections between their parameters and spectral properties of the corresponding kernel matrices. More specifically, we construct a family of probability-dependent operators and build estimators by matching eigenvalues and eigenfunctions of the operator associated to a probability distribution to those of the matrix associated to a data sample. Thus given a mixture distribution $p(x) = \sum_{g=1}^{G} \pi^g p^g(x)$, we use a Gaussian kernel $K(x,y) = e^{-\frac{\|x-y\|^2}{2\omega^2}}$ to construct the integral operator

$$\mathcal{G}_p^\omega f(y) = \int e^{-\frac{\|x-y\|^2}{2\omega^2}} f(x)\, p(x) dx$$

which will be the principal object of this paper. Our framework will rely on three key observations about the spectral properties of this operator and its connection to the sampled data.

Observation 1. (Single component) For the Gaussian distribution $p = N(\mu, \Sigma)$, we can analytically express eigenfunctions and eigenvalues of $\mathcal{G}_p^\omega$ in terms of the mean μ and the covariance Σ. This will allows us to reverse this dependence and explicitly express μ and Σ in terms of the spectral properties of $\mathcal{G}_p^\omega$.

Observation 2. (Mixture of components)

Let p be a mixture distribution $p(x) = \sum_{g=1}^{G} \pi^g p^g(x)$. Note that by linearity

$$\mathcal{G}_p^\omega f(y) = \sum_{g=1}^{G} \pi^g \int e^{-\frac{\|x-y\|^2}{2\omega^2}} f(x)\, p^g(x) dx$$

$$= \sum_{g=1}^{G} \pi^g \mathcal{G}_{p^g}^\omega f(y)$$

It can be seen (Theorem 1) that given enough separation between the mixture components, top eigenfunctions of the individual components $\mathcal{G}_{p^g}^\omega$ are approximated by top eigenfunctions of $\mathcal{G}_p^\omega$. That will allow us to connect eigenfunctions/eigenvalues of the mixture to eigenfunctions/eigenvalues of the individual components. A specific example of this is given in Fig. 2, which will be discussed in detail in Section 4.

Observation 3. (Estimation from data) The eigenfunctions and eigenvalues of $\mathcal{G}_p^\omega$ can be approximated given data sampled from $p(x)$ by eigenvectors and eigenvalues of empirical kernel matrices.

To highlight the effectiveness of our methodology consider the distribution in Fig. 1, where the density given by a mixture of two normal distributions $p = 0.9\, N(-3, 1^2) + 0.1\, N(0, 0.3^2)$ and a histogram obtained by sampling 1000 points are shown. From the Table 1, we see that the spectroscopic estimator has no difficulty providing reasonably accurate estimates for the mixing coefficients π^1, π^2, means μ^1, μ^2 and variances σ^1, σ^2 for each component, despite the fact that the mixture is unbalanced. We also see that these estimates can be further improved by using the spectroscopic estimate to initialize EM.

We note that, while EM is a computationally efficient and algorithmically attractive method, it is a local optimization procedure and the quality of the achieved maximum and accuracy of the resulting estimate are sensitive to initialization (see, e.g., [10]). If the initial value happens to be close to the global maximum, fast convergence can be guaranteed. However, finding such "lucky" regions of the parameter space may be nontrivial. To emphasize that point, consider the bottom two rows of Table 1, where the results of k-means clustering ($k = 2$) and EM initialized by k-means are shown. We see that k-means consistently provides a poor starting point as the energy minimizing configuration splits the large component, ignoring the small one. EM, initialized with k-means, stays at a local maximum and cannot provide an accurate estimate for the mixture. On the other hand, EM initialized with our method, converges to the correct solution.

We should note that our method requires sufficient separation between the components to provide accurate results. However there does not exist a computationally feasible method for estimating parameters of a mixture distribution in several dimensions without a separation assumption.

The rest of the paper is structured as follows: in Sec-

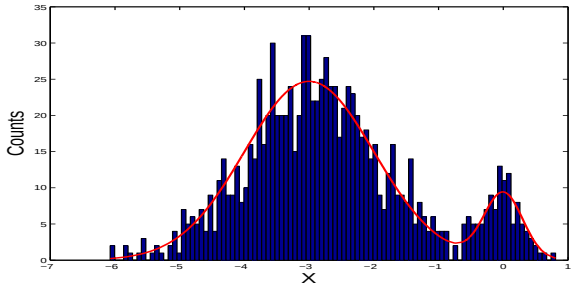


Figure 1. Histogram of 1000 data points sampled from $0.9N(-3, 1^2) + 0.1N(0, 0.3^2)$ and the distribution (red line).

True Parameters	$\pi^1 = 0.9$	$\pi^2 = 0.1$	$\mu^1 = -3$	$\mu^2 = 0$	$\sigma^1 = 1$	$\sigma^2 = 0.3$
Spectroscopic Estimator	0.86 (0.01)	0.14 (0.01)	-2.98 (0.23)	-0.02 (0.08)	1.12 (0.54)	0.34 (0.10)
EM [**SE** initialization]	0.90 (0.01)	0.10 (0.01)	-3.01 (0.04)	0.00 (0.03)	1.00 (0.03)	0.30 (0.02)
k-means [random samples]	0.68 (0.03)	0.32 (0.03)	-3.42 (0.06)	-1.17 (0.16)	0.74 (0.03)	0.90 (0.03)
EM [k-means initialization]	0.78 (0.07)	0.22 (0.07)	-3.17 (0.09)	-0.93 (0.56)	0.92 (0.05)	0.95 (0.39)

Table 1. Mixture Gaussian parameters and corresponding estimators from Spectroscopic Estimation, and EM (initialized by SE), k-means (random initialization) and EM (initialized by k-means). The mean and the (standard deviation) of each estimator over 50 runs are shown.

tion 2, we describe our approach in the simplest setting of a one-dimensional component in $\mathbb{R}$. In Section 3, we analyze a single component in $\mathbb{R}^d$, in Section 4, we deal with a general case of a mixture distribution and state a basic theoretical result for the mixture. In section 5, we show some experimental results on a simulated mixture distribution with three components in $\mathbb{R}^5$ and show some experimental results on the USPS handwritten digit dataset. We conclude in Section 6.

2. Setting Up the Framework: Single Component in $\mathbb{R}$

We start the discussion by demonstrating the basis of our approach on the problem of estimating parameters of a single univariate Gaussian distribution $p(x) = N(\mu, \sigma^2)$. We first establish a connection between eigenfunctions and eigenvalues of the convolution operator $\mathcal{G}_p^\omega f(y) = \int_{\mathbb{R}} e^{-\frac{(x-y)^2}{2\omega^2}} f(x)\, p(x) dx$ and the parameters μ and σ^2. We show these parameters can be estimated from sampled data. We will need the following

Proposition 1 *(Refinement of a result in [13]) Let $\beta = 2\sigma^2/\omega^2$ and let $H_i(x)$ be the i-th order Hermite polynomial. Then eigenvalues and eigenfunctions of $\mathcal{G}_p^\omega$ for $i = 0, 1, \cdots$ are given by*

$$\lambda_i = \frac{\sqrt{2}}{(1+\beta+\sqrt{1+2\beta})^{1/2}} \left(\frac{\beta}{1+\beta+\sqrt{1+2\beta}}\right)^i \quad (1)$$

$$\phi_i(x) = \frac{(1+2\beta)^{1/8}}{\sqrt{2^i i!}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\frac{\sqrt{1+2\beta}-1}{2}\right) \times H_i\left(\left(\frac{1+2\beta}{4}\right)^{\frac{1}{4}} \frac{x-\mu}{\sigma}\right) \quad (2)$$

Since $H_0(x) = 1$, and putting $C = (1+2\beta)^{1/8}$

$$\phi_0(x) = C \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\frac{\sqrt{1+4\sigma^2/\omega^2}-1}{2}\right) \quad (3)$$

We observe that that the maximum value of $|\phi_0(x)|$ is taken at the mean of the distribution μ, hence $\mu = \text{argmax}_x |\phi_0(x)|$. We also observe that $\frac{\lambda_1}{\lambda_0} = \frac{2\sigma^2}{\omega^2}\left(1 + \frac{2\sigma^2}{\omega^2} + \sqrt{1+\frac{4\sigma^2}{\omega^2}}\right)^{-1}$. Taking $r = \lambda_1/\lambda_0$, we derive

$$\sigma^2 = \frac{r\omega^2}{(1-r)^2}. \quad (4)$$

Thus we have established an explicit connection between spectral properties of $\mathcal{G}_p^\omega$ and parameters of $p(x)$. We now present **Algorithm 1** for estimating μ and σ^2 from a sample $x_1, \ldots, x_n$ from $p(x)$.

- **Step 1.** Construct kernel matrix K_n, $(K_n)_{ij} = \frac{1}{n} e^{-\frac{(x_i - x_j)^2}{2\omega^2}}$. K_n serves as the empirical version of the operator $\mathcal{G}_p^\omega$. Compute the top eigenvector v_0 of K_n and the top two eigenvalues $\lambda_0(K_n), \lambda_1(K_n)$.

Actual value	$\mu = 0$	$\sigma = 1$
SE $(\hat{\mu}, \hat{\sigma})$	0.000 (0.014)	1.005 (0.012)
Std Est $(\bar{x}, s)$	0.002 (0.011)	1.001 (0.007)

Table 2. Average(standard deviation) of spectroscopic estimator SE$(\hat{\mu}, \hat{\sigma})$ and the standard estimator Std Est$(\bar{x}, s)$ of 100 simulation run. In each run, estimators are calculated from 1000 $i.i.d$ samples of $N(0, 1)$.

- **Step 2.** Construct estimators $\hat{\mu}$ and $\hat{\sigma}^2$ for mean and variance as follows:

$$\hat{\mu} = x_k, \qquad k = \operatorname*{argmax}_i |(v_0)_i|$$

$$\hat{\sigma}^2 = \frac{\omega^2 \hat{r}}{(1-\hat{r})^2},$$

where $\hat{r} = \frac{\lambda_0(K_n)}{\lambda_1(K_n)}$.

These estimators are constructed by substituting top eigenvector of K_n for the top eigenfunction of $\mathcal{G}_p^\omega$ and eigenvalues of K_n for the corresponding eigenvalues of $\mathcal{G}_p^\omega$.

It is well-known (e.g., [6]) that eigenvectors and eigenvalues of K_n approximate and converge to eigenfunctions and eigenvalues of $\mathcal{G}_p^\omega$ at the rate $\frac{1}{\sqrt{n}}$ as $n \to \infty$, which implies consistency of the estimators. The accuracy of $\hat{\mu}$ and $\hat{\sigma}^2$ depends on how well the empirical operator K_n approximates the underlying operator $\mathcal{G}_p$.

The Table 2 reports the average and the standard deviation of our spectroscopic estimators $(\hat{\mu}, \hat{\sigma}^2)$ compared the standard estimators $(\bar{x}, s^2)$ for one hundred repetitions of the simulation. We see that our spectroscopic estimators are comparable to the standard estimators for mean and variance of a single Gaussian.

3. Setting Up the Framework: Single Component in $\mathbb{R}^d$

In this section we extend our framework to estimating a single multivariate Gaussian $p = N(\mu, \Sigma)$ in $\mathbb{R}^d$. Let $\Sigma = \sum_{i=1}^d \sigma_i^2 u_i u_i^t$ be the spectral decomposition of the covariance matrix Σ. As before we put $\mathcal{G}_p^\omega f(x) = \int_{\mathbb{R}^d} e^{-\frac{\|x-y\|^2}{2\omega^2}} f(y)\, p(y) dy$. Since the kernel $e^{-\frac{\|x-y\|^2}{2\omega^2}}$ is invariant under rotations, it follows that the operator $\mathcal{G}_p^\omega$ can be decomposed as: $\mathcal{G}_p^\omega = \oplus_{i=1}^d \mathcal{G}_{p_i}^\omega$, where p_i is an 1-dimensional Gaussian with variance σ_i^2 and mean $\langle \mu, u_i \rangle$ along the direction of u_i.

It is easy to see that given two operators $\mathcal{F}$, $\mathcal{H}$, the spectrum of their direct sum $\mathcal{F} \oplus \mathcal{H}$ consists of pairwise products $\lambda\mu$, where λ and μ are their respective eigenvalues. The corresponding eigenfunction of the product is $e_{[\lambda,\mu]}(x, y) = e_\lambda(x)\, e_\mu(y)$.

Applying this result, we see that eigenvalues and eigenfunctions of of $\mathcal{G}_p^\omega$ can be written as products

$$\lambda_{[i_1,\ldots,i_d]}(\mathcal{G}_p^\omega) = \prod_{j=1}^d \lambda_{i_j}(\mathcal{G}_{p_j}^\omega)$$

$$\phi_{[i_1,\ldots,i_d]}(\mathcal{G}_p^\omega)(x) = \prod_{j=1}^d \phi_{i_j}(\mathcal{G}_{p_j}^\omega)(\langle x, u_j \rangle)$$

Where $[i_1, \ldots, i_d]$ is a multindex over all components. It can be seen that $\phi_{[0,\ldots,0]}$ is (up to a scaling factor) a Gaussian with the same mean μ as the original distribution $p(x)$. Thus μ can be estimated as the point with maximum value $\phi_{[0,\ldots,0]}$ in the same way as for 1-dimensional distributions.

Consider now ϕ_I, where $I = [\underbrace{0, \ldots, 0}_{i-1}, 1, 0, \ldots, 0]$. Since $H_2(x) = 2x$, Eq. 1 implies that $\frac{\phi_I(x)}{\phi_{[0,\ldots,0]}(x)}$ is a linear function in x with the gradient pointing in the direction of u_i. That allows us to estimate the principal directions. The resulting **Algorithm 2** for estimating μ and Σ is presented below:

Step 1. Construct kernel matrix K_n, $(K_n)_{st} = \frac{1}{n} e^{-\frac{\|x_s - x_t\|^2}{2\omega^2}}$. K_n serves as the empirical version of the operator $\mathcal{G}_p^\omega$. Compute eigenvalues $\lambda(K_n)$ and eigenvectors $v(K_n)$ of K_n. Denote the top eigenvector by v_0 and the corresponding eigenvalue by λ_0.

Step 2. Identify each eigenvector v_i, $v_i \neq v_0$, $i = 1, \ldots, d$ such that the values of $\frac{v_i}{v_0}$ are approximately linear in x, that is

$$\frac{v_i(x_s)}{v_0(x_s)} \approx a^T x_s + b, \qquad a, b \in \mathbb{R}^d$$

The corresponding principal direction u_i is estimated by $\hat{u}_i = \frac{a}{\|a\|}$. Let the corresponding eigenvalue be λ_i.

Step 3. Construct estimators $\hat{\mu}$ and $\hat{\Sigma}$ for mean and variance as follows:

$$\hat{\mu} = x_k, \qquad k = \operatorname*{argmax}_i |(v_0)_i|$$

$$\hat{\Sigma} = \sum_{i=1}^d \hat{\sigma}_i^2 \hat{u}_i \hat{u}_i^t,$$

where $\hat{\sigma}_i^2 = \frac{\omega^2 \hat{r}_i}{(1-\hat{r}_i)^2}$ and $\hat{r}_i = \frac{\lambda_0}{\lambda_i}$.

4. Spectroscopic Estimation for Mixtures of Gaussians

We now extend our framework to the case of a mixture of several multivariate Gaussian distributions with potentially different covariance matrices and mixture coefficients. To illustrate our approach we sample 1000 points from two different Gaussian distributions $N(2,1^2)$ and $N(-2,1^2)$ and from their mixture $0.5N(2,1^2)+0.5N(-2,1^2)$. The histogram of the mixture density is shown in the top left panel of Fig 2, and histograms of each mixture component are shown in the right top panels. Taking the bandwidth $\omega = 0.3$, we construct three kernel matrices K^1, K^2 and K for a sample from each of the components and the mixture distribution respectively. The middle and lower left panels show the top two eigenvectors of K, while the middle and lower right panels show the top eigenvector of K^1 and K^2 respectively.

The key observation is to notice the similarity between the left and right panels. That is, the top eigenvectors of the mixture are nearly identical to the top eigenvectors of each of the components. Thus knowing eigenvectors of the mixture allows us to approximate top eigenvectors (and the corresponding eigenvalues) for each of the components. Having access to these eigenvectors and using our Algorithms 1,2, allows us to estimate parameters of each of the mixture components.

This phenomenon is easily understood from the point of view of operator theory. The leading eigenfunctions of operators defined by each mixture component are approximately the eigenfunctions of the operators defined on the mixture distribution. To be explicit, let us consider the Gaussian convolution operator $\mathcal{G}_p^\omega$ defined by the mixture distribution $p(x) = \pi^1 p^1 + \pi^2 p^2$, with Gaussian components $p^1 = N(\mu^1, \Sigma^2)$ and $p^2 = N(\mu^2, \Sigma^2)$ and the Gaussian kernel $K(x,y)$ with bandwidth ω. The corresponding operators are $\mathcal{G}_{p^1}^\omega$ and $\mathcal{G}_{p^2}^\omega$ and $\mathcal{G}_p^\omega = \pi^1 \mathcal{G}_{p^1}^\omega + \pi^2 \mathcal{G}_{p^2}^\omega$ respectively. Consider an eigenfunction $\phi^1(x)$ of $\mathcal{G}_{p^1}^\omega$ with eigenvalue λ^1, $\mathcal{G}_{p^1}^\omega \phi^1 = \lambda^1 \phi^1$. We have

$$\mathcal{G}_p^\omega \phi^1(y) = \pi^1 \lambda^1 \phi^1(y) + \pi^2 \int K(x,y)\phi^1(x)p^2(x)dx.$$

It can be shown that eigenfunction $\phi^1(x)$ of $\mathcal{G}_{p^1}^\omega$ is centered at μ^1 and decays exponentially away from μ^1. Therefore, assuming the separation $\|\mu^1 - \mu^2\|$ is large enough, the second summand $\pi^2 \int K(x,y)\phi^1(x)p^2(x)dx \approx 0$ for all y uniformly, and hence $\mathcal{G}_p^\omega \phi^1 \approx \pi^1 \lambda^1 \phi^1$. When the approximation holds the top eigenfunctions of $\mathcal{G}_p^\omega$ are approximated by top eigenfunctions of either $\mathcal{G}_{p^1}^\omega$ or $\mathcal{G}_{p^2}^\omega$.

Theorem 1 *Given a d-dimensional mixture of two Gaussians* $p(\mathbf{x}) = \sum_{i=1}^2 \pi^i p^i(x)$ *where* π_i *is mixing weight and* p_i *is the density corresponding to* $N(\mu_i, \sigma^2 I)$. *Define* $\beta = 2\sigma^2/w^2$ *and* $\xi = \sqrt{2}\sigma/\sqrt{\sqrt{1+2\beta}-1}$, *then the first eigenfunction* (ϕ_0^1 *with an eigenvalue* λ_0^1) *of* $\mathcal{G}_{p_1}^w$ *is approximately an eigenfunction of* $\mathcal{G}_p^w$ *in the following sense: For any* $\epsilon > 0$ *we have that for all* y

$$\mathcal{G}_p^w \phi_0^1(y) = \pi_1 \lambda_0^1 (\phi_0^1(y) + T(y)) \quad \textit{and} \quad |T(y)| \le \epsilon$$

assuming that the separation satisfies

$$\frac{\|\mu_1 - \mu_2\|^2}{\xi^2 + \sigma^2} \ge 2\log\left(\frac{\pi_2}{\pi_1}\right) + 2\log\left(\frac{1}{\epsilon}\right) + \frac{d}{4}\log(1+2\beta)$$

We do not provide a proof of Theorem 1 for lack of space. A more general version of the theorem for several Gaussians with different covariance matrices can also be given along the same lines. Together with some perturbation analysis ([5]) it is possible to provide bounds on the resulting eigenvalues and eigenfunctions of the operator.

We now observe that for the operator $\mathcal{G}_{p^g}^\omega$, the top eigenfunction is the only eigenfunction with no sign change. Therefore, such eigenfunction of $\mathcal{G}_p^\omega$ corresponds to exactly one component of the mixture distribution. This immediately suggest a strategy for identifying components of the mixture: we look for eigenfunctions of $\mathcal{G}_p^\omega$ that have *no sign change*. Once these eigenfunctions of $\mathcal{G}_p^\omega$ are identified, each eigenfunction of $\mathcal{G}_p^\omega$ can be assigned to a group determined an eigenfunction with no sign change. As a result, the eigenvalues and eigenfunctions in each group only depend on one of the component p^g and mixing weight π^g. By reversing the relationship between parameters and eigenvalues/eigenfunctions, parameter estimations for each mixing component can be constructed based only on the eigenvalues/eigenvectors in the corresponding group.

4.1. Algorithm for Estimation of a Mixture of Gaussians

Following the discussion above, we now describe the resulting algorithm for estimating a multidimensional mixture of Gaussians $p(x) = \sum_{g=1}^G \pi^g N(\mu^g, \Sigma^g)$, from a sample $x_1, \ldots, x_n \in \mathbb{R}^d$, first giving the following

Definition 1 *For vectors* $d, e \in \mathbb{R}^n$*), we define*
1. ϵ**-support** *of* d *is the set of indices* $\{i\text{: } |d_i| \ge \epsilon,$ $i = 1, \cdots, n\}$.
2. d *has* **no sign changes up to precision** ϵ*, if* d *is*

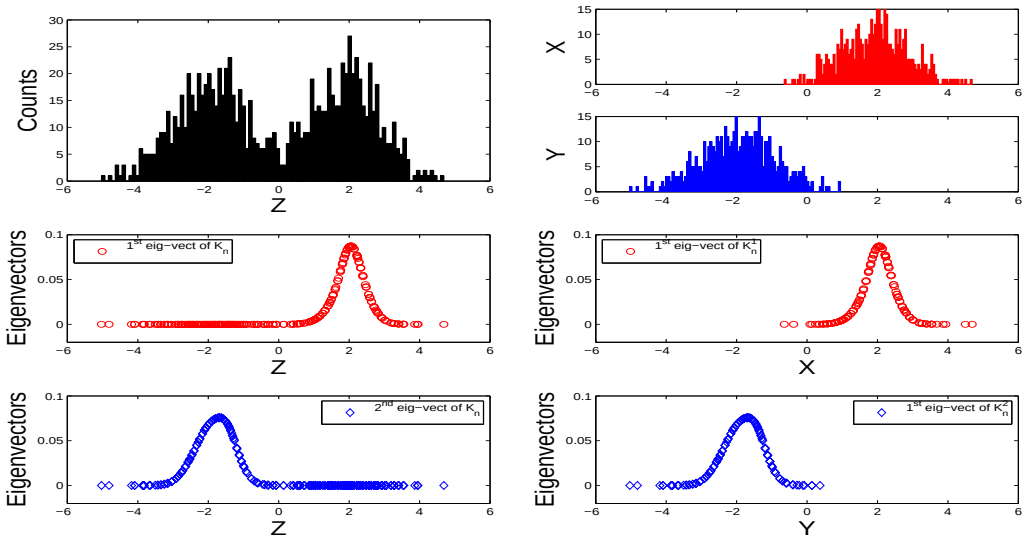


Figure 2. Eigenvectors of a Gaussian kernel matrix ($\omega = 0.3$) of 1000 data sampled from a Mixture Gaussian distribution $P = 0.5N(2, 12) + 0.5N(-2, 12)$. Top left panel: Histogram of the data. Middle left panel: First eigenvector of K_n. Bottom left panel: Second eigenvector of K_n. Top right panel: Histograms of data from each component. Middle right panel: First eigenvector of K_n^1. Bottom right panel: First eigenvector of K_n^2.

either positive or negative on the ϵ-support of e. $\{i : |e_i| \geq \epsilon\} \subset \{i : |d_i| \geq \epsilon\}$.

Algorithm 3. Spectroscopic estimation of a Gaussian mixture distribution.

Input: Data $x_1, \ldots, x_n \in \mathbb{R}^d$. **Parameters:** Kernel bandwidth $\omega > 0$, threshold $\epsilon > 0$.[1]
Output: Number of components $\hat{G}$. Estimated mean $\hat{\mu}^g \in \mathbb{R}^d$, mixing weight $\hat{\pi}^g$, $g = 1, \ldots, \hat{G}$ and covariance matrix Σ^g for each component.

- **Step 1.** Constructing K_n, the empirical approximation to $\mathcal{G}_p^\omega$:

  Put $(K_n)_{ij} = \frac{1}{n} \exp\left(-\frac{\|x_i - x_j\|^2}{2\omega^2}\right)$, $i, j = (1, \ldots, n)$. Compute the (leading) eigenvalues $\lambda_1, \lambda_2, \ldots$ and eigenvectors $v_1, v_2, \ldots$ of K_n.

- **Step 2.** Estimating the number of components G:

  Identify all eigenvectors of K_n, which have no sign changes up to precision ϵ. Estimate G by the number ($\hat{G}$) of such eigenvectors and denote those eigenvectors and the corresponding eigenvalues by $v_0^1, v_0^2, \ldots, v_0^{\hat{G}}$ and $\lambda_0^1, \lambda_0^2, \ldots, \lambda_0^{\hat{G}}$ respectively.

[1] In our implementation of the algorithm we choose $\epsilon = \max_j |(v_i)_j|/n$ for each eigenvector v_i. In the description of the algorithm we will use the same ϵ for simplicity.

- **Step 3.** Estimating the mean μ^g and the mixing weight π^g of each component:

  For the g'th component, $g = 1, \ldots, \hat{G}$, estimate the mean and the mixing weight as follows:

$$\hat{\mu}^g = x_k, \quad \text{where} \quad k = \underset{i}{\text{argmax}} \, |(v_0^g)_i|$$

$$\hat{\pi}^g = \frac{n^g}{\sum_{h=1}^{\hat{G}} n^h},$$

  where $n^h =$ cardinality of ϵ-support of v_0^h.

To estimate the covariance matrix Σ^g of each component p^g: we first all eigenvectors such that $\frac{v(x_s)}{v_0^g(x_s)}$ is approximately a linear function of x_s on the ϵ-support of v_0^g. Then we can apply the estimation methods described in **Algorithm 2**, **Step 3** on the ϵ-support of v_0^g.

5. Simulations and Experiments

Simulation: multivariate Gaussian mixture distribution.
A simulation on five dimensional data is carried out to test the proposed algorithm. The first two variables X_1 and X_2 are a mixture three Gaussian components $p(X) = \sum_{g=1}^3 \pi^g N(\mu^g, \Sigma^g)$ with mixing weights and group means shown in Table 3 and covariance matrices:

$$\Sigma^1 = \begin{pmatrix} 0.5 & -0.25 \\ -0.25 & 0.5 \end{pmatrix}, \quad \Sigma^2 = \begin{pmatrix} 0.5 & 0.25 \\ 0.25 & 0.5 \end{pmatrix},$$

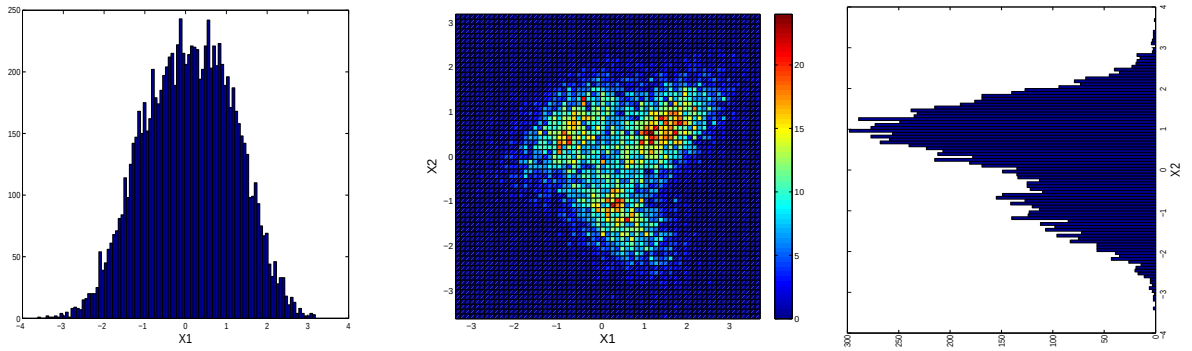


Figure 3. Left: Histogram of the first coordinate X_1; Middle: Two dimensional histogram of the first two coordinates (X_1, X_2). Right: Histogram of X_2.

$$\Sigma^3 = \begin{pmatrix} 0.5 & -0.25 \\ -0.25 & 0.5 \end{pmatrix}$$

The remaining three variables are Gaussian noise $N(0, 0.1I)$. In each simulation run, 3000 data points are sampled. The histogram of X_1, two-dimensional histogram of X_1 and X_2, and histogram of X_2 for one simulation run are shown in Figure 3. We see that it is impossible to identify the number of components by investigating the one-dimensional histograms. The Algorithm 3 with $\omega = 0.1$ was used to estimate the number of components G, mixing weights π^g. The simulation is run 50 times and the algorithm accurately estimated the number of groups in 46 of the 50 runs. Two times the number of groups was estimated as 2 and two times as 4. The average and standard deviation of the estimators of mixing weights and means for the 46 runs are reported in Table 3. We see that the estimates for mixing weights are close to the true values and the estimated group means are close to the estimates from labeled data. Covariance estimates, which we do not show due to space limitations, also show reasonable accuracy.

USPS ZIP code data.
To apply our method to some real-world data we choose a subset of the USPS handwritten digit dataset, consisting of 16x16 grayscale images. In this experiment, 658 "3"s, 652 "4"s, and 556 "5"s in the training data are pooled together as our sample (size 1866). The Spectroscopic estimation algorithm using a Gaussian kernel with bandwidth 2 is applied to the sample . Here we do not use the algorithm to estimate mean and variance of each component, since we do not expect the distribution of the 256 dimensional data to like a Gaussian distribution. Instead, we investigate the eigenvectors with no sign change over $\{x : |v(x)| > \epsilon\}$. We expect (1) the data corresponding to large absolute values of each of such eigenvectors present one mode

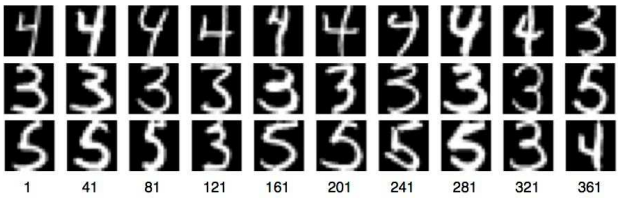


Figure 4. Images ordered by the three eigenvectors v_1, v_{16} and v_{49} identified by Algorithm 3. The images are the digits corresponding to the 1^{st}, 41^{st}, 81^{st}, $\cdots$, 361^{st} largest entries of $|v_1|$ (first row), $|v_{16}|$ (second row) and $|v_{49}|$ (third row).

	"3" (T)	"4" (T)	"5" (T)
"3" (P)	625	0	45
"4" (P)	17	640	32
"5" (P)	16	12	479

Table 4. Confusion matrix of clustering results for USPS handwritten digits. Each cell shows the number of data points belonging both in the **T**rue group (e.g. "3") and the **P**redicted group (e.g. "3")

(cluster) and (2) those data points are in the same digit group.

In the output of our algorithm, three eigenvectors v_1, v_{16} and v_{49} of K_n satisfy the condition of no sign change over $\{x : |v(x)| > \epsilon\}$ with $\epsilon = \max(v)/n$. We first rank the data by an decreasing order of $|v|$ and show the 1^{st}, 41^{st}, 81^{st}, $\cdots$, 361^{st} digits in Figure 4. All digits with larger value of $|v_1|$ belong to the group of "4"s, and other digits ("3" and "5") correspond to smaller values of $|v_1|$. Similarly, larger values of $|v_{16}|$ are in the group of "3"s and $|v_{49}|$ for "5"s.

By assigning digits to their component defined by one of the eigenvectors (v_1, v_{16}, v_{49}) we obtain the clustering results shown in the confusion Table 4. We see that

Parameter value	$\pi^1 = 0.4$		$\pi^2 = 0.3$		$\pi^3 = 0.3$	
Spectroscopy (STD)	0.40 (0.03)		0.30 (0.03)		0.30 (0.03)	
Parameter value	$\mu_1^1 = 1$	$\mu_2^1 = 1$	$\mu_1^2 = 0$	$\mu_2^2 = -1$	$\mu_1^3 = -1$	$\mu_2^3 = 1$
Spectroscopy (STD)	1.00 (0.12)	1.00 (0.19)	0.01 (0.20)	-0.94 (0.21)	-0.96 (0.22)	0.99 (0.22)
$\bar{x}(STD)$ of each group	1.00 (0.02)	1.00 (0.022)	-0.00 (0.03)	-1.00 (0.02)	-1.00 (0.02)	0.99 (0.03)

Table 3. Estimation of mixing weight and mean of each component

the overall accuracy of clustering is 93.46%. This clustering method can be thought of as an extension of the framework provided in this paper. While this method is closely related to spectral clustering, the procedures for choosing eigenvectors are different.

6. Conclusion

In this paper we have presented *Data Spectroscopy*, a new framework for inferring parameters of certain families of probability distributions from data. In particular we have analyzed the case of a mixture of Gaussian distributions and shown how to detect and estimate its components under the assumption of reasonable component separation. The framework is based on the spectral properties of data-dependent convolution operators and extends intuitions from spectral clustering and Kernel PCA. We have developed algorithms and have shown promising experimental results on simulated and real-world datasets.

We think that our approach provides new connections between spectral methods and inference of distributions from data, which may lead to development of algorithms for using labeled and unlabeled data in problems of machine learning.

7. Acknowledgments

The authors thank Yoonkyung Lee for helpful discussions and suggestions. Mikhail Belkin was partially supported by NSF Early Career Award 0643916. Bin Yu was partially supported by NSF grant DMS-0605165, ARO grant W911NF-05-1-0104, NSFC grant 60628102, a grant from MSRA, and a Guggenheim Fellowship in 2006.

References

[1] M. Belkin, P. Niyogi, *Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering*, NIPS 2001.

[2] S. Dasgupta, *Learning mixtures of Gaussians*, FOCS 1999.

[3] A. Dempster, N. Laird, D. Rubin, *Maximum-likelihood from incomplete data via the em algorithm*, Journal of Royal Statistics Society, Ser. B, 39 (1997), pp. 1–38.

[4] R.Kannan, H.Salmasian, S.Vempala, *The spectral method for general mixture models*, COLT 2005.

[5] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, 1966.

[6] V. Koltchinskii and E. Giné, *Random matrix approximation of spectra of integral operators*, Bernoulli, 6 (2000), pp. 113 – 167.

[7] S. MacEachern and P. Muller, *Estimating mixture of Dirichlet process models*, Journal of Computational and Graphical Statistics, 7 (1998), pp. 223–238.

[8] A. Ng, M. Jordan, and Y. Weiss, *On spectral clustering: Analysis and an algorithm*, NIPS 2001.

[9] K. Pearson, *Contributions to the mathematical theory of evolution*, Phil. Trans. Royal Soc., 185A, (1894), pp. 71–110.

[10] R. Redner and H. Walker, *Mixture densities, maximum likelihood and the em algorithm*, SIAM Review, 26 (1984), pp. 195–239.

[11] B. Schölkopf, A. J. Smola, and K.-R. Müller, *Kernel principal component analysis*, Advances in kernel methods: support vector learning, (1999), pp. 327–352.

[12] C.Williams, M.Seeger, *The effect of the input density distribution on kernel-based classifiers*, ICML2000.

[13] H. Zhu, C. Williams, R. Rohwer, and M. Morcinie, *Gaussian regression and optimal finite dimensional linear models*, in Neural networks and machine learning, C. Bishop, ed., 1998.

A Generalization of Haussler's Convolution Kernel — Mapping Kernel

Kilho Shin YSHIN@CMUJ.JP
Carnegie Mellon CyLab Japan, 3-3 Higashi-kawasaki-cho, Chuo-ku, Kobe, Hyogo, Japan

Tetsuji Kuboyama KUBOYAMA@GAKUSHUIN.AC.JP
Gakushuin University, Mejiro, Toshima-ku, Tokyo, Japan

Abstract

Haussler's convolution kernel provides a successful framework for engineering new positive semidefinite kernels, and has been applied to a wide range of data types and applications. In the framework, each data object represents a finite set of finer grained components. Then, Haussler's convolution kernel takes a pair of data objects as input, and returns the sum of the return values of the predetermined primitive positive semidefinite kernel calculated for all the possible pairs of the components of the input data objects. On the other hand, the *mapping kernel* that we introduce in this paper is a natural generalization of Haussler's convolution kernel, in that the input to the primitive kernel moves over a predetermined subset rather than the entire cross product. Although we have plural instances of the mapping kernel in the literature, their positive semidefiniteness was investigated in case-by-case manners, and worse yet, was sometimes incorrectly concluded. In fact, there exists a simple and easily checkable necessary and sufficient condition, which is generic in the sense that it enables us to investigate the positive semidefiniteness of an arbitrary instance of the mapping kernel. This is the first paper that presents and proves the validity of the condition. In addition, we introduce two important instances of the mapping kernel, which we refer to as the *size-of-index-structure-distribution* kernel and the *edit-cost-distribution* kernel. Both of them are naturally derived from well known (dis)similarity measurements in the literature (*e.g.* the maximum agreement tree, the edit distance), and are reasonably expected to improve the performance of the existing measures by evaluating their distributional features rather than their peak (maximum/minimum) features.

1. Introduction

Haussler's convolution kernel (Haussler, 1999) has been used as a general framework to tailor known primitive kernels to the context of specific applications. In this section, we first review a degenerated form of Haussler's convolution kernel, which proves in fact to be equivalent to the general form of Haussler's convolution kernel (see 2.2). Let each data point x in a space χ be associated with a finite subset χ'_x of a common space χ'. Furthermore, we assume that a kernel $k : \chi' \times \chi' \to \mathbb{R}$ is given. Then, Haussler's convolution kernel $K : \chi \times \chi \to \mathbb{R}$ is defined as follows (see 2.2).

$$K(x, y) = \sum_{(x',y') \in \chi'_x \times \chi'_y} k(x', y') \tag{1}$$

Haussler proved that, if $k(x', y')$ is positive semidefinite, then so is $K(x, y)$. Haussler's convolution kernel is known to have a wide range of application (Lodhi et al., 2001; Collins & Duffy, 2001; Suzuki et al., 2004).

On the other hand, the *mapping kernel* is a natural generalization of Haussler's convolution kernel, and is defined by Eq. (2) for $\{M_{x,y} \subseteqq \chi'_x \times \chi'_y \mid (x, y) \in \chi^2\}$. The problem that the present paper addresses is to determine whether the mapping kernel is positive semidefinite.

$$K(x, y) = \sum_{(x',y') \in M_{x,y}} k(x', y') \tag{2}$$

The main contribution of the present paper is to present a necessary and sufficient condition for the mapping kernel $K(x, y)$ defined by Eq. (2) to be positive semidefinite for all possible choices of positive semidefinite $k(x', y')$. More specifically, we prove that the condition is that the *mapping system* $\{M_{x,y} \mid (x, y) \in \chi^2\}$ is *transitive*, *i.e.*, $(x', y') \in M_{x,y} \wedge (y', z') \in M_{y,z} \Rightarrow (x', z') \in M_{x,z}$. Haussler's convolution kernel is indeed the special case of the mapping kernel for $\{M_{x,y} = \chi'_x \times \chi'_y\}$, which is apparently transitive.

We see plural instances of the mapping kernel in the literature, and some of them were mistreated in respective manners.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

- Although the *elastic tree kernel* (Kashima & Koyanagi, 2002) was treated as an instance of Haussler's convolution kernel, it is, in fact, an instance of the mapping kernel. Therefore, the positive semidefiniteness of the kernel should not have been determined based on Haussler's theorem.
- The *codon-improved kernel* (Zien et al., 2000) was claimed to be unconditionally positive semidefinite, since it was viewed as an instance of the polynomial kernel. The kernel, in fact, is an instance of the mapping kernel under certain settings of weights.

That is to say, the positive semidefiniteness of the aforementioned kernels were concluded on wrong grounds, and in fact, the conclusion regarding the codon-improved kernel is wrong — in reality, it is not necessarily positive semidefinite.

The kernels introduced in (Menchetti et al., 2005) and (Kuboyama et al., 2006) are also instances of the mapping kernel. In contrast to the elastic and codon-improved kernels, their positive semidefiniteness was properly investigated, albeit in specific manners.

This is the first paper that recognizes the mapping kernel as a generic class of kernels, and presents a necessary and sufficient condition that a mapping kernel becomes positive semidefinite. Furthermore, the condition is simple, intuitive and easy to check, and therefore, would make engineering of new instances of the mapping kernel easier, more efficient and more effective to a large extent.

As the second contribution of the present paper, we take advantage of the mapping kernel, and present a way to augment a couple of well-known frameworks to engineer similarity functions for discretely structured objects (*e.g.* strings, trees, general graphs).

It is known that the *maximum* sizes of shared substructures of the objects can be used as a good measure of similarities of the objects. The *maximum agreement subtree* is a good example. Also, the *edit distance* has been applied to various types of objects. An edit distance between two data objects is generally defined as the minimum cost of edit scripts that transform one object into the other.

These two frameworks are common in that they only focus on the maximum/minimum values of the similarity measures (*i.e.* the sizes of shared substructures and the costs of edit scripts), and therefore, only those substructures with the maximum sizes or those edit scripts with the minimum costs can contribute to the similarity functions. It is, however, reasonably presumable that distributional features of the measurements may carry useful information with regard to similarities of objects, and more accurate similarity functions can be engineered by evaluating the distributional features.

Based on the aforementioned consideration, we introduce two novel classes of kernels (similarity functions) each evaluating the distributional features of the sizes of shared substructures or the costs of edit scripts. Also, we show a general way to view them as mapping kernels. By virtue of our simple criteria for positive semidefinite mapping kernels, we can easily determine whether instances of the new kernel classes are positive semidefinite, and, if they are, we can take advantage of sophisticated classifiers such as support vector machines (SVM). In 3.1 and 3.2, we see that the examples of distribution-based similarity functions derived from maximum agreement subtrees and general tree edit distances are positive semidefinite, while those derived from *maximum refinement trees* (Hein et al., 1996) and *less-constrained tree edit distance* (Lu et al., 2001) are not.

2. The Mapping Kernel

In this section, as a preliminary, we quickly review the positive semidefinite kernel (2.1) and Haussler's convolution kernel (2.2). Then, we describe our main theorem with regard to the mapping kernel (2.3).

2.1. The Positive Semidefinite Kernel

A kernel $K : \chi \times \chi \longrightarrow \mathbb{R}$ is said to be positive semidefinite, if, and only if, for arbitrary $x_1, \ldots, x_n \in \chi$, the corresponding Gram matrix $G = [K(x_i, x_j)]_{i,j=1,\ldots,n}$ is a positive semidefinite matrix. Positive semidefiniteness of kernels is a critical condition for reproducing kernel Hilbert spaces to exist. In simpler cases where a data point space χ is finite, this condition is equivalent to the property that there exists a mapping $\Phi : \chi \longrightarrow \mathbb{R}^N$ such that $K(x, y) = \Phi(x)\Phi(y)^{\mathsf{T}}$.

In this paper, by a positive semidefinite matrix, we mean a real symmetric matrix (*i.e.* $A^{\mathsf{T}} = A$) that satisfies one of, hence, all of the mutually equivalent conditions stated below, where $\dim A = n$.

- $(c_1, \ldots, c_n)A(c_1, \ldots, c_n)^{\mathsf{T}} \geq 0$ for $\forall (c_1, \ldots, c_n) \in \mathbb{R}^n$.
- A has only non-negative real eigenvalues.
- There exists an n-dimensional orthogonal matrix P (*i.e.* $P^{\mathsf{T}}P = E_n$) such that $P^{\mathsf{T}}AP$ is a diagonal matrix with non-negative elements.
- $A = B^{\mathsf{T}}B$ for some $m \times n$ real matrix B.

2.2. Haussler's Convolution Kernel

Hausler's theorem (Haussler, 1999, Theorem 1) asserts the positive semidefiniteness of Haussler's R-convolution kernel, and Theorem 1 presents its special case for $D = 1$.

Theorem 1. *Let $k : \chi' \times \chi' \to \mathbb{R}$ be a positive semidefinite kernel. Given a relation $R \subseteq \chi' \times \chi$, $K : \chi \times \chi \to \mathbb{R}$ defined by Eq. (3) is also positive semidefinite.*

$$K(x, y) = \sum_{(x',x)\in R} \sum_{(y',y)\in R} k(x', y') \tag{3}$$

It is interesting to note that Haussler's theorem for $D > 1$ is obtained as a corollary to Theorem 1.

Corollary 1. (Haussler, 1999) *Let $k_d : \chi'_d \times \chi'_d \longrightarrow \mathbb{R}$ be positive semidefinite kernels for $d = 1, \ldots, D$. Given a relation $R \subset \chi'_1 \times \cdots \times \chi'_D \times \chi$, the kernel $K : \chi \times \chi \longrightarrow \mathbb{R}$ defined below is also positive semidefinite.*

$$K(x, y) = \sum_{(x'_1,\ldots,x'_D,x)\in R} \sum_{(y'_1,\ldots,y'_D,y)\in R} \prod_{d=1}^{D} k_d(x'_d, y'_d)$$

2.3. Definition and Main Theorem

Letting χ'_x denote $\{x' \in \chi' \mid (x', x) \in R\}$, Eq. (1) gives an equivalent form of Eq. (3). On the other hand, the *mapping kernel* is defined so that (x', y') moves over a subset $M_{x,y}$ of $\chi'_x \times \chi'_y$ rather than the entire cross product $\chi'_x \times \chi'_y$ (Eq. (2)).

The present paper shows that the mapping kernel is positive semidefinite for all possible choices of positive semidefinite underlying kernels k, if, and only if, $\{M_{x,y} \mid x, y \in \chi\}$ is *transitive* (Definition 2).

Therefore, for an arbitrary non-transitive $\{M_{x,y}\}$, a positive semidefinite underlying kernel $k(x', y')$ exists such that the resulting $K(x, y)$ is not positive semidefinite (4.1.2). On the other hand, $K(x, y)$ may be positive semidefinite even for a non-transitive $\{M_{x,y}\}$ and a positive semidefinite $k(x', y')$ (Example 1).

Example 1. The (k, m)-mismatch kernel $K_{(k,m)}(x, y)$ is positive semidefinite (Leslie et al., 2004). When χ'_x and χ'_y denote the sets of k-mers in x and y, $K_{(k,m)}(x, y)$ can be regarded as a mapping kernel for the non-transitive $\{M_{x,y}\}$ defined as follows.

$$M_{x,y} = \{(x', y') \mid K_{(k,m)}(x', y') \neq 0\} \subseteq \chi'_x \times \chi'_y$$

$$K_{(k,m)}(x, y) = \sum_{(x',y')\in M_{x,y}} K_{(k,m)}(x', y')$$

The result is formalized as follows.

Definition 1. A *mapping system* $\mathcal{M}$ is a triplet $(\chi, \{\chi'_x \mid x \in \chi\}, \{M_{x,y} \subseteq \chi'_x \times \chi'_y \mid (x, y) \in \chi^2\})$ such that $|M_{x,y}| < \infty$ and $(y', x') \in M_{y,x}$ if $(x', y') \in M_{x,y}$.

Definition 2. A mapping system $(\chi, \{\chi'_x\}, \{M_{x,y}\})$ is said to be *transitive*, if, and only if, $(x'_1, x'_2) \in M_{x_1,x_2} \wedge (x'_2, x'_3) \in M_{x_2,x_3} \Rightarrow (x'_1, x'_3) \in M_{x_1,x_3}$ holds for arbitrary $x_i \in \chi$ and $x'_i \in \chi'_{x_i}$ $(i = 1, 2, 3)$.

Definition 3. An *evaluating system* $\mathcal{E}$ for a mapping system $(\chi, \{\chi'_x\}, \{M_{x,y}\})$ is a triplet $(\chi', k, \{\gamma_x \mid x \in \chi\})$ with a positive semidefinite *underlying kernel* $k : \chi' \times \chi' \to \mathbb{R}$ and *projections* $\gamma_x : \chi'_x \to \chi'$.

Definition 4. For a mapping system $\mathcal{M} = (\chi, \{\chi'_x\}, \{M_{x,y}\})$ and an evaluating system $\mathcal{E} = (\chi', k, \{\gamma_x\})$ for $\mathcal{M}$, the *mapping kernel* with respect to $\mathcal{M}$ and $\mathcal{E}$ is defined by Eq. (4).

$$K(x, y) = \sum_{(x',y')\in M_{x,y}} k(\gamma_x(x'), \gamma_y(y')) \tag{4}$$

Now, our main theorem is described as follows, and its proof is given in Section 4.

Theorem 2. *For a mapping system $\mathcal{M}$, the following are equivalent to each other.*

1. *$\mathcal{M}$ is transitive.*
2. *For an arbitrary evaluating system $\mathcal{E}$ for $\mathcal{M}$, the mapping kernel with respect to $\mathcal{M}$ and $\mathcal{E}$ is positive semidefinite.*

It is possible to prove (1) $\Rightarrow$ (2) of Theorem 2 as a corollary to Theorem 1. Nevertheless, our direction in the present paper is opposite — we like to view Theorem 1 as a trivial corollary to Theorem 2. In fact, we will prove Theorem 2 without assuming Theorem 1 in Section 4.

3. Similarity Functions Based on Distributions

In this section, we introduce two new classes of the mapping kernel. The kernels are expected to improve the classification performance of known similarity measurements by evaluating their distributional features.

3.1. Size-of-index-structure-distribution Kernels

When some structures are commonly derived from two data objects, the structures may carry information with regard to similarities between the data objects. In this paper, we call such structures *index structures*.

The *agreement subtree* is a good example of the index structure, when data objects are represented as

trees. An agreement subtree between plural input trees is usually defined as a subtree *homeomorphically* included in all the input trees (Berry & Nicolas, 2004). In the present paper, we assume that the input trees are a pair of trees. Even when we fix the input tree pair, there may exist more than one agreement subtree, and the maximum size of the agreement subtrees can be naturally viewed as a measure of similarities between the input trees. The *maximum agreement subtrees* (MAST) problem is the problem to determine at least one agreement subtree with the *maximum size* among the possible agreement subtrees for the input trees. The MAST problem has been extensively studied from the application point of view (*e.g.* evolutionary trees (Hein et al., 1996; Berry & Nicolas, 2004), shape-axis trees (Pelillo, 2002)) as well as from the algorithm efficiency point of view.

When using the size of the maximum agreement subtrees as a similarity measurement between trees, we discard those agreement subtrees smaller in size than the maximum ones, and therefore, they do not contribute to the final evaluation at all. It is, however, reasonable to think that distributional features of the sizes of agreement subtrees may carry useful information with regard to similarities of the trees.

Based on the aforesaid consideration, we introduce the kernel of Eq. (5), which evaluates distributional features of the sizes of agreement subtrees. In Eq. (5), we let $\mathsf{AST}(x, y)$ denote the set of the agreement subtrees between x and y, and $f : \mathbb{N} \to \mathbb{R}_+ = \{y \geq 0 \mid y \in \mathbb{R}\}$ be an increasing function.

$$K(x, y) = \sum_{t \in \mathsf{AST}(x,y)} f(\mathsf{size\_of}(t)) \tag{5}$$

If x and y are rooted trees of bounded degree, and if $f(n) = \alpha^n$ or $f(n) = n$, for example, there exist polynomial-time efficient algorithms to calculate $K(x, y)$.

Beside the advantages due to the distributional features, the kernel could provide the advantage of using sophisticated classifiers such as SVM (Cristianini & Shawe-Taylor, 2000). In fact, our contribution asserts that $K(x, y)$ is positive semidefinite as follows. First, $K(x, y)$ can be viewed as a mapping kernel under the following notation.

- χ'_x is the set of the subtrees of x.
- $M_{x,y} = \{(x', y') \in \chi'_x \times \chi'_y \mid x' \cong y'\}$, where $x' \cong y'$ means that they are homeomorphic as trees.
- $k(x', y') = \begin{cases} f(\mathsf{size\_of}(x')) & \text{if } \mathsf{size\_of}(x') = \mathsf{size\_of}(y') \\ 0 & \text{otherwise.} \end{cases}$

It is easy to see that $\{M_{x,y}\}$ is transitive and $k(x', y')$ is positive semidefinite. Hence,

$$K(x, y) = \sum_{t \in \mathsf{AST}(x,y)} f(\mathsf{size\_of}(t)) = \sum_{(x',y') \in M_{x,y}} k(x', y')$$

is positive semidefinite by Theorem 2.

Besides the maximum agreement subtree, the *maximum refinement subtree* (Hein et al., 1996; Berry & Nicolas, 2004), *maximum subtree isomorphism* (Pelillo, 2002; Aoki et al., 2003) and *maximum agreement supertree* (Jansson et al., 2005) are also used as index structures for trees. As for general graphs, the *maximal common clique* included in an input pair of graphs is also studied in association with MAST in (Pelillo, 2002).

For each of those index structures, we can define kernels in the same way as for MAST. We have only to replace $\mathsf{AST}(x, y)$ in Eq. (5) with the set of the respective index structures. Moreover, except for the maximum refinement subtree, through the same discussion as for MAST, the kernels prove to be positive semidefinite.

Interestingly, Theorem 2 also implies that the kernels defined based on the minimum refinement subtree are not necessarily positive semidefinite. The minimum refinement subtree for $x' \subseteqq x$ and $y' \subseteqq y$ is defined as the minimum tree t such that both x' and y' can be derived from t through a sequence of *edge contractions*, and the maximum refinement subtree problem (*a.k.a.* the maximum compatible tree problem) is the problem to find a minimum refinement subtree with the largest size. Different from the agreement subtree, the relation of having a refinement is not an equivalence relation — even if x' and y', and y' and z', have refinement subtrees, x' and z' do not necessarily have a refinement subtree. This implies that the corresponding $M_{x,y}$ is not necessarily transitive. Therefore, Theorem 2 asserts that the corresponding $K(x, y)$ is not necessarily positive semidefinite.

3.2. Edit-cost-distribution Kernels

The *Edit distance* is also used as an effective measure of similarities between discrete data structures (*e.g.* (Wagner & Fischer, 1974) for strings, (Barnard et al., 1995) for trees, (Bunke, 1997) for general graphs).

Let x be an object consisting of one or more components. For example, a string consists of one or more characters which are laid out on a line. For another example, a graph consists of one or more vertices and edges, and each edge connects a vertex to another. We first give a general definition of *edit operations*, *edit*

scripts, *edit costs* and *edit distances* for such objects.

An edit operation is an operation on a component of x, and is one of (i) substituting a component b for a component a of x (denoted by $\langle a \to b\rangle$), (ii) deleting a component a of x (denoted by $\langle a \to \bullet\rangle$), and (iii) inserting a component a into x (denoted by $\langle \bullet \to a\rangle$). An edit script is a sequence of zero or more edit operations which transforms an object into another. When a cost $\gamma\langle a \to b\rangle \in \mathbb{R}$ is given for each edit operation $\langle a \to b\rangle$[1], the cost $\gamma(\sigma)$ of an edit script σ is the sum of the costs of the edit operations that comprise σ. Finally, an edit distance $d(x, y)$ between objects x and y is defined by:

$$d(x, y) = \min\{\gamma(\sigma) \mid \sigma \text{ transforms } x \text{ into } y\}.$$

Therefore, those edit scripts with larger costs than the minimum cost do not contribute to the final edit distance. In contrast, by introducing kernels by Eq. (6) with a decreasing function $f : \mathbb{R}_+ \to \mathbb{R}_+$, we try to take advantage of the information that those discarded edit scripts potentially carry.

$$K(x, y) = \sum_{\sigma \text{ transforms } x \text{ into } y} f(\gamma(\sigma)) \qquad (6)$$

It is important to note that there exists a natural interpretation of Eq. (6). In a natural setting where the cost $\gamma\langle a \to b\rangle$ is defined as the negative logarithm of the probability that the substitution of b for a (a or b could be $\bullet$) would occur (*e.g.* (Li & Jiang, 2005; Salzberg, 1997)), we let $f(x) = e^{-x}$. For an edit script $\sigma = \langle x'_1 \to y'_1\rangle \cdots \langle x'_n \to y'_n\rangle$ transforming x into y, $f(\gamma(\sigma))$ is evaluated as follows.

$$\begin{aligned} f(\gamma(\sigma)) &= e^{-\gamma(\sigma)} = e^{-\sum_{i=1}^{n} \gamma\langle x'_i \to y'_i\rangle} \\ &= e^{-\sum_{i=1}^{n} -\log \Pr(x'_i \to y'_i)} = \prod_{i=1}^{n} \Pr(x'_i \to y'_i) \end{aligned}$$

Hence, $K(x, y)$ by Eq. (6) equals the total probability that x would be transformed into y.

Usage of sophisticated classifiers such as SVM is another potential advantage of the kernels of the form of Eq. (6). In fact, as shown below, the kernels can be viewed as mapping kernels, if we can pose the following four assumptions.

[1]Usually, components are labeled with elements of an alphabet, and costs of edit operations are defined on the labels rather than on the components. However, for simplicity, we assume that the cost function is defined over the space of objects in the present paper. In addition, to make the resulting edit distance be a distance metric, the costs are often assumed to be a distance metric.

1. The cost function is symmetric (*i.e.* $\gamma\langle a \to b\rangle = \gamma\langle b \to a\rangle$).
2. We let $f(x) = e^{-cx}$ for some positive constant c.
3. In order to avoid calculating infinite sums, we take only *irreducible* edit scripts into consideration in calculating Eq. (6) — Assume that $\sigma = \langle x'_1 \to y'_1\rangle \ldots \langle x'_n \to y'_n\rangle$ transforms x into y. σ is irreducible, if, and only if, (1) x'_i (resp. y'_i) is either a component of x (resp. y) or $\bullet$ and (2) exactly one edit operation $\langle x'_i \to y'_i\rangle$ is applied to each component of x and y.
4. If two irreducible edit scripts differ from each other only in the order of the included edit operations, they are identified in calculating Eq. (6), that is, they are evaluated only once.

For $\sigma = \langle x'_1 \to y'_1\rangle \ldots \langle x'_n \to y'_n\rangle$, we assume that x'_i and y'_i are respectively components of x and y, if, and only if, $i \in \{1, \ldots, m(\sigma)\}$, and call $\langle x'_1 \to y'_1\rangle \cdots \langle x'_{m(\sigma)} \to y'_{m(\sigma)}\rangle$ the *core* of σ. Then, $\gamma(\sigma)$ and $K(x, y)$ are evaluated as follows.

$$\begin{aligned} \gamma(\sigma) &= \sum_{i=1}^{m(\sigma)} (\gamma\langle x'_i \to y'_i\rangle - \gamma\langle x'_i \to \bullet\rangle - \gamma\langle \bullet \to y'_i\rangle) \\ &\quad + \sum_{x' \in x} \gamma\langle x' \to \bullet\rangle + \sum_{y' \in y} \gamma\langle \bullet \to y'\rangle \end{aligned}$$

$$\begin{aligned} K(x, y) = &\prod_{\xi \in x} f(\gamma\langle \xi \to \bullet\rangle) \cdot \prod_{\eta \in y} f(\gamma\langle \bullet \to \eta\rangle) \cdot \\ &\left[\sum_{\sigma} \left(\prod_{i=1}^{m(\sigma)} \frac{f(\gamma\langle x'_i \to y'_i\rangle)}{f(\gamma\langle x'_i \to \bullet\rangle) f(\gamma\langle \bullet \to y'_i\rangle)} \right)\right] \end{aligned} \qquad (7)$$

In Eq. (7), the first two factors of the right-hand side are functions of x and y, and therefore, we denote them by $g(x)$ and $g(y)$, respectively. On the other hand, the last factor is a function of $x' = (x'_1, \ldots, x'_{m(\sigma)})$ and $y' = (y'_1, \ldots, y'_{m(\sigma)})$, and is denoted by $k(x', y')$. We define $M_{x,y}$ as follows.

$$\begin{aligned} M_{x,y} = \{&((x'_1, \ldots, x'_m), (y'_1, \ldots, y'_m)) \mid \\ &\exists\sigma[\langle x'_1 \to y'_1\rangle \cdots \langle x'_m \to y'_m\rangle \text{ is the core of } \sigma]\} \end{aligned}$$

Then, the following holds

$$\begin{aligned} K(x, y) &= g(x) \cdot g(y) \cdot \left(\sum_{(x', y') \in M_{x,y}} k(x', y') \right) \\ &= g(x) \cdot g(y) \cdot \bar{K}(x, y) \end{aligned}$$

In particular, $\bar{K}(x, y)$ is a mapping kernel, and $K(x, y)$ is positive semidefinite, if, and only if, so is $\bar{K}(x, y)$,

since $g(x)$ cannot take the value 0. The kernel $\bar{K}(x,y)$, however, is not necessarily positive semidefinite, even if $k(x',y')$ is positive semidefinite, since $\{M_{x,y}\}$ is not necessarily transitive. We will investigate this problem taking the *tree edit distance* as an example.

For the tree edit distance, the edit operations act on vertices of trees. For a pair (x',y') to be the core of some irreducible tree edit script, it is necessary and sufficient that φ defined by $\varphi(x'_i) = y'_i$ preserves the ancestor-descendent relation and the sibling (left-to-right) relation (Tai, 1979). Therefore, $M_{x,y}$ for the general tree edit distance is defined as follows, where $x'_i < x'_j$ means x'_j is an ancestor of x'_i and $x'_i \prec x'_j$ means x'_j is located on the right side of x'_i.

$$M_{x,y} = \{((x'_1,\ldots,x'_m),(y'_1,\ldots,y'_m)) \mid [x'_i < x'_j \Leftrightarrow y'_i < y'_j] \wedge [x'_i \prec x'_j \Leftrightarrow y'_i \prec y'_j]\} \quad (8)$$

It is straightforward to verify that $\{M_{x,y}\}$ is transitive. Therefore, Theorem 2 asserts that, if $k(x',y')$ is positive semidefinite, so is $\bar{K}(x,y)$ for this $\{M_{x,y}\}$.

On the other hand, two subclasses of the general tree edit distance have been proposed. They are *constrained* (*a.k.a.* structure-preserving) tree edit distance (Zhang, 1995) and *less-constrained* (*a.k.a.* alignable) tree edit distance (Lu et al., 2001).

Those subclasses of the general tree edit distance determine respective $M_{x,y}$, which are generally proper subsets of those define by (8). Since $\{M_{x,y}\}$ for the constrained tree edit distance is easily verified to be transitive, the resulting $\bar{K}(x,y)$ turns out positive semidefinite by virtue of Theorem 2. In contrast to the constrained edit distance, $\{M_{x,y}\}$ for the less-constrained tree edit distance is not transitive. Therefore, Theorem 2 implies that $\bar{K}(x,y)$ is not necessarily positive semidefinite.

4. Proof of Theorem 2

4.1. Key Lemma

Let X^{ij} be m-dimensional square matrices parameterized by $(i,j) = \{1,\ldots,n\}^2$, and let X denote the derived mn-dimensional square matrix $[X^{ij}]_{i,j=1,\ldots,n}$ — the $(m(i-1)+k, m(j-1)+l)$-element of X, denoted by X^{ij}_{kl}, is defined to be the (k,l)-element of X^{ij}.

Furthermore, for an m-dimensional square matrix A, $\mathrm{smry}_A(X)$ denotes the n-dimensional square matrix $[\mathrm{tr}(A^\mathsf{T} X^{ij})]_{i,j=1,\ldots,n}$. Note that the (i,j)-element of $\mathrm{smry}_A(X)$ is given by Eq. (9).

$$\mathrm{tr}(A^\mathsf{T} X^{ij}) = \sum_{k=1}^{m}\sum_{l=1}^{m} A_{kl} X^{ij}_{kl} \quad (9)$$

Proposition 1. *For an m-dimensional square matrix A, the following are equivalent to each other.*

1. *A is positive semidefinite.*
2. *$\mathrm{smry}_A(X)$ is positive semidefinite for an arbitrary mn-dimensional positive semidefinite matrix X.*

Proof. First, we prove the assertion assuming that A is diagonal, whose I-th diagonal element is α_I.

The condition 2 implies 1, since we see $\alpha_I \geq 0$ for any I by letting X be the sparse matrix such that X_{kl} is 1, if $k = l = I$, and 0, otherwise.

On the other hand, the converse follows from Eq. (10), since $\mathrm{smry}_A(X) = Z^\mathsf{T} Z$ holds for the $m^2 n \times n$ matrix Z such that $Z_{mn(I-1)+m(k-1)+i,j} = \sqrt{\alpha_I} Y^{kj}_{iI}$, where Y is an mn-dimensional matrix such that $X = Y^\mathsf{T} Y$.

$$\begin{aligned} \mathrm{tr} A^\mathsf{T} X^{ij} &= \sum_{I=1}^{m} \alpha_I \left(\sum_{k=1}^{n}\sum_{l=1}^{m} Y^{ki}_{lI} Y^{kj}_{lI} \right) \quad (10) \\ &= \sum_{I=1}^{m}\sum_{k=1}^{n}\sum_{l=1}^{m} (\sqrt{\alpha_I} Y^{ki}_{lI})(\sqrt{\alpha_I} Y^{kj}_{lI}) \end{aligned}$$

The general cases for non-diagonal A reduces to the diagonal case, since, for P such that $P^\mathsf{T} A P$ is diagonal, $\mathrm{smry}_A(X) = \mathrm{smry}_{P^\mathsf{T} A P}(\tilde{X})$ holds for $\tilde{X} = [P^\mathsf{T} X^{ij} P]_{i,j=1,\ldots,n}$. □

4.2. (1) Implies (2)

Investigating whether K is positive semidefinite is equivalent to investigating whether the Gram matrices for finite subsets of χ are positive semidefinite. Therefore, without any loss of generality, we may assume that χ is a finite set $\{x_1,\ldots,x_n\}$. Since M_{x_i,x_j} are finite, we may also assume χ'_{x_i} are finite.

We slightly extend the definition of $(\chi', k, \{\gamma_x\})$ by adding a new element $\bullet \in \chi'$ such that $k(\bullet,\bullet) = k(\bullet,x') = k(x',\bullet) = 0$ hold for an arbitrary $x' \in \chi'$. Even after the extension, $(\chi', k, \{\gamma_x\})$ still remains an evaluating system for $\mathcal{M}$.

Next, we define $\bar{\chi}', \bar{M}$ and $\{\bar{\gamma}_x\}$ as follows: $\bar{\chi}'$ is the disjoint union $\bigsqcup_{i=1}^{n} \chi'_{x_i}$; $\bar{x}'$ denotes the image of $x' \in \chi'_x$ in $\bar{\chi}'$; $\bar{M} = \{(\bar{x}',\bar{y}') \mid (x',y') \in M_{x,y} \wedge x,y \in \chi\}$; $\bar{\gamma}_x : \bar{\chi}' \longrightarrow \chi'$ satisfies that $\bar{\gamma}_x(\bar{x}') = \gamma_x(x')$, if $x' \in \chi'_x$, and $\bar{\gamma}_x(\bar{x}') = \bullet$, otherwise. Then, the mapping kernel K with respect to $\mathcal{M}$ and $\mathcal{E}$ is rewritten as follows.

$$K(x,y) = \sum_{(\bar{x}',\bar{y}') \in \bar{M}} k(\bar{\gamma}_x(\bar{x}'), \bar{\gamma}_y(\bar{y}'))$$

Furthermore, $K(x_i,x_j) = \mathrm{tr}(A^\mathsf{T} X^{ij})$ holds, when we define m-dimensional matrices A and X^{ij} for $\bar{\chi}' =$

$\{\bar{x}'_1, \ldots, \bar{x}'_m\}$. $A_{kl} = 1$ if $(\bar{x}'_k, \bar{x}'_l) \in \bar{M}$, and $A_{kl} = 0$ otherwise; $X^{ij}_{kl} = k(\bar{\gamma}_{x_i}(\bar{x}'_k), \bar{\gamma}_{x_j}(\bar{x}'_l))$.

To show the assertion, it suffices to prove A is positive semidefinite by Proposition 1 ($X = [X^{ij}]_{i,j=1,\ldots,n}$ is positive semidefinite by definition). A is symmetric, since $(x', y') \in M_{x,y} \Leftrightarrow (y', x') \in M_{y,x}$ holds. The hypothesis that $\{M_{x,y}\}$ is transitive implies that $\{1, \ldots, m\}$ is decomposed into $U_1 \sqcup \cdots \sqcup U_M$ such that: $U_a \cap U_b = \emptyset$, if $a \neq b$; $(\bar{x}'_k, \bar{x}'_l) \in \bar{M}$, if, and only if, $k, l \in U_a$ for some $a \in \{1, \ldots, M\}$. Therefore, $A = \bigoplus_{a=1}^{M} A[U_a]$ holds, and therefore, A is positive semidefinite, since so are $A[U_a]$.

4.3. (2) Implies (1)

We prove the cotraposition of the assertion. If $\mathcal{M}$ is not transitive, A includes at least one of the following sub-matrices (without any loss of generality, we may assume $k < l < b$), where $A[i_1, \ldots, i_n]$ denote the n-dimensional matrix whose (α, β)-element is A_{i_α, i_β}.

$$A[k, l] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{11}$$

$$A[k, l] = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{12}$$

$$A[k, l, b] = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{13}$$

Note that any of them has a negative eigenvalue, since $\det A < 0$ holds.

We will see that there exists an instance of $\mathcal{E} = (\chi', k, \{\gamma_x\})$ such that $\mathrm{smry}_A(X)$, which is the Gram matrix for χ, is not positive semidefinite, if any of the above three cases occurs. In the remaining of this section, we will give a proof only for the case where Eq. (13) holds. The assertion for the simpler cases, that is, where either Eq. (11) or (12) holds, can be proved in almost the same way.

Let i, j and a denote the indices such that $x'_k \in \chi'_{x_i}$, $x'_l \in \chi'_{x_j}$ and $x'_b \in \chi'_{x_a}$ (be reminded that $\bar{\chi}'$ is defined as the disjoint union of χ'_x for $x \in \chi$). The indices are not necessarily different from each other. Further, let column vectors $\vec{e}_1$, $\vec{e}_2$ and $\vec{e}_3$ be an orthogonal basis of $\mathbb{R}^3$ such that the following holds.

$$[\vec{e}_1, \vec{e}_2, \vec{e}_3]^\mathsf{T} A[k, l, b][\vec{e}_1, \vec{e}_2, \vec{e}_3] = \begin{bmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_3 \end{bmatrix}$$

We assume $\alpha_1 < 0$ without any loss of generality, and define positive semidefinite K as follows.

$$K = [\vec{e}_1, \vec{e}_2, \vec{e}_3] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} [\vec{e}_1, \vec{e}_2, \vec{e}_3]^\mathsf{T}$$

$$\begin{aligned} &\therefore \ \mathrm{tr}(A[k, l, b]^\mathsf{T} K) \\ &= \ \mathrm{tr}\left(\begin{bmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) = \alpha_1 < 0 \end{aligned}$$

Now, we define $\mathcal{E} = (\chi', k, \{\gamma_x\})$ as follows.

- $\chi' = \{\bullet, \xi, \eta, \zeta\}$
- $\begin{bmatrix} k(\xi,\xi) & k(\xi,\eta) & k(\xi,\zeta) \\ k(\eta,\xi) & k(\eta,\eta) & k(\eta,\zeta) \\ k(\zeta,\xi) & k(\zeta,\eta) & k(\zeta,\zeta) \end{bmatrix} = K$
- $\gamma_x(x') = \begin{cases} \xi, & \text{if } x = x_i \text{ and } x' = x'_k, \\ \eta, & \text{if } x = x_j \text{ and } x' = x'_l, \\ \zeta, & \text{if } x = x_a \text{ and } x' = x'_b, \\ \bullet, & \text{otherwise.} \end{cases}$

Below, we investigate three cases: the indices take the same value, that is, $i = j = a$; two of the indices coincide with each other, where we can assume $i = j \neq a$ without loss of generality: the indices are different from one another, that is, $i \neq j \neq a \neq i$. For each case, we see that some diagonally located submatrix of $\mathrm{smry}_A(X)$ is not positive semidefinite. This implies that $\mathrm{smry}_A(X)$ itself is not positive semidefinite.

Case $i = j = a$**:** The submatrix $\mathrm{smry}_A(X)[i]$ is not positive semidefinite.

$$\mathrm{smry}_A(X)[i] = \mathrm{tr}(A[k, l, b]^\mathsf{T} K) < 0$$

Case $i = j \neq a$**:** We will show that $\mathrm{smry}_A(X)[i, k]$ is not positive semidefinite.

$$\begin{aligned} \mathrm{tr}(A^\mathsf{T} X^{ii}) &= \mathrm{tr}(A[k, l, b]^\mathsf{T}[1, 2] K[1, 2]) \\ \mathrm{tr}(A^\mathsf{T} X^{ia}) &= A[k, l, b]^\mathsf{T}{}_{1,3} K_{1,3} + A[k, l, b]^\mathsf{T}{}_{2,3} K_{2,3} \\ \mathrm{tr}(A^\mathsf{T} X^{ai}) &= A[k, l, b]^\mathsf{T}{}_{3,1} K_{3,1} + A[k, l, b]^\mathsf{T}{}_{3,2} K_{3,2} \\ \mathrm{tr}(A^\mathsf{T} X^{aa}) &= A[k, l, b]^\mathsf{T}{}_{3,3} K_{3,3} \end{aligned}$$

$$\therefore \mathrm{tr}\left(\mathrm{smry}_A(X)[i, a] \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right) = \mathrm{tr}(A[k, l, b]^\mathsf{T} K) < 0$$

By Proposition 1 $\mathrm{smry}_A(X)[i, a]$ turns out not to be positive semidefinite.

Case $i \neq j \neq a \neq i$**:** For $\alpha, \beta = 1, 2, 3$, the (α, β)-element of $\mathrm{smry}_A(X)[i, j, a]$ coincides with

$A[k,l,b]^{\mathsf{T}}{}_{\alpha,\beta} K_{\alpha,\beta}$.

$$\begin{aligned} & \mathrm{tr}\left(\mathrm{smry}_A(X)[i,j,a]\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\right) \\ = \;& \mathrm{tr}(A[k,l,b]^{\mathsf{T}} K) < 0 \end{aligned}$$

By Proposition 1, $\mathrm{smry}_A(X)[i,j,a]$ turns out not to be positive semidefinite.

References

Aoki, K. F., Yamaguchi, A., Okuno, Y., Akutsu, T., Ueda, N., Kanehisa, M., & Mamitsuka, H. (2003). Efficient tree-matching methods for accurate carbohydrate database query. *Genome Informatics*, *14*, 134 – 143.

Barnard, D., Clarke, G., & Duncan, N. (1995). *Tree-to-tree correction for document trees* (Technical Report 95-375). Queen's University, Kingston, Ontario K7L 3N6 Canada.

Berry, V., & Nicolas, F. (2004). Maximum Agreement and Compatible Supertrees (Extended Abstract). *CPM* (pp. 205–219).

Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, *18*, 689–694.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001]* (pp. 625–632). MIT Press.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.

Haussler, D. (1999). *Convolution kernels on discrete structures*UCSC-CRL 99-10). Dept. of Computer Science, University of California at Santa Cruz.

Hein, J., Jiang, T., Wang, L., & Zhang, K. (1996). On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, *71*, 153 – 169.

Jansson, J., Ng, J. H. K., Sadakane, K., & Sung, W. K. (2005). Rooted maximum agreement supertrees. *Algorithmica*, 293 – 307.

Kashima, H., & Koyanagi, T. (2002). Kernels for semi-structured data. *the 9th International Conference on Machine Learning (ICML 2002)* (pp. 291–298).

Kuboyama, T., Shin, K., & Kashima, H. (2006). Flexible tree kernels based on counting the number of tree mappings. *Proc. of Machine Learning with Graphs*.

Leslie, C. S., Eskin, E., Cohen, A., Weston, J., & Noble, W. S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, *20*.

Li, H., & Jiang, T. (2005). A class of edit kernels for svms to predict translation initiation sites in eukaryotic mrnas. *Trans. on Comput. Syst. Bio. II*, *LNBI 3680*, 48 – 58.

Lodhi, H., Shawe-Taylor, J., Cristianini, N., & Watkins, C. J. C. H. (2001). Text classificatio using string kernels. *Advances in Neural Information Processing Systems*, *13*.

Lu, C. L., Su, Z.-Y., & Tang, G. Y. (2001). A New Measure of Edit Distance between Labeled Trees. *LNCS* (pp. pp. 338–348). Springer-Verlag Heidelberg.

Menchetti, S., Costa, F., & Frasconi, P. (2005). Weighted decomposition kernel. *Proc. of the 22nd International Conference on Machine Learning*.

Pelillo, M. (2002). Matching free trees, maximal cliques, and monotone game dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*, 1535 – 1541.

Salzberg, S. L. (1997). A method for identifying splice sites and translational staqr sites in eukaryotic mrna. *Computer Applications in the Biosciences*, *13*, 365 – 376.

Suzuki, J., Isozaki, H., & Maeda, E. (2004). Convolution kernels with feature selection for natural language processing tasks. *Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)* (pp. 119–126).

Tai, K. C. (1979). The Tree-to-Tree Correction Problem. *JACM*, *26*, 422–433.

Wagner, R., & Fischer, M. (1974). The string-to-string correction problem. *JACM*, *21*, 168–173.

Zhang, K. (1995). Algorithms for the constrained editing distance between ordered labeled trees and related problems. *PR*, *28*, 463–474.

Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T., & Müller, K. R. (2000). Engineering support vector machne kernels that recognize translation initiation sites. *Bioinformatics*, *16*, 799 – 807.

mStruct: A New Admixture Model for Inference of Population Structure in Light of Both Genetic Admixing and Allele Mutations

Suyash Shringarpure SUYASH@CS.CMU.EDU
Eric P. Xing EPXING@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

Traditional methods for analyzing population structure, such as the *Structure* program, ignore the influence of mutational effects. We propose *mStruct*, an admixture of population-specific mixtures of inheritance models, that addresses the task of structure inference and mutation estimation jointly through a hierarchical Bayesian framework, and a variational algorithm for inference. We validated our method on synthetic data, and used it to analyze the HGDP-CEPH cell line panel of microsatellites used in (Rosenberg et al., 2002) and the HGDP SNP data used in (Conrad et al., 2006). A comparison of the structural maps of world populations estimated by *mStruct* and *Structure* is presented, and we also report potentially interesting mutation patterns in world populations estimated by *mStruct*, which is not possible by *Structure*.

1. Introduction

The deluge of genomic polymorphism data, such as the genome-wide multilocus genotype profiles of variable number of tandem repeats (i.e., microsatellites) and single nucleotide polymorphisms (i.e., SNPs), has fueled the long-standing interest in analyzing patterns of genetic variations.to reconstruct the ancestral structures of modern human populations, because such genetic ancestral information can shed light on the evolutionary history of modern populations and provide guidelines for more accurate association studies and other population genetics problems.

One of the state-of-the-art methods for population structure analysis based on multilocus genotype data is the program *Structure*, whose basic form is based on a statistical formalism known as the admixture model (Pritchard et al., 2000). Admixtures are instances of a more general class of hierarchical Bayesian models known as *mixed membership models* (Erosheva et al., 2004), which postulate that genetic markers of each individual are *iid* (Pritchard et al., 2000) or spatially coupled (Falush et al., 2003) samples from multiple population-specific fixed-dimensional multinomial distributions (which we will call *allele frequency profiles* (Falush et al., 2003), or AP) of marker alleles. Under this assumption, the *admixture* model identifies each ancestral population by a specific AP (that defines a unique allele frequency distribution for each ancestral population for each marker) and displays the fraction of contributions from each AP in a modern individual chromosome as a *structural map*. Figure 1 shows an example of a structural map of four modern populations inferred from a portion of the HapMap multi-population dataset by *Structure*. In this *population structural map*, each individual is represented as a thin vertical line which shows the fraction of the individual's chromosome which originated from each ancestral population, as given by a unique AP. This method has been successfully applied to human genetic data in (Rosenberg et al., 2002) and has unraveled impressive patterns in the genetic structures of world population.

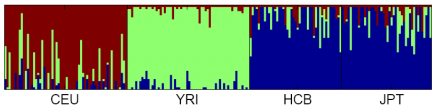


Figure 1. Population structural map inferred by *Structure* on HapMap data consisting of 4 populations. The colors represent different populations

However, since an AP merely represents the *frequency* of alleles in an ancestral population, rather than the actual allelic content or haplotypes of the alleles themselves, the admixture models developed so far based on AP do not model genetic changes due to mutations from the ancestral alleles. Indeed, a serious pitfall of the model underlying *Structure*, as pointed out in (Excoffier & Hamilton, 2003), is that there is no

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

mutation model for modern individual alleles with respect to hypothetical common prototypes in the ancestral populations, i.e, every unique allele in the modern population is assumed to have a distinct ancestral frequency, rather than allowing the possibility of it just being a descendent of some common ancestral allele. Thus, while *Structure* aims to provide ancestry information for each individual and each locus, there is no explicit representation of "ancestors" as a physical set of "founding alleles". Therefore, the inferred population structural map emphasizes revealing the contributions of *abstract* population-specific allele frequency profiles, which does not directly reflect individual diversity or the extent of genetic changes with respect to the founders. Therefore, *Structure* does not enable inference of the founding genetic patterns, the age of the founding alleles, or the population divergence time (Excoffier & Hamilton, 2003).

Another important issue in determining population structure is to look for the presence of admixture, a basic assumption of the *Structure* model. However, as we shall see later, on the HGDP data, it produces results that cluster individuals cleanly into one allele frequency profile or the other, thus leading us to conclude that there was little or no admixture between the human populations. While such a partitioning of individuals would be desirable for clustering them into groups, it does not offer us any biological insight into the intermixing of the populations.

In this paper, we present *mStruct* (for *structure under mutations*), based on an admixture of population-specific mixtures of inheritance model (AdMim). AdMim is an *admixture of mixtures* model, which represents each ancestral population as a mixture of ancestral alleles each with its own inheritance process, and each modern individual as an "ancestry proportion vector" (ancestry vector or *map vector*) that indicates membership proportions among the ancestral populations. By a simple but important extension to the LDA-like (Blei et al., 2003) admixture model used by *Structure*, *mStruct* facilitates estimation of both the *structural map* of populations (incorporating mutations) and the mutation rates of either SNP or microsatellite alleles. A new variational inference algorithm was developed for inference and learning. We compare our method with *Structure* on both synthetic genotype data, and on the microsatellite and SNP genotype data of world populations (Rosenberg et al., 2002; Conrad et al., 2006). Our results show the presence of significant levels of admixture among the founding populations. We also report interesting genetic divergence in world populations revealed by the mutation patterns we estimated.

2. The Statistical Model

2.1. Representation of Populations

To reveal the genetic composition of each modern individual in terms of contributions from hypothetical ancestral populations via statistical inference on multilocus genotype data, one must first choose an appropriate representation of ancestral populations. Below, we begin with a brief description of a commonly used method, followed by a new method that we propose.

2.1.1. Population-Specific *Allele Frequency* Profiles

Due to the polymorphic nature of genetic markers, an intuitive statistic to characterize a population is the frequencies of all observed alleles at all loci. For example, we can represent an ancestral population k by a unique set of population-specific *multinomial* distributions, $\boldsymbol{\beta}^k \equiv \{\vec{\beta}_i^k \ ; \ i = 1 : I\}$, where $\vec{\beta}_i^k = [\beta_{i,1}^k, \ldots, \beta_{i,L_i'}^k]$ is the vector of multinomial parameters, also known as the *allele frequency profile* (Falush et al., 2003), or AP, of the allele distribution at locus i in ancestral population k; L_i' denotes the total number of observed marker alleles at locus i, and I denotes the total number of marker loci. This representation, known as *population-specific ancestry proportion profile*, is used by the program *Structure*.

2.1.2. Population-Specific *Mixtures of Ancestral Alleles*

A problem with the population-specific AP profile representation is that it ignores the possibility of mutations underlying the alleles observed in modern populations with respect to their ancestral alleles. To capture this, we propose to represent a population by a genetically more realistic statistical model known as the *population-specific mixtures of ancestral alleles (MAA)*. For each locus i, an MAA for ancestral population k is a triple $\{\mu_i^k, \delta_i^k, \vec{\beta}_i^k\}$ consisting of a set of *ancestral* (or founder) alleles $\mu_i^k = (\mu_{i,1}^k, \ldots, \mu_{i,L_i}^k)$, which can differ from their descendent alleles in the modern population; a mutation rate δ_i^k associated with the locus, which can be further generalized to be allele-specific if necessary; and an AP $\vec{\beta}_i^k$ which now represents the frequencies of the *ancestral* alleles. Here L_i denotes the total number of ancestral alleles at loci i.

An MAA is strictly more expressive than an AP, because the incorporation of a mutation model helps to capture details about the population structure which an AP cannot; and the MAA reduces to the AP when the mutation rates become zero and the founders are identical to their descendents. As we show shortly, with an MAA, one can examine the mutation parameters corresponding to each ancestral population via

Bayesian inference from genotype data; this might enable us to infer the age of alleles, and also estimate population divergence times.

Let $i \in \{1, \ldots, I\}$ index the position of a locus in the study genome, $n \in \{1, \ldots, N\}$ index an individual in the study population, and $e \in \{0, 1\}$ index the two possible parental origin of an allele (in this study we do not require strict phase information of the two alleles, so the index e is merely used to indicate diploid data). Under an MAA specific to an ancestral population k, the correspondence between a marker allele X_{i,n_e} and a founder $\mu^k_{i,l} \in \mu^k_i$ is not directly observable. For each allele founder $\mu^k_{i,l}$, we associate with it an inheritance model $p(\cdot|\mu^k_{i,l}, \delta^k_{i,l})$ from which descendants can be sampled. Then, given specifications of the ancestral population from which X_{i,n_e} is derived (denoted by hidden indicator variable Z_{i,n_e}), the conditional distribution of X_{i,n_e} under MAA follows a mixture of population-specific inheritance model: $p(x_{i,n_e} = l' \mid Z_{i,n_e} = k) = \sum_{l=1}^{L} \beta^k_{i,l} p(x_{i,n_e}|\mu^k_{i,l}, \delta^k_{i,l})$. Comparing to the counterpart of this function under AP: $p(x_{i,n_e} = l' \mid Z_{i,n_e} = k) = \beta^k_{i,l'}$, we can see that the latter cannot explicitly model allele diversities in terms of molecular evolution from the founders.

2.2. A New Admixture Model for Population Structure

The concept of admixture arises when modeling objects (e.g., human beings) each comprising multiple instances of some attributes (e.g., marker alleles), each of which comes from a (possibly different) source distribution $P_k(\cdot|\Theta_k)$, according to an individual-specific *admixing coefficient vector* (a.k.a. *map vector*) $\vec{\theta}$. The *map vector* represents the normalized contribution from each of the source distributions $\{P_k \; ; \; k = 1 : K\}$ to the study object. For example, for every individual, the alleles at all marker loci may be inherited from founders in different ancestral populations, each represented by a unique distribution of founding alleles and the way they can be inherited. Formally, this scenario can be captured in the following generative process:

1. For each individual n, draw the admixing vector: $\vec{\theta_n} \sim P(\cdot|\alpha)$, where $P(\cdot|\alpha)$ is a pre-chosen map prior.
2. For each marker allele $x_{i,n_e} \in \mathbf{x}_n$
   - 2.1: draw the latent *ancestral-population-origin* indicator $z_{i,n_e} \sim \text{Multinomial}(\cdot|\vec{\theta_n})$;
   - 2.2: draw the allele $x_{i,n_e}|z_{i,n_e} = k \sim P_k(\cdot|\Theta_k)$.

As discussed in the previous section, an ancestral population can be either represented as an AP or as an MAA. These two different representations lead to two different probability distributions for $P_k(\cdot|\Theta_k)$ in the last sampling step above, and thereby two different admixtures of very different characteristics.

2.2.1. The Existing Model

In *Structure*, the ancestral populations are represented by a set of population-specific APs. Thus the distribution $P_k(\cdot|\Theta_k)$ from which an observed allele can be sampled is a multinomial distribution defined by the rates of all observed alleles in the ancestral population, i.e., $x_{i,n_e}|z_{i,n_e} = k \sim \text{Multinomial}(\cdot|\vec{\beta}^k_i)$. Using this probability distribution in the general admixture scheme outlined above, we can see that *Structure* essentially implements an *admixture of population-specific allele rates* model. But a serious pitfall of using such a model, as pointed out in (Excoffier & Hamilton, 2003), is that there is no error model for individual alleles with respect to the common prototypes, i.e, every unique measurement at a particular allele is assumed to be a new allele, rather than allowing the possibility of it just being the mutation of some common ancestral allele at that marker.

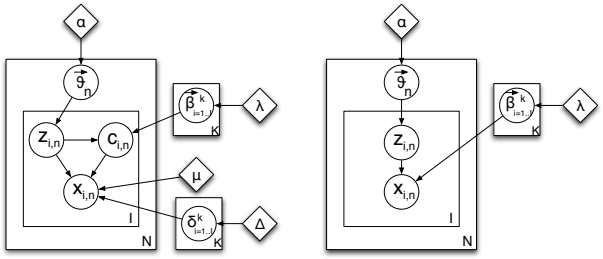
2.2.2. The Proposed Model

We propose to represent each ancestral population by a set of population-specific MAAs. Under this representation, now the distribution $P_k(\cdot|\Theta_k)$ from which an observed allele can be sampled becomes a mixture of inheritance models, each defined on a specific founder. The ensuing sampling module to be plugged into the general admixture scheme outlined above (to replace step 2.2) becomes a two-step generative process:

- 2.2a: draw the latent founder indicator $c_{i,n_e}|z_{i,n_e} = k \sim \text{Multinomial}(\cdot|\vec{\beta}^k_i)$;
- 2.2b: draw the allele $x_{i,n_e}|c_{i,n_e} = l, z_{i,n_e} = k \sim P_m(\cdot|\mu^k_{i,l}, \delta^k_{i,l})$,

where $P_m()$ is a mutation model that can be flexibly defined based on whether the genetic markers are microsatellites or single nucleotide polymorphisms. We call this model an *admixture of population-specific inheritance models* (AdMim), while the previous model is technically only an admixture of population specific allele frequency profiles. Figure 2(a) shows a graphical model the overall generative scheme for AdMim, in comparison with the admixture of population-specific allele rates discussed earlier. From the figure, we can clearly see that *Structure* is virtually identical to an LDA model, while *mStruct* is an extended LDA model which allows noisy observations.

For simplicity of presentation, in the model described above we assume that for a particular individual, the genetic markers at each locus are conditionally *iid* samples from a set of population-specific fixed-dimensional mixture of inheritance models, and that the set of founder alleles at a particular locus is the same for all ancestral populations($\mu^k_i = \mu_i$). Also our model assumes Hardy-Weinberg equilibrium within populations. The simplifying assumptions of



(a) *mStruct* (b) *Structure 2.1*

Figure 2. Graphical Models: the circles represent random variables and diamonds represent hyperparameters.

unlinked loci, no linkage disequilibrium between loci within populations can be easily removed by incorporating Markovian dependencies over ancestral indicators Z_{i,n_e} and Z_{i+1,n_e} of adjacent loci, and over other parameters such as the allele frequencies $\vec{\beta}_i^k$ in exactly the same way as in *Structure*. We can also introduce Markovian dependencies over mutation rates at adjacent loci, which might be desirable to better reflect the dynamics of molecular evolution in the genome. We defer such extensions to a later paper.

2.3. Mutation Model

As described above, our model is applicable to almost all kinds of genetic markers by plugging in an appropriate allele mutation model (i.e., inheritance model) $P_m()$. We now discuss two mutation models, for microsatellites and SNPs, respectively.

2.3.1. Microsatellite Mutation Model

Microsatellites are the repeats of a small sequences in DNA about 1-4 base pairs in length which are usually represented as integer counts. The choice of a suitable microsatellite mutation model is important, for both computational and interpretation purposes. Below we discuss the mutation model that we use and the biological interpretation of the parameters of the mutation model. We begin with a stepwise mutation model for microsatellites widely used in forensic analysis (Valdes et al., 1993; Lin et al., 2006).

This model defines a conditional distribution of a progeny allele b given its progenitor allele a, both of which take continuous values:

$$p(b|a) = \frac{1}{2}\xi(1-\delta)\delta^{|b-a|-1}, \qquad (1)$$

where ξ is the mutation rate (probability of any mutation), and δ is the factor by which mutation decreases as distance between the two alleles increases. Although this mutation distribution is not stationary (i.e. it does not ensure allele frequencies to be constant over the generations), it is simple and commonly used in forensic inference. To some degree δ can be regarded as a parameter that controls the probability of unit-distance mutation, as can be seen from the following identity: $p(b+1|a)/p(b|a) = \delta$.

In practice, the two-parameter stepwise continuous mutation model described above complicates the inference process. We propose a discrete microsatellite mutation model that is a simplification of Eq. 1, but captures its main idea. We posit that: $P(b|a) \propto \delta^{|b-a|}$. It is not hard to show that normalizing this probability mass function gives us the mutation model as:

$$P(b|a) = \frac{1-\delta}{1-\delta^a+\delta}\delta^{|b-a|}. \qquad (2)$$

We can interpret δ as a variance parameter, the factor by which probability drops as a fuction of the distance between the mutated version b of the allele a.

Determination of founder set at each locus: According to our model assumptions, there can be a different number of founder alleles at each locus. This number is typically smaller than the number of alleles observed at each marker since the founder alleles are "ancestral". To estimate the appropriate number and allele states of founders, we fit finite mixtures of microsatellite mutation models, and use the Bayesian Information Criterion (BIC) to determine the cardinality of the mixture.

Choice of mutation prior: In our model, the δ parameter, as explained above, is a population-specific parameter that controls the probability of stepwise mutations. Being a parameter that controls the variance of the mutation distribution, there is a possibility that inference on the model will encourage higher values of δ to improve the log-likelihood, in the absence of any prior distribution on δ. To avoid this situation, and to allow more meaningful and realistic results to emerge from the inference process, we impose on δ a beta prior that will be biased towards smaller values of δ. The beta prior will be a fixed one and will not be among the parameters we estimate.

2.3.2. SNP Mutation Model

SNPs, or single nucleotide polymorphisms, represent the largest class of individual differences in DNA. In general, there is a well-defined correlation between the age of the mutation producing a SNP allele and the frequency of the allele. For SNPs, we use a simple pointwise mutation model, rather than more complex block models. Thus, the observations in SNP data are only binary in nature (0/1). So, given the observed allele b, we say that the probability of it being derived from the founder allele a is given by:

$$P(b|a) = \delta^{\mathcal{I}[b=a]} \times (1-\delta)^{\mathcal{I}[b\neq a]}; \quad a,b \in \{0,1\}. \qquad (3)$$

In this case, the mutation parameter δ is the probability that the observed allele is not identical to the

founder allele, but derived from it due to a mutation.

2.4. Inference and Parameter Estimation

2.4.1. Probability Distribution on the Model

For notational convenience, we will ignore the diploid nature of observations in the analysis that follows. With the understanding that the analysis is carried out for the n^{th} individual, we will drop the subscript n. Also, we overload the indicator variables z_i and c_i to be both, arrays with only one element equal to 1 , as well as scalars with a value equal to the index at which the array forms have 1s. In other words: $z_i \in 1,\ldots,K, c_i \in 1,\ldots,L, z_{i,k} = \mathcal{I}[z_i = k]$, and $c_{i,l} = \mathcal{I}[c_i = l]$.

The joint probability distribution of the the data and the relevant variables under the AdMim model can then be written as:

$$p\left(\mathbf{x},\mathbf{z},\mathbf{c},\vec{\theta}|\alpha,\boldsymbol{\beta},\mu,\delta\right) = p\left(\vec{\theta}|\alpha\right)\prod_{i=1}^{I} p\left(z_i|\vec{\theta}\right) p\left(c_i|z_i,\vec{\beta}_i^{k=1\cdot K}\right).$$

The marginal likelihood of the data can be computed by summing/integrating out the latent variables. However, a closed-form solution to this summation/integration is not possible, and indeed exact inference on hidden variables such as the map vector $\vec{\theta}$, and estimation of model parameters such as the mutation rates δ under AdMim is intractable.(Pritchard et al., 2000) developed an MCMC algorithms for approximate inference for their admixture model underlying *Structure*. We choose to apply a computationally more efficient approximate inference method known as variational inference (Jordan et al., 1999).

2.5. Variational Inference

We use a mean-field approximation for performing inference on the model. This approximation method approximates an intractable joint posterior $p()$ of the all hidden variables in the model by a product of marginal distributions $q() = \prod q_i()$, each over only a single hidden variable. The optimal parameterization of $q_i()$ for each variable is obtained by minimizing the Kullback-Leibler divergence between the variational approximation q and the true joint posterior p. Using results from the the Generalised Mean Field theory (Xing et al., 2003), we can write the variational distributions of the latent variables as follows:

$$q(\vec{\theta}) \propto \prod_{k=1}^{K} \theta_k^{\alpha_k - 1 + \sum_{i=1}^{I}\langle z_{i,k}\rangle}$$

$$q(c_i) \propto \prod_{l=1}^{L}\left(\prod_{k=1}^{K}\left(\beta_{i,l}^k f(x_i|\mu_{i,l},\delta_i^k)\right)^{\langle z_{i,k}\rangle}\right)^{c_{i,l}}$$

$$q(z_i) \propto \prod_{k=1}^{K}\left(e^{\langle log(\theta_k)\rangle}\left(\prod_{l=1}^{L}\beta_{i,l}^k f(x_i|\mu_{i,l},\delta_i^k)^{\langle c_{i,l}\rangle}\right)\right)^{z_{i,k}}.$$

In the distributions above, the '$\langle\rangle$' are used to indicate the expected values of the enclosed random variables. A close inspection of the above formulae reveals that these variational distributions have the form $q(\vec{\theta}) \sim \text{Dirichlet}(\gamma_1,\ldots,\gamma_K)$, $q(z_i) \sim \text{Multinomial}(\rho_{i,1},\ldots,\rho_{i,K})$, and $q(c_i) \sim \text{Multinomial}(\xi_{i,1},\ldots,\xi_{i,L})$, respectively, where the parameters $\gamma_k, \rho_{i,k}$ and $\xi_{i,l}$ are given by the following equations:

$$\gamma_k = \alpha_k + \sum_{i=1}^{I}\langle z_{i,k}\rangle$$

$$\rho_{i,k} = \frac{e^{\langle log(\theta_k)\rangle}\left(\prod_{l=1}^{L}\beta_{i,l}^k f(x_i|\mu_{i,l},\delta_i^k)^{\langle c_{i,l}\rangle}\right)}{\sum_{k=1}^{K}\left(e^{\langle log(\theta_k)\rangle}\left(\prod_{l=1}^{L}\beta_{i,l}^k f(x_i|\mu_{i,l},\delta_i^k)^{\langle c_{i,l}\rangle}\right)\right)}$$

$$\xi_{i,k} = \frac{\prod_{k=1}^{K}\left(\beta_{i,l}^k f(x_i|\mu_{i,l},\delta_i^k)\right)^{\langle z_{i,k}\rangle}}{\sum_{k=1}^{K}\left(\prod_{k=1}^{K}\left(\beta_{i,l}^k f(x_i|\mu_{i,l},\delta_i^k)\right)^{\langle z_{i,k}\rangle}\right)}$$

and they have the properties: $\langle log(\theta_k)\rangle = \gamma_k$, $\langle z_{i,k}\rangle = \rho_{i,k}$ and $\langle c_{i,l}\rangle = \xi_{i,l}$, which suggest that they can be computed via fixed point iterations. It can be shown that this iteration will converge to a local optimum, similar to what happens in an EM algorithm. Empirically, a near global optimal can be obtained by multiple random restarts of the fixed point iteration. Typically, such a mean-field variational inference converges much faster then sampling (Xing et al., 2003).

3. Hyperparameter Estimation

The parameters of our model, i.e., $\{\mu,\delta,\boldsymbol{\beta}\}$, and the Dirichlet hyperparameter α, can be estimated by maximizing the lower bound on the log-likelihood as a function of the current values of the hyperparameters, via a variational EM algorithm. Due to space limits, details of this empirical Bayes estimation scheme are available in the extended version.

4. Experiments and Results

We validated our model on a synthetic microsatellite dataset where the simulated values of the hidden map vector θ_n of each individual and the population parameters $\{\mu^k,\delta^k,\vec{\beta}^k\}$ of each ancestral population are known as ground truth. The goal is to assess the performance of *mStruct* in terms of accuracy and consistency of the estimated map vectors and population parameters, and test of the correctness of the inference and estimation algorithms we developed. We also conduct empirical analysis using *mStruct* of two real datasets: the HGDP-CEPH cell line panel of microsatellite loci and the HGDP SNP data, in comparison with the *Structure* program (version 2.1).

4.1. Validations on Synthetic Data

We simulated 20 microsatellite genotype datasets using the AdMim generative process described in section 2.2, with 100 diploid individuals from 2 ances-

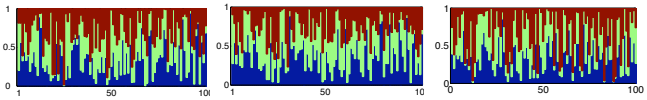


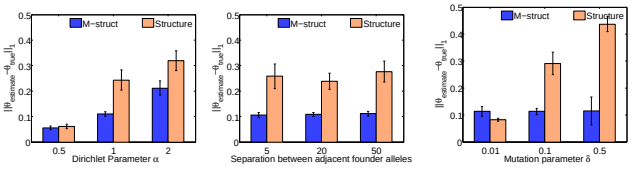
Figure 3. Ancestry spectra for a 3-population simulated dataset. First panel shows the true ancestry proportion vectors. Middle panel shows the estimate by *mStruct*. Right panel shows the estimate from *Structure* .

tral populations, at 50 genotype loci. Each locus has 4 founding alleles, separated by adjustable distances; the mutation parameter at each locus for both populations had default value 0.1, but can be varied to simulate different degrees of divergence. The founding allele frequencies, $\vec{\beta}_i^k$, were drawn from a flat Dirichlet prior with parameter 1. The map vectors θ_n were sampled from a symmetric beta distribution with parameter α, allowing different levels of admixing. We examine the accuracies of several estimates of interest under a number of different simulation conditions, and for each condition we report the statistics of the accuracies across 20 iid synthetic datasets. Due to space limitations, we only report two experiments below; the additional results on accuracy of ancestral alleles recovery and the ancestral-allele frequency estimation are available in the full paper.

4.1.1. Accuracy of Population Map Estimate

The map vector θ_n reflects the proportions of contributions from different ancestral population to the maker-alleles of each individual. The display of the map vectors of all individuals in a study population gives a *Map* of population structure (see, e.g., Fig. 1 in the introduction), which has been the main output of the *Structure* program . We compare the accuracy of the estimated θ_n w.r.t. the ground truth recorded during the simulation in terms of their L1 distances.

Figure 3 shows an example of this comparison, and we can see that *mStruct* is visually more accurate than *Structure*. Figure 4 shows the accuracy of the Map estimate by *mStruct* on synthetic datasets simulated with different properties, in comparison with that of *Structure*. Fig. 4(a) shows that, under different degrees of biases of population admixing induced by the Beta prior of θ_n, *mStruct* consistently outperforms *Structure*. Specifically, as the value of the Beta prior hyperparameter α increases, fewer individuals tend to belong completely to only one population, and more and more individuals become highly admixed. As the figure shows, the performance of both methods degrades as we progress toward this end; however, the severity of degradation of *mStruct* is much less than that of *Structure*. *mStruct* remains robust and performs better than *Structure* as the separations between founding alleles decreases (Fig.4(b)), which tends to increasingly confound the ancestral origins of modern alleles. Finally, Fig. 4(c) shows how the presence of mutations affects the performance of both methods. At very low values of the mutation parameters, the performances of both models are comparable; but as the mutation parameter increases in magnitude, the performance of *Structure* degrades significantly. On the other hand, the decrease in accuracy for *mStruct* is hardly noticeable. This shows that our model is resistant to the confounding effect of large mutations.



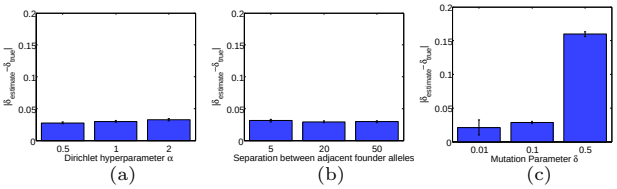
(a) (b) (c)

Figure 4. Accuracy of θ est. under different conditions.

4.1.2. Accuracy of Parameter Estimation.

An important aspect of guarantee and utility we desire for our model and inference algorithm is that it should offer consistent estimates of the population parameters $\{\mu^k, \delta^k, \vec{\beta}^k\}$ underlying the composition of the ancestral population and their inheritance processes. These estimates offer important insight of the evolutionary history and dynamics of modern population genotype data. We have extensively investigated the robustness and accuracy of all these estimates. Due to space limitations, here we only report highlights of mutation rate estimation.

Mutation parameter estimation: We evaluate the performance at recovery of δ^k's by a simple distance measure, (L1 distance measure), between the true and inferred values. We expect that using the beta prior described earlier improves the recovery of the population-specific mutation parameters. As shown in Figure 5, the estimates of δ^k's are robust and remain low-bias under different degree of admixing (due to changing α) and different ancestor dispersion (due to changing distances among the μ^k's). The accuracy decreases as the value of the mutation parameter itself increases, but remains respectable, as shown in Figure. 5(c).



(a) (b) (c)

Figure 5. Accuracy of microsatellite mutation para. est.

4.2. Empirical Analysis of Real Datasets

The HGDP-CEPH cell line panel (Cann et al., 2002; Cavalli-Sforza, 2005) used in (Rosenberg et al., 2002)contains genotype information from 1056 indvid-

uals from 52 populations at 377 autosomal microsatellite loci, along with geographical and population labels. The HGDP SNP data (Conrad et al., 2006) contains the SNPs genotypes at 2834 loci of 927 unrelated individuals that overlap with the HGDP-CEPH data. To make results for both types of data comparable, we chose the set of only those individuals present in both datasets. As in (Rosenberg et al., 2002), the choice of the total number of ancestral populations is left to the user, and here we only show results of $K = 4$ due to space limitations.

4.2.1. Structural Maps from HGDP Data

We compare the structural maps inferred from both the microsatellite and the SNP data using *mStruct* and *Structure* (top panels in Figure 6). The structural maps produced by both programs are quite similar in the case of SNPs, but are very different for microsatellites. The most obvious difference between the maps produced by both programs is the degree of admixing that the individuals in the program are assigned. *Structure* assigns each geographical population to a distinct profile. Thus, it seems to predict very little admixing effect in modern human populations. While useful for clustering, this might result in loss of potentially useful information about actual evolutionary history of populations. In contrast, the structure map produced by *mStruct* for microsatellites suggests that all populations share a common ancestral population with a unique extra component that characterizes their particular genotypes. It is interesting to note that clustering individuals by the ancestry proportion vectors due to *mStruct* will produce exactly the same clustering partitions as that due to *Structure*. The structural maps produced in the case of SNP data are quite similar for both softwares, with results from *mStruct* again predicting more admixture than *Structure*. It is also interesting to see that the ancestry proportions for European and Middle Eastern regions are more distinct from each other in *mStruct* than in *Structure*, allowing for better separation of the two geographical regions. A possible cause for the inconsistency between the results produced by *mStruct* for SNP data and microsatellite data could be the large difference between their mutation rates, or due to the choice of a simplistic SNP mutation model.This issue will be explored in more detail in the full version of the paper.

4.2.2. Analysis of the Mutation Spectrums

Now we report a preliminary analysis of the evolutionary dynamics reflected by the estimated mutation spectrums of different ancestral populations (denoted "am-spectrum"), and of different modern geographical populations (denoted "gm-spectrum"), which is not possible by *Structure*. For the am-spectrum, we compute the mean mutation rates over all loci and founding alleles for each ancestral population as estimated by *mStruct*. We estimate the gm-spectrum as follows: for every individual, a mutation rate is computed as the per-locus number of observed alleles that are attributed to mutations, weighted by the mutation rate corresponding to the ancestral allele chosen for that locus. This can be computed by observing the population-indicator (Z) and the allele-indicator (C) for each individual. We then compute the population mutation rates by averaging mutation rates of all individuals having the same geographical label.

As shown in the gm-spectrums in Figure 6 (lower subpanels on the right), the mutation rates for African populations are indeed higher than those of other modern populations. This indicates that they diverged earlier, a common hypothesis of human migration. Other trends in the gm-spectrums also reveal interesting insights, which we do not have space to discuss. The am-spectrums of SNP data in Figure 6 suggest that the founder ancestral population that dominates modern African populations has a higher mutation rate than the other ancestral population, indicating that is the older of the two ancestral populations. The mutation estimates are largely consistent for both microsatellites and SNPs in comparative order, but vastly different in numerical values.

4.3. Model Selection

As with all probabilistic models, we face a tradeoff between model complexity and the log-likelihood value that the model achieves. In our case, complexity is controlled by the number of ancestral populations we pick, K.

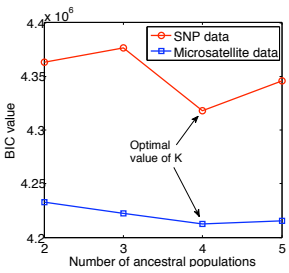
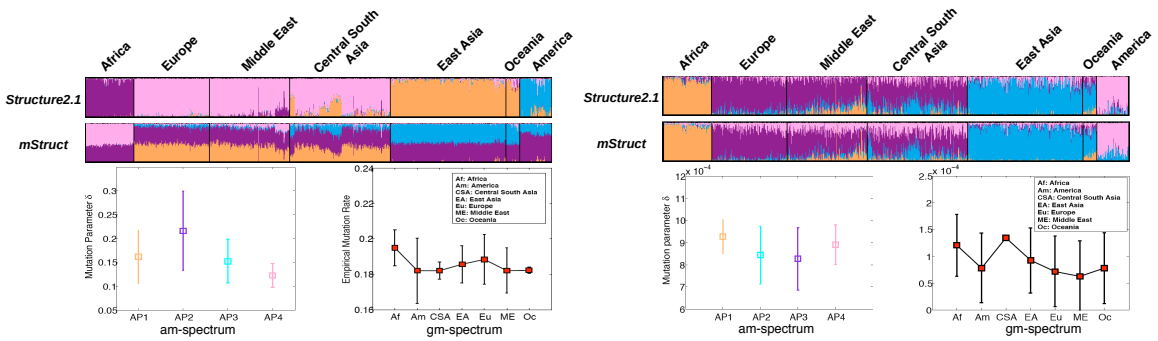


Figure 7. BIC scores for K=2 to 5.

Unlike non-parametric or infinite dimensional models (e.g., Dirichlet processes etc.), for models of fixed dimension, it is not clear in general as to what value of K gives us the best balance between model complexity and log-likelihood. In such cases, different information criteria are often used to determine the optimal model complexity. To determine what number of ancestral populations fit the HGDP SNP and microsatellite data best, we computed BIC scores for K=2 to K=5 for both kinds of data separately. The results are shown in Figure 7. The BIC curves for both SNPs and microsatellites suggest K=4 as the best fit for the data.



(a) from Microsatellite data (b) from SNP data

Figure 6. Structural maps, mutation spectrums, from the HGDP data via *mStruct* and *Structure*.

5. Discussions

We have developed *mStruct*, which allows estimation of genetic contributions of ancestral populations in each modern individual in light of both population admixture and allele mutation. The variational inference algorithm that we developed allows tractable approximate inference on the model. The ancestral proportions of each individual enable representing population structure in a way that is both visually easy to interpret, as well as amenable to further computational analysis. In conjunction with geographical location, the inferred ancestry proportions could be used to detect migrations,sub-populations etc quite easily. Moreover, the ability to estimate population and locus specific mutation rates also allows us to substantiate evolutionary dynamics claims based on high/low mutation rates in certain geographical population, or on high/low mutation rates at certain loci in the genome. While the estimates of mutation rates that *mStruct* provides are not on an absolute scale, the comparison of their relative magnitudes is certainly informative. As of now, there remain a number of possible extensions to the methodology we presented so far. It would be instructive to see the impact of allowing linked loci as in (Falush et al., 2003). We have not yet addressed the issue of the most suitable choice of mutation process, but instead have chosen one that is reasonable and computationally tractable. It would be interesting to combine *mStruct* with the nonparametric Bayesian models based on the Dirichlet processes such as (Sohn & Xing, 2007). Our model might be described as a noisy-channel version of the LDA model, where the observations are modified instances of the original alleles. It is not hard to imagine applications of this model to other tasks such as image modeling or IR tasks involving noisy data, with minor changes in the distributions from which the observations are sampled.

Acknowledgements

This research was supported in part by NSF Grant CCF-0523757 and DBI-0546594.

References

Blei, D., Ng, A., & Jordan, M. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Cann, H., de Toma, C., Cazes, L., Legrand, M., Morel, V., Piouffre, L., Bodmer, J., Bodmer, W., Bonne-Tamir, B., Cambon-Thomsen, A., et al. (2002). A Human Genome Diversity Cell Line Panel. *Science*, *296*, 261–262.

Cavalli-Sforza, L. (2005). The Human Genome Diversity Project: past, present and future. *Nat Rev Genet*, *6*, 333–40.

Conrad, D., Jakobsson, M., Coop, G., Wen, X., Wall, J., Rosenberg, N., & Pritchard, J. (2006). A worldwide survey of haplotype variation and linkage disequilibrium in the human genome. *Nat Genet*, *38*, 1251–1260.

Erosheva, E., Fienberg, S., & Lafferty, J. (2004). Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences*, *101*, 5220–5227.

Excoffier, L., & Hamilton, G. (2003). Comment on Genetic Structure of Human Populations. *Science*, *300*, 1877–1877.

Falush, D., Stephens, M., & Pritchard, J. (2003). Inference of Population Structure Using Multilocus Genotype Data Linked Loci and Correlated Allele Frequencies. *Genetics*, *164*, 1567–1587.

Jordan, M., Ghahramani, Z., Jaakkola, T., & Saul, L. (1999). An Introduction to Variational Methods for Graphical Models. *Machine Learning*, *37*, 183–233.

Lin, T., Myers, E., & Xing, E. (2006). Interpreting anonymous DNA samples from mass disasters–probabilistic forensic inference using genetic markers. *Bioinformatics*, *22*, e298.

Pritchard, J., Stephens, M., & Donnelly, P. (2000). Inference of Population Structure Using Multilocus Genotype Data. *Genetics*, *155*, 945–959.

Rosenberg, N., Pritchard, J., Weber, J., Cann, H., Kidd, K., Zhivotovsky, L., & Feldman, M. (2002). Genetic Structure of Human Populations. *Science*, *298*, 2381–2385.

Sohn, K., & Xing, E. (2007). Spectrum: joint bayesian inference of population structure and recombination events. *Bioinformatics*, *23*, i479.

Valdes, A., Slatkin, M., & Freimer, N. (1993). Allele Frequencies at Microsatellite Loci: The Stepwise Mutation Model Revisited. *Genetics*, *133*, 737–749.

Xing, E., Jordan, M., & Russell, S. (2003). A generalized mean field algorithm for variational inference in exponential families. *Uncertainty in Artificial Intelligence (UAI2003). Morgan Kaufmann Publishers.*

Expectation-Maximization for Sparse and Non-Negative PCA

Christian D. Sigg CHRSIGG@INF.ETHZ.CH
Joachim M. Buhmann JBUHMANN@INF.ETHZ.CH
Institute of Computational Science, ETH Zurich, 8092 Zurich, Switzerland

Abstract

We study the problem of finding the dominant eigenvector of the sample covariance matrix, under additional constraints on the vector: a cardinality constraint limits the number of non-zero elements, and non-negativity forces the elements to have equal sign. This problem is known as sparse and non-negative principal component analysis (PCA), and has many applications including dimensionality reduction and feature selection. Based on expectation-maximization for probabilistic PCA, we present an algorithm for any combination of these constraints. Its complexity is at most quadratic in the number of dimensions of the data. We demonstrate significant improvements in performance and computational efficiency compared to other constrained PCA algorithms, on large data sets from biology and computer vision. Finally, we show the usefulness of non-negative sparse PCA for unsupervised feature selection in a gene clustering task.

1. Introduction

Principal component analysis (PCA) provides a lower dimensional approximation of high dimensional data, where the reconstruction error (measured by Euclidean distance) is minimal. The first principal component (PC) is the solution to

$$\arg\max_{\mathbf{w}} \mathbf{w}^\top \mathbf{C}\mathbf{w}, \text{ subject to } \|\mathbf{w}\|_2 = 1, \qquad (1)$$

where $\mathbf{C} \in \mathbb{R}^{D\times D}$ is the positive semi-definite covariance matrix of the data. It is straightforward to show that the first PC is the dominant eigenvector of $\mathbf{C}$, i.e. the eigenvector corresponding to the largest eigenvalue. The first PC maximizes the variance of the projected data, while the second PC again maximizes the variance, under the constraint that it is orthogonal to the first, and so on.

Constrained PCA and its Applications. We consider problem (1) under two additional constraints on $\mathbf{w}$: Sparsity $\|\mathbf{w}\|_0 \leq K$[1] and non-negativity $\mathbf{w} \succeq \mathbf{0}$. Constraining PCA permits a trade-off between maximizing *statistical fidelity* on the one hand, and facilitating *interpretability* and *applicability* on the other (d'Aspremont et al., 2007). Although it is often the case that PCA provides a good approximation with few PCs, each component is usually a linear combination of all original features. Enforcing *sparsity* facilitates identification of the relevant influence factors and is therefore an unsupervised feature selection method. In applications where a fixed penalty is associated with each included dimension (e.g. transaction costs in finance), a small loss in variance for a large reduction in cardinality can lead to an overall better solution. Enforcing *non-negativity* renders PCA applicable to domains where only positive influence of features is deemed appropriate (e.g. due to the underlying physical process). Moreover, the total variance is explained *additively* by each component, instead of the mixed sign structure of unconstrained PCA. Often non-negative solutions already show some degree of sparsity, but a combination of both constraints enables precise control of the cardinality. Sparse PCA has been successfully applied to gene ranking (d'Aspremont et al., 2007), and non-negative sparse PCA has been compared favorably to non-negative matrix factorization for image parts extraction (Zass & Shashua, 2006).

Related Work. Problem (1) is a *concave programming* problem, and is NP-hard if either sparsity or non-negativity is enforced (Horst et al., 2000). Although an efficient global optimizer is therefore unlikely, local optimizers often find good or even optimal solutions in practice, and global optimality can be tested

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] See final paragraph of this section for a definition of our notation.

in $O(D^3)$ (d'Aspremont et al., 2007), where D is the dimensionality of the data. As is evident from writing the objective function of (1) as

$$\mathbf{w}^\top \mathbf{C}\mathbf{w} = \sum_{i=1}^{D}\sum_{j=1}^{D} C_{ij} w_i w_j, \tag{2}$$

setting w_k to zero excludes the k-th column and row of $\mathbf{C}$ from the summation. For a *given* sparsity pattern $\mathcal{S} = \{i|w_i \neq 0\}$, the optimal solution is the dominant eigenvector of the corresponding submatrix of $\mathbf{C}$. For sparse PCA, the computationally hard part is therefore to identify the optimal sparsity pattern, and any solution can potentially be improved by keeping $\mathcal{S}$ only and recomputing the weights, a process called *variational renormalization* by Moghaddam et al. (2006).

Sparse PCA methods can be characterized by the following two paradigms:

1. Relaxation of the hard cardinality constraint $\|\mathbf{w}\|_0 \leq K$ into a convex constraint $\|\mathbf{w}\|_1 \leq B$, thus approximating the combinatorial problem by continuous optimization of (1) on a convex feasible region.

2. Direct combinatorial optimization of $\mathcal{S}$. Due to the potentially exponential runtime of exact methods, heuristics such as greedy search have to be employed for large values of D.

Cadima and Jolliffe (1995) proposed thresholding the $(D-K)$ smallest elements of the dominant eigenvector to zero, which has complexity $O(D^2)$. Better results have been achieved by the SPCA algorithm of Zou et al. (2004), which is based on iterative elastic net regression. Combinatorial optimization was introduced by Moghaddam et al. (2006), who derived an exact branch-and-bound method and a greedy algorithm, that computes the full sparsity path $1 \leq K \leq D$ in $O(D^4)$. Based on a semi-definite relaxation of the sparse PCA problem, d'Aspremont et al. (2007) proposed PathSPCA, which reduces the complexity of each greedy step to $O(D^2)$, and renders computation of the full regularization path possible in $O(D^3)$. Finally, Sriperumbudur et al. (2007) formulate sparse PCA as a d.c. program (Horst et al., 2000) and provide an iterative algorithm called DC-PCA, where each iteration consists of solving a quadratically constrained QP with complexity $O(D^3)$.

Non-negative (sparse) PCA was proposed by Zass and Shashua (2006). In contrast to the methods discussed so far, their algorithm (called NSPCA) optimizes the cumulative variance of L components jointly, versus a sequential approach that computes one component after another. Orthonormality of the components is enforced by a penalty in the objective function (see section 4 for a discussion about orthogonality for non-negative components), and the desired sparsity is again expressed in terms of the whole set of L components.

Our Contribution. To our knowledge, there is no algorithm either for sparse or non-negative sparse PCA that achieves competitive results in less than $O(D^3)$. In this paper, we propose an $O(D^2)$ algorithm that enforces sparsity, or non-negativity or both constraints simultaneously in the same framework, which is rooted in expectation-maximization for a probabilistic generative model of PCA (see next section). As for the combinatorial algorithms, the desired cardinality can be expressed directly as $K = |\mathcal{S}|$, instead of a bound B on the l_1 norm of $\mathbf{w}$ (which requires searching for the appropriate value). Although computing the full regularization path is also of order $O(D^3)$, our method *directly* computes a solution for any K in $O(D^2)$, in contrast to forward greedy search which needs to build up a solution incrementally. As is the case with SPCA, our method works on the data matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$ (N is the number of samples), instead of the covariance matrix $\mathbf{C}$. To summarize, the low complexity combined with an efficient treatment of the $D \gg N$ case enables an application of our method to large data sets of high dimensionality.

Notation. Vectors are indexed as $\mathbf{w}_{(t)}$, and elements of vectors as w_i. $\|\mathbf{w}\|_1 = \sum_i |w_i|$ and $\|\mathbf{w}\|_0 = |\mathcal{S}|$, where $\mathcal{S} = \{i|w_i \neq 0\}$. $\|\mathbf{w}\|_0$ is also called the *cardinality* of $\mathbf{w}$. $\mathbf{I}$ is the identity matrix, $\mathbf{0}$ a vector of zero elements, and $\mathbf{w} \succeq \mathbf{0} \Leftrightarrow \forall i : w_i \geq 0$. $\mathbf{x} \circ \mathbf{y}$ denotes element-wise multiplication of $\mathbf{x}$ and $\mathbf{y}$, and $\mathrm{tr}(\mathbf{X}) = \sum_i X_{ii}$ is the trace of matrix $\mathbf{X}$. $\mathbb{E}[.]$ is the expectation operator, and $\mathcal{N}$ denotes a Gaussian distribution.

2. EM for Probabilistic PCA

Tipping and Bishop (1999) and independently Roweis (1998) proposed a generative model for PCA, where the full covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{D\times D}$ of the Gaussian distribution is approximated by its first L eigenvectors (in terms of magnitude of the respective eigenvalues). The latent variable $\mathbf{y} \in \mathbb{R}^L$ (in the principal component subspace) is distributed according to a zero mean, unit covariance Gaussian

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{3}$$

The observation $\mathbf{x} \in \mathbb{R}^D$, conditioned on the value of the latent variable $\mathbf{y}$, is linear-Gaussian distributed

according to

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{W}\mathbf{y} + \boldsymbol{\mu}, \sigma^2\mathbf{I}), \tag{4}$$

where the matrix $\mathbf{W} \in \mathbb{R}^{D\times L}$ spans the principal subspace, and $\boldsymbol{\mu} \in \mathbb{R}^D$ is the mean of the data. To simplify the presentation, we will assume centered data from now on.

The EM equations for probabilistic PCA have the following form. The E-step keeps track of

$$\mathbb{E}[\mathbf{y}_{(n)}] = \mathbf{M}_{(t)}^{-1}\mathbf{W}_{(t)}^\top\mathbf{x}_{(n)} \tag{5}$$

$$\mathbb{E}[\mathbf{y}_{(n)}\mathbf{y}_{(n)}^\top] = \sigma_t^2\mathbf{M}_{(t)}^{-1} + \mathbb{E}[\mathbf{y}_{(n)}]\mathbb{E}[\mathbf{y}_{(n)}]^\top, \tag{6}$$

where $\mathbf{M} \in \mathbb{R}^{L\times L}$ is defined as

$$\mathbf{M} = \mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I}. \tag{7}$$

The M-step equations are

$$\mathbf{W}_{(t+1)} = \left[\sum_{n=1}^N \mathbf{x}_{(n)}\mathbb{E}[\mathbf{y}_{(n)}]^\top\right]\left[\sum_{n=1}^N \mathbb{E}[\mathbf{y}_{(n)}\mathbf{y}_{(n)}^\top]\right]^{-1} \tag{8}$$

$$\sigma_{t+1}^2 = \frac{1}{ND}\sum_{n=1}^N \Big[\|\mathbf{x}_{(n)}\|_2^2 - 2\mathbb{E}[\mathbf{y}_{(n)}]^\top\mathbf{W}_{(t+1)}^\top\mathbf{x}_{(n)} + \operatorname{tr}\Big(\mathbb{E}[\mathbf{y}_{(n)}\mathbf{y}_{(n)}^\top]\mathbf{W}_{(t+1)}^\top\mathbf{W}_{(t+1)}\Big)\Big]. \tag{9}$$

In order to efficiently incorporate constraints into the EM algorithm (see next section), we make three simplifications: take the limit $\sigma^2 \to 0$, consider a one-dimensional subspace and normalize $\|\mathbf{w}_{(t)}\|_2$ to unity. The first simplification reduces probabilistic PCA to standard PCA. Computing several components will be treated in section 4, and the unity constraint on $\|\mathbf{w}_{(t)}\|_2$ is easily restored after each EM iteration. The E-step now amounts to

$$\mathbb{E}[y_n] = \mathbf{w}_{(t)}^\top\mathbf{x}_{(n)}, \tag{10}$$

and the M-step is

$$\mathbf{w}_{(t+1)} = \frac{\sum_{n=1}^N \mathbf{x}_{(n)}\mathbb{E}[y_n]}{\sum_{n=1}^N \mathbb{E}[y_n]^2}. \tag{11}$$

These two equations have the following interpretation (Roweis, 1998): The E-step orthogonally projects the data onto the current estimate of the subspace, while the M-step re-estimates the projection to minimize squared reconstruction error for fixed subspace coordinates. We summarize this result in algorithm 1, which iteratively computes the solution to eq. (1). Due to the fact that so far only $\|\mathbf{w}\|_2 = 1$ is enforced, convergence to the global optimum doesn't depend on the initial estimate $\mathbf{w}_{(1)}$. This will no longer be the case for additional constraints.

Algorithm 1 Iterative Computation of First PC

Input: Data $\mathbf{X} \in \mathbb{R}^{N\times D}$, initial estimate $\mathbf{w}_{(1)}$, ε
Algorithm:
$t \leftarrow 1$
repeat
  $\mathbf{y} = \mathbf{X}\mathbf{w}_{(t)}$
  $\mathbf{w}_{(t+1)} = \arg\min_{\mathbf{w}} \sum_{n=1}^N \|\mathbf{x}_{(n)} - y_n\mathbf{w}\|_2^2$
  $\mathbf{w}_{(t+1)} \leftarrow \mathbf{w}_{(t+1)}/\|\mathbf{w}_{(t+1)}\|_2$
  $t \leftarrow t + 1$
until $|\mathbf{w}_{(t+1)}^\top\mathbf{w}_{(t)}| > 1 - \varepsilon$
Output: $\mathbf{w}$

3. Constrained PCA

Consider the minimization step in algorithm 1, which can be written as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} J(\mathbf{w}) := h\mathbf{w}^\top\mathbf{w} - 2\mathbf{f}^\top\mathbf{w}, \tag{12}$$

with $h = \sum_{n=1}^N y_n^2$ and $\mathbf{f} = \sum_{n=1}^N y_n\mathbf{x}_{(n)}$. Eq. (12) is a quadratic program (QP), and is convex due to the non-negativity of h. Furthermore, because the Hessian is a scaled identity matrix, the problem is also isotropic. The unique global optimum is found by analytical differentiation of the objective function

$$\nabla J \stackrel{!}{=} 0 \Rightarrow \mathbf{w}^* = \frac{\mathbf{f}}{h}, \tag{13}$$

which of course is identical to eq. (11).

3.1. Sparsity

It is well known (Tibshirani, 1996) that solving a QP under an additional constraint on $\|\mathbf{w}\|_1$ favors a sparse solution. This constraint corresponds to restricting the feasible region to an l_1 diamond:

$$\mathbf{w}^\circ = \arg\min_{\mathbf{w}} \left(h\mathbf{w}^\top\mathbf{w} - 2\mathbf{f}^\top\mathbf{w}\right) \quad \text{s.t.} \quad \|\mathbf{w}\|_1 \le B, \tag{14}$$

where the upper bound B is chosen such that $\mathbf{w}^\circ$ has the desired cardinality. The l_1 constrained QP is again convex, and because the objective function is isotropic, it implies that $\mathbf{w}^\circ$ is the feasible point minimizing l_2 distance to the unconstrained optimum $\mathbf{w}^*$.

We derive an efficient and optimal algorithm for eq. (14), where the desired cardinality can be specified directly by the number K of non-zero dimensions. Observe that $\mathbf{w}^\circ$ must have the same sign structure as $\mathbf{f}$, therefore we can transform the problem such that both $\mathbf{w}^*$ and $\mathbf{w}^\circ$ come to lie in the non-negative orthant. The algorithm (illustrated in fig. 1) approaches $\mathbf{w}^\circ$

with axis-aligned steps in the direction of the largest element of the negative gradient

$$-\nabla J(\mathbf{w}) \propto \mathbf{w}^* - \mathbf{w}, \tag{15}$$

until the boundary of the feasible region is hit or the gradient vanishes. Because the elements of $\mathbf{w}$ become positive one after another, and their magnitude increases monotonically, B is set implicitly by terminating the gradient descent once the cardinality of the solution vector is K. Finally, the solution is transformed back into the original orthant of $\mathbf{w}^*$.

Proposition 3.1 *Axis-aligned gradient descent with infinitesimal stepsize terminates at the optimal feasible point* $\mathbf{w}^\circ$.

Proof. Optimality is trivial if $\mathbf{w}^*$ lies within the feasible region, so we consider the case where the l_1 constraint is active. The objective function in eq. (14) is equivalent to

$$\|\mathbf{w}^* - \mathbf{w}\|_2^2 = \sum_{d=1}^{D} \left(w_d^* - w_d\right)^2. \tag{16}$$

The gradient descent procedure invests all available coefficient weight B into decreasing the largest term(s) of this sum, which follows from eq. (15). We show equivalence of $\mathbf{w}^\circ$ to the gradient descent solution $\mathbf{v}$ by contradiction. Suppose the computation of $\mathbf{w}^\circ$ follows a different strategy, so at least one summation term $(w_l^* - w_l^\circ)^2$ is larger than $\max_d (w_d^* - v_d)^2$. However, subtracting a small amount from w_s° $(s \neq l)$ and adding it to w_l° doesn't change $\|\mathbf{w}^\circ\|_1$ but decreases the objective, which is a contradiction. □

Implementation of axis-aligned gradient descent amounts to sorting the elements of $-\nabla J(\mathbf{w})$ in descending order (an $O(D \log D)$ operation), and iterating over its first K elements. At each iteration $k \in \{1, \ldots, K\}$, the first k elements of $\mathbf{w}$ are manipulated, resulting in complexity $O(K^2)$ for the whole loop. Algorithm 2 provides a full specification of the method. Because EM is a local optimizer, the initial direction $\mathbf{w}_{(1)}$ must be chosen carefully to achieve good results. For sparse PCA, initialization with the unconstrained first principal component gave best results (see section 5). Initialization is therefore the most expensive operation of the algorithm with its $O(D^2)$ complexity. For the $D \gg N$ case, it can be reduced to $O(N^2)$ by working with $\mathbf{X}\mathbf{X}^\top$ instead of $\mathbf{X}^\top\mathbf{X}$. As initialization is independent of K, $\mathbf{w}_{(1)}$ can be cached and re-used when varying the sparsity parameter. The number of EM iterations t until convergence also depends on D and K, but our experiments (see section 5) suggest that dependence is weak and sub-linear. On average, $t < 10$ iterations were sufficient to achieve convergence.

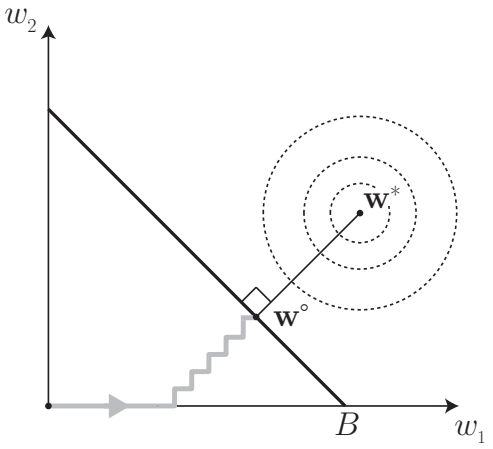


Figure 1. Starting at the origin, $\mathbf{w}^\circ$ is approached by axis-aligned steps in the direction of the largest element of the negative gradient. As dimensions enter the solution vector one after another, and the corresponding weights w_i increase monotonically, the bound B is set implicitly by terminating once $\|\mathbf{w}\|_0 = K$.

3.2. Non-Negativity

Enforcing non-negativity is achieved in the same way as sparsity. Here, the the feasible region is constrained to the non-negative orthant, which is again a convex domain:

$$\begin{aligned} \mathbf{w}^\circ \quad &= \quad \arg\min_{\mathbf{w}} \left(h\mathbf{w}^\top\mathbf{w} - 2\mathbf{f}^\top\mathbf{w}\right) \\ &\text{s.t.} \quad \mathbf{w} \succeq \mathbf{0}. \end{aligned} \tag{17}$$

Eq. (17) implies that choosing $w_i = 0$ for $f_i < 0$ is optimal. The non-negativity constraint can then be dropped, and optimization for the other elements of $\mathbf{w}$ proceeds as before.

The first PC is invariant to a change of sign. However, this symmetry is broken if the non-negativity constraint is enforced. As an extreme example, non-negative EM fails if the initial projection $\mathbf{w}_{(1)}$ is a dominant eigenvector that only consists of non-positive elements - the minimum of eq. (17) is the zero vector. But changing the sign of $\mathbf{w}_{(1)}$ implies that the non-negativity constraint becomes inactive, and the algorithm terminates immediately with the optimal solution. We choose to initialize EM for non-negative PCA with a random unit vector in the non-negative orthant, which exploits the benefit of random restarts.

For non-negative sparse PCA, the feasible region is defined as the intersection of the non-negative orthant

Algorithm 2 EM for Sparse PCA

Input: $\mathbf{X} \in \mathbb{R}^{N \times D}$, $K \in \{1, \ldots, D\}$, ε
Algorithm:
$t \leftarrow 1$
$\mathbf{w}_{(t)} \leftarrow$ first principal component of $\mathbf{X}$
repeat
  $\mathbf{y} \leftarrow \mathbf{X}\mathbf{w}_{(t)}$
  $\mathbf{w}^* \leftarrow \sum_{n=1}^{N} y_n \mathbf{x}_{(n)} / \sum_{n=1}^{N} y_n^2$
  $\mathbf{s} \leftarrow$ elements $|w_i^*|$ sorted in descending order
  $\boldsymbol{\pi} \leftarrow$ indices of sorting order
  $\mathbf{w}_{(t+1)} \leftarrow \mathbf{0}$
  for $k = 1$ **to** K **do**
    Add $(s_k - s_{k+1})$ to elements $1, \ldots, k$ of $\mathbf{w}_{(t+1)}$
  end for
  Permute elements of $\mathbf{w}_{(t+1)}$ according to $\boldsymbol{\pi}^{-1}$
  $\mathbf{w}_{(t+1)} \leftarrow \mathbf{w}_{(t+1)} \circ \operatorname{sign}(\mathbf{w}^*) / \|\mathbf{w}_{(t+1)}\|_2$
  $t \leftarrow t + 1$
until $|\mathbf{w}_{(t+1)}^\top \mathbf{w}_{(t)}| > 1 - \varepsilon$
Output: $\mathbf{w}$

and the l_1 diamond. As the intersection of two convex sets is again convex, the combined constraints can be treated in the same framework. We establish convergence of our method in the following proposition:

Proposition 3.2 *EM for sparse and non-negative PCA converges to a local minimum of the l_2 reconstruction error.*

Proof. Given a feasible $\mathbf{w}_{(t)}$ (either by proper initialization or after one EM iteration), both the E-step and the M-step never increase l_2 reconstruction error. Orthogonal projection $\mathbf{y} = \mathbf{X}\mathbf{w}$ in the E-step is the l_2 optimal choice of subspace coordinates for given $\mathbf{w}$. Error minimization w.r.t. $\mathbf{w}$ in the M-step either recovers $\mathbf{w}_{(t)}$ as it is feasible, or provides an improved $\mathbf{w}_{(t+1)}$. □

4. Several Components

A full eigen decomposition of the covariance matrix $\mathbf{C}$ provides all r PCs, where r is the rank of $\mathbf{C}$. Sorted in descending order of eigenvalue magnitude, each eigenvector maximizes the variance of the projected data, under the constraint that it is orthogonal to all other components considered so far. For sparse PCA, we compute more than one component by means of iterative deflation: having identified the first component $\mathbf{w}_{(1)}$, project the data to its orthogonal subspace using

$$\mathbf{P} = \mathbf{I} - \mathbf{w}_{(1)}\mathbf{w}_{(1)}^\top, \tag{18}$$

re-run EM to identify $\mathbf{w}_{(2)}$, and so on. Although deflation suffers from numerical errors that accumulate over each iteration, this inaccuracy is not a serious problem as long as the desired number of components L is small compared to r (which is true in many applications of PCA).

Desiring non-negativity and orthogonality implies that each feature can be part of at most one component:

$$w_i^{(l)} > 0 \Rightarrow w_i^{(m)} = 0 \tag{19}$$

for $m \neq l$, i.e. the sparsity patterns have to be disjoint: $\mathcal{S}_l \bigcap \mathcal{S}_m = \emptyset$, for $l \neq m$ and $\mathcal{S}_l = \{i | w_i^{(l)} > 0\}$. This constraint might be too strong for some applications, where it can be relaxed to require a minimum angle between components. This *quasi*-orthogonality is enforced by adding a quadratic penalty term

$$\alpha \mathbf{w}^\top \mathbf{V}\mathbf{V}^\top \mathbf{w}, \tag{20}$$

to eq. (17), where $\mathbf{V} = \left[\mathbf{w}_{(1)}\mathbf{w}_{(2)} \cdots \mathbf{w}_{(l-1)}\right]$ contains previously identified components as columns, and α is a tuning parameter. Because $\mathbf{V}\mathbf{V}^\top$ is also positive semi-definite, the QP remains convex, but the Hessian is no longer isotropic. We have used the standard Matlab QP solver, but there exist special algorithms for this case in the literature (Sha et al., 2007).

5. Experimental Results

We report performance and efficiency of our method in comparison to three algorithms: SPCA[2] and PathSPCA[3] for cardinality constrained PCA, and NSPCA[4] for non-negative sparse PCA. SPCA was chosen because it has conceptual similarities to our algorithm: both are iterative methods that solve an l_1 constrained convex program, and both use the data matrix instead of the covariance matrix. PathSPCA was chosen because it is (to our knowledge) the most efficient combinatorial algorithm. We are not aware of any other non-negative PCA algorithm besides NSPCA.

The data sets considered in the evaluation are the following:

1. CBCL face images (Sung, 1996): 2429 gray scale images of size 19×19 pixels, which have been used in the evaluation of (Zass & Shashua, 2006).
2. Leukemia data (Armstrong et al., 2002): Expression profiles of 12582 genes from 72 patients. Sim-

[2]We use the Matlab implementation of SPCA by Karl Sjöstrand, available at `http://www2.imm.dtu.dk/~kas/software/spca/index.html`.

[3]Available from the authors at `http://www.princeton.edu/~aspremon/PathSPCA.htm`.

[4]Available from the authors at `http://www.cs.huji.ac.il/~zass/`.

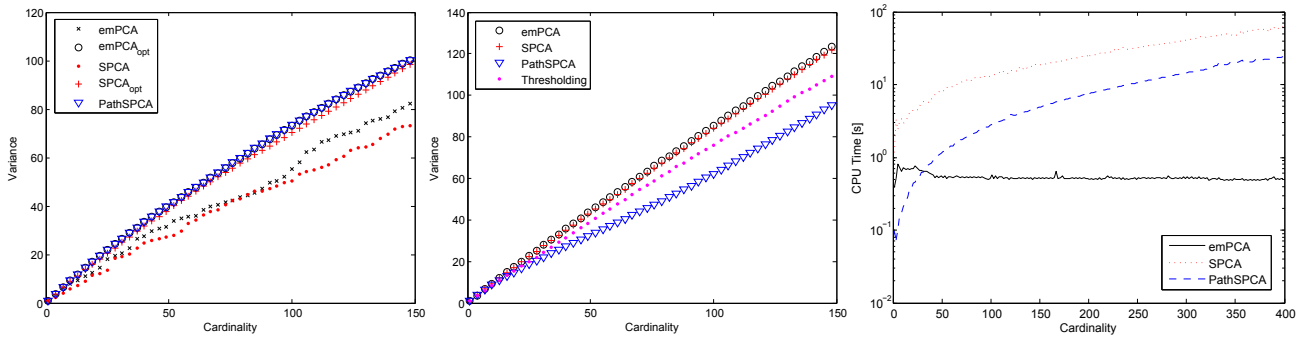


Figure 2. Left: Variance versus cardinality trade-off curves for the face image data. "opt" subscripts denote variance after recomputing optimal weights for a given sparsity pattern (which is not necessary for PathSPCA). *Middle:* Variance versus cardinality trade-off curves for the gene expression data. Performance of simple thresholding was included for reference. *Right:* Running times of Matlab implementations on the gene expression data, which include renormalization for SPCA and emPCA.

ilar data sets have been used in the evaluation of (Zou et al., 2004) and (d'Aspremont et al., 2007).

The two data sets cover the $N > D$ and $D \gg N$ case and are large enough such that differences in computational complexity can be established with confidence. Both were standardized such that each dimension has zero mean and unit variance.

5.1. Sparse PCA

Figure 2 (left) plots explained variance versus cardinality for SPCA, PathSPCA and our algorithm (called emPCA) on the face image data set. Variational renormalization is necessary for SPCA and emPCA to close the performance gap to PathSPCA, which computes optimal weights for a specific sparsity pattern by construction. Figure 2 (middle) shows analogous results for the gene expression data. As a reference, we have also plotted results for simple thresholding (after renormalization).

To complement theoretical analysis of computational complexity, we have also measured running times of reference Matlab implementations, provided by their respective authors (SPCA is a re-implementation of the author's R code in Matlab). CPU time was measured using Matlab's `tic` and `toc` timer constructs, running on an Intel Core 2 Duo processor at 2.2GHz with 3GB of RAM. Our focus is not to report absolute numbers, but rather demonstrate the dependency on the choice of K. Figure 2 (right) plots the running times versus cardinality on the gene expression data. The PathSPCA curve is well explained by the incremental forward greedy search. SPCA is harder to analyze, due to its active set optimization scheme: at each iteration of the algorithm, active features are reexamined and possibly excluded, but might be added again later on. emPCA is only marginally affected by the choice of K, but shows an increased number of EM iterations for $10 \leq K \leq 25$, which was observed on other data sets as well.

5.2. Non-Negative PCA

The impact of the non-negativity constraint on the explained variance depends on the sign structure of $\mathbf{w}^*$. Because the first principal component for the face image data happens to lie in the non-negative orthant, we projected the data onto its orthogonal subspace such that the constraint becomes active. Figure 3 (left) shows the variance versus cardinality trade-off curves for non-negative sparse PCA. For NSPCA, the sparsity penalty β was determined for each K using bisection search, which was aborted when the relative length of the parameter search interval was below a threshold of 10^{-5}. Both the variance achieved and the number of cardinalities for which a solution was found strongly depend on the value of α, which corresponds to a unit norm penalty (for the case of a single component). For smaller values of α the performance of NSPCA is comparable to emPCA, but only solutions close to the full cardinality are found. Increasing the magnitude of α makes it possible to sweep the whole cardinality path, but the performance degrades.

Because both algorithms are initialized randomly, we chose the best result after ten restarts. Running times for both methods showed no strong dependency on K. Average times for $K \in \{1, \ldots, 100\}$ were $0.4s$ for emPCA ($0.15s$ standard deviation) and $24s$ for NSPCA ($14.7s$ standard deviation).

We already motivated in section 4 that requiring orthogonality between several non-negative components can be restrictive. If the first PC happens to lie in the

non-negative orthant, the constraints have to be modified such that more than one component can satisfy them. We have explored the following two strategies:

1. Enforcing orthogonality, but constraining the cardinality of each component.

2. Relaxing the orthogonality constraint, by enforcing a minimum angle between components instead.

There is a methodological difficulty in comparing the performance of NSPCA and emPCA. The former maximizes cumulative variance of all components jointly, while our algorithm computes them sequentially, maximizing the variance under the constraint that subsequent components are orthogonal to previous ones (see section 4). We therefore expect emPCA to capture more variance in the first components, while NSPCA is expected to capture larger cumulative variance. Figure 3 (middle) shows the results of applying the first strategy to the face image data. The NSPCA sparsity penalty β was tuned to achieve a joint cardinality of 200 for all components. For emPCA we distributed the active features evenly among components by setting $K = 20$ for all of them. As in figure 3 (left), emPCA captures significantly more of the variance, suggesting that the way NSPCA incorporates sparsity seriously degrades performance. This observation was confirmed for various values of K and L.

Finally, figure 3 (right) reports results for the second strategy, where a minimum angle of 85 degrees was enforced between components. Here, the complementary objectives of NSPCA and emPCA match with our prior expectations. Again, various values for L and minimum angle lead to essentially the same behavior.

5.3. Unsupervised Gene Selection

We apply emPCA to select a subset of genes of the leukemia data, and measure subset relevance by following the evaluation methodology of Varshavsky et al. (2006). For each gene subset, we cluster the data using k-means ($k = 3$), and compare the cluster assignments to the true labeling of the data, which differentiates between three types of leukemia (ALL, AML and MLL). Agreement is measured using Jaccard scores (Varshavsky et al., 2006), where a value of one signifies perfect correspondence between cluster assignment and label. We compare emPCA to simple ranking of the CE criterion as proposed by the authors, which has shown competitive performance to other popular gene selection methods. Figure 4 shows that selecting 70 genes according to the first non-negative sparse PC

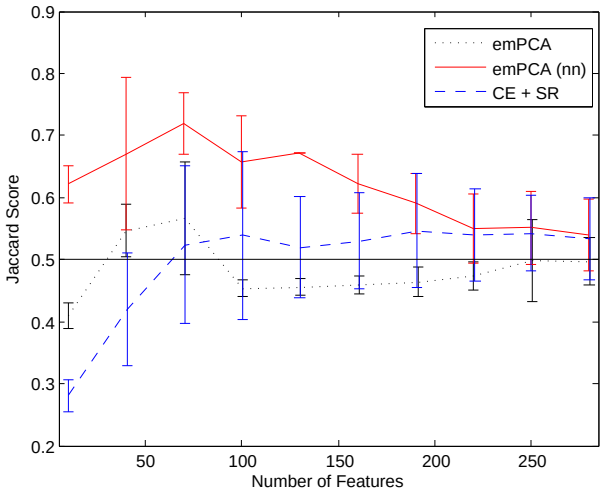


Figure 4. Mean and standard deviation for Jaccard scores after subset selection and k-means clustering ($k = 3$), averaged over 100 random initializations of the centroids (see text). A small amount of jitter has been added to better distinguish error bars.

results in a significantly better Jaccard score than a clustering of the full data set.

6. Conclusions

We have presented a novel algorithm for constrained principal component analysis, based on expectation-maximization for probabilistic PCA. Our method is applicable to a broad range of problems: it includes sparsity, non-negativity or both kinds of constraints, it has an efficient formulation for $N > D$ and $D \gg N$ type of data, and it enforces either strict or quasi-orthogonality between successive components. Desired sparsity is directly specified in the number of non-zero elements, instead of a bound on the l_1 norm of the vector. We have demonstrated on popular data sets from biology and computer vision that our method achieves competitive results for sparse problems, and that it shows significant improvements for non-negative sparse problems. Its unmatched computational efficiency enables a constrained principal component analysis of substantially larger data sets and lower requirements on available computation time.

Although our algorithm is rooted in expectation-maximization for a generative model of PCA, constraints are added at the optimization stage. In the future, we will study how to include them in the model itself, which would enable a Bayesian analysis and data-driven determination of the proper sparsity and number of components. Secondly, we intend to examine whether our algorithm can be extended to the related

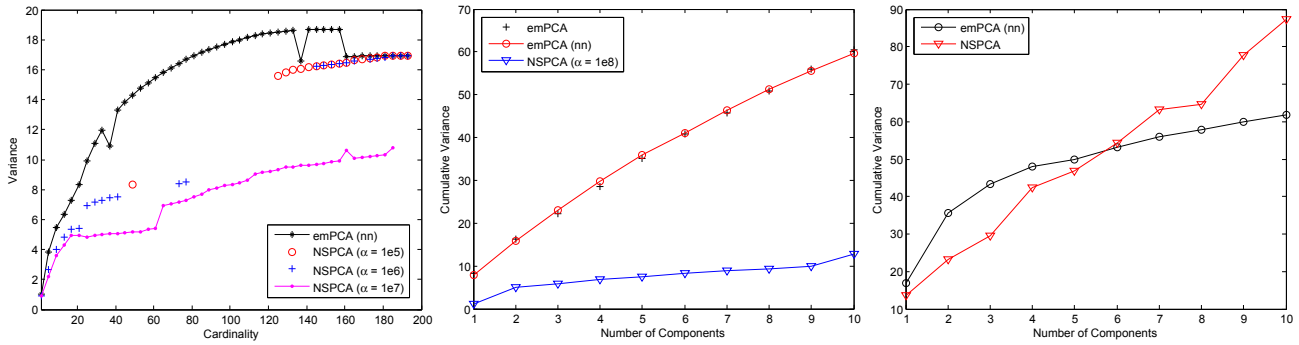


Figure 3. Left: Variance versus cardinality trade-off curves for non-negative sparse PCA methods on face image data. For NSPCA, the sparsity penalty β was determined using bisection search (see text). Values indicate better result after ten random restarts. *Middle:* Cumulative variance versus number of orthogonal components. For NSPCA, β was tuned to achieve a joint cardinality of 200 for all components. For emPCA, we set $K = 20$ for every component. emPCA (without non-negativity constraints) is plotted for reference. *Right:* Cumulative variance versus number of quasi-orthogonal components. A minimum angle of 85 degrees was enforced between components.

problem of constrained linear discriminant analysis.

A Matlab implementation of emPCA is available at `http://www.inf.ethz.ch/personal/chrsigg/icml2008`.

Acknowledgements

We thank Wolfgang Einhäuser-Treyer, Peter Orbanz and the anonymous reviewers for their valuable comments on the manuscript. This work was in part funded by CTI grant 8539.2;2 ESPP-ES.

References

Armstrong, S., Staunton, J., Silverman, L., Pieters, R., den Boer, M., Minden, M., Sallan, S., Lander, E., Golub, T., & Korsmeyer, S. (2002). MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics*, *30*, 41–47.

Cadima, J., & Jolliffe, I. (1995). Loadings and correlations in the interpretation of principal components. *Applied Statistics*, 203–214.

d'Aspremont, A., Bach, F., & El Ghaoui, L. (2007). Full regularization path for sparse principal component analysis. *Proceedings of the International Conference on Machine Learning*.

Horst, R., Pardalos, P., & Thoai, N. (2000). *Introduction to global optimization*. Kluwer Acad. Publ.

Moghaddam, B., Weiss, Y., & Avidan, S. (2006). Spectral bounds for sparse PCA: Exact and greedy algorithms. *Advances in Neural Information Processing Systems*.

Roweis, S. (1998). EM algorithms for PCA and sensible PCA. *Advances in Neural Information Processing Systems*.

Sha, F., Lin, Y., Saul, L., & Lee, D. (2007). Multiplicative Updates for Nonnegative Quadratic Programming. *Neural Computation*, *19*, 2004–2031.

Sriperumbudur, B., Torres, D., & Lanckriet, G. (2007). Sparse eigen methods by d.c. programming. *Proceedings of the International Conference on Machine Learning*.

Sung, K.-K. (1996). *Learning and example selection for object and pattern recognition*. Doctoral dissertation, MIT, Artificial Intelligence Laboratory and Center for Biological and Computational Learning, Cambridge, MA.

Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal statistical society, series B*, *58*, 267–288.

Tipping, M., & Bishop, C. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, *21*, 611–622.

Varshavsky, R., Gottlieb, A., Linial, M., & Horn, D. (2006). Novel Unsupervised Feature Filtering of Biological Data. *Bioinformatics*, *22*.

Zass, R., & Shashua, A. (2006). Nonnegative sparse PCA. *Advances in Neural Information Processing Systems*.

Zou, H., Hastie, T., & Tibshirani, R. (2004). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*.

Sample-Based Learning and Search with Permanent and Transient Memories

David Silver SILVER@CS.UALBERTA.CA
Richard S. Sutton SUTTON@CS.UALBERTA.CA
Martin Müller MMUELLER@CS.UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, Alberta

Abstract

We present a reinforcement learning architecture, Dyna-2, that encompasses both sample-based learning and sample-based search, and that generalises across states during both learning and search. We apply Dyna-2 to high performance Computer Go. In this domain the most successful planning methods are based on sample-based search algorithms, such as UCT, in which states are treated individually, and the most successful learning methods are based on temporal-difference learning algorithms, such as Sarsa, in which linear function approximation is used. In both cases, an estimate of the value function is formed, but in the first case it is transient, computed and then discarded after each move, whereas in the second case it is more permanent, slowly accumulating over many moves and games. The idea of Dyna-2 is for the transient planning memory and the permanent learning memory to remain separate, but for both to be based on linear function approximation and both to be updated by Sarsa. To apply Dyna-2 to 9×9 Computer Go, we use a million binary features in the function approximator, based on templates matching small fragments of the board. Using only the transient memory, Dyna-2 performed at least as well as UCT. Using both memories combined, it significantly outperformed UCT. Our program based on Dyna-2 achieved a higher rating on the Computer Go Online Server than any handcrafted or traditional search based program.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

1. Introduction

Reinforcement learning can be subdivided into two fundamental problems: *learning* and *planning*. Informally, the goal of learning is for an agent to improve its policy from its interactions with the environment. The goal of planning is for an agent to improve its policy without further interaction with its environment. The agent can deliberate, reason, ponder, think or search, so as to find the best behaviour in the available computation time. Sample-based methods can be applied to both problems. During learning, the agent samples experience from the real world: it executes an action at each time-step and observes its consequences. During planning, the agent samples experience from a *model* of the world: it *simulates* an action at each computational step and observes its consequences. We propose that an agent can both learn and plan effectively using sample-based reinforcement learning algorithms. We use the game of 9×9 Go as an example of a large-scale, high-performance application in which learning and planning both play significant roles.

In the domain of Computer Go, the most successful learning methods have used sample-based reinforcement learning to extract domain knowledge from games of self-play (Schraudolph et al., 1994; Dahl, 1999; Enzenberger, 2003; Silver et al., 2007). The value of a position is approximated by a multi-layer perceptron, or a linear combination of binary features, that form a compact representation of the state space. Temporal difference learning is used to update the value function, slowly accumulating knowledge from the complete history of experience.

The most successful planning methods use sample-based search to identify the best move in the current position. 9×9 Go programs based on the UCT algorithm (Kocsis & Szepesvari, 2006) have now achieved master level (Gelly & Silver, 2007; Coulom, 2007). The UCT algorithm begins each new move with no domain

knowledge, but rapidly learns the values of positions in a temporary search tree. Each state in the tree is explicitly represented, and the value of each state is learned by Monte-Carlo simulation, from games of self-play that start from the current position.

In this paper we develop an architecture, Dyna-2, that combines these two approaches. Like the Dyna architecture (Sutton, 1990), the agent updates a value function both from real experience, and from simulated experience that is sampled using a model of the world. The new idea is to maintain two separate memories: a permanent learning memory that is updated from real experience; and a transient planning memory that is updated from simulated experience. Both memories use linear function approximation to form a compact representation of the state space, and both memories are updated by temporal-difference learning.

2. Reinforcement Learning

We consider sequential decision-making processes, in which at each time-step t the agent receives a state s_t, executes an action a_t according to its current policy $\pi_t(s, a)$, and then receives a scalar reward r_{t+1}.

2.1. Sample-Based Learning

Most efficient reinforcement learning methods use a *value function* as an intermediate step for computing a policy. In episodic tasks the action-value function $Q^\pi(s, a)$ is the expected total reward from state s after taking action a and then following policy π.

In large domains, it is not possible or practical to learn a value for each individual state. In this case, it is necessary to approximate the value function using features $\phi(s, a)$ and parameters θ. A simple and successful approach (Sutton, 1996) is to use a linear function approximation $Q(s, a) = \phi(s, a)^T\theta$. We note that table lookup is a special case of linear function approximation, using binary features $\phi(s, a) = e(s, a)$, where $e(s, a)$ is a unit vector with a one in the single component corresponding to (s, a) and zeros elsewhere.

The TD(λ) algorithm (Sutton, 1988) estimates the value of the current state from the value of subsequent states. The λ parameter determines the temporal span over which values are updated. At one extreme, TD(0) bootstraps the value of a state from its immediate successor. At the other extreme, TD(1) updates the value of a state from the final return; it is equivalent to Monte-Carlo evaluation (Sutton & Barto, 1998). TD(λ) can be incrementally computed by maintaining a vector of *eligibility traces* z_t.

The Sarsa algorithm (Rummery & Niranjan, 1994) combines temporal difference evaluation with policy improvement. An action-value function is estimated by the TD(λ) algorithm, and the policy is improved by selecting actions according to an ϵ-greedy policy. The action-value function is updated from each tuple (s, a, r, s', a') of experience, using the TD(λ) update rule,

$$\delta_t = r_{t+1} + Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \quad (1)$$

$$z_t = \lambda z_{t-1} + \phi(s_t, a_t) \quad (2)$$

$$\theta_t = \theta_{t-1} + \alpha\delta_t z_t(s) \quad (3)$$

2.2. Sample-Based Search

Sample-based planning applies sample-based reinforcement learning methods to simulated experience. This requires a sample model of the world: a state transition generator $A_t(s, a) \in \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ and reward generator $B_t(s, a) \in \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. The effectiveness of sample-based planning depends on the accuracy of the model (Paduraru, 2007). In *sample-based search*, experience is simulated from the real state s, so as to identify the best action from this state.

Monte-Carlo simulation is a simple but effective method for sample-based search. Multiple episodes are simulated, starting from the real state s, and following a random policy. The action-values $Q(s, a)$ are estimated by the empirical average of the returns of all episodes in which action a was taken from the real state s. After simulation is complete, the agent selects the greedy action $\text{argmax}_a\, Q(s, a)$, and proceeds to the next real state.

Monte-Carlo tree search constructs a search tree containing all state–action pairs that have been visited by the agent. Each simulation consists of two distinct phases: greedy action selection while within the tree, and then random action selection until termination. If a simulated state s is fully represented in the search tree, i.e. all actions from s have already been tried, then the agent selects the greedy action $\text{argmax}_a\, Q(s, a)$. Otherwise, the agent selects actions at random. After each simulation, the action-values $Q(s, a)$ of all states and actions experienced in the episode are updated to the empirical average return following each state–action pair. In practice, only one new state–action pair is added per episode, resulting in a tree-like expansion.

The UCT algorithm (Kocsis & Szepesvari, 2006) improves the greedy action selection in Monte-Carlo tree search. Each state of the search tree is treated as a multi-armed bandit, and actions are chosen using the UCB algorithm for balancing exploration and exploita-

tion (Auer et al., 2002).

2.3. Dyna

The Dyna architecture (Sutton, 1990) combines sample-based learning with sample-based planning. The agent learns a model of the world from real experience, and updates its action-value function from both real and simulated experience. Before each real action is selected, the agent performs some sample-based planning. For example, the Dyna-Q algorithm remembers all previous states, actions and transitions. During planning, experience is simulated by sampling states, actions and transitions from the empirical distribution. A Q-learning update is applied to update the action-value function after each sampled transition, and after each real transition.

2.4. Tracking

Traditional learning methods focus on finding a single best solution to the learning problem. In reinforcement learning one may seek an algorithm that converges on the optimal value function (or optimal policy). However, in large domains the agent may not have sufficient resources to perfectly represent the optimal value function. In this case we can actually achieve better performance by *tracking* the current situation rather than *converging* on the best overall parameters. The agent can specialise its value function to its current region of the state space, and update its representation as it moves through the state space. The potential for specialisation means that tracking methods may outperform converging methods, even in stationary domains (Sutton et al., 2007).

3. Permanent and Transient Memories

We define a *memory* to be the set of features and corresponding parameters used by an agent to estimate the value function. In our architecture, the agent maintains two distinct memories: a *permanent memory* (ϕ, θ) updated during sample-based learning, and a *transient memory* $(\bar{\phi}, \bar{\theta})$ updated during sample-based search. The value function is a linear combination of the transient and permanent memories, such that the transient memory tracks a local correction to the permanent memory,

$$Q(s,a) = \phi(s,a)^T\theta \quad (4)$$

$$\bar{Q}(s,a) = \phi(s,a)^T\theta + \bar{\phi}(s,a)^T\bar{\theta} \quad (5)$$

where $Q(s,a)$ is a permanent value function, and $\bar{Q}(s,a)$ is a combined value function.

We refer to the distribution of states and actions encountered during real experience as the *learning distribution*, and the distribution encountered during simulated experience as the *search distribution*. The permanent memory is updated from the learning distribution and converges on the best overall representation of the value function, based on the agent's past experience. The transient memory is updated from the search distribution and tracks the local nuances of the value function, based on the agent's expected future experience.

4. Dyna-2

Algorithm 1 Episodic Dyna-2

procedure LEARN
  Initialise A, B ▷ Transition and reward models
  $\theta \leftarrow 0$ ▷ Clear permanent memory
  loop
    $s \leftarrow s_0$ ▷ Start new episode
    $\bar{\theta} \leftarrow 0$ ▷ Clear transient memory
    $z \leftarrow 0$ ▷ Clear eligibility trace
    SEARCH(s)
    $a \leftarrow \pi(s; \bar{Q})$ ▷ e.g. ϵ-greedy
    while s is not terminal **do**
      Execute a, observe reward r, state s'
      $(A, B) \leftarrow$ UPDATEMODEL(s, a, r, s')
      SEARCH(s')
      $a' \leftarrow \pi(s'; \bar{Q})$
      $\delta \leftarrow r + Q(s', a') - Q(s, a)$ ▷ TD-error
      $\theta \leftarrow \theta + \alpha(s, a)\delta z$ ▷ Update weights
      $z \leftarrow \lambda z + \phi$ ▷ Update eligibility trace
      $s \leftarrow s', a \leftarrow a'$
    end while
  end loop
end procedure

procedure SEARCH(s)
  while time available **do**
    $\bar{z} \leftarrow 0$ ▷ Clear eligibility trace
    $a \leftarrow \bar{\pi}(s; \bar{Q})$ ▷ e.g. ϵ-greedy
    while s is not terminal **do**
      $s' \leftarrow A(s, a)$ ▷ Sample transition
      $r \leftarrow B(s, a)$ ▷ Sample reward
      $a' \leftarrow \bar{\pi}(s'; \bar{Q})$
      $\bar{\delta} \leftarrow r + \bar{Q}(s', a') - \bar{Q}(s, a)$ ▷ TD-error
      $\bar{\theta} \leftarrow \bar{\theta} + \bar{\alpha}(s, a)\bar{\delta}\bar{z}$ ▷ Update weights
      $\bar{z} \leftarrow \bar{\lambda}\bar{z} + \bar{\phi}$ ▷ Update eligibility trace
      $s \leftarrow s', a \leftarrow a'$
    end while
  end while
end procedure

The Dyna-2 architecture can be summarised as Dyna with Sarsa updates, permanent and transient memories, and linear function approximation (see Algorithm 1). The agent updates its permanent memory from real experience. Before selecting a real action, the agent executes a sample-based search from the current state. The search procedure simulates complete episodes from the current state, sampled from the model, until no more computation time is available. The transient memory is updated during these simulations to learn a local correction to the permanent memory; it is cleared at the beginning of each real episode.

A particular instance of Dyna-2 must specify learning parameters: a policy π to select real actions; a set of features ϕ for the permanent memory; a temporal difference parameter λ; and a learning rate $\alpha(s, a)$. Similarly, it must specify the equivalent search parameters: a policy $\bar{\pi}$ to select actions during simulation; a set of features $\bar{\phi}$ for the transient memory; a temporal difference parameter $\bar{\lambda}$; and a learning rate $\bar{\alpha}(s, a)$.

The Dyna-2 architecture subsumes a large family of learning and search algorithms. If there is no transient memory, $\bar{\phi} = \emptyset$, then the search procedure has no effect and may be skipped. This results in the linear Sarsa algorithm.

If there is no permanent memory, $\phi = \emptyset$, then Dyna-2 reduces to a sample-based search algorithm. For example, Monte-Carlo tree search is achieved by choosing table lookup $\bar{\phi}(s, a) = e(s, a)$[1]; using a simulation policy that is greedy within the tree, and then uniform random until termination; and selecting learning parameters $\bar{\lambda} = 1$ and $\bar{\alpha}(s, a) = 1/n(s, a)$, where $n(s, a)$ counts the number of times that action a has been selected in state s. The UCT algorithm replaces the greedy phase of the simulation policy with the UCB rule for action selection.

Finally, we note that real experience may be accumulated offline prior to execution. Dyna-2 may be executed on any suitable training environment (e.g. a helicopter simulator) before it is applied to real data (e.g. a real helicopter). The permanent memory is updated offline, but the transient memory is updated online. Dyna-2 provides a principled mechanism for combining offline and online knowledge(Gelly & Silver, 2007); the permanent memory provides prior knowledge and a baseline for fast learning. Our examples of Dyna-2 in Computer Go operate in this manner.

[1]The number of entries in the table can increase over time, to give a tree-like expansion.

5. Dyna-2 in Computer Go

In domains with spatial coherence, binary features can be constructed to exploit spatial structure at multiple levels (Sutton, 1996). The game of Go exhibits strong spatial coherence: expert players describe positions using a broad vocabulary of shapes (Figure 1a). A simple way to encode basic shape knowledge is through a large set of *local shape features* which match a particular configuration within a small region of the board (Silver et al., 2007). We define the feature vector $\phi^{square(m)}$ to be the vector of local shape features for $m \times m$ square regions, for all possible configurations and square locations. For example, Figure 1a shows several local shape features of size 3×3. Combining local shape features of different sizes builds a representation spanning many levels of generality: we define the multi-level feature vector $\phi^{square(m,n)} = [\phi^{square(m)}; ...; \phi^{square(n)}]$. In 9×9 Go there are nearly a million $\phi^{square(1,3)}$ features, about 200 of which are non-zero at any given time.

Local shape features can be used as a permanent memory, to represent general domain knowledge. For example, local shape features can be learned offline, using temporal difference learning and training by self-play (Silver et al., 2007; Gelly & Silver, 2007). However, local shape features can also be used as a transient memory[2], by learning online from simulations from the current state. The representational power of local shape features is significantly increased when they can track the short-term circumstances (Sutton et al., 2007). A local shape may be bad in general, but good in the current situation (Figure 1b). By training from simulated experience, starting from the current state, we can focus learning on what works well *now*.

We apply the Dyna-2 algorithm to 9×9 Computer Go using local shape features $\phi(s, a) = \bar{\phi}(s, a) = \phi^{square(1,3)}(s \circ a)$, where $s \circ a$ indicates the *afterstate* following action a in state s (Sutton & Barto, 1998). We use a self-play model, an ϵ-greedy policy, and default parameters of $\lambda = \bar{\lambda} = 0.4$, $\alpha(s, a) = 0.1/|\phi(s, a)|$, $\bar{\alpha}(s, a) = 0.1/|\bar{\phi}(s, a)|$, and $\epsilon = 0.3$. We modify the Dyna-2 algorithm slightly to utilise the logistic function and to minimise a cross-entropy loss function, by replacing the value function approximation in (4) and (5),

$$Q(s, a) = \sigma(\phi(s, a)^T \theta) \quad (6)$$

$$\bar{Q}(s, a) = \sigma(\phi(s, a)^T \theta + \bar{\phi}(s \circ a)^T \bar{\theta}) \quad (7)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic function.

[2]Symmetric local shape features share weights in the permanent memory, but not in the transient memory.

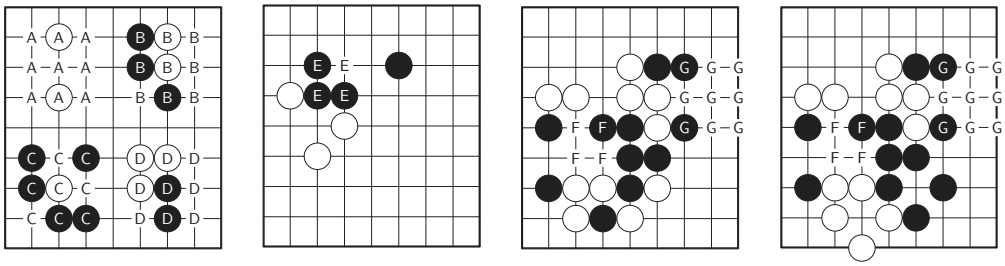


Figure 1. a) Examples of 3×3 local shape features, matching common shapes known to Go players: the *one-point jump* (A), *hane* (B), *net* (C) and *turn* (D). b) The *empty triangle* (E) is normally considered a bad shape; it can be learned by the permanent memory. However, in this position the empty triangle makes a good shape, known as *guzumi*; it can be learned by the transient memory. c) White threatens to cut blacks stones apart at F and G. 2×2 and 3×3 local shape features can represent the local consequences of cutting at F and G respectively. d) A position encountered when searching from (c): Dyna-2 is able to generalise, using local shape features in its transient memory, and can re-use its knowledge about cutting at F and G. UCT considers each state uniquely, and must re-search each continuation.

In addition we ignore local shape features consisting of entirely empty intersections; we clear the eligibility trace for exploratory actions; and we use the default policy described in (Gelly et al., 2006) after the first $D = 10$ moves of each simulation. We refer to the complete algorithm as *Dyna-2-Shape*, and implement this algorithm in our program *RLGO*, which executes almost 2000 complete episodes of simulation per second on a 3 GHz processor.

For comparison, we implemented the UCT algorithm, based on the description in (Gelly et al., 2006). We use an identical default policy to the Dyna-2-Shape algorithm, to select moves when outside of the search tree, and a first play urgency of 1. We evaluate both programs by running matches against GnuGo, a standard benchmark program for Computer Go.

We compare the performance of local shape features in the permanent memory alone; in the transient memory alone; and in both the permanent and transient memories. We also compare the performance of local shape features of different sizes (see Figure 3). Using only the transient memory, Dyna-2-Shape outperformed UCT by a small margin. Using Dyna-2-Shape with both permanent and transient memories provided the best results, and outperformed UCT by a significant margin.

Local shape features would normally be considered naive in the domain of Go: the majority of shapes and tactics described in Go textbooks span considerably larger regions of the board than 3×3 squares. Indeed, when used only in the permanent memory, the local shape features win just 5% of games against GnuGo. However, when used in the transient memory, even the $\phi^{square(1,2)}$ features achieve performance comparable to UCT. Unlike UCT, the transient memory can generalise in terms of local responses: for example, it quickly learns the importantance of black connecting when white threatens to cut (Figures 1c and 1d).

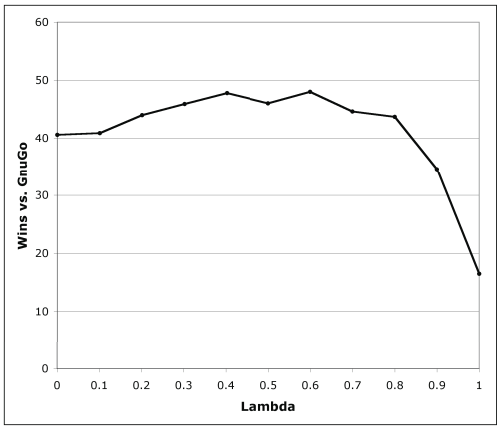


Figure 2. Winning rate of RLGO against GnuGo 3.7.10 (level 0) in 9×9 Go, using Dyna-2-Shape with 1000 simulations/move, for different values of $\bar{\lambda}$. Each point represents the winning percentage over 1000 games.

We also study the effect of the temporal difference parameter $\bar{\lambda}$ in the search procedure (see Figure 2). We see that bootstrapping ($\bar{\lambda} < 1$) provides significant benefits. Previous work in sample-based search has largely been restricted to Monte-Carlo methods (Tesauro & Galperin, 1996; Kocsis & Szepesvari, 2006; Gelly et al., 2006; Gelly & Silver, 2007; Coulom, 2007). Our results suggest that generalising these approaches to temporal difference learning methods may provide significant benefits when value function approximation is used.

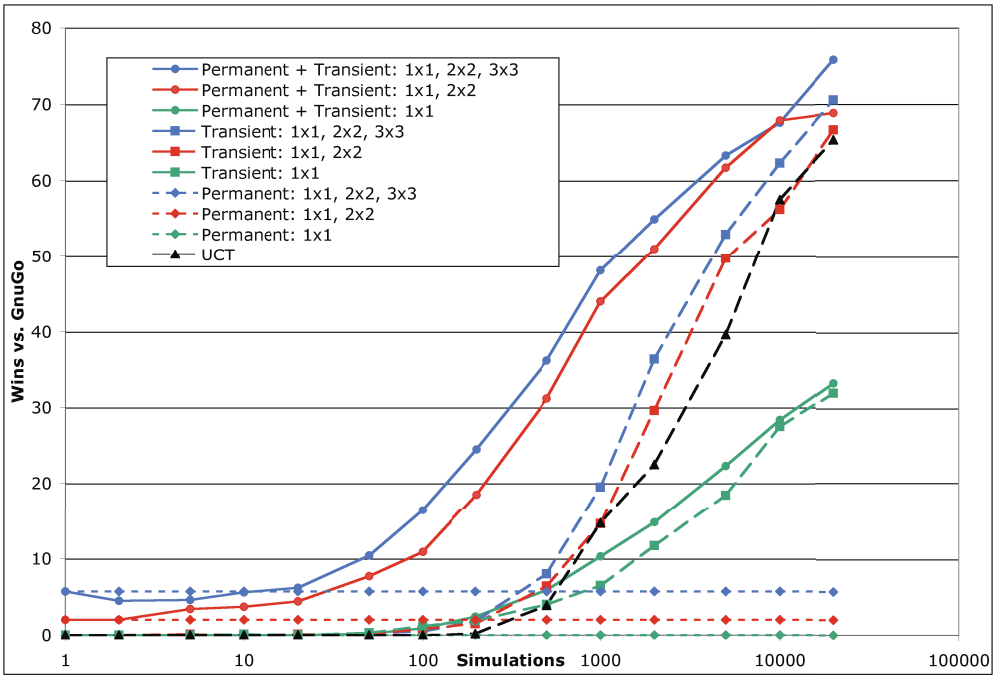


Figure 3. Winning rate of RLGO against GnuGo 3.7.10 (level 0) in 9×9 Go, using Dyna-2-Shape for different simulations/move. Local shape features are used in either the permanent memory (dotted lines), the transient memory (dashed lines), or both memories (solid lines). The permanent memory is trained offline from 100,000 games of self-play. Local shape features varied in size from 1×1 up to 3×3. Each point represents the winning percentage over 1000 games.

6. Dyna-2 and Heuristic Search

In games such as Chess, Checkers and Othello, master level play has been achieved by combining a heuristic evaluation function with α-β search. The heuristic is typically approximated by a linear combination of binary features, and can be learned offline by temporal-difference learning and self-play (Baxter et al., 1998; Schaeffer et al., 2001; Buro, 1999). Similarly, in the permanent memory of our architecture, the value function is approximated by a linear combination of binary features, learned offline by temporal-difference learning and self-play (Silver et al., 2007). Thus it is natural to compare Dyna-2 with approaches based on α-β search.

Dyna-2 combines a permanent memory with a transient memory, using sample-based search. In contrast, the classical approach uses the permanent memory $Q(s, a)$ as an evaluation function for α-β search. A hybrid approach is also possible, in which the combined value function $\bar{Q}(s, a)$ is used as an evaluation function for α-β search, including both permanent and transient memories. This can be viewed as searching with a dynamic evaluation function that evolves according to the current context. We compare all three approaches in Figure 4.

Dyna-2 outperformed classical search by a wide margin. In the game of Go, the consequences of a particular move (for example, playing good shape as in Figure 1a) may not become apparent for tens or even hundreds of moves. In a full-width search these consequences remain beyond the horizon, and will only be recognised if represented by the evaluation function. In contrast, sample-based search only uses the permanent memory as an initial guide, and learns to identify the consequences of particular patterns in the current situation. The hybrid approach successfully combines this knowledge with the precise lookahead provided by full-width search.

Using the hybrid approach, our program RLGO established an Elo rating of 2130 on the Computer Go Online Server, more than any handcrafted or traditional search program.

7. Related work

The Computer Go program MoGo uses the heuristic UCT algorithm (Gelly & Silver, 2007) to achieve *dan*-level performance. This algorithm can be viewed as an instance of Dyna-2 with local shape features in the permanent memory, and table lookup in the transient memory. It uses a step-size of $\bar{\alpha}(s, a) =$

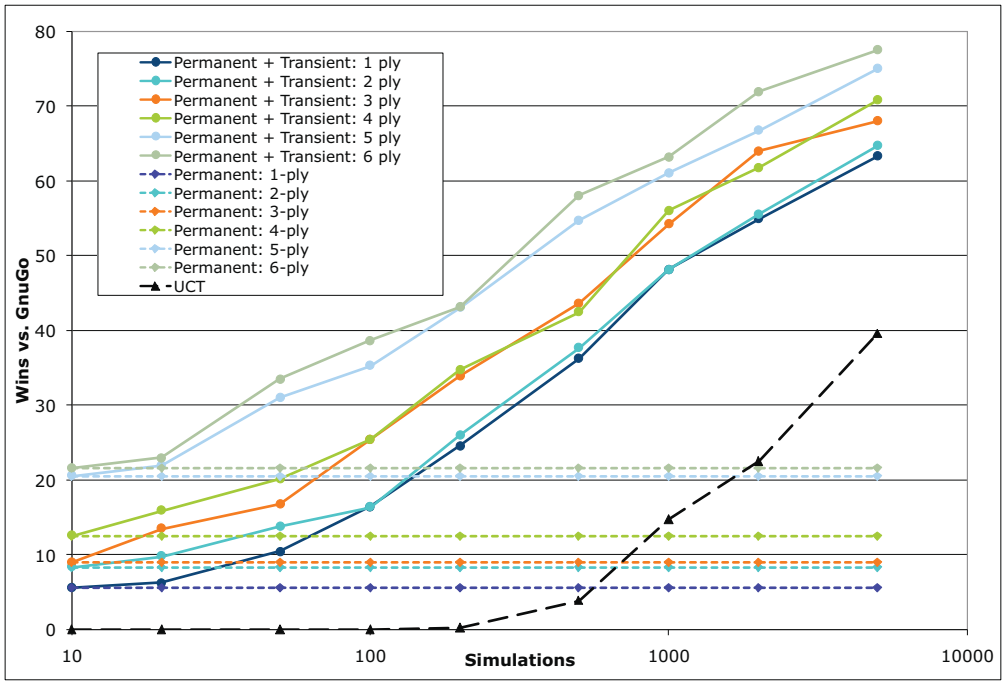


Figure 4. Winning rate of RLGO against GnuGo 3.7.10 (level 0) in 9×9 Go, using Dyna-2-Shape. A full-width α-β search is used for move selection, using a value function based on either the permanent memory (dotted lines), or both memories (solid lines). A 1-ply search corresponds to the usual move selection procedure in Dyna-2. For comparison, a 5-ply search takes approximately the same computation time as 1000 simulations. The permanent memory is trained offline from 100,000 games of self-play. Each point represents the winning percentage over 1000 games.

$1/(n_{prior}(s,a)+n(s,a))$. The confidence in the permanent memory is specified by n_{prior} in terms of *equivalent experience*, i.e. the worth of the permanent memory, measured in episodes of simulated experience.

In addition, MoGo uses the *Rapid Action Value Estimate* (RAVE) algorithm in its transient memory (Gelly & Silver, 2007). This algorithm can also be viewed as a special case of the Dyna-2 architecture, but using features of the full history h_t and not just the current state s_t and action a_t.

We define a history to be a sequence of states and actions $h_t = s_1 a_1 ... s_t a_t$, including the current action a_t. An individual RAVE feature $\phi_{sa}^{RAVE}(h)$ is a binary feature of the history h that matches a particular state s and action a. The binary feature is on iff s occurs in the history and a matches the current action a_t,

$$\phi_{sa}^{RAVE}(s_1 a_1 ... s_t a_t) = \begin{cases} 1 & \text{if } a_t = a \text{ and } \exists i \text{ s.t. } s_i = s; \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Thus the RAVE algorithm provides a simple abstraction over classes of related histories. The implementation of RAVE used in MoGo makes two additional simplifications. First, MoGo estimates a value for each RAVE feature independently of any other RAVE features, set to the average outcome of all simulations in which the RAVE feature ϕ_{sa}^{RAVE} is active. Second, for action selection, MoGo only evaluates the single RAVE feature $\phi_{s_t a_t}^{RAVE}$ corresponding to the current state s_t and candidate action a_t. This somewhat reduces the generalisation power of RAVE, but allows for a particularly efficient update procedure.

8. Conclusion

Reinforcement learning is often considered a slow procedure. Outstanding examples of success have, in the past, learned a value function from months of offline computation. However, this does not need to be the case. Many reinforcement learning methods are fast, incremental, and scalable. When such a reinforcement learning algorithm is applied to simulated experience, using a transient memory, it becomes a high performance search algorithm. This search procedure can be made more efficient by generalising across states; and it can be combined with long-term learning, using a permanent memory.

Monte-Carlo tree search algorithms, such as UCT, have recently received much attention. However, this

is just one example of a sample-based search algorithm. There is a spectrum of algorithms that vary from table-lookup to function approximation; from Monte-Carlo learning to bootstrapping; and from permanent to transient memories. Function approximation provides rapid generalisation in large domains; bootstrapping is advantageous in the presence of function approximation; and permanent and transient memories allow general knowledge about the past to be combined with specific knowledge about the expected future. By varying these dimensions, we have achieved a significant improvement over the UCT algorithm.

In 9×9 Go, programs based on extensions to the UCT algorithm have achieved *dan*-level performance. Our program RLGO, based on the Dyna-2 architecture, is the strongest program not based on UCT, and suggests that the full spectrum of sample-based search methods merits further investigation. For larger domains, such as 19×19 Go, generalising across states becomes increasingly important. Combining state abstraction with sample-based search is perhaps the most promising avenue for achieving human-level performance in this challenging domain.

References

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, *47*, 235–256.

Baxter, J., Tridgell, A., & Weaver, L. (1998). Experiments in parameter learning using temporal differences. *International Computer Chess Association Journal*, *21*, 84–99.

Buro, M. (1999). From simple features to sophisticated evaluation functions. *First International Conference on Computers and Games* (pp. 126–145).

Coulom, R. (2007). Computing Elo ratings of move patterns in the game of Go. *Computer Games Workshop*.

Dahl, F. (1999). Honte, a Go-playing program using neural nets. *Machines that learn to play games* (pp. 205–223). Nova Science.

Enzenberger, M. (2003). Evaluation in Go by a neural network using soft segmentation. *10th Advances in Computer Games Conference* (pp. 97–108).

Gelly, S., & Silver, D. (2007). Combining online and offline learning in UCT. *17th International Conference on Machine Learning* (pp. 273–280).

Gelly, S., Wang, Y., Munos, R., & Teytaud, O. (2006). *Modification of UCT with patterns in Monte-Carlo Go* (Technical Report 6062). INRIA.

Kocsis, L., & Szepesvari, C. (2006). Bandit based Monte-Carlo planning. *15th European Conference on Machine Learning* (pp. 282–293).

Paduraru, C. (2007). Planning with approximate and learned MDP models. Master's thesis, University of Alberta.

Rummery, G., & Niranjan, M. (1994). *On-line Q-learning using connectionist systems* (Technical Report CUED/F-INFENG/TR 166). Cambridge University Engineering Department.

Schaeffer, J., Hlynka, M., & Jussila, V. (2001). Temporal difference learning applied to a high-performance game-playing program. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (pp. 529–534).

Schraudolph, N., Dayan, P., & Sejnowski, T. (1994). Temporal difference learning of position evaluation in the game of Go. *Advances in Neural Information Processing 6* (pp. 817–824).

Silver, D., Sutton, R., & Müller, M. (2007). Reinforcement learning of local shape in the game of Go. *20th International Joint Conference on Artificial Intelligence* (pp. 1053–1058).

Sutton, R. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *7th International Conference on Machine Learning* (pp. 216–224).

Sutton, R. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems 8* (pp. 1038–1044).

Sutton, R., & Barto, A. (1998). *Reinforcement learning: an introduction*. MIT Press.

Sutton, R., Koop, A., & Silver, D. (2007). On the role of tracking in stationary environments. *17th International Conference on Machine Learning* (pp. 871–878).

Tesauro, G., & Galperin, G. (1996). On-line policy improvement using Monte-Carlo search. *Advances in Neural Information Processing 9* (pp. 1068–1074).

An RKHS for Multi-View Learning and Manifold Co-Regularization

Vikas Sindhwani VSINDHW@US.IBM.COM
Mathematical Sciences, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA

David S. Rosenberg DROSEN@STAT.BERKELEY.EDU
Department of Statistics, University of California Berkeley, CA 94720 USA

Abstract

Inspired by co-training, many multi-view semi-supervised kernel methods implement the following idea: find a function in each of multiple Reproducing Kernel Hilbert Spaces (RKHSs) such that (a) the chosen functions make similar predictions on unlabeled examples, and (b) the average prediction given by the chosen functions performs well on labeled examples. In this paper, we construct a single RKHS with a data-dependent "co-regularization" norm that reduces these approaches to standard supervised learning. The reproducing kernel for this RKHS can be explicitly derived and plugged into any kernel method, greatly extending the theoretical and algorithmic scope of co-regularization. In particular, with this development, the Rademacher complexity bound for co-regularization given in (Rosenberg & Bartlett, 2007) follows easily from well-known results. Furthermore, more refined bounds given by localized Rademacher complexity can also be easily applied. We propose a co-regularization based algorithmic alternative to manifold regularization (Belkin et al., 2006; Sindhwani et al., 2005a) that leads to major empirical improvements on semi-supervised tasks. Unlike the recently proposed transductive approach of (Yu et al., 2008), our RKHS formulation is truly semi-supervised and naturally extends to unseen test data.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

1. Introduction

In semi-supervised learning, we are given a few labeled examples together with a large collection of unlabeled data from which to estimate an unknown target function. Suppose we have two hypothesis spaces, $\mathcal{H}^1$ and $\mathcal{H}^2$, each of which contains a predictor that well-approximates the target function. We know that *predictors that agree with the target function also agree with each other on unlabeled examples*. Thus, any predictor in one hypothesis space that does not have an "agreeing predictor" in the other can be safely eliminated from consideration. Due to the resulting reduction in the complexity of the joint learning problem, one can expect improved generalization performance.

These conceptual intuitions and their algorithmic instantiations together constitute a major line of work in semi-supervised learning. One of the earliest approaches in this area was "co-training" (Blum & Mitchell, 1998), in which $\mathcal{H}^1$ and $\mathcal{H}^2$ are defined over different representations, or "views", of the data, and trained alternately to maximize mutual agreement on unlabeled examples. More recently, several papers have formulated these intuitions as joint complexity regularization, or *co-regularization*, between $\mathcal{H}^1$ and $\mathcal{H}^2$ which are taken to be Reproducing Kernel Hilbert Spaces (RKHSs) of functions defined on the input space $\mathcal{X}$. Given a few labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{i\in L}$ and a collection of unlabeled data $\{\boldsymbol{x}_i\}_{i\in U}$, co-regularization learns a prediction function,

$$f_\star(\boldsymbol{x}) = \frac{1}{2}\big(f^1_\star(\boldsymbol{x}) + f^2_\star(\boldsymbol{x})\big) \tag{1}$$

where $f^1_\star \in \mathcal{H}^1$ and $f^2_\star \in \mathcal{H}^2$ are obtained by solving the following optimization problem,

$$(f^1_\star, f^2_\star) = \operatorname*{argmin}_{f^1\in\mathcal{H}^1, f^2\in\mathcal{H}^2} \gamma_1||f^1||^2_{\mathcal{H}^1} + \gamma_2||f^2||^2_{\mathcal{H}^2} + \mu\sum_{i\in U}[f^1(\boldsymbol{x}_i) - f^2(\boldsymbol{x}_i)]^2 + \sum_{i\in L} V\left(y_i, f(\boldsymbol{x}_i)\right) \tag{2}$$

In this objective function, the first two terms measure complexity by the RKHS norms $\|\cdot\|^2_{\mathcal{H}_1}$ and $\|\cdot\|^2_{\mathcal{H}_2}$ in $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively, the third term enforces agreement among predictors on unlabeled examples, and the final term evaluates the empirical loss of the mean function $f = (f^1 + f^2)/2$ on the labeled data with respect to a loss function $V(\cdot,\cdot)$. The real-valued parameters γ_1, γ_2, and μ allow different tradeoffs between the regularization terms. L and U are index sets over labeled and unlabeled examples respectively.

Several variants of this formulation have been proposed independently and explored in different contexts: linear logistic regression (Krishnapuram et al., 2005), regularized least squares classification (Sindhwani et al., 2005b), regression (Brefeld et al., 2006), support vector classification (Farquhar et al., 2005), Bayesian co-training (Yu et al., 2008), and generalization theory (Rosenberg & Bartlett, 2007).

The main theoretical contribution of this paper is the construction of a new "co-regularization RKHS," in which standard supervised learning recovers the solution to the co-regularization problem of Eqn. 2. Theorem 2.2 presents the RKHS and gives an explicit formula for its reproducing kernel. This "co-regularization kernel" can be plugged into any standard kernel method giving convenient and immediate access to two-view semi-supervised techniques for a wide variety of learning problems. Utilizing this kernel, in Section 3 we give much simpler proofs of the results of (Rosenberg & Bartlett, 2007) concerning bounds on the Rademacher complexity and generalization performance of co-regularization. As a more algorithmic application, in Section 4 we consider the semi-supervised learning setting where examples live near a low-dimensional manifold embedded in a high dimensional ambient euclidean space. Our approach, manifold co-regularization (CoMR), gives major empirical improvements over the manifold regularization (MR) framework of (Belkin et al., 2006; Sindhwani et al., 2005a).

The recent work of (Yu et al., 2008) considers a similar reduction. However, this reduction is strictly transductive and does not allow prediction on unseen test examples. By contrast, our formulation is truly semi-supervised and provides a principled out-of-sample extension.

2. An RKHS for Co-Regularization

We start by reformulating the co-regularization optimization problem, given in Eqn. 1 and Eqn. 2, in the following equivalent form where we directly solve for the final prediction function $f_\star$:

$$f_\star = \operatorname*{argmin}_f \min_{\substack{f=f^1+f^2 \\ f^1\in\mathcal{H}^1, f^2\in\mathcal{H}^2}} \frac{\gamma_1}{2}||f^1||^2_{\mathcal{H}^1} + \frac{\gamma_2}{2}||f^2||^2_{\mathcal{H}^2} + \frac{\mu}{2}\sum_{i\in U}[f^1(\boldsymbol{x}_i) - f^2(\boldsymbol{x}_i)]^2 + \frac{1}{2}\sum_{i\in L} V\left(y_i, \frac{1}{2}f(\boldsymbol{x}_i)\right) \quad (3)$$

Consider the sum space of functions, $\tilde{\mathcal{H}}$, given by,

$$\begin{aligned}\tilde{\mathcal{H}} &= \mathcal{H}^1 \oplus \mathcal{H}^2 \quad (4)\\ &= \{f | f(\boldsymbol{x}) = f^1(\boldsymbol{x}) + f^2(\boldsymbol{x}), f^1 \in \mathcal{H}^1, f^2 \in \mathcal{H}^2\}\end{aligned}$$

and impose on it a data-dependent norm,

$$\|f\|^2_{\tilde{\mathcal{H}}} = \min_{\substack{f=f^1+f^2 \\ f^1\in\mathcal{H}^1, f^2\in\mathcal{H}^2}} \gamma_1\|f^1\|^2_{\mathcal{H}^1} + \gamma_2\|f^2\|^2_{\mathcal{H}^2} + \mu\sum_{i\in U}\left[f^1(\boldsymbol{x}_i) - f^2(\boldsymbol{x}_i)\right]^2 \quad (5)$$

The minimization problem in Eqn. 3 can then be posed as standard supervised learning in $\tilde{\mathcal{H}}$ as follows,

$$f_\star = \operatorname*{argmin}_{f\in\tilde{\mathcal{H}}} \gamma\|f\|^2_{\tilde{\mathcal{H}}} + \frac{1}{2}\sum_{i\in L} V\left(y_i, \frac{1}{2}f(\boldsymbol{x}_i)\right) \quad (6)$$

where $\gamma = \frac{1}{2}$. Of course, this reformulation is not really useful unless $\tilde{\mathcal{H}}$ itself is a valid new RKHS. Let us recall the definition of an RKHS.

Definition 2.1 (RKHS). *A reproducing kernel Hilbert space (RKHS) is a Hilbert Space $\mathcal{F}$ that possesses a reproducing kernel,* i.e., *a function $k : \mathcal{X}\times\mathcal{X} \to \mathcal{R}$ for which the following hold: (a) $k(\boldsymbol{x}, .) \in \mathcal{F}$ for all $\boldsymbol{x} \in \mathcal{X}$, and (b) $\langle f, k(\boldsymbol{x}, .)\rangle_{\mathcal{F}} = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$ and $f \in \mathcal{F}$, where $\langle\cdot,\cdot\rangle_{\mathcal{F}}$ denotes inner product in $\mathcal{F}$.*

In Theorem 2.2, we show that $\tilde{\mathcal{H}}$ is indeed an RKHS, and moreover we give an explicit expression for its reproducing kernel. Thus, it follows that although the domain of optimization in Eqn. 6 is nominally a function space, by the Representer Theorem we can express it as a finite-dimensional optimization problem.

2.1. Co-Regularization Kernels

Let $\mathcal{H}^1, \mathcal{H}^2$ be RKHSs with kernels given by k^1, k^2 respectively, and let $\tilde{\mathcal{H}} = \mathcal{H}^1 \oplus \mathcal{H}^2$ as defined in Eqn. 4. We have the following result.

Theorem 2.2. *There exists an inner product on $\tilde{\mathcal{H}}$ for which $\tilde{\mathcal{H}}$ is a RKHS with norm defined by Eqn. 5 and reproducing kernel $\tilde{k} : \mathcal{X}\times\mathcal{X} \to \mathcal{R}$ given by,*

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{z}) = s(\boldsymbol{x}, \boldsymbol{z}) - \mu\mathbf{d}_{\boldsymbol{x}}^T H \mathbf{d}_{\boldsymbol{z}} \quad (7)$$

where $s(\boldsymbol{x}, \boldsymbol{z})$ is the (scaled) sum of kernels given by,

$$s(\boldsymbol{x}, \boldsymbol{z}) = \gamma_1^{-1} k^1(\boldsymbol{x}, \boldsymbol{z}) + \gamma_2^{-1} k^2(\boldsymbol{x}, \boldsymbol{z}),$$

and $\mathbf{d}_{\boldsymbol{x}}$ is a vector-valued function that depends on the difference in views measured as,

$$\mathbf{d}_{\boldsymbol{x}} = \gamma_1^{-1} \mathbf{k}^1_{U\boldsymbol{x}} - \gamma_2^{-1} \mathbf{k}^2_{U\boldsymbol{x}},$$

where $\mathbf{k}^i_{U\boldsymbol{x}} = \left[k^i(\boldsymbol{x}, \boldsymbol{x}_j), j \in U\right]^T$, and H is a positive-definite matrix given by $H = (I + \mu S)^{-1}$. Here, S is the gram matrix of $s(\cdot, \cdot)$, i.e., *$S = \left(\gamma_1^{-1} K^1_{UU} + \gamma_2^{-1} K^2_{UU}\right)$ where $K^i_{UU} = k^i(U, U)$ denotes the Gram matrices of k^i over unlabeled examples.*

We give a rigorous proof in Appendix A.

2.2. Representer Theorem

Theorem 2.2 says that $\tilde{\mathcal{H}}$ is a valid RKHS with kernel $\tilde{k}$. By the Representer Theorem, the solution to Eqn.6 is given by

$$f_\star(\boldsymbol{x}) = \sum_{i \in L} \alpha_i \tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}) \tag{8}$$

The corresponding components in $\mathcal{H}^1, \mathcal{H}^2$ can also be retrieved as,

$$f^1_\star(\boldsymbol{x}) = \sum_{i \in L} \alpha_i \gamma_1^{-1} \left(k^1(\boldsymbol{x}_i, \boldsymbol{x}) - \mu \mathbf{d}^T_{\boldsymbol{x}_i} H k^1_{U\boldsymbol{x}}\right) \tag{9}$$

$$f^2_\star(\boldsymbol{x}) = \sum_{i \in L} \alpha_i \gamma_2^{-1} \left(k^2(\boldsymbol{x}_i, \boldsymbol{x}) + \mu \mathbf{d}^T_{\boldsymbol{x}_i} H k^2_{U\boldsymbol{x}}\right) \tag{10}$$

Note that $\mathcal{H}^1$ and $\mathcal{H}^2$ are defined on the same domain $\mathcal{X}$ so that taking the mean prediction is meaningful. In a two-view problem one may begin by defining $\mathcal{H}^1, \mathcal{H}^2$ on different view spaces $\mathcal{X}^1, \mathcal{X}^2$ respectively. Such a problem can be mapped to our framework by extending $\mathcal{H}^1, \mathcal{H}^2$ to $\mathcal{X} = \mathcal{X}^1 \times \mathcal{X}^2$ by re-defining $f^1(\boldsymbol{x}^1, \boldsymbol{x}^2) = f^1(\boldsymbol{x}^1), f^1 \in \mathcal{H}^1$; similarly for $\mathcal{H}^2$. While we omit these technical details, it is important to note that in such cases, Eqns. 9 and 10 can be reinterpreted as predictors defined on $\mathcal{X}^1, \mathcal{X}^2$ respectively.

3. Bounds on Complexity and Generalization

By eliminating all predictors that do not collectively agree on unlabeled examples, co-regularization intuitively reduces the complexity of the learning problem. It is reasonable then to expect better test performance for the same amount of labeled training data. In (Rosenberg & Bartlett, 2007), the size of the co-regularized function class is measured by its empirical Rademacher complexity, and tight upper and lower bounds are given on the Rademacher complexity of the co-regularized hypothesis space. This leads to generalization bounds in terms of the Rademacher complexity. In this section, we derive these complexity bounds in a few lines using Theorem 2.2 and a well-known result on RKHS balls. Furthermore, we present improved generalization bounds based on the theory of localized Rademacher complexity.

3.1. Rademacher Complexity Bounds

Definition 3.1. *The empirical Rademacher complexity of a function class $\mathcal{A} = \{f : \mathcal{X} \to \mathcal{R}\}$ on a sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell \in \mathcal{X}$ is defined as*

$$\hat{R}_\ell(\mathcal{A}) = E^\sigma \left[\sup_{f \in \mathcal{A}} \left| \frac{2}{\ell} \sum_{i=1}^{\ell} \sigma_i f(\boldsymbol{x}_i) \right| \right],$$

where the expectation is with respect to $\sigma = \{\sigma_1, \ldots, \sigma_\ell\}$, and the σ_i are i.i.d. Rademacher random variables, that is, $P(\sigma_i = 1) = P(\sigma_i = -1) = \frac{1}{2}$.

Let $\mathcal{H}$ be an arbitrary RKHS with kernel $k(\cdot, \cdot)$, and denote the standard RKHS supervised learning objective function by $Q(f) = \sum_{i \in L} V(y_i, f(\boldsymbol{x}_i)) + \lambda ||f||^2_{\mathcal{H}}$. Let $f_\star = \operatorname{argmin}_{f \in \mathcal{H}} Q(f)$. Then $Q(f_\star) \leq Q(0) = \sum_{i \in L} V(y_i, 0)$. It follows that $\|f_\star\|^2_{\mathcal{H}} \leq Q(0)/\lambda$. Thus if we have some control *a priori* on $Q(0)$, then we can restrict the search for $f_\star$ to a ball in $\mathcal{H}$ of radius $r = \sqrt{Q(0)/\lambda}$.

We now cite a well-known result about the Rademacher complexity of a ball in an RKHS (see *e.g.* (Boucheron et al., 2005)). Let $\mathcal{H}_r := \{f \in \mathcal{H} : ||f||_{\mathcal{H}} \leq r\}$ denote the ball of radius r in $\mathcal{H}$. Then we have the following:

Lemma 3.2. *The empirical Rademacher complexity on the sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell \in \mathcal{X}$ for the RKHS ball $\mathcal{H}_r$ is bounded as follows: $\frac{1}{\sqrt[4]{2}} \frac{2r}{\ell} \sqrt{\mathrm{tr} K} \leq \hat{R}_\ell(\mathcal{H}_r) \leq \frac{2r}{\ell} \sqrt{\mathrm{tr} K}$ where $K = \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{i,j=1}^{\ell}$ is the kernel matrix.*

For the co-regularization problem described in Eqns. 3 and 6, we have $f_\star \in \tilde{\mathcal{H}}_r$ where $r^2 = \ell \sup_y V(0, y)$, where ℓ is number of labeled examples. We now state and prove bounds on the Rademacher complexity of $\tilde{\mathcal{H}}_r$. The bounds here are exactly the same as those given in (Rosenberg & Bartlett, 2007). However, while they have a lengthy "bare-hands" approach, here we get the result as a simple corollary of Theorem 2.2 and Lemma 3.2.

Theorem 3.3. *The empirical Rademacher complexity on the labeled sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell \in \mathcal{X}$ for the RKHS ball $\tilde{\mathcal{H}}_r$ is bounded as follows:*

$$\frac{1}{\sqrt[4]{2}} \frac{2r}{\ell} \sqrt{\mathrm{tr} \tilde{K}} \leq \hat{R}_\ell(\tilde{\mathcal{H}}_r) \leq \frac{2r}{\ell} \sqrt{\mathrm{tr} \tilde{K}},$$

where
$\tilde{K} = \gamma_1^{-1} K^1_{LL} + \gamma_2^{-1} K^2_{LL} - \mu D^T_{UL} (I + \mu S)^{-1} D_{UL}$ *and*
$D_{UL} = \left(\gamma_1^{-1} K^1_{UL} - \gamma_2^{-1} K^2_{UL}\right)$

Proof. Note that $\tilde{K}$ is just the kernel matrix for the co-regularization kernel $\tilde{k}(\cdot,\cdot)$ on the labeled data. Then the bound follows immediately from Lemma 3.2. □

3.2. Co-Regularization Reduces Complexity

The co-regularization parameter μ controls the extent to which we enforce agreement between f^1 and f^2. Let $\tilde{\mathcal{H}}(\mu)$ denote the co-regularization RKHS for a particular value of μ. From Theorem 3.3, we see that the Rademacher complexity for a ball of radius r in $\tilde{\mathcal{H}}(\mu)$ decreases with μ by an amount determined by

$$\Delta(\mu) = \operatorname{tr}\left[\mu D^T_{UL} (I + \mu S)^{-1} D_{UL}\right] \quad (11)$$

$$= \sum_{i=1}^{\ell} \rho^2 \left(\mathbf{k}^1_{U\boldsymbol{x}_i} \mathbf{k}^2_{U\boldsymbol{x}_i}\right) \quad (12)$$

where $\rho(\cdot,\cdot)$ is a metric on $\mathcal{R}^{|U|}$ defined by
$\rho^2(\mathbf{s},\mathbf{t}) = \mu(\gamma_1^{-1}\mathbf{s} - \gamma_2^{-1}\mathbf{t})' (I + \mu S)^{-1} (\gamma_1^{-1}\mathbf{s} - \gamma_2^{-1}\mathbf{t})$

We see that the complexity reduction, $\Delta(\mu)$, grows with the ρ-distance between the two different representations of the labeled points. Note that the metric ρ is determined by S, which is the weighted sum of the gram matrices of the two kernels on unlabeled data.

3.3. Generalization Bounds

With Theorem 2.2 allowing us to express multi-view co-regularization problems as supervised learning in a data-dependent RKHS, we can now bring a large body of theory to bear on the generalization performance of co-regularization methods. We start by quoting the theorem proved in (Rosenberg & Bartlett, 2007). Next, we state an improved bound based on localized Rademacher complexity. Below, we denote the unit ball in $\tilde{\mathcal{H}}$ by $\tilde{\mathcal{H}}_1$.

Condition 1. The loss $V(\cdot,\cdot)$ is Lipschitz in its first argument, i.e., there exists a constant A such that $\forall y, \hat{y}_1, \hat{y}_2$: $|V(\hat{y}_1, y) - V(\hat{y}_2, y)| \leq A\,|\hat{y}_1 - \hat{y}_2|$

Theorem 3.4. *Suppose $V : \mathcal{Y}^2 \to [0,1]$ satisfies Condition 1. Then conditioned on the unlabeled data, for any $\delta \in (0,1)$, with probability at least $1-\delta$ over the sample of labeled points $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ drawn i.i.d. from P, we have for any predictor $f \in \tilde{\mathcal{H}}_1$ that*

$$P[V(\varphi(\boldsymbol{x}), y)] \leq \frac{1}{\ell}\sum_{i=1}^{\ell} V(\varphi(\boldsymbol{x}_i), y_i) + 2B\hat{R}_\ell(\tilde{\mathcal{H}}_1) + \frac{1}{\sqrt{\ell}}\left(2 + 3\sqrt{\ln(2/\delta)/2}\right)$$

We need two more conditions for the localized bound:

Condition 2. For any probability distribution P, there exists $f_\star \in \tilde{\mathcal{H}}_1$ satisfying $P[V(f_\star(\boldsymbol{x}), y)] = \inf_{f \in \tilde{\mathcal{H}}_1} P[V(f(\boldsymbol{x}), y)]$

Condition 3. There is a constant $B \geq 1$ such that for every probability distribution P and every $f \in \tilde{\mathcal{H}}_1$ we have, $P(f - f_*)^2 \leq BP(V[f(\boldsymbol{x}), y] - V[f_\star(\boldsymbol{x}), y])$

In the following theorem, let P_ℓ denote the empirical probability measure for the labeled sample of size ℓ.

Theorem 3.5. *[Cor. 6.7 from (Bartlett et al., 2002)] Assume that $\sup_{\boldsymbol{x}\in\mathcal{X}} k(\boldsymbol{x},\boldsymbol{x}) \leq 1$ and that V satisfies the 3 conditions above. Let $\hat{f}$ be any element of $\tilde{\mathcal{H}}_1$ satisfying $P_\ell V[\hat{f}(\boldsymbol{x}), y] = \inf_{f\in\tilde{\mathcal{H}}_1} P_\ell V[f(\boldsymbol{x}), y]$. There exist a constant c depending only on A and B s.t. with probability at least $1 - 6e^{-\nu}$,*

$$P\left(V[\hat{f}(\boldsymbol{x}), y] - V[f_\star(\boldsymbol{x}), y]\right) \leq c\left(\hat{r}^* + \frac{\nu}{\ell}\right),$$

where $\hat{r}^* \leq \min_{0\leq h\leq \ell}\left(\frac{h}{\ell} + \frac{1}{\ell}\sqrt{\sum_{i>h}\lambda_i}\right)$ and where $\lambda_1, \ldots, \lambda_\ell$ are the eigenvalues of the labeled-data kernel matrix $\tilde{K}_{LL}$ in decreasing order.

Note that while Theorem 3.4 bounds the gap between expected and empirical performance of an arbitrary $f \in \tilde{\mathcal{H}}_1$, Theorem 3.5 bounds the gap between the empirical loss minimizer over $\tilde{\mathcal{H}}_1$ and true risk minimizer in $\tilde{\mathcal{H}}_1$. Since the localized bound only needs to account for the capacity of the function class in the neighborhood of f_*, the bounds are potentially tighter. Indeed, while the bound in Theorem 3.4 is in terms of the trace of the kernel matrix, the bound in Theorem 3.5 involves the tail sum of kernel eigenvalues. If the eigenvalues decay very quickly, the latter is potentially much smaller.

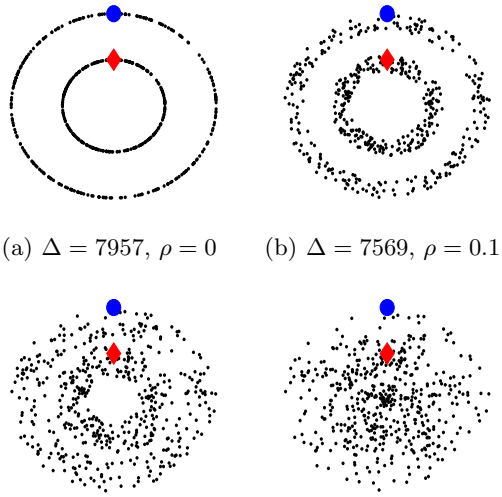
4. Manifold Co-Regularization

Consider the picture shown in Figure 1(a) where there are two classes of data points in the plane ($\mathcal{R}^2$) lying on one of two concentric circles. The large, colored points are labeled while the smaller, black points are unlabeled. The picture immediately suggests two notions of distance that are very natural but radically different. For example, the two labeled points are close in the *ambient* euclidean distance on $\mathcal{R}^2$, but infinitely apart in terms of *intrinsic* geodesic distance measured along the circles.

Suppose for this picture one had access to two kernel functions, k^1, k^2 that assign high similarity to nearby points according to euclidean and geodesic distance respectively. Because of the difference in ambient and intrinsic representations, by co-regularizing the associated RKHSs one can hope to get good reductions in

complexity, as suggested in section 3.2. In Figure 1, we report the value of complexity reduction (Eqn. 12) for four point clouds generated at increasing levels of noise off the two concentric circles. When noise becomes large, the ambient and intrinsic notions of distance converge and the amount of complexity reduction decreases.

Figure 1. Complexity Reduction $\Delta(\mu = 1)$ (Eqn. 12) with respect to noise level ρ. The choice of k^1, k^2 is discussed in the following subsections.



(a) $\Delta = 7957$, $\rho = 0$ (b) $\Delta = 7569$, $\rho = 0.1$

(c) $\Delta = 3720$, $\rho = 0.2$ (d) $\Delta = 3502$, $\rho = 0.5$

The setting where data lies on a low-dimensional submanifold $\mathcal{M}$ embedded in a higher dimensional ambient space $\mathcal{X}$, as in the concentric circles case above, has attracted considerable research interest recently, almost orthogonal to multi-view efforts. The main assumption underlying manifold-motivated semi-supervised learning algorithms is the following: *two points that are close with respect to geodesic distances on $\mathcal{M}$ should have similar labels.* Such an assumption may be enforced by an *intrinsic* regularizer that emphasizes complexity along the manifold.

Since $\mathcal{M}$ is truly unknown, the intrinsic regularizer is empirically estimated from the point cloud of labeled and unlabeled data. In the graph transduction approach, an *nn*-nearest neighbor graph $\mathcal{G}$ is constructed which serves as an empirical substitute for $\mathcal{M}$. The vertices $\mathcal{V}$ of this graph are the set of labeled and unlabeled examples. Let $\mathcal{H}_\mathcal{I}$ denote the space of all functions mapping $\mathcal{V}$ to $\mathcal{R}$, where the subscript $\mathcal{I}$ implies "intrinsic." Any function $f \in \mathcal{H}_\mathcal{I}$ can be identified with the vector $\boldsymbol{f} = [f(\boldsymbol{x}_i), \boldsymbol{x}_i \in \mathcal{V}]^T$. One can impose a norm $\|\boldsymbol{f}\|_\mathcal{I}^2 = \sum_{ij} W_{ij} \left[f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)\right]^2$ on $\mathcal{H}_\mathcal{I}$ that provides a natural measure of smoothness for $\boldsymbol{f}$ with respect to the graph. Here, W denotes the adjacency matrix of the graph. When $\mathcal{X}$ is a euclidean space, a typical W is given by $W_{ij} = \exp(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2})$ if i and j are nearest neighbors and 0 otherwise. In practice, one may use a problem dependent similarity matrix to set these edge weights. This norm can be conveniently written as a quadratic form $\boldsymbol{f}^T M \boldsymbol{f}$, where M is the graph Laplacian matrix defined as $M = D - W$, and D is a diagonal degree matrix with entrees $D_{ii} = \sum_j W_{ij}$.

It turns out that $\mathcal{H}_\mathcal{I}$ with the norm $\|\cdot\|_\mathcal{I}$ is an RKHS whose reproducing kernel $k_\mathcal{I} : \mathcal{V} \times \mathcal{V} \to \mathcal{R}$ is given by $k_\mathcal{I}(\boldsymbol{x}_i, \boldsymbol{x}_j) = M^\dagger_{ij}$, where $M^\dagger$ denotes the pseudo-inverse of the Laplacian. Given $\mathcal{H}_\mathcal{I}$ with its reproducing kernel, graph transduction solves the standard RKHS regularization problem, $f_\star = \operatorname{argmin}_{f \in \mathcal{H}_I} \gamma \|f\|_\mathcal{I}^2 + \sum_{i \in L} V(y_i, f(\boldsymbol{x}_i))$, where y_i is the label associated with the node $\boldsymbol{x}_i$. Note that the solution $f_\star$ is only defined over $\mathcal{V}$, the set of labeled and unlabeled examples. Since graph transduction does not provide a function whose domain is the ambient space $\mathcal{X}$, it is not clear how to make predictions on unseen test points $\boldsymbol{x} \in \mathcal{X}$. Possessing a principled "out-of-sample extension" distinguishes semi-supervised methods from transductive procedures.

4.1. Ambient and Intrinsic Co-Regularization

We propose a co-regularization solution for out-of-sample prediction. Conceptually, one may interpret the manifold setting as a multi-view problem where each labeled or unlabeled example appears in two "views": (a) an ambient view in $\mathcal{X}$ in terms of euclidean co-ordinates $\boldsymbol{x}$ and (b) an intrinsic view in $\mathcal{G}$ as a vertex index i. Let $\mathcal{H}_\mathcal{A} : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ be an RKHS defined over the ambient domain with an associated kernel $k_\mathcal{A} : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$. We can now combine ambient and intrinsic views by co-regularizing $\mathcal{H}_\mathcal{A}, \mathcal{H}_\mathcal{I}$. This can be done by setting $k^1 = k_\mathcal{A}$ and $k^2 = k_\mathcal{I}$ in Eqn. 7 and solving Eqn. 6. The combined prediction function $f_\star$ given by Eqn. 8 is the mean of an ambient component $f^1_\star$ given by Eqn. 9 and an intrinsic component $f^2_\star$ given by Eqn. 10. Even though $f_\star$ is transductive and only defined on labeled and unlabeled examples, the ambient component $f^1_\star$ can be used for out-of-sample prediction. Due to co-regularization, this ambient component is (a) smooth in $\mathcal{H}_\mathcal{X}$ and (b) in agreement with a smooth function on the data manifold. We call this approach manifold co-regularization, and abbreviate it as CoMR.

4.2. Manifold Regularization

In the manifold regularization (MR) approach of (Belkin et al., 2006; Sindhwani et al., 2005a), the

following optimization problem is solved:

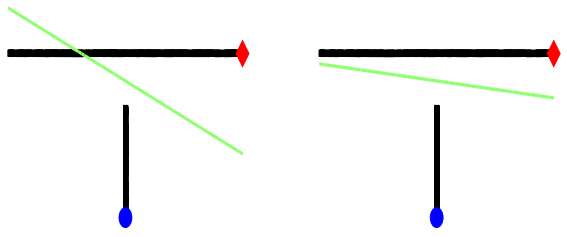
$$f^\star = \underset{f\in\mathcal{H}_\mathcal{A}}{\operatorname{argmin}}\, \gamma_1\|f\|^2_{\mathcal{H}_A} + \gamma_2 \boldsymbol{f}^T M \boldsymbol{f} + \sum_{i\in L} V(y_i, f(\boldsymbol{x}_i)) \quad (13)$$

where $\boldsymbol{f} = [f(\boldsymbol{x}_i), i \in L, U]^T$. The solution, $f^\star$ is defined on $\mathcal{X}$, and can therefore be used for out-of-sample prediction.

In Figure 2, we show a simple two-dimensional dataset where MR provably fails when $\mathcal{H}_\mathcal{A}$ is the space of linear functions on $\mathcal{R}^2$. The LINES dataset consists of two classes spread along perpendicular lines. In MR the intrinsic regularizer is enforced directly on $\mathcal{H}_\mathcal{A}$. It can be easily shown that the intrinsic norm of a linear function $f(\boldsymbol{x}) = \mathbf{w}^T\boldsymbol{x}$ along the perpendicular lines is exactly the same as the ambient norm, *i.e.*, $\|f\|^2_{\mathcal{H}_\mathcal{I}} = \|f\|^2_{\mathcal{H}_\mathcal{A}} = \mathbf{w}^T\mathbf{w}$. Due to this, MR simply ignores unlabeled data and reduces to supervised training with the regularization parameter $\gamma_1 + \gamma_2$.

The linear function that gives maximally smooth predictions on one line also gives the maximally non-smooth predictions on the other line. One way to remedy this restrictive situation is to introduce slack variables $\boldsymbol{\xi} = (\xi_i)_{i\in L\cup U}$ in Eqn. 13 with an ℓ_2 penalty, and instead solve: $f^\star = \operatorname{argmin}_{\boldsymbol{f}\in\mathcal{H}_\mathcal{A},\boldsymbol{\xi}}\, \gamma_1\|f\|^2_{\mathcal{H}_A} + \gamma_2(\boldsymbol{f}+\boldsymbol{\xi})^T M(\boldsymbol{f}+\boldsymbol{\xi}) + \mu\|\boldsymbol{\xi}\|^2 + \sum_{i\in L} V(y_i, f(\boldsymbol{x}_i))$. Re-parameterizing $\mathbf{g} = \boldsymbol{f} + \boldsymbol{\xi}$, we can re-write the above problem as, $f^\star = \operatorname{argmin}_{f\in\mathcal{H}_\mathcal{A}, g\in\mathcal{H}_\mathcal{I}}\, \gamma_1\|f\|^2_{\mathcal{H}_A} + \gamma_2\|g\|^2_{\mathcal{H}_\mathcal{I}} + \mu\|\boldsymbol{f}-\mathbf{g}\|^2 + \sum_{i\in L} V(y_i, f(\boldsymbol{x}_i))$, which may be viewed as a variant of the co-regularization problem in Eqn. 2 where empirical loss is measured for f alone. Thus, this motivates the view that CoMR adds extra slack variables in the MR objective function to better fit the intrinsic regularizer. Figure 2 shows that CoMR achieves better separation between classes on the LINES dataset.

Figure 2. Decision boundaries of MR and CoMR (using the quadratic hinge loss) on the LINES dataset



(a) MR (b) CoMR

4.3. Experiments

In this section, we compare MR and CoMR. Similar to our construction of the co-regularization kernel, (Sindhwani et al., 2005a) provide a data-dependent kernel that reduces manifold regularization to standard supervised learning in an associated RKHS. We write the manifold regularization kernel in the following form,

$$\tilde{k}^{mr}(\boldsymbol{x},\boldsymbol{z}) = \bar{s}(\boldsymbol{x},\boldsymbol{z}) - \bar{\mathbf{d}}_{\boldsymbol{x}}^T \bar{H} \bar{\mathbf{d}}_{\boldsymbol{z}} \quad (14)$$

where we have, $\bar{s} = \gamma_1^{-1}k^1(\boldsymbol{x},\boldsymbol{z})$, $\bar{\mathbf{d}}_{\boldsymbol{x}} = \gamma_1^{-1}k^1_{U\boldsymbol{x}}$ and $\bar{H} = \left(\gamma_1^{-1}\bar{K^1} + \gamma_2^{-1}\bar{K^2}\right)^{-1}$, where $\bar{K^1}$ is the Gram Matrix of $k^1 = k_\mathcal{A}$ over labeled and unlabeled examples, and $\bar{K^2} = M^\dagger$. We use the notation $\bar{s}, \bar{\mathbf{d}}, \bar{H}$ so that the kernel can be easily compared with corresponding quantities in the co-regularization kernel Eqn. 7. In this section we empirically compare this kernel with the co-regularization kernel of Eqn. 7 for exactly the same choice of k^1, k^2. Semi-supervised classification experiments were performed on 5 datasets described in table 1.

Table 1. Datasets with d features and c classes. 10 random data splits were created with l labeled, u unlabeled, t test, and v validation examples.

DATASET	d	c	l	u	t	v
LINES	2	2	2	500	250	250
G50C	50	2	50	338	112	50
USPST	256	10	50	1430	477	50
COIL20	241	20	40	1320	40	40
PCMAC	7511	2	50	1385	461	50

The LINES dataset is a variant of the two-dimensional problem shown in Figure 2 where we added random noise around the two perpendicular lines. The G50C, USPST, COIL20, and PCMAC datasets are well known and have frequently been used for empirical studies in semi-supervised learning literature. They were used for benchmarking manifold regularization in (Sindhwani et al., 2005a) against a number of competing methods. G50C is an artificial dataset generated from two unit covariance normal distributions with equal probabilities. The class means are adjusted so that the Bayes error is 5%. COIL20 consists of 32×32 gray scale images of 20 objects viewed from varying angles. USPST is taken from the test subset of the USPS dataset of images containing 10 classes of handwritten digits. PCMAC is used to setup binary text categorization problems drawn from the 20-newsgroups dataset.

For each of the 5 datasets, we constructed random splits into labeled, unlabeled, test and validation sets. The sizes of these sets are given in table 1. For all datasets except LINES, we used Gaussian am-

Table 2. Error Rates (in percentage) on Test Data

DATASET	MR	COMR
LINES	7.7 (1.2)	**1.0** (1.5)
G50C	5.8 (2.8)	5.5 (2.3)
USPST	18.2 (1.5)	**14.1** (1.6)
COIL20	23.8 (11.1)	**14.8** (8.8)
PCMAC	11.9 (3.4)	**8.9** (2.6)

Table 3. Error Rates (in percentage) on Unlabeled Data

DATASET	MR	COMR
LINES	7.5 (1.0)	**1.3** (2.0)
G50C	6.6 (0.8)	6.9 (1.0)
USPST	18.6 (1.4)	**13.3** (1.0)
COIL20	37.5 (6.0)	**14.8** (3.3)
PCMAC	11.0 (2.4)	**9.4** (1.9)

bient kernels $k^1(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2})$, and intrinsic graph kernel whose gram matrix is of the form $K_2 = (M^p + 10^{-6}I)^{-1}$. Here, M is the normalized Graph Laplacian constructed using nn nearest neighbors and p is an integer. These parameters are tabulated in Table 4 for reproducibility. For more details on these parameters see (Sindhwani et al., 2005a).

We chose squared loss for $V(\cdot,\cdot)$. Manifold regularization with this choice is also referred to as Laplacian RLS and empirically performs as well as Laplacian SVMs. For multi-class problems, we used the one-versus-rest strategy. γ_1, γ_2 were varied on a grid of values: $10^{-6}, 10^{-4}, 10^{-2}, 1, 10, 100$ and chosen with respect to validation error. The chosen parameters are also reported in Table 4. Finally, we evaluated the MR solution and the ambient component of CoMR on an unseen test set. In Tables 2 and 3 we report the mean and standard deviation of error rates on test and unlabeled examples with respect to 10 random splits. We performed a paired t-test at 5% significance level to assess the statistical significance of the results. Results shown in bold are statistically significant.

Our experimental protocol makes MR and CoMR exactly comparable. We find that CoMR gives major empirical improvements over MR on all datasets except G50C where both methods approach the Bayes error rate.

5. Conclusion

In this paper, we have constructed a single, new RKHS in which standard supervised algorithms are immediately turned into multi-view semi-supervised learners. This construction brings about significant theoretical simplifications and algorithmic benefits, which we have demonstrated in the context of generalization analysis and manifold regularization respectively.

Table 4. Parameters Used. Note $\mu = 1$ for CoMR. Linear kernel was used for LINES dataset.

DATASET	nn	σ	p	MR γ_1, γ_2	COMR γ_1, γ_2
LINES	10	–	1	$0.01, 10^{-6}$	$10^{-4}, 100$
G50C	50	17.5	5	$1, 100$	$10, 10$
USPST	10	9.4	2	$0.01, 0.01$	$10^{-6}, 10^{-4}$
COIL20	2	0.6	1	10^{-4},10^{-6}	$10^{-6}, 10^{-6}$
PCMAC	50	2.7	5	$10, 100$	$1, 10$

References

Bartlett, P. L., Bousquet, O., & Mendelson, S. (2002). Localized rademacher complexities. *COLT 02* (pp. 44–58). Springer-Verlag.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, *7*, 2399–2434.

Bertinet, A., & Thomas-Agnan, C. (2004). *Reproducing kernel hilbert spaces in probability and statistics*. Kluwer Academic Publishers.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *COLT*.

Boucheron, S., Bousquet, O., & Lugosi, G. (2005). Theory of classification: a survey of some recent advances. *ESAIM: P&S*, *9*, 323–375.

Brefeld, U., Gärtner, T., Scheffer, T., & Wrobel, S. (2006). Efficient co-regularised least squares regression. *ICML* (pp. 137–144).

Farquhar, J. D. R., Hardoon, D. R., Meng, H., Taylor, J. S., & Szedmák, S. (2005). Two view learning: SVM-2K, theory and practice. *NIPS*.

Krishnapuram, B., Williams, D., Xue, Y., & A. Hartemink, L. C. (2005). On semi-supervised classification. *NIPS*.

Rosenberg, D., & Bartlett, P. L. (2007). The Rademacher complexity of co-regularized kernel classes. *AISTATS*.

Sindhwani, V., Niyogi, P., & Belkin, M. (2005a). Beyond the point cloud: From transductive to semi-supervised learning. *ICML*.

Sindhwani, V., Niyogi, P., & Belkin, M. (2005b). A co-regularization approach to semi-supervised learning with multiple views. *ICML Workshop on Learning in Multiple Views* (pp. 824–831).

Yu, S., Krishnapuram, B., Rosales, R., Steck, H., & Rao, R. (2008). Bayesian co-training. *NIPS*.

A. Proof of Theorem 2.2

This theorem generalizes Theorem 5 in (Bertinet & Thomas-Agnan, 2004).

The Product Hilbert Space We begin by introducing the product space,

$$\mathcal{F} = \mathcal{H}^1 \times \mathcal{H}^2 = \{(f^1, f^2) : f^1 \in \mathcal{H}^1, f^2 \in \mathcal{H}^2\},$$

and an inner product on $\mathcal{F}$ defined by,

$$\left\langle (f^1, f^2), (g^1, g^2) \right\rangle_{\mathcal{F}} = \gamma_1 \langle f^1, g^1 \rangle_{\mathcal{H}^1} + \gamma_2 \langle f^2, g^2 \rangle_{\mathcal{H}^2} + \mu \sum_{i \in U} \left(f^1(\boldsymbol{x}_i) - f^2(\boldsymbol{x}_i)\right)\left(g^1(\boldsymbol{x}_i) - g^2(\boldsymbol{x}_i)\right) \quad (15)$$

It's straightforward to check that $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ is a valid inner product. Moreover, we have the following:

Lemma A.1. *$\mathcal{F}$ is a Hilbert space.*

Proof. (Sketch.) We need to show that $\mathcal{F}$ is complete. Let (f_n^1, f_n^2) be a Cauchy sequence in $\mathcal{F}$. Then f_n^1 is Cauchy in $\mathcal{H}^1$ and f_n^2 is Cauchy in $\mathcal{H}^2$. By the completeness of $\mathcal{H}^1$ and $\mathcal{H}^2$, we have $f_n^1 \to f^1$ in $\mathcal{H}^1$ and $f_n^2 \to f^2$ in $\mathcal{H}^2$, for some $(f^1, f^2) \in \mathcal{F}$. Since $\mathcal{H}^1$ and $\mathcal{H}^2$ are RKHSs, convergence in norm implies pointwise convergence, and thus the co-regularization part of the distance also goes to zero. □

$\tilde{\mathcal{H}}$ is a Hilbert Space Recall the definition of $\tilde{\mathcal{H}}$ in Eqn. 4. Define the map $u : \mathcal{F} \to \tilde{\mathcal{H}}$ by $u(f^1, f^2) = \frac{1}{2}\left(f^1 + f^2\right)$. The map's kernel $N := u^{-1}(0)$ is a closed subspace of $\mathcal{F}$, and thus its orthogonal complement $N^\perp$ is also a closed subspace. We can consider $N^\perp$ as a Hilbert space with the inner product that is the natural restriction of $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ to $N^\perp$. Define $v : N^\perp \to \tilde{\mathcal{H}}$ as the restriction of u to $N^\perp$. Then v is a bijection, and we define an inner product on $\tilde{\mathcal{H}}$ by

$$\langle f, g \rangle_{\tilde{\mathcal{H}}} = \langle v^{-1}(f), v^{-1}(g) \rangle_{\mathcal{F}}. \quad (16)$$

We conclude that $\tilde{\mathcal{H}}$ is a Hilbert space isomorphic to $N^\perp$.

The Co-Regularization Norm Fix any $f \in \tilde{\mathcal{H}}$, and note that $u^{-1}(f) = \left\{v^{-1}(f) + n \mid n \in N\right\}$. Since $v^{-1}(f)$ and N are orthogonal, it's clear by the Pythagorean theorem that $v^{-1}(f)$ is the element of $u^{-1}(f)$ with minimum norm. Thus

$$||f||_{\tilde{\mathcal{H}}}^2 = ||v^{-1}(f)||_{\mathcal{F}}^2 = \min_{(f^1, f^2) \in u^{-1}(f)} ||(f^1, f^2)||_{\mathcal{F}}^2$$

We see that the inner product on $\tilde{\mathcal{H}}$ induces the norm claimed in the theorem statement.

We next check the two conditions for validity of an RKHS (see Definition 2.1).

(a) $\tilde{k}(\boldsymbol{z}, \cdot) \in \tilde{\mathcal{H}} \quad \forall \boldsymbol{z} \in \mathcal{X}$ Recall from Eqn. 7 that the co-regularization kernel is defined as

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{z}) = \gamma_1^{-1} k^1(\boldsymbol{x}, \boldsymbol{z}) + \gamma_2^{-1} k^2(\boldsymbol{x}, \boldsymbol{z}) - \mu \left(\gamma_1^{-1} \mathbf{k}_{U\boldsymbol{x}}^1 - \gamma_2^{-1} \mathbf{k}_{U\boldsymbol{x}}^2\right)^T \beta_{\boldsymbol{z}}$$

where $\beta_{\boldsymbol{z}} = H\mathbf{d}_z = (I + \mu S)^{-1}\left(\gamma_1^{-1}\mathbf{k}_{U\boldsymbol{z}}^1 - \gamma_2^{-1}\mathbf{k}_{U\boldsymbol{z}}^2\right)$. Define $h^1(\boldsymbol{x}) = \gamma_1^{-1}k^1(\boldsymbol{x}, \boldsymbol{z}) - \mu\gamma_1^{-1}\mathbf{k}_{U\boldsymbol{x}}^1\beta_{\boldsymbol{z}}$ and $h^2(\boldsymbol{x}) = \gamma_2^{-1}k^2(\boldsymbol{x}, \boldsymbol{z}) + \mu\gamma_2^{-1}k^2(\boldsymbol{x}, U)\beta_{\boldsymbol{z}}$. Note that, $h^1 \in \text{span}\left\{k^1(\boldsymbol{z}, \cdot), k^1(\boldsymbol{x}_1, \cdot), \ldots, k^1(\boldsymbol{x}_u, \cdot)\right\} \subset \mathcal{H}^1$, and similarly, $h^2 \in \mathcal{H}^2$. It's clear that $\tilde{k}(\boldsymbol{z}, \cdot) = \left[h^1(\cdot) + h^2(\cdot)\right]$, and thus $\tilde{k}(\boldsymbol{z}, \cdot) \in \tilde{\mathcal{H}}$.

(b) Reproducing Property For convenience, we collect some basic properties of h^1 and h^2 in the following lemma:

Lemma A.2 (Properties of h^1 and h^2)**.** *Writing $h^1(U)$ for the column vector with entries $h^1(\boldsymbol{x}_i)$ $\forall i \in U$, and similarly for other functions on $\mathcal{X}$, we have the following:*

$$\left\langle f^1, h^1 \right\rangle_{\mathcal{H}^1} = \gamma_1^{-1} f^1(\boldsymbol{z}) - \mu\gamma_1^{-1} f^1(U)^T \beta_{\boldsymbol{z}} \quad (17)$$

$$\left\langle f^2, h^2 \right\rangle_{\mathcal{H}^2} = \gamma_2^{-1} f^2(\boldsymbol{z}) + \mu\gamma_2^{-1} f^2(U)^T \beta_{\boldsymbol{z}} \quad (18)$$

$$h^1(U) - h^2(U) = \beta_{\boldsymbol{z}} \quad (19)$$

Proof. The first two equations follow from the definitions of h^1 and h^2 and the reproducing kernel property. The last equation is derived as follows:

$$\begin{aligned} h^1(U) - h^2(U) =& \gamma_1^{-1}k^1(U, \boldsymbol{z}) - \mu\gamma_1^{-1}k^1(U, U)\beta_{\boldsymbol{z}} \\ & - \gamma_2^{-1}k^2(U, \boldsymbol{z}) - \mu\gamma_2^{-1}k^2(U, U)\beta_{\boldsymbol{z}} \\ =& \mathbf{d}_z - \mu S(I + \mu S)^{-1}\mathbf{d}_z \\ =& \left(I - \mu S(I + \mu S)^{-1}\right)\mathbf{d}_z \\ =& (I + \mu S)^{-1}\,\mathbf{d}_z = \beta_{\boldsymbol{z}} \end{aligned}$$

where the last line follows from the Sherman-Morrison-Woodbury inversion formula. □

Since $\tilde{k}(\boldsymbol{z}, \cdot) = h_1(\cdot) + h_2(\cdot)$, it is clear that $(h_1, h_2) = v^{-1}\left(\tilde{k}(\boldsymbol{z}, \cdot)\right) + n$, for some $n \in N$. Since $v^{-1}(f) \in N^\perp$, we have

$$\begin{aligned} \left\langle f, \tilde{k}(\boldsymbol{z}, \cdot) \right\rangle_{\tilde{\mathcal{H}}} =& \left\langle v^{-1}(f), v^{-1}(\tilde{k}(\boldsymbol{z}, \cdot)) \right\rangle_{\mathcal{F}} \\ =& \left\langle v^{-1}(f), (h_1, h_2) - n \right\rangle_{\mathcal{F}} \\ =& \left\langle v^{-1}(f), (h_1, h_2) \right\rangle_{\mathcal{F}} \\ =& \gamma_1 \left\langle f^1, h^1 \right\rangle_{\mathcal{H}_1} + \gamma_2 \left\langle f_2, h_2 \right\rangle_{\mathcal{H}_2} \\ & + \mu \left[h^1(U) - h^2(U)\right]^T \left[f^1(U) - f^2(U)\right] \\ =& f^1(\boldsymbol{z}) + f^2(\boldsymbol{z}) - \mu \left[f^1(U) - f^2(U)\right]^T \beta_{\boldsymbol{z}} \\ & + \mu \left[f^1(U) - f^2(U)\right]^T \beta_{\boldsymbol{z}} \text{ (from A.2)} \\ =& f(\boldsymbol{z}) \end{aligned}$$

□

The Asymptotics of Semi-Supervised Learning in Discriminative Probabilistic Models

Nataliya Sokolovska SOKOLOVSKA@TELECOM-PARISTECH.FR
Olivier Cappé CAPPE@TELECOM-PARISTECH.FR
LTCI, TELECOM ParisTech and CNRS, 46 rue Barrault, 75013 Paris, France

François Yvon YVON@LIMSI.FR
Université Paris-Sud 11 and LIMSI-CNRS, 91403 Orsay, France

Abstract

Semi-supervised learning aims at taking advantage of unlabeled data to improve the efficiency of supervised learning procedures. For discriminative models however, this is a challenging task. In this contribution, we introduce an original methodology for using unlabeled data through the design of a simple semi-supervised objective function. We prove that the corresponding semi-supervised estimator is asymptotically optimal. The practical consequences of this result are discussed for the case of the logistic regression model.

1. Introduction

In most real-world pattern classification problems (e.g., for text, image or audio data), unannotated data is plentiful and can be collected at almost no cost, whereas labeled data are comparatively rarer, and more costly to gather. A sensible question is thus to find ways to exploit the unlabeled data in order to improve the performance of supervised training procedures. Many proposals have been made in the recent years to devise effective semi-supervised training schemes (see (Chapelle et al., 2006) for an up-to-date panorama). In this contribution, we focus on methods applicable to probabilistic classifiers, that is, classifiers designed to provide a probabilistic confidence measure associated with each decision. These classifiers do not necessarily perform better than other alternatives – particularly since probabilistic classification and minimum error classification are related, but different, tasks – but are important in some applications, for instance when it comes to predicting the generalization error, dealing with uneven error costs, ranking, combining decisions from multiple sources, etc.

Probabilistic generative models fare easily with the use of unlabeled data, usually through Expectation-Maximization (see, e.g., (Nigam et al., 2000; Klein & Manning, 2004) for successful implementations of this idea). It is however an extensively documented fact that discriminative models perform better than Generative models for classification tasks (Ng & Jordan, 2002). Integrating unlabeled data into discriminative models is however a much more challenging issue. Put in probabilistic terms, when learning to predict an output y from an observation x, a discriminative model attempts to fit $P(y|x;\theta)$, where θ denotes the parameter. The role to be played by any available prior knowledge about the marginal probability $P(x)$ in this context is not obvious. Several authors indeed claim that knowledge of $P(x)$ is basically useless (Seeger, 2002; Lasserre et al., 2006), although one of the contribution of this paper will be to show that this intuition relies on the implicit assumption that the model is *well-specified*, in the sense of allowing a perfect fit of the conditional probability.

The most common approach is to make the unknown parameter vector θ depend on the unlabeled data, either directly or indirectly. One way to achieve this goal is to use the unlabeled data to enforce constraints on the shape of $P(y|x)$: the *cluster assumption*, for instance, stipulates that the decision boundary should be located in low density regions (Seeger, 2002; Chapelle & Zien, 2005). (Grandvalet & Bengio, 2004) use this intuition to devise a semi-supervised training method (termed *entropy regularization*), which combines the usual log-likelihood term with an entropy-based penalty; see also (Jiao et al., 2006), who extend this methodology to Conditional Random Fields, (Laf-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

ferty et al., 2001), or (Corduneanu & Jaakkola, 2003) for related ideas. This approach, as any attempt to distort the supervised training criterion with supplementary terms faces two risks: (i) to turn a well-behaved convex optimization problem into a non-convex one, fraught with local optima, thus making the results highly dependent of a proper initialization; (ii) to loose the asymptotic consistency property of the usual (conditional maximum likelihood) estimator. As a result, these methods are not guaranteed to improve over a trivial baseline which would only use the available annotated data. They furthermore require a fine tuning of the various optimization parameters (Mann & McCallum, 2007). The cluster assumption is also used in graph-based methods, which exploit the intuition that unlabeled data points should receive the same label as their labeled neighbors: in (Zhu & Ghahramani, 2002), a neighborhood graph is used to iteratively propagate labels from labeled to unlabeled data points until convergence.

(Lasserre et al., 2006) explores yet another avenue, introducing two sets of parameters: one for the conditional $P(y|x;\theta)$, and one for the marginal $P(x;\nu)$: the case where θ and ν are unrelated corresponds to the purely discriminative model, where unlabeled data are of no help; taking $\theta = \nu$ recovers the traditional generative model; introducing (via their Bayesian prior distribution) dependencies between (θ, ν) allows to build a full range of hybrid models. Finally, we also mention (Mann & McCallum, 2007) who try to also exploit prior knowledge on the distribution of the labels Y, which may be available in some specific applications.

In this paper, we try to challenge the view that unlabeled data cannot help purely discriminative models by exhibiting a semi-supervised estimator of the parameter θ which is asymptotically optimal and, in some situations, preferable to the usual maximum (conditional) likelihood estimator. To this aim, we make the simplifying assumption that the marginal $P(x)$ is fully known, which is true in the limit of infinitely many unlabeled data. An interesting observation about the proposed method is that it is most efficient when the Bayes error is very small which correlates well with the intuition underlying most semi-supervised approaches that unlabeled data is most useful if one can assume that the classes are "well-separated". In addition to the asymptotic results, we also discuss a number of empirical findings pertaining to logistic regression.

This paper is organized as follows: in Section 2, we introduce our formal framework and formulate the main result of the paper (Theorem 1), which is first exposed in its full generality, then particularized to the case of the logistic regression. Experiments with the logistic regression model are discussed in Section 3. Concluding remarks and perspectives close the paper.

2. Semi-Supervised Estimator

Let $g(y|x;\theta)$ denote the conditional probability density function (pdf) corresponding to a discriminative probabilistic model parameterized by $\theta \in \Theta$. In the following, we will always assume that the class variable Y takes its values in a finite set, $\mathcal{Y}$, with a special interest for the binary case where $\mathcal{Y} = \{0, 1\}$. We will further assume that the input (or explanatory) variable X also takes its values in a finite set $\mathcal{X}$, which may be arbitrary large.

The training procedure has access to a set of n i.i.d. *labeled* observations, $(X_i, Y_i)_{1 \leq i \leq n}$, as well as to a potentially unlimited number of unlabeled observations, where the quantity of unlabeled data is so large that we can consider that the marginal probability of X is fully known.

Finally, for a function $f : \mathbb{R}^p \mapsto \mathbb{R}$, we denote by $\nabla_z f(z_\star)$ the $p \times 1$ gradient vector and by $\nabla_{z^\mathrm{T}} \nabla_z f(z_\star)$ the $p \times p$ Hessian matrix in $z_\star$. When $f : \mathbb{R}^p \mapsto \mathbb{R}^r$, the notation $\nabla_{z^\mathrm{T}} f(z_\star)$ will be used to denote the $r \times p$ Jacobian matrix in $z_\star$.

2.1. Preliminary: A Simple Case

We first consider the case where the "model" of interest is very basic and simply consists in estimating the complete joint probability of X and Y, which is denoted by $\pi(x, y)$. We will also denote by $\eta(y|x)$ and $q(x)$, respectively, the conditional and the marginal probabilities associated with π. Although this case is not directly of interest for real-life statistical learning tasks, it highlights the role played by the knowledge of the marginal q in semi-supervised learning.

It is well known that the maximum-likelihood estimator of $\pi(x, y)$ defined by

$$\hat{\pi}_n(x, y) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{X_i = x, Y_i = y\} \qquad (1)$$

is asymptotically efficient with asymptotic variance $\upsilon(x, y) = \pi(x, y)(1 - \pi(x, y))$ (assuming that $0 < \pi(x, y) < 1$).

Assume now that we are given $q(x)$, the marginal distribution of X, and that $0 < q(x) < 1$. It is easily checked that the maximum-likelihood estimator of $\pi(x, y)$ subject to the marginal constraint that

$\sum_{y\in\mathcal{Y}}\pi(x,y)=q(x)$ is given by

$$\hat{\pi}_n^s(x,y)=\frac{\sum_{i=1}^n \mathbf{1}\{X_i=x,Y_i=y\}}{\sum_{i=1}^n \mathbf{1}\{X_i=x\}}q(x) \qquad (2)$$

where the superscript s stands for "semi-supervised" and the ratio is recognized as the maximum-likelihood estimate of the *conditional probability* $\eta(y|x)$. As $\hat{\pi}_n^s(x,y)$ is a ratio of two simple estimators, its asymptotic variance can be computed using the δ-method, yielding

$$\upsilon^s(x,y)=\pi(x,y)(1-\pi(x,y)/q(x))$$

As $0<\pi(x,y)\leq q(x)<1$, $\upsilon^s(x,y)$ is less than $\upsilon(x,y)$. Hence, in general the semi-supervised estimator $\hat{\pi}_n^s(x,y)$ and $\hat{\pi}_n(x,y)$ are not asymptotically equivalent, and $\hat{\pi}_n^s(x,y)$ is preferable. More precisely, $\upsilon^s(x,y)/\upsilon(x,y)=(1-\pi(x,y)/q(x))/(1-\pi(x,y))$ which tends to zero as $\pi(x,y)$ gets closer to $q(x)$. In other words, the performance of $\hat{\pi}_n^s(x,y)$ is all the more appreciable, compared to that of $\hat{\pi}_n(x,y)$, that y is a frequent label for x. In this case, knowledge of the marginal $q(x)$ makes it possible to obtain a precise estimate of $\hat{\pi}_n^s(x,y)\approx q(x)$ even with a very limited number of observations of x.

2.2. General Discriminative Model

We now consider the extension of the previous simple observation to the case of a general discriminative probabilistic model; the main difference being the fact that a given parametric model $\{g(y|x;\theta)\}_{\theta\in\Theta}$ will generally not be able to fit exactly the actual conditional distribution $\eta(y|x)$ of the data. As in the fully-specified case above, it is nonetheless possible to exhibit a semi-supervised estimator which is asymptotically optimal and preferable to the usual conditional maximum likelihood estimator defined by

$$\hat{\theta}_n=\arg\min_{\theta\in\Theta}\frac{1}{n}\sum_{i=1}^n \ell(Y_i|X_i;\theta) \qquad (3)$$

where $\ell(y|x;\theta)=-\log g(y|x;\theta)$ denotes the inverse of the conditional log-likelihood function.

Under the (classical) assumptions of Theorem 1 below, $\frac{1}{n}\sum_{i=1}^n \ell(Y_i|X_i;\theta)$ tends, uniformly in θ, to $\mathrm{E}_\pi[\ell(Y|X;\theta)]$ and thus the limiting value of $\hat{\theta}_n$ is given by

$$\theta_\star=\arg\min_{\theta\in\Theta}\mathrm{E}_\pi[\ell(Y|X;\theta)] \qquad (4)$$

The maximum likelihood estimator in (3) may also be interpreted as $\hat{\theta}_n=\arg\min_{\theta\in\Theta}\mathrm{E}_{\hat{\pi}_n}[\ell(Y|X;\theta)]$ where

$$\hat{\pi}_n(x,y)=\frac{1}{n}\sum_{i=1}^n \mathbf{1}\{X_i=x,Y_i=y\}$$

denotes the empirical measure associated with the sample $(X_i,Y_i)_{1\leq i\leq n}$, which also coincides with the maximum likelihood estimate of $\pi(x,y)$ defined in (1).

If we now assume that the marginal $q(x)$ is available, we know that $\hat{\pi}_n(x,y)$ is dominated (asymptotically) by the estimator $\hat{\pi}_n^s(x,y)$ defined in (2), which we here particularize to

$$\hat{\pi}_n^s(x,y)=\begin{cases}\frac{\sum_{i=1}^n \mathbf{1}\{X_i=x,Y_i=y\}}{\sum_{i=1}^n \mathbf{1}\{X_i=x\}}q(x) & \text{if } \sum_{i=1}^n \mathbf{1}\{X_i=x\}>0\\ 0 & \text{otherwise}\end{cases} \qquad (5)$$

By analogy with the construction used in the absence of information on q, we now define the corresponding semi-supervised estimator as $\hat{\theta}_n^s=\arg\min_{\theta\in\Theta}\mathrm{E}_{\hat{\pi}_n^s}[\ell(Y|X;\theta)]$, where the notation $\mathrm{E}_{\hat{\pi}_n^s}[f(Y,x)]=\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\hat{\pi}_n^s(x,y)f(x,y)$ is used somewhat loosely here as it may happen that, for finite n, $\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\hat{\pi}_n(x,y)<1$, although $\hat{\pi}_n(x,y)$ sums to one with probability one, for sufficiently large n. It is easily checked that $\hat{\theta}_n^s$ may also be rewritten as

$$\hat{\theta}_n^s=\arg\min_{\theta\in\Theta}\sum_{i=1}^n \frac{q(X_i)}{\sum_{j=1}^n \mathbf{1}\{X_j=X_i\}}\ell(Y_i|X_i;\theta) \qquad (6)$$

Eq. (6) is a weighted version of (3) where the weight given to observations that share the same input x is common and reflects our prior knowledge on the marginal $q(x)$.

Theorem 1 *Let the joint probability of X and Y factorize as $\pi(x,y)=\eta(y|x)q(x)$, where q is known, and define the following matrices*

$$H(\theta_\star)=\mathrm{E}_q\left(\mathrm{V}_\eta\left[\nabla_\theta\ell(Y|X;\theta_\star)|X\right]\right) \qquad (7)$$

$$I(\theta_\star)=\mathrm{E}_q\left[\nabla_\theta\ell(Y|X;\theta_\star)\left\{\nabla_\theta\ell(Y|X;\theta_\star)\right\}^{\mathrm{T}}\right] \qquad (8)$$

$$J(\theta_\star)=\mathrm{E}_q\left[\nabla_{\theta^{\mathrm{T}}}\nabla_\theta\ell(Y|X;\theta_\star)\right] \qquad (9)$$

Assume that (1) $\mathcal{X}$ and $\mathcal{Y}$ are finite sets; (2) $\pi(x,y)>0$ for all $(x,y)\in\mathcal{X}\times\mathcal{Y}$; (3) for all $(x,y)\in\mathcal{X}\times\mathcal{Y}$, $\ell(y|x;\theta)$ is bounded on Θ; (4) $\theta_\star$ is the unique minimizer of $\mathrm{E}_\pi[\ell(Y|X;\theta)]$ on Θ; (5) for all $(x,y)\in\mathcal{X}\times\mathcal{Y}$, $\ell(y|x;\theta)$ is twice continuously differentiable on Θ; (6) the matrices $H(\theta_\star)$ and $J(\theta_\star)$ are non singular.

Then, $\hat{\theta}_n$ and $\hat{\theta}_n^s$ are consistent and asymptotically normal estimators of $\theta_\star$, which satisfy

$$\sqrt{n}\left(\hat{\theta}_n-\theta_\star\right)\xrightarrow{L}\mathcal{N}\left(0,J^{-1}(\theta_\star)I(\theta_\star)J^{-1}(\theta_\star)\right) \qquad (10)$$

$$\sqrt{n}\left(\hat{\theta}_n^s-\theta_\star\right)\xrightarrow{L}\mathcal{N}\left(0,J^{-1}(\theta_\star)H(\theta_\star)J^{-1}(\theta_\star)\right) \qquad (11)$$

Furthermore, $\hat{\theta}_n^s$ is asymptotically efficient.

Theorem 1 asserts that the asymptotic covariance matrix associated with $\hat{\theta}_n^s$ is optimal. Understanding the relations between $H(\theta_\star)$ and $I(\theta_\star)$ is thus important to assess the asymptotic performance achievable by *any* semi-supervised training method which assumes prior knowledge of $q(x)$. Indeed, the well-known Rao-Blackwell variance decomposition shows that

$$I(\theta_\star) - H(\theta_\star) = \mathrm{V}_q\left(\mathrm{E}_\eta\left[\nabla_\theta \ell(Y|X;\theta_\star)|X\right]\right)$$

As a result, the difference between both estimators will mostly depend on whether $\mathrm{E}_\eta\left[\nabla_\theta \ell(Y|X;\theta_\star)|X=x\right]$ varies significantly or not around 0 as a function of x, given that, by definition, $\theta_\star$ is such that $\mathrm{E}_q\left(\mathrm{E}_\eta\left[\nabla_\theta \ell(Y|X;\theta_\star)|X\right]\right) = 0$.

Note that in the particular case where the model is *well-specified*, in the sense that $\theta_\star$ is such that $g(y|x;\theta_\star) = \eta(y|x)$ for all $(x,y) \in \mathcal{X}\times\mathcal{Y}$, not only is $\mathrm{E}_q\left(\mathrm{E}_\eta\left[\nabla_\theta \ell(Y|X;\theta_\star)|X\right]\right)$ null but one indeed has the stronger result that *for all* $x \in \mathcal{X}$, $\mathrm{E}_\eta\left[\nabla_\theta \ell(Y|X;\theta_\star)|X=x\right] = 0$. This is the only case for which $H(\theta_\star) = I(\theta_\star)$, and hence, where both estimators are asymptotically equivalent; it is also well known that in this case $J(\theta_\star) = I(\theta_\star)$ so that all asymptotic covariance matrices coincide with the usual expression of the inverse of the Fisher information matrix for θ. Theorem 1 gives formal support to the intuition that it is impossible to improve over the classic maximum likelihood estimator for large n's when the model is well-specified, even when the marginal q is known.

The results of Theorem 1 are stated in terms of parameter estimation which is usually not the primary interest for statistical learning tasks. Due to the non-differentiability of the 0–1 loss, it is not directly possible to derive results pertaining to the error probability from Theorem 1. One may however state the following result in terms of the *logarithmic risk*, in which the negated log-likelihood $\ell(y|x;\theta)$ is interpreted as a loss function.

Corollary 2 *In addition to the assumptions of Theorem 1, assume that $\ell(y|x;\theta)$ has bounded second derivative on Θ. Then, the logarithmic risk admits the following asymptotic equivalent:* $\mathrm{E}_{\pi^{\otimes n}}\{\mathrm{E}_\pi[\ell(Y|X;\hat{\theta}_n)]\} = \mathrm{E}_\pi[\ell(Y|X;\theta_\star)] + \frac{1}{2n}\operatorname{trace}\left\{I(\theta_\star)J^{-1}(\theta_\star)\right\} + o\left(\frac{1}{n}\right)$, *where* $\mathrm{E}_{\pi^{\otimes n}}$ *denotes the expectation with respect to the training data* $(X_i,Y_i)_{1\le i\le n}$*; for the semi-supervised estimator* $\hat{\theta}_n^s$*, the first order term is given by* $\frac{1}{2n}\operatorname{trace}\left\{H(\theta_\star)J^{-1}(\theta_\star)\right\}$.

As a final comment on Theorem 1, note that the form of the semi-supervised estimator in (6) shows that $\hat{\theta}_n^s$ will be consistent also in the presence of covariate shift (i.e., when the marginal distribution of the training sample differs from q), whereas the logistic regression estimates can only be consistent in this case if we assume that the model is well-specified (Shimodaira, 2000). In the presence of covariate shift however, the expressions of the asymptotic covariance matrices will be different.

2.3. Application to Logistic Regression

To gain further insight into the results summarized in Theorem 1, we consider the example of the logistic regression model with binary labels Y and input variables X in $\mathbb{R}^p$; the parameter θ is thus p-dimensional. In this model, the negative log-likelihood function is given by $\ell(y|x;\theta) = -y\theta^{\mathrm{T}}x + \log(1+\mathrm{e}^{\theta^{\mathrm{T}}x})$[1]. Thus, the estimation equation which implicitly defines the value of the optimal fit $\theta_\star$ as the value for which $\mathrm{E}_\pi\left[\nabla_\theta \ell(Y|X;\theta_\star)\right] = 0$ may be rewritten as

$$\mathrm{E}_q\left[X\left(g(1|X;\theta_\star) - \eta(1|X)\right)\right] = 0 \tag{12}$$

Similar direct calculations yield

$$H(\theta_\star) = \mathrm{E}_q\left[\eta(1|X)(1-\eta(1|X))XX^{\mathrm{T}}\right] \tag{13}$$

$$I(\theta_\star) = \mathrm{E}_q\big[\big\{\eta(1|X)(1-\eta(1|X)) + (\eta(1|X) - g(1|X;\theta_\star))^2\big\}XX^{\mathrm{T}}\big] \tag{14}$$

$$J(\theta_\star) = \mathrm{E}_q\left[g(1|X;\theta_\star)\{1-g(1|X;\theta_\star)\}XX^{\mathrm{T}}\right] \tag{15}$$

$J(\theta_\star)$ is the Fisher information matrix traditionally found in logistic regression. Interestingly, $H(\theta_\star)$ is recognized as the Fisher information matrix for $\theta_\star$ corresponding to the fully supervised logistic regression model in the well-specified case (i.e. assuming that $g(y|x;\theta_\star) = \eta(y|x)$), although we made no such assumption here.

For logistic regression, the difference

$$I(\theta_\star) - H(\theta_\star) = \mathrm{E}_q\left[\{\eta(1|X) - g(1|X;\theta_\star)\}^2 XX^{\mathrm{T}}\right]$$

is clearly a term that is all the more significant that the fit achievable by the model is poor. The second important factor that can lead to substantial differences between the asymptotic performances of $\hat{\theta}_n$ and $\hat{\theta}_n^s$ is revealed by the following observation: for a given distribution π, the largest (in a matrix sense) achievable value for $I(\theta_\star)$ is given by

$$I(\theta_\star) = \mathrm{E}_q\left[\max\{\eta(1|X), 1-\eta(1|X)\}XX^{\mathrm{T}}\right]$$

whereas $H(\theta_\star)$ in (13) may be rewritten as

$$H(\theta_\star) = \mathrm{E}_q\big[\max\{\eta(1|X),1-\eta(1|X)\}\min\{\eta(1|X),1-\eta(1|X)\}XX^{\mathrm{T}}\big]$$

[1] Or $\log(1+\mathrm{e}^{-\theta^{\mathrm{T}}yx})$ when the labels are coded as $\{-1,1\}$ rather than $\{0,1\}$.

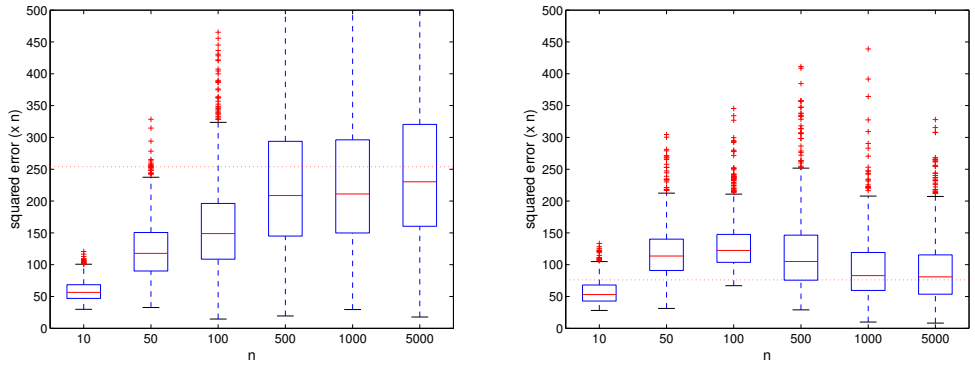


Figure 1. Boxplots of the scaled squared parameter estimation error as a function of the number of observations. Left: for logistic regression, $n\|\hat{\theta}_n - \theta_\star\|^2$; right: for the semi-supervised estimator, $n\|\hat{\theta}^s_n - \theta_\star\|^2$.

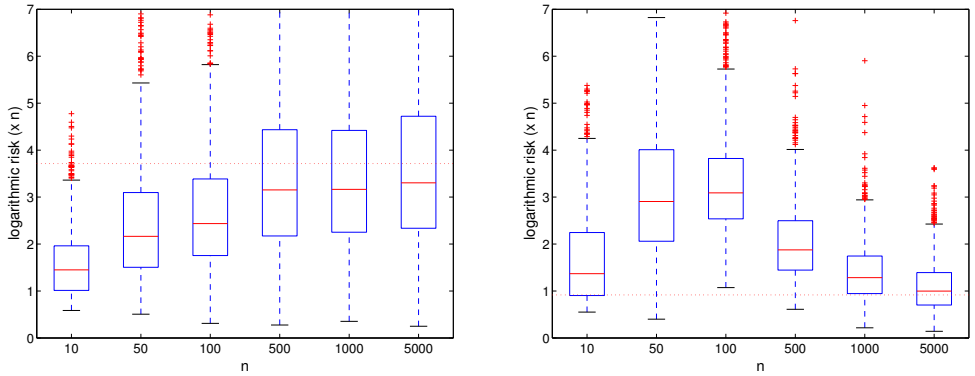


Figure 2. Boxplots of the scaled excess logarithmic risk as a function of the number of observations. Left: for logistic regression, $n(\mathrm{E}_{\pi^{\otimes n}}\{\mathrm{E}_\pi[\ell(Y|X;\hat{\theta}_n)]\} - \mathrm{E}_\pi[\ell(Y|X;\theta_\star)])$; right: for the semi-supervised estimator, $n(\mathrm{E}_{\pi^{\otimes n}}\{\mathrm{E}_\pi[\ell(Y|X;\hat{\theta}^s_n)]\} - \mathrm{E}_\pi[\ell(Y|X;\theta_\star)])$.

Hence the difference between $I(\theta_\star)$ and $H(\theta_\star)$ can only become very significant in cases where $\min\{\eta(1|X = x), 1 - \eta(1|X = x)\}$ is small, that is, when the probability of incorrect decision is small, for some values of x. The overall effect will be all the more significant that this situation happens for many values of x, or, in other words, that the Bayes error associated with π is small.

3. Experiments

3.1. A Small Scale Experiment

We consider here experiments on artificial data which correspond to the case of binary logistic regression discussed in Section 2.3. We focus on a small-scale problem where it is possible to exactly compute error probabilities and risks so as to completely bypass the empirical evaluation of trained classifiers. This setting makes it possible to obtain an accurate assessment of the performance as the only source of Monte Carlo error lies in the choice of the training corpus. More precisely, we consider the case where each observation consists of a vector of $p = 10$ positive counts which sums to $k = 3$. Hence the logistic regression parameter θ is ten-dimensional and the set $\mathcal{X}$ of possible count vectors contains exactly $\frac{(p+k-1)!}{(p-1)!k!} = 220$ different vectors.

In this case, it is well-known that one can simulate data from well-specified logistic models by resorting to mixture of multinomial distributions. Denote by α_1 the prior probability of class 1, and by β_0 and β_1 the vectors of multinomial parameters. Count vectors X generated from the mixture of multinomial have marginal probabilities $q(x) = \alpha_1 \,\mathrm{mult}(x;\beta_1) + (1-\alpha_1)\,\mathrm{mult}(x;\beta_0)$ and conditional probabilities $\mathrm{P}(Y = 1|X = x) = \{1 + \exp -[(\log \beta_1 - \log \beta_0)^{\mathrm{T}} x + \log \frac{\alpha_1}{1-\alpha_1}]\}^{-1}$, where the log is to be understood componentwise. In the following, we take $\alpha_1 = 0.5$, i.e., balanced classes, so as to avoid the bias term.

In order to generate misspecified scenarios, we simply

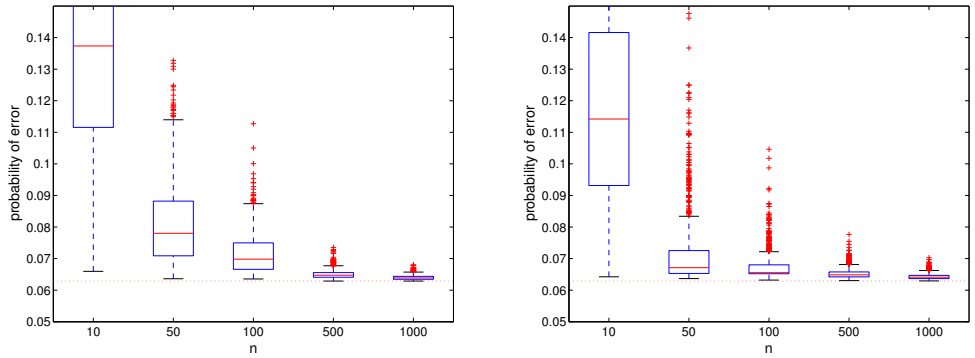


Figure 3. Boxplots of the probability of error as a function of the number of observations for a well-specified model. Left: for the logistic regression; right: for the semi-supervised estimator.

flipped the labels of a few (three in the case shown on figures. 1–2) x's taken among the most likely ones. This label flipping transformation leaves the Bayes error unchanged to that of the underlying unperturbed logistic model but the performance achievable by logistic regression is of course reduced. Figures 1 and 2 correspond to a case where the underlying unperturbed logistic model has a Bayes error of 1.7% and the probability of error associated with the best fitting logistic model is of 9.4%. Remember that in these figures, the only source of randomness is due to the choice of the training sample, which is repeated 1000 times independently for each size of the training sample, from $n = 10$ to $n = 5000$ observations.

As logistic regression is very sensitive to the use of regularization for small sample sizes (here, when n is less than one thousand), both (3) and (6) were regularized by adding a L^2 penalty term of the form $\rho_n \|\theta\|^2$, where ρ_n has been calibrated independently for each value of n. This being said, the optimal regularization parameter was always found to be within a factor 2 of $\rho_n = 1/n$ for (3) and $\rho_n = \frac{1}{n} \sum_{\{x:\sum_{i=1}^{1} \mathbf{1}\{X_i=x\}>0\}} q(x)$ for (6). The effect of regularization is also negligible for the two rightmost boxplots in each graph (i.e., when n is greater than 1000). On figures 1 and 2, the superimposed horizontal dashed lines correspond to the theoretical averages computed from Theorem 1 and Corollary 2, respectively.

When n is larger than one thousand, figures 1 and 2 perfectly correlate with the theory which predicts some advantage for the semi-supervised estimator as we are considering a case where the Bayes error is small and the model misspecification is significant. For large values of n, the semi-supervised estimator not only achieves better average performance but also does so more constantly, with a reduced variability. For smaller values of n, the picture is more contrasted, particularly when n ranges from 50 to 100 where the semi-supervised estimator may perform comparatively worse than the logistic regression. In this example, in terms of the probability of error, the semi-supervised estimator performs marginally better than logistic regression when $n = 10$ and $n = 5000$ (although the difference is bound to be very small in the latter case) and somewhat worse in between.

As expected, the difference between both approaches for large values of n decreases for scenarios with larger error probabilities. In those scenarios, the semi-supervised estimator performs worse than logistic regression for smaller values of n and equivalently for large values of n. A finding of interest is the fact that for well-specified models (i.e., with data generated from a multinomial mixture model) with low Bayes error, the semi-supervised approach does perform better than logistic regression, *for small values of* n. This effect can be significant even when considering the probability of error of the trained classifiers, as exemplified on Figure 3 in a case where the Bayes error is 6.3%. This observation is promising and deserves further investigation as the analysis of Section 2 only explains the behavior observed for large values of n, which in the case of well-specified models results in the two approaches being equivalent.

3.2. Text Classification Experiment

To evaluate our methodology on a more realistic test bed, we have used a simple binary classification task, consisting in classifying mails as spam or ham based on their textual content. The corpus used is the SpamAssassin corpus (Mason, 2002), which contains approximately 6 000 documents. Adapting our technique to real-world data requires to provide an estimate for the marginal $q(x)$. This was carried out by performing a discrete quantification of the data vectors as fol-

lows. We first use unsupervised clustering techniques to partition the available unlabeled collection of documents in k clusters. More specifically, we used a mixture of multinomial model as in (Nigam et al., 2000) with $k = 10$ components. We then simply adapt (6) by replacing $q(X_i)$ by the empirical frequency of the cluster to which X_i belongs, likewise the denominator $\sum_{j=1}^{n} \mathbf{1}\{X_j = X_i\}$ is replaced by the number *of training documents* belonging to the same cluster as X_i. We believe that this methodology is very general and makes the proposed approach applicable to a large variety of data. In effect, observations belonging to clusters which are underrepresented in the training corpus have higher relative weights, while the converse if true for observations belonging to overrepresented clusters. Note that, at this stage, no attempts have been made at tuning the number k of clusters, although intuition suggests that it would probably be reasonable to increase k (slowly) with n.

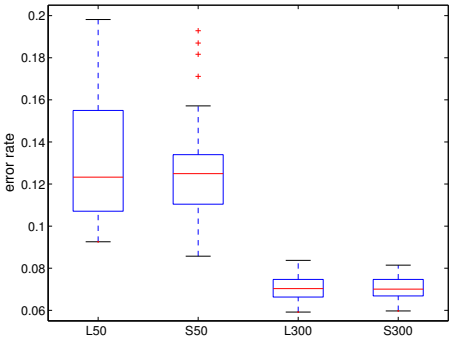


Figure 4. Boxplots of the error rates for, L50: logistic regression with $n = 50$; S50: semi-supervised estimator with $n = 50$; L300 and S300, idem with $n = 300$.

We tested the method with $n = 50$ and $n = 300$ randomly chosen training documents, the remaining mails serving as the test set; each trial gave rise to 50 Monte Carlo replications. For each value of n, the best regularization parameter was determined experimentally both for the usual logistic regression and the semi-supervised estimator. Each document is here represented as a count vector of dimension 1500. The resulting error rates are plotted as boxplots on Figure 4. Although the difference between both methods is certainly not very significant in this preliminary experiment, we note that, as in the simple case of Section 3.1, the semi-supervised estimator provides a more less variable performance when n is small.

4. Conclusion

In this contribution, we have tried to address the problem of semi-supervised learning without using any prior idea on what type of information is to be provided by the unlabeled data. The result of Theorem 1 provides both proper theoretical support for the claim that the unlabeled data does not matter asymptotically *when the model is well-specified* and a better understanding of the cases where the unlabeled data does matter. In particular, it confirms the intuition that unlabeled data is most useful when the Bayes error is small. One advantage of the proposed method is that it does not compromise the simplicity of the maximum likelihood approach because the weighted semi-supervised criterion stays convex. In addition, one could easily incorporate prior knowledge as used in other semi-supervised approaches: for instance the "cluster assumption" can be implemented by modifying (5) so as to incorporate a Bayesian prior that connects conditional probabilities for neighboring values of the input vector. In Section 3.2, we suggested a means by which the method can be extended to larger scales problem, including applications in which the feature vector is either continuous or has a more complex structure. We are in particular currently investigating the extension of the proposed approach to the case of sequence labelling with conditional random fields. Another open issue is the theoretical analysis of the behavior of the proposed criterion when n is small, which cannot be deduced from the asymptotic analysis presented here.

Appendix: Sketch of Proofs

First note that (10) is the well-known result that pertains to the behavior of the maximum likelihood estimator in misspecified models – see, for instance, (White, 1982) or Lemma 1 of (Shimodaira, 2000).

Now, the fact that $\hat{\theta}_n^s = \arg\min_{\theta\in\Theta} \mathrm{E}_{\hat{\pi}_n^s}[\ell(Y|X;\theta)]$ implicitly defines the semi-supervised estimator $\hat{\theta}_n^s$ as a function of the maximum-likelihood estimator of the conditional probabilities

$$\hat{\eta}_n(y|x) = \frac{\sum_{i=1}^{n} \mathbf{1}\{X_i = x, Y_i = y\}}{\sum_{i=1}^{n} \mathbf{1}\{X_i = x\}}$$

In our setting, the conditional probability η may be represented by a finite dimensional vector block defined by $\boldsymbol{\eta} = (\boldsymbol{\eta}(x_1), \ldots, \boldsymbol{\eta}(x_d))^{\mathrm{T}}$, where $\boldsymbol{\eta}(x_i) = (\eta(y_1|x_i), \ldots, \eta(y_k|x_i))^{\mathrm{T}}$, $\{x_1, \ldots, x_d\}$ denote the elements of $\mathcal{X}$, and, $\{y_0, \ldots, y_k\}$ denote the elements of $\mathcal{Y}$. As usual in polytomous regression models, we omit one of the possible values of Y (by convention, y_0) due to the constraint that $\sum_{y\in\mathcal{Y}} \eta(y|x) = 1$, for all $x \in \mathcal{X}$. The estimator $\hat{\boldsymbol{\eta}}_n$ is defined similarly with $\hat{\eta}_n(y|x)$ substituted for $\eta_n(y|x)$. $\hat{\boldsymbol{\eta}}_n$ is the maximum likelihood estimator of $\boldsymbol{\eta}$ and it is asymptotically efficient with

asymptotic covariance matrix given by $K^{-1}(\boldsymbol{\eta})$, the inverse of the Fisher information matrix for $\boldsymbol{\eta}$, block-defined by

$$K^{-1}(\boldsymbol{\eta}) = \operatorname{diag}\left(K^{-1}(x_1;\boldsymbol{\eta}), \ldots, K^{-1}(x_d;\boldsymbol{\eta})\right)$$

where

$$K^{-1}(x_i;\boldsymbol{\eta}) = q(x_i)^{-1}\left\{\operatorname{diag}\left(\boldsymbol{\eta}(x_i)\right) - \boldsymbol{\eta}(x_i)\boldsymbol{\eta}^{\mathrm{T}}(x_i)\right\}$$

To obtain the asymptotic behavior of the semi-supervised estimator $\hat{\theta}_n^s$, remark that $\hat{\theta}_n^s$ is obtained as a function ψ of $\hat{\boldsymbol{\eta}}_n$, where ψ is implicitly defined by the optimality equation $s(\boldsymbol{\eta}, \psi(\boldsymbol{\eta})) = 0$ where s is the (negative of the) score function defined by

$$s(\boldsymbol{\eta}, \theta) = \nabla_\theta \mathrm{E}_\pi\left[\nabla_\theta \ell(Y|X;\theta)\right] = \sum_{x\in\mathcal{X}} q(x) \sum_{y\in\mathcal{Y}} \eta(y|x) \nabla_\theta \ell(y|x;\theta) \quad (16)$$

Because $\theta_\star = \psi(\boldsymbol{\eta})$ and $\hat{\theta}_n^s = \psi(\hat{\boldsymbol{\eta}}_n)$, $\hat{\theta}_n^s$ is an asymptotically efficient estimator of $\theta_\star$ with asymptotic covariance matrix given by $\nabla_{\boldsymbol{\eta}^{\mathrm{T}}}\psi(\boldsymbol{\eta}) K^{-1}(\boldsymbol{\eta}) \left\{\nabla_{\boldsymbol{\eta}^{\mathrm{T}}}\psi(\boldsymbol{\eta})\right\}^{\mathrm{T}}$. The Jacobian matrix $\nabla_{\boldsymbol{\eta}^{\mathrm{T}}}\psi(\boldsymbol{\eta})$ may be evaluated thanks to the implicit function theorem as

$$\nabla_{\boldsymbol{\eta}^{\mathrm{T}}}\psi(\boldsymbol{\eta}) = \left\{\nabla_{\theta^{\mathrm{T}}} s(\boldsymbol{\eta}, \theta_\star)\right\}^{-1} \nabla_{\boldsymbol{\eta}^{\mathrm{T}}} s(\boldsymbol{\eta}, \theta_\star)$$

From the definition of the score function in (16), it is obvious that $\nabla_{\theta^{\mathrm{T}}} s(\boldsymbol{\eta}, \theta_\star) = J(\theta_\star)$. In order to calculate $\nabla_{\boldsymbol{\eta}^{\mathrm{T}}} s(\boldsymbol{\eta}, \theta_\star)$, we differentiate the rightmost expression in (16) using the fact that $\eta(y_0|x) = 1 - \sum_{y\neq y_0} \eta(y|x)$ to obtain

$$\frac{\partial s(\boldsymbol{\eta}, \theta)}{\partial \eta(x|y)} = q(x)\left[\nabla_\theta \ell(y|x;\theta) - \nabla_\theta \ell(y_0|x;\theta)\right]$$

The expression given in Theorem 1 or the asymptotic variance of $\hat{\theta}_n^s$ follows by computing the product $\nabla_{\boldsymbol{\eta}^{\mathrm{T}}} s(\boldsymbol{\eta}, \theta_\star) K^{-1}(x_i;\boldsymbol{\eta}) \left\{\nabla_{\boldsymbol{\eta}^{\mathrm{T}}} s(\boldsymbol{\eta}, \theta_\star)\right\}^{\mathrm{T}}$ – which factories into blocks of size k – and using the fact that $\eta(y_0|x) = 1 - \sum_{y\neq y_0} \eta(y|x)$.

Corollary 2 is based on the classical asymptotic expansion of $\mathrm{E}_\pi[\ell(Y|X;\hat{\theta}_n)] - \mathrm{E}_\pi[\ell(Y|X;\theta_\star)]$ as $\frac{1}{2}(\hat{\theta}_n - \theta_\star)^T J(\theta_\star)(\hat{\theta}_n - \theta_\star) + o_p(\frac{1}{n})$, see, for instance, (Bach, 2006).

References

Bach, F. (2006). Active learning for misspecified generalized linear models. *NIPS*.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. MIT Press.

Chapelle, O., & Zien, A. (2005). Semi-supervised classification by low density separation. *In Proc. of the Tenth International Workshop on Artificial Intelligence and Statistics*.

Corduneanu, A., & Jaakkola, T. (2003). On information regularization. *In Proc. of the 19th conference on Uncertainty in Artificial Intelligence (UAI)*.

Grandvalet, Y., & Bengio, Y. (2004). Semi-supervised learning by entropy minimization. *NIPS*.

Jiao, F., Wang, S., Lee, C. H., Greiner, R., & Schuurmans, D. (2006). Semi-supervised conditional random fields for improved sequence segmentation and labeling. *ACL/COLING*.

Klein, D., & Manning, C. D. (2004). Corpus-based induction of syntactic structure: models of dependency and constituency. *ACL*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. *ICML*.

Lasserre, J. A., Bishop, C. M., & Minka, T. P. (2006). Principled hybrids of generative and discriminative models. *IEEE CVPR*.

Mann, G., & McCallum, A. (2007). Simple, robust, scalable semi-supervised learning via expectation regularization. *ICML*.

Mason, J. (2002). SpamAssassin corpus.

Ng, A., & Jordan, M. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *NIPS*.

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

Seeger, M. (2002). *Learning with labeled and unlabeled data* (Technical Report). University of Edinburgh, Institute for Adaptive and Neural Computation.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, *90*, 227–244.

White, H. (1982). Maximum likelihood estimation in misspecified models. *Econometrica*, *50*, 1–25.

Zhu, X., & Ghahramani, Z. (2002). *Learning from labeled and unlabeled data with label propagation* (Technical Report). Carnegie Mellon University.

Tailoring Density Estimation via Reproducing Kernel Moment Matching

Le Song LE.SONG@NICTA.COM.AU
Xinhua Zhang XINHUA.ZHANG@NICTA.COM.AU
Alex Smola ALEX.SMOLA@NICTA.COM.AU
Statistical Machine Learning, NICTA, 7 London Circuit, Canberra, ACT 2601, Australia
Arthur Gretton ARTHUR.GRETTON@TUEBINGEN.MPG.DE
Bernhard Schölkopf BERNHARD.SCHOELKOPF@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tuebingen, Germany

Abstract

Moment matching is a popular means of parametric density estimation. We extend this technique to nonparametric estimation of mixture models. Our approach works by embedding distributions into a reproducing kernel Hilbert space, and performing moment matching in that space. This allows us to tailor density estimators to a function class of interest (i.e., for which we would like to compute expectations). We show our density estimation approach is useful in applications such as message compression in graphical models, and image classification and retrieval.

1. Introduction

Density estimation is a key element of statistician's toolbox, yet it remains a challenging problem. A popular class of methods relies on mixture models, such as Parzen windows (Parzen, 1962; Silverman, 1986) or mixtures of Gaussians or other basis functions (McLachlan & Basford, 1988). These models are normally learned using the likelihood. However, density estimation is often not the ultimate goal but rather an intermediate step in solving another problem. For instance, we may ultimately want to compute the expectation of a random variable or functions thereof. In this case it is not clear whether likelihood is the ideal objective, especially when the training sample size is small.

A second class of density estimators employ exponential family models and are based on the duality between maximum entropy and maximum likelihood estimation (Barndorff-Nielsen, 1978; Dudík et al., 2004; Altun & Smola, 2006). These methods match the moments of the estimators to those of the data, which helps focus the models on certain aspects of the data for particular applications. However, these parametric moment based methods can be too limited in terms of the class of distributions. Furthermore, exponential families tend to require highly nontrivial integration of high-dimensional distributions to ensure proper normalization. We desire to overcome these drawbacks and extend this technique to a larger class of models.

In this paper, we generalize moment matching to nonparametric mixture models. Our major aim is to tailor these density estimators for a particular function class, and provide uniform convergence guarantees for approximating the function expectations. The key idea is if we have good knowledge of the function class, we can tightly couple the density estimation with this knowledge. Rather than performing a full density estimation where we leave the function class and subsequent operations arbitrary, we restrict our attention to a smaller set of functions and the expectation operator. By exploiting this kind of domain knowledge, we make the hard density estimation problem easier.

Our approach is motivated by the fact that distributions can be represented as points in the marginal polytope in reproducing kernel Hilbert spaces (RKHS) (Wainwright & Jordan, 2003; Smola et al., 2007). By projecting data and density estimators into RKHS via kernel mean maps, we match them in that space (also referred to as the feature space). Choosing the kernel determines how much information about the density is retained by the kernel mean map, and thus which aspects (*e.g.*, moments) of a density are considered important in the matching process. The matching process, and thus our density estimation procedure, amounts to the solution of a convex quadratic program. We demonstrate the application of our approach in experiments, and show that it can lead to improvements in more complicated applications such as particle filtering and image processing.

2. Background

Let $\mathcal{X}$ be a compact domain and $X = \{x_1, \ldots, x_m\}$ be a sample of size m drawn independently and identically distributed (iid.) from a distribution p over $\mathcal{X}$. We aim to find an approximation $\hat{p}$ of p based on the sample X.

Let $\mathcal{H}$ be a reproducing kernel Hilbert space (RKHS) on

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

$\mathcal{X}$ with kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ and associated feature map $\phi : \mathcal{X} \mapsto \mathcal{H}$ such that $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$. By design $\mathcal{H}$ has the reproducing property, that is, for any $f \in \mathcal{H}$ we have $f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}}$. A kernel k is called *universal* if $\mathcal{H}$ is dense in the space of bounded continuous functions $C^0(\mathcal{X})$ on the compact domain $\mathcal{X}$ in the L_∞ norm (Steinwart, 2002). Examples of such kernels include Gaussian kernel $\exp(-\|x - x'\|^2 / 2\theta^2)$ and Laplace kernel $\exp(-\|x - x'\| / 2\theta^2)$.

The *marginal polytope* is the range of the expectation of the feature map ϕ under all distributions in a set $\mathcal{P}$, *i.e.*, $\mathcal{M} := \{\mu[p] | p \in \mathcal{P}, \mu[p] := \mathbb{E}_{x \sim p}[\phi(x)]\}$ (Wainwright & Jordan, 2003). The map $\mu : \mathcal{P} \mapsto \mathcal{H}$ associates a distribution to an element in the RKHS. For universal kernels, the elements of the marginal polytope uniquely determine distributions:

Theorem 1 (Gretton et al. (2007)) *Let k be universal and $\mathcal{P}$ denote the set of Borel probability measures p on $\mathcal{X}$ with $\mathbb{E}_{x \sim p}[k(x, x)] < \infty$. Then the map μ is injective.*

3. Kernel Moment Matching

Given a finite sample X from p, $\mu[p]$ can be approximated by the empirical mean map $\mu[X] := \frac{1}{m} \sum_{i=1}^m \phi(x_i)$. This suggests that a good estimate $\hat{p}$ of p should be chosen such that $\mu[\hat{p}]$ matches $\mu[X]$: this is the key idea of the paper. The flow of reasoning works as follows:

$$\begin{aligned} \text{density } p \rightarrow \text{ sample } X \rightarrow \text{ empirical mean } \mu[X] \\ \rightarrow \text{ density estimation via } \mu[\hat{p}] \approx \mu[X] \end{aligned} \quad (1)$$

The first line of this reasoning was established in (Altun & Smola, 2006, Theorem 15). Let $R_m(\mathcal{H}, p)$ be the Rademacher average (Bartlett & Mendelson, 2002) associated with p and $\mathcal{H}$ via

$$R_m(\mathcal{H}, p) := \frac{1}{m} \mathbb{E}_X \mathbb{E}_\omega \left[\sup_{\|f\|_{\mathcal{H}} \leq 1} \left| \sum_{i=1}^m \omega_i f(x_i) \right| \right].$$

where $\omega \in \{\pm 1\}$ is uniformly random. We use it to bound the deviation between empirical means and expectations:

Theorem 2 (Altun & Smola (2006)) *Assume $\|f\|_\infty \leq R$ for all $f \in \mathcal{H}$ with $\|f\|_{\mathcal{H}} \leq 1$. Then for $\epsilon > 0$ with probability at least $1 - \exp(-\epsilon^2 m R^{-2}/2)$ we have $\|\mu[p] - \mu[X]\|_{\mathcal{H}} \leq 2R_m(\mathcal{H}, p) + \epsilon$.*

This ensures that $\mu[X]$ is a good proxy for $\mu[p]$. To carry out the last step of (1) we assume the density estimator $\hat{p}$ is a mixture of a set of candidate densities p_i (or prototypes):

$$\hat{p} = \sum_{i=1}^n \alpha_i p_i \text{ where } \alpha^\top \mathbf{1} = 1 \text{ and } \alpha_i \geq 0, \quad (2)$$

where $\mathbf{1}$ is a vector of all ones. Here the goal is to obtain good estimates for the coefficients α_i and to obtain performance guarantees which specify how well $\hat{p}$ is capable of estimating p. This can be cast as an optimization problem:

$$\min_\alpha \|\mu[X] - \mu[\hat{p}]\|_{\mathcal{H}}^2 \text{ s.t. } \alpha^\top \mathbf{1} = 1 \,, \alpha_i \geq 0. \quad (3)$$

To prevent overfitting, we add a regularizer $\Omega[\alpha]$, such as $\frac{1}{2}\|\alpha\|^2$, and weight it by a regularization constant $\lambda > 0$. Using the expansion of $\hat{p}$ in (2) we obtain a quadratic program (QP) for α

$$\min_\alpha \tfrac{1}{2} \alpha^\top (\mathbf{Q} + \lambda \mathbf{I}) \alpha - \mathbf{l}^\top \alpha \text{ s.t. } \alpha^\top \mathbf{1} = 1 \,, \alpha_i \geq 0, \quad (4)$$

where $\mathbf{I}$ is the identity matrix. $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and $\mathbf{l} \in \mathbb{R}^n$ are given by

$$\mathbf{Q}_{ij} = \langle \mu[p_i], \mu[p_j] \rangle_{\mathcal{H}} = \mathbb{E}_{x \sim p_i, x' \sim p_j}[k(x, x')], \quad (5)$$

$$\mathbf{l}_j = \langle \mu[X], \mu[p_j] \rangle_{\mathcal{H}} = \tfrac{1}{m} \sum_{i=1}^m \mathbb{E}_{x \sim p_j}[k(x_i, x)]. \quad (6)$$

By construction $\mathbf{Q} \succeq 0$ is positive semidefinite, hence the program (4) is convex. We will discuss examples of kernels k and prototypes p_i where $\mathbf{Q}$ and $\mathbf{l}$ have closed form in Section 5. In many cases, the prototypes p_i also contain tunable parameters. We can also optimize them via gradient methods. Before doing so, we first explain our theoretical basis for tailoring density estimators.

4. Tailoring Density Estimators

Given functions $f \in \mathcal{H}$, a key question is to bound how well the expectations of f with respect to p can be approximated by $\hat{p}$. We have the following lemma:

Lemma 3 *Let $\epsilon > 0$ and $\epsilon' := \|\mu[X] - \mu[\hat{p}]\|_{\mathcal{H}}$. Under the assumptions of Theorem 2 we have with probability at least $1 - \exp(-\epsilon^2 m R^{-2}/2)$*

$$\sup_{\|f\|_{\mathcal{H}} \leq 1} |\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{x \sim \hat{p}}[f(x)]| \leq 2R_m(\mathcal{H}, p) + \epsilon + \epsilon'.$$

Proof In the RKHS, we have $\mathbb{E}_{x \sim p}[f(x)] = \langle f, \mu[p] \rangle_{\mathcal{H}}$ and $\mathbb{E}_{x \sim \hat{p}}[f(x)] = \langle f, \mu[\hat{p}] \rangle_{\mathcal{H}}$. Hence the LHS of the bound equates to $\sup_{\|f\|_{\mathcal{H}} \leq 1} |\langle \mu[p] - \mu[\hat{p}], f \rangle|$, which is given by $\|\mu[p] - \mu[\hat{p}]\|_{\mathcal{H}}$. Using the triangle inequality, our assumption on $\mu[\hat{p}]$ and Theorem 2 completes the proof. ■

This means that we have good control over the behavior of the expectations, as long as the function class is "smooth" on $\mathcal{X}$ in terms of the Rademacher average. It also means that $\|\mu[X] - \mu[\hat{p}]\|_{\mathcal{H}}$ is a sensible objective to minimize if we are only interested in approximating well the expectations over functions f.

This bound also provides the basis for tailoring density estimators. Essentially, if we have good knowledge of the function class used in an application, we can choose the corresponding RKHS or the mean map. This is equivalent to filtering the data and extracting only certain moments. Then the density estimator $\hat{p}$ can focus on matching p only up to these moments.

5. Examples

We now give concrete examples of density estimation. A number of existing methods are special cases of our setting.

Discrete Prototype or Discrete Kernel The simplest case is to represent p by a convex combination of Dirac

Table 1. Expansions for $\mathbf{Q}_{ij}$ and $\mathbf{l}_j$ when using Gaussian prototypes and various kernels in combination. Let $c := \langle x_i, x_j \rangle + 1$.

Kernel	$\mathbf{Q}_{ij}$	$\mathbf{l}_j$
Linear kernel $\langle x, x' \rangle$	$\langle x_i, x_j \rangle$	$\frac{1}{m}\sum_{i=1}^m \langle x_i, x_j \rangle$
Degree 2 polynomial kernel $(\langle x, x' \rangle + 1)^2$	$c^2 + \operatorname{tr}\Sigma_i\Sigma_j + x_i^\top \Sigma_j x_i + x_j^\top \Sigma_i x_j$	$\frac{1}{m}\sum_{i=1}^m (c^2 + x_i^\top \Sigma_j x_i)$
Degree 3 polynomial kernel $(\langle x, x' \rangle + 1)^3$	$c^3 + 6x_i^\top \Sigma_i\Sigma_j x_j + 3c(\operatorname{tr}\Sigma_i\Sigma_j + x_i^\top \Sigma_j x_i + x_j^\top \Sigma_i x_j)$	$\frac{1}{m}\sum_{i=1}^m (c^3 + 3c x_i^\top \Sigma_j x_i)$
Gaussian RBF kernel $e^{-\frac{1}{2\theta^2}\|x-x'\|^2}$	$\theta^d \left\|\Sigma_i + \Sigma_j + \theta^2\mathbf{I}\right\|^{-\frac{1}{2}} e^{-\frac{1}{2}\|x_i - x_j\|^2_{(\Sigma_i+\Sigma_j+\theta^2\mathbf{I})}}$	$\frac{1}{m}\theta^d \left\|\Sigma_j + \theta^2\mathbf{I}\right\|^{-\frac{1}{2}} \sum_{i=1}^m e^{-\frac{1}{2}\|x_i - x_j\|^2_{(\Sigma_j+\theta^2\mathbf{I})}}$

measures $p_i(x) = \delta_{x_i}$. Particle filters (Doucet et al., 2001) use this choice when approximating distributions. For instance, we could choose x_i to be the set of training points. In this case $\mathbf{Q}$ defined in (5) equals the kernel matrix and $\mathbf{l}$ is the vector of empirical kernel averages:

$$\mathbf{Q}_{ij} = k(x_i, x_j) \text{ and } \mathbf{l}_j = \frac{1}{m}\sum_{i=1}^m k(x_i, x_j). \quad (7)$$

The key difference between an unweighted set as used in particle filtering and our setting is that our expansion is specifically optimized towards good estimates with respect to functions drawn from $\mathcal{H}$.

The problem of data squashing (DuMouchel et al., 1999) can likewise be seen as a special case of kernel mean matching. Here one aims to approximate a potentially large set X by a smaller set $X' = \{(x_1, \alpha_1), \ldots, (x_n, \alpha_n)\}$ of *weighted* observations. We want to discard X and only retain X' for all further processing. If $\|\mu[X] - \mu[X']\|_{\mathcal{H}}$ is small, we expect X' to be a good proxy for X.

Instead of using generic kernels k and discrete measures δ_{x_i} as prototypes for density estimation, we may reverse their roles. That is, we may pick generic densities p_i and a Dirac kernel $k(x, x') = \delta(x = x')$. Note this is only well defined for discrete domains $\mathcal{X}$.[1] In this case the mean operator simply maps a distribution into itself and we obtain $\langle \mu[p], \mu[p'] \rangle_{\mathcal{H}} = \int_{\mathcal{X}} p(x)p'(x)\mathrm{d}x$. Using (5) we have

$$\mathbf{Q}_{ij} = \int_{\mathcal{X}} p_i(x)p_j(x)\mathrm{d}x \text{ and } \mathbf{l}_j = \frac{1}{m}\sum_{i=1}^m p_j(x_i). \quad (8)$$

Gaussian Prototype In general we will neither pick discrete prototypes nor discrete kernels for density estimation. We now give explicit expressions for Gaussian prototypes

$$p_i(x) = (2\pi)^{-\frac{d}{2}} |\Sigma_i|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\|x - x_i\|^2_{\Sigma_i}\right), \quad (9)$$

where d is the dimension of the data, $\Sigma_i \succ 0$ is a covariance matrix, and $\|x - x'\|^2_{\Sigma_i} := (x - x')^\top \Sigma_i^{-1}(x - x')$ is the squared Mahalanobis distance. When used in conjunction with different kernels, we have the expansions in Table 1.

Other Prototypes and Kernels Other combinations of kernels and prototypes also lead to closed form expansions. For instance, similar expressions also holds for a Laplacian kernel. However, this involves more tedious integrals of the form $\int e^{-\lambda(|x|+|x-a|)}\mathrm{d}x = \lambda^{-1} + e^{-\lambda|a|}$. Another example is to use indicator functions on unit intervals centered at x_i as p_i and a Gaussian RBF kernel. In this case, both $\mathbf{Q}$ and $\mathbf{l}$ can be expressed using the error function (erf).

[1] On continuous domains such a kernel does not correspond to an RKHS since the point evaluation is not continuous.

Furthermore, Jebara et al. (2004) introduced kernels on probability functions which effectively used definition (5). While they were not motivated by the connection between kernels and density estimation, their results for rich classes of densities, such as HMMs, can be used directly to compute our $\mathbf{Q}$ and $\mathbf{l}$.

6. Related Work

Our work is related to the density estimators of Vapnik & Mukherjee (2000) and Shawe-Taylor & Dolia (2007). The main difference lies in the function space chosen to measure the approximation properties. The former uses the Banach space of functions of bounded variation, while the latter uses the space $L_1(q)$, where q denotes a distribution over test functions. For spherically invariant distributions over test functions our approach and the latter approach are identical, with a key difference (to our advantage) that our optimization is a simple QP which does not require constraint sampling to make the optimization feasible.

Support Vector Density Estimation The model of Vapnik & Mukherjee (2000) can be summarized as follows: let $F[\hat{p}]$ be the cumulative distribution function of $\hat{p}$ and let $F[X]$ be its empirical counterpart. Assume $\hat{p}$ is given by (2), and that we have a regularizer $\Omega[\alpha]$ as previously discussed. In this case the Support Vector Density Estimation problem can be written as

$$\min_{\alpha \text{ feasible}} \frac{1}{m}\sum_{i=1}^m |F[\hat{p}](x_i) - F[X](x_i)| + \lambda\Omega[\alpha]. \quad (10)$$

That is, we minimize the ℓ_1 distance between the empirical and estimated cumulative distribution functions when evaluated on the set of observations X.

To integrate this into our framework we need to extend our setting from Hilbert spaces to Banach spaces. Denote by $\mathcal{B}$ a Banach space, let $\mathcal{X}$ be a domain furnished with probability measures p, p', and let $\phi : \mathcal{X} \mapsto \mathcal{B}$ be a feature map into $\mathcal{B}$. Analogously, we define the mean map $\mu : \mathcal{P} \mapsto \mathcal{B}$ as $\mu[p] := \mathbb{E}_{x\sim p(x)}[\phi(x)]$. Moreover, we define a distance between distributions p and p' via $D(p, p') := \|\mu[p] - \mu[p']\|_{\mathcal{B}}$. If we choose $\phi(x) = \left(\chi_{(-\infty,x]}(x_1), \ldots, \chi_{(-\infty,x]}(x_m)\right)^\top$ where χ is the indicator function, and use the ℓ_1^m norm on ϕ we recover SV density estimation as a special case.

Expected Deviation Estimation Shawe-Taylor & Dolia (2007) defined a distance between distributions as follows: let $\mathcal{H}$ be a set of functions on $\mathcal{X}$ and q be a probability distribution over $\mathcal{F}$. Then the distance between two distributions p and p' is given by

$$D(p, p') := \mathbb{E}_{f\sim q(f)} \left| \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{x\sim p'}[f(x)] \right|. \quad (11)$$

That is, we compute the average distance between p and p' with respect to a distribution of test functions.

Lemma 4 *Let $\mathcal{H}$ be a reproducing kernel Hilbert space, $f \in \mathcal{H}$, and assume $q(f) = q(\|f\|_{\mathcal{H}})$ with finite $\mathbb{E}_{f\sim q}[\|f\|_{\mathcal{H}}]$. Then $D(p, p') = C \|\mu[p] - \mu[p']\|_{\mathcal{H}}$ for some constant C which depends only on $\mathcal{H}$ and q.*

Proof Note that by definition $\mathbb{E}_{x\sim p}[f(x)] = \langle \mu[p], f\rangle_{\mathcal{H}}$. Using linearity of the inner product, Equation (11) equals

$$\int |\langle \mu[p] - \mu[p'], f\rangle_{\mathcal{H}}| \, \mathrm{d}q(f)$$

$$= \|\mu[p] - \mu[p']\|_{\mathcal{H}} \int \left| \left\langle \frac{\mu[p] - \mu[p']}{\|\mu[p] - \mu[p']\|_{\mathcal{H}}}, f \right\rangle_{\mathcal{H}} \right| \mathrm{d}q(f),$$

where the integral is independent of p, p'. To see this, note that for any p, p', $\frac{\mu[p]-\mu[p']}{\|\mu[p]-\mu[p']\|_{\mathcal{H}}}$ is a unit vector which can be turned into, say, the first canonical basis vector by a rotation which leaves the integral invariant, bearing in mind that q is rotation invariant. ■

The above result covers a large number of interesting function classes. To go beyond Hilbert spaces, let $\phi : \mathcal{X} \mapsto \mathcal{B}$ be the transformation from x into $f(x)$ for all $f \in \mathcal{H}$ and $\|z\|_{\mathcal{B}} := \mathbb{E}_{f\sim q(f)}[|z_f|]$ be the $L_1(q)$ norm. Then (11) can also be written as $\|\mu[p] - \mu[p']\|_{\mathcal{B}}$, where μ is the mean map into Banach spaces. Its main drawback is the nontrivial computation for constraint sampling (de Farias & Roy, 2004) and the additional uniform convergence reasoning required. In Hilbert spaces no such operations are needed.

7. Experiments

We focus on two aspects: first, our method performs well as a density estimator *per se*; and second, it can be tailored towards the expectation over a particular function class.

7.1. Methods for Comparison

Gaussian Mixture Model (GMM)[2] The density was represented as a convex sum of Gaussians. GMM was initialized with k-means clustering. The centers, covariances and mixing proportions of the Gaussians were optimized using the EM algorithm. We used diagonal covariances in all our experiments. We always employed 50 random restarts for k-means, and returned the results from the best restart.

Parzen Windows (PZ) The density was represented as an average of a set of normalized RBF functions, with each centered on a data point. The bandwidths of the RBF functions were identical and tuned via the likelihood using leave-one-out cross validation.

Reduced Set Density Estimation (RSDE)[3] Girolami & He (2003) compressed a Parzen window estimator using RBF functions of larger bandwidths. The reduced representation was produced by minimizing an integrated squared distance between the two densities.

[2]GMM codes from: http://www.datalab.uci.edu/resources/gmm/

[3]PZ and RSDE from: http://ttic.uchicago.edu/∼ihler/code/

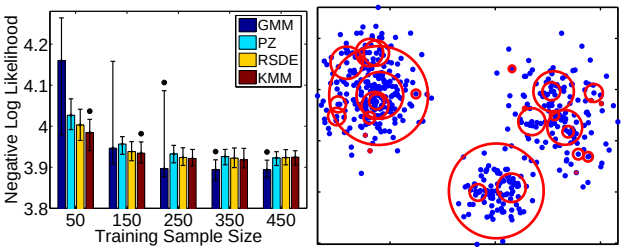


Figure 1. **Left**: Negative log-likelihood using a mixture of 3 Gaussians. The height of the bars represents the median of the scores from 100 repeats, and the whiskers correspond to the quartiles. We mark the best method with a black dot above the whiskers. **Right**: Sparsity of KMM solution. Blue dots are data points. Red circles represent prototypes selected by KMM. The size of a circle is proportional to the magnitude of its weight α_i.

Kernel Moment Matching (KMM) In applying our method, we used Gaussians with diagonal covariances as our prototypes p_i. The regularization parameter λ in our algorithm was fixed at 10^{-10} throughout. Since KMM may be tailored for different RKHS, we instantiated it with the four different kernels in Table 1. We denote them as LIN, POL2, POL3 and RBF, respectively. Our choice of kernel corresponded to the function class where we evaluated the expectations. The initialization of the prototypes will be further discussed below.

7.2. Evaluation Criterion

We compared various density estimators in terms of two criteria: negative log-likelihood and discrepancy between function expectations on test data. Since different algorithms are optimized using different criteria, we expect that each will win with respect to the criterion it employs. The benefit of our density estimator is that we can explicitly tailor for different classes of functions. For this reason, we will focus on the second criterion.

Given a function f, the discrepancy between function expectations is computed as follows: (*i*) Evaluate function expectation using test points, *i.e.*, $\frac{1}{m}\sum_{i=1}^{m} f(x_i)$; (*ii*) Evaluate function expectation using estimated density $\hat{p}$, *i.e.*, $\mathbb{E}_{x\sim\hat{p}}[f(x)]$. (*iii*) Calculate $\left|\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \mathbb{E}_{x\sim\hat{p}}[f(x)]\right|$ and normalize it by $\left|\frac{1}{m}\sum_{i=1}^{m} f(x_i)\right|$.

We will compare various methods either by repeated random instantiation or random split of the data (which we will make clear in context). For both cases, we will perform paired sign tests at the significance level of 0.01 on the results obtained from the randomizations. We will always present the median of the results in subsequent tables, and highlight in boldface those statistically equivalent methods that are significantly better than the rest.

7.3. Synthetic Dataset

In this experiment, we use synthetic datasets to compare various methods as the sample size changes. We also show

Table 2. Sparsity of the solution of RSDE and KMM. We show the median of the number of retained prototypes from 100 random initializations. Also shown are median percentages of retained prototypes after optimization.

Sample Size	50	150	250	350	450
RSDE	12 (25.0%)	35 (23.3%)	62 (24.8%)	90 (25.9%)	124 (27.5%)
KMM	**10 (20.0%)**	**21 (14.0%)**	**30 (12.0%)**	**36 (10.3%)**	**42 (9.3%)**

that KMM leads to sparse solutions.

Data Generation We generated 2 dimensional mixtures of 3 Gaussians with centers $(0,0)^\top$, $(3,3)^\top$ and $(-6,4)^\top$, and covariances $0.8^2\mathbf{I}$, $1.2^2\mathbf{I}$ and $\mathbf{I}$ respectively. The mixing proportions were 0.2, 0.3 and 0.5 respectively. We varied the training sample size while always testing on samples of size 1000. For each fixed training size, we randomly instantiated the experiments 100 times.

Experimental Protocol All training data points were used as prototypes for RSDE and KMM. Their initial covariances were set to be identical, and were initialized in both cases using the approach of RSDE. We used the RBF instance of KMM and set the bandwidth θ of the kernel to be the same as that for the prototypes. GMM used 3 centers.

Negative Log Likelihood The results are plotted in Figure 1. GMM performs best in general, while KMM is superior for small sample sizes. This is not surprising since we used a correct generative model of the data for GMM. When the sample size is small (less than 30 data points for each cluster), GMM is susceptible to local maxima and does not result in good estimates.

Sparsity of the Solution KMM also leads to sparse solutions (*e.g.*, Figure 1). When using all data points as candidate prototypes, KMM automatically prunes away most of them and results in a much reduced representation of the data. In terms of both likelihood and sparsity, KMM is superior to other reduction method such as RSDE (Table 2).

7.4. UCI Dataset

We used 15 UCI datasets to compare various methods based on the discrepancies between function expectations.

Data Description We only included real-valued dimensions of the data, and normalized each dimension of the data separately to zero mean and unit variance. For each dataset, we randomly shuffled the data points for 50 times. In each shuffle, we used the first half of the data for training and the remaining data for testing. In each shuffle, we randomly generated 100 functions f to evaluate the discrepancy criterion, *i.e.*, $f = \sum_{i=1}^{m_0} w_i k(x_i, \cdot)$ where $m_0 \in \mathbb{N}$ was uniformly random in $[1, m]$, $w_i \in \mathbb{R}$ was uniformly random in $[-1, 1]$, and the x_i were uniformly sampled from the test points. Thus, each method resulted in 5000 numbers for each dataset.

Experimental Protocol Both GMM and KMM used 10 prototypes and diagonal covariances, and both were initialized using k-means clustering. We used all four instances of KMM, namely LIN, POL2, POL3 and RBF, for the experiments, depending on the function class where we evaluated the expectations. When we used the RBF instance of KMM, we set the bandwidth θ of the kernel to the median of the distances between data points. Besides optimizing the mixing proportions of the prototypes of KMM, we also used conjugate gradient descent to optimize the center positions and covariances of the prototypes.

Table 3. Negative log-likelihood on test points as computed by various density estimators over randomizations.

Data	PZ	GMM	RSDE	LIN	POL2	POL3	RBF
covertype	11.48	**11.22**	14.97	11.64	208.29	45.23	62.37
ionosphere	**28.09**	36.58	56.68	29.55	69.92	68.79	46.49
sonar	**78.29**	119.16	122.35	**78.13**	129.81	92.35	112.21
australian	**3.32**	5.82	8.82	3.40	4.64	22.73	7.27
specft	43.01	**42.61**	43.16	**42.90**	231.76	87.28	105.04
wdbc	**25.17**	42.97	48.44	25.98	248.73	63.61	88.61
wine	19.68	21.17	22.95	**19.43**	70.15	47.99	48.94
satimage	**18.49**	39.10	59.88	20.18	158.31	121.27	52.41
segment	**-1.43**[a]	5.71	36.74	-1.07	154.25	128.74	28.38
vehicle	**10.98**	**11.99**	32.85	**11.34**	170.66	200.22	83.35
svmguide2	**27.85**	39.67	40.07	**27.92**	204.30	59.22	36.08
vowle	11.75	**6.24**	25.59	11.77	108.43	47.45	26.18
housing	**3.68**	7.44	15.53	3.81	16.07	90.51	39.88
bodyfat	**16.38**	20.06	21.96	**16.59**	87.23	171.53	53.33
abalone	**2.53**	2.57	10.17	2.75	19.15	21.77	16.29
mix3 100	2.42	**2.09**	**2.12**	2.55	2.49	2.43	2.41
mix3 500	1.91	1.92	1.92	1.94	1.93	1.91	**1.91**
mix3 1000	**1.86**	1.87	1.88	1.88	1.87	1.87	1.86

[a] Some numbers are negative, which is possible since unlike probability mass function, density can take values greater than 1.

Negative Log Likelihood Kernel moment matching can be a very different objective from the likelihood (Table 3). Except for the LIN instance, KMM results in much larger negative log-likelihood. This suggests that if the purpose of density estimation is to approximate the function expectations, likelihood is no longer a good criterion. We confirm this observation in our next experiment.

Discrepancy between Function Expectations We used four classes of functions corresponding to the RKHS of the LIN, POL2, POL3 and RBF instances of KMM. For nonlinear functions KMM clearly outperforms other density estimators, while for linear functions KMM has equivalent performance to PZ and GMM (Table 4). These results are not surprising, since KMM is explicitly optimized for approximating the function expectations well. Note that PZ is the second best for polynomial functions. This is reasonable since PZ retains all training points in the density, and should perform better than compressed representations such as GMM and RSDE. We also applied this new experimental protocol to the synthetic mixture of 3 Gaussians from the last section. We instantiate the synthetic data with 3 different sample sizes: 100, 500 and 1000. The results are shown in the last three rows of Table 3 and 4, which are consistent with those for UCI data. A closer view of the difference between GMM and KMM using "covertype" dataset is shown in Figure 2. We chose to compare GMM and KMM because they are initialized similarly.

As an aside, we remark that PZ and GMM also match the

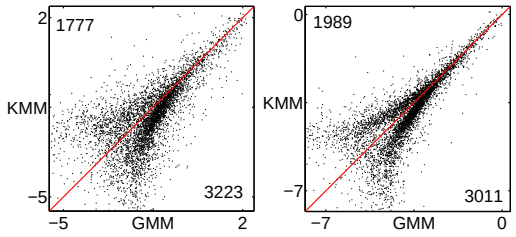


Figure 2. Scatter plot of the discrepancies between function expectations (in log scale) for the 'covertype' dataset, with GMM discrepancies on the horizontal axis and KMM discrepancies on the vertical axis. **Left**: plot for polynomial functions ($d = 2$); **Right**: plot for RBF functions. The distribution of the points is skewed below the red diagonal line, which means KMM is better. The numbers near the corners count respectively the number of points falling above and below the red diagonal.

empirical mean $\frac{1}{m}\sum_{j=1}^{m} x_j$. This is obvious for PZ. For GMM, in each EM iteration, the centers μ_i and weights α_i of each p_i are updated via $\mu_i \leftarrow \sum_{j=1}^{m} \tau_j^i x_j / \sum_{j=1}^{m} \tau_j^i$ and $\alpha_i \leftarrow \frac{1}{m}\sum_{j=1}^{m} \tau_j^i$. Here τ_j^i is the probability of x_j being generated by p_i. It follows that $\mathbb{E}_p[x] = \sum_i \alpha_i \mu_i$ also matches $\frac{1}{m}\sum_{j=1}^{m} x_j$.

8. Applications

In this section, we employ KMM for two different applications: message compression in graphical models, and image processing. The common feature of these two applications is that they involve density estimation for computing the expectation of a function, which is the relevant setting for KMM.

8.1. Message Compression

We use density estimation to compress messages in graphical models. This is of particular interest for applications such as distributed inference in sensor networks. It is our desire to compress the messages to cater for limited power supply and communication bandwidth. We will use a particle filtering example to compare GMM and KMM only, since they performed best in our earlier experiments.

We model a one dimensional time series y_t ($t = 1 \ldots 100$) as being conditionally independent given an unobserved state s_t, which is itself Markovian. This system evolves as follows:

$$s_t = f(s_{t-1}) = 1 + \sin(0.04\pi t) + 0.5 s_{t-1} + \xi \quad (12)$$

$$y_t = g(s_t) = \begin{cases} 0.2 s_t^2 + \zeta, & \text{if } t < 50, \\ 0.5 s_t - 2 + \zeta, & \text{otherwise.} \end{cases} \quad (13)$$

The random variables ξ and ζ represent process and measurement noise, respectively, and are modeled as mixtures of Gaussians,

$$\xi \sim \tfrac{1}{5}\sum_{i=1}^{5} \mathcal{N}(\mu_i, \sigma), \quad \zeta \sim \mathcal{N}(0, \sigma). \quad (14)$$

Throughout this experiment, we fix σ to 0.2 and choose μ_i to be $\{-1.5, -0.5, 0.5, 1.5, 2\}$. We initialize s_0 with ξ.

Note that our setting is a modification of de Freitas's demo[4] where we only change the process noise from a unimodal gamma distribution to a more complicated mixture model.

The task of particle filtering (Doucet et al., 2001) is to infer the hidden state given past and current observations. This can be carried out by estimating the filtering density $p(s_t|Y_t) := p(s_t|y_1, \ldots, y_t)$ recursively in a two-stage procedure. First, the current filtering density $p(s_t|Y_t)$ is propagated into the future via the transition density $p(s_{t+1}|s_t)$ to produce the prediction density $p(s_{t+1}|Y_t)$, *i.e.*,

$$\mathbb{E}_{s_t \sim p(s_t|Y_t)}[p(s_{t+1}|s_t)] := \int p(s_{t+1}|s_t) p(s_t|Y_t) \mathrm{d}s_t. \quad (15)$$

Second, $p(s_t|Y_t)$ is updated via Bayes' law,

$$p(s_{t+1}|Y_{t+1}) \propto p(y_{t+1}|s_{t+1}) p(s_{t+1}|Y_t). \quad (16)$$

The integral in (15) is usually intractable since the filtering density $p(s_t|Y_t)$ can take a complicated form. Therefore, $p(s_t|Y_t)$ is often approximated with a set of samples called particles. For distributed inference, it is these samples that need to be passed around. We want to compress the samples using density estimation such that we still do well in computing $\mathbb{E}_{s_t \sim p(s_t|Y_t)}[p(s_{t+1}|s_t)]$. In our example, $p(s_{t+1}|s_t)$ takes the form

$$p(s_{t+1}|s_t) \propto \sum_{i=1}^{5} \exp\left(-\tfrac{(s_{t+1} - f(s_t) - \mu_i)^2}{2\sigma^2}\right). \quad (17)$$

In terms of variable s_t, $p(s_{t+1}|s_t)$ is in the RKHS with kernel $k(x, x') = \exp\left(-\frac{(x-x')^2}{2(2\sigma)^2}\right)$. We can customize KMM using this kernel, and compress messages by targeting a good approximation of $\mathbb{E}_{s_t \sim p(s_t|Y_t)}[p(s_{t+1}|s_t)]$.

We use 5 centers for both GMM and KMM to compress the messages. We compare the filtering results with the true states. The error is measured as the root mean square of the deviations. The results for compressing different numbers of particles are reported in Table 5. We find that filtering results after compression even outperform those obtained from the full set of particles (PF). In particular, the results for KMM are slightly better than those for GMM. By compression, we have extracted the information most essential to statistical inference, and actually made the inference more robust. If the compression is targeted to $\mathbb{E}_{s_t \sim p(s_t|Y_t)}[p(s_{t+1}|s_t)]$ (as we do in KMM), we can simply get better results.

The shortcomings of general purpose density estimation also arise in the more general settings of message passing and belief propagation. This is due to the way messages are constructed: given a clique, the incoming messages are multiplied by the clique potential and all variables not in the receiver are integrated out. In most cases, this makes the outgoing messages very complicated, causing significant computational problems. Popular methods include

[4] http://www.cs.ubc.ca/∼nando/software/upf_demos.tar.gz

Table 4. Discrepancy between function expectations over randomizations. Smaller numbers are not necessarily statistical significant.

Data	Linear Functions				Polynomials ($d=2$)				Polynomials ($d=3$)				RBF Functions			
	PZ	GMM	RSDE	LIN	PZ	GMM	RSDE	POL2	PZ	GMM	RSDE	POL3	PZ	GMM	RSDE	RBF
covertype	**2.003**	**2.003**	10.280	**2.003**	0.185	0.194	0.396	**0.150**	**0.418**	0.539	1.240	0.412	0.073	0.023	0.071	**0.020**
ionosphere	**2.006**	**2.006**	17.995	**2.006**	**0.159**	0.232	0.383	0.169	**0.615**	0.664	1.659	0.626	0.120	0.024	0.142	**0.022**
sonar	**2.000**	**2.000**	12.288	**2.000**	0.971	0.354	0.933	**0.242**	0.691	0.745	2.558	**0.673**	0.857	0.030	0.873	**0.029**
australian	**2.000**	**2.000**	14.217	**2.000**	**0.369**	0.380	0.587	0.380	**0.832**	0.837	1.031	0.833	0.089	0.028	0.106	**0.024**
specft	**2.000**	**2.000**	3.594	**2.000**	0.891	0.515	0.522	**0.488**	0.922	0.878	1.265	**0.867**	0.903	0.067	0.904	**0.062**
wdbc	**2.004**	**2.004**	16.447	**2.004**	0.209	0.233	0.406	**0.166**	0.519	0.612	1.362	**0.512**	0.482	0.027	0.456	**0.023**
wine	**2.017**	**2.017**	9.489	**2.017**	0.822	0.236	1.027	**0.211**	**0.679**	0.718	2.782	**0.682**	0.471	0.040	0.545	**0.039**
satimage	**2.000**	**2.000**	27.561	**2.000**	0.146	0.126	0.533	**0.122**	0.260	0.281	1.230	**0.256**	0.307	0.028	0.359	**0.026**
segment	**2.003**	**2.003**	23.388	**2.003**	**0.258**	0.245	0.803	**0.263**	**0.590**	0.572	1.021	**0.588**	0.053	0.025	0.247	**0.022**
vehicle	**2.005**	**2.005**	26.331	**2.005**	**0.126**	0.135	0.780	**0.119**	**0.496**	**0.478**	1.686	**0.493**	0.095	0.028	0.325	**0.027**
svmguide2	**2.005**	**2.005**	7.248	**2.005**	3.468	0.247	3.341	**0.183**	0.866	0.782	2.603	**0.729**	0.798	0.019	0.808	**0.018**
vowle	**2.000**	**2.000**	12.913	**2.000**	**0.131**	0.150	0.642	**0.131**	**0.348**	0.394	1.741	**0.352**	0.028	0.019	0.111	**0.018**
housing	**2.000**	**2.000**	7.668	**2.000**	**0.117**	0.126	0.399	**0.121**	**0.393**	0.421	0.890	**0.391**	0.044	0.027	0.091	**0.025**
bodyfat	**2.000**	**2.000**	7.295	**2.000**	0.288	0.243	0.595	**0.242**	1.029	**1.017**	1.200	**1.015**	0.430	0.038	0.432	**0.037**
abalone	**2.005**	**2.005**	17.010	**2.005**	0.105	0.101	0.234	**0.103**	0.629	0.636	3.308	**0.628**	0.049	0.044	0.294	**0.043**
mix3 100	**2.000**	**2.000**	2.164	**2.000**	**0.153**	**0.152**	0.164	**0.152**	0.248	**0.242**	0.271	0.244	0.046	**0.044**	0.046	**0.044**
mix3 500	**2.000**	**2.000**	2.069	**2.000**	0.064	0.062	0.064	**0.062**	0.094	**0.092**	0.097	**0.091**	0.020	0.019	0.020	**0.019**
mix3 1000	**2.000**	**2.000**	2.035	**2.000**	0.052	**0.051**	0.051	**0.050**	0.082	0.081	0.082	**0.080**	0.015	**0.014**	**0.015**	**0.014**

Table 5. Root mean square error and standard deviation of the filtering results before and after particle compression. We randomly instantiated the system 50 times and concatenate the times to produce the results. Statistical tests are done by viewing each time point as a data point.

Particle #	PF	GMM	KMM
100	0.683±0.114	**0.558±0.084**	**0.546±0.072**
500	0.679±0.111	**0.556±0.076**	**0.530±0.070**
1000	0.685±0.111	0.556±0.082	**0.526±0.070**

particle filtering, which uses a discrete approximation of the messages, and expectation propagation, which uses a single Gaussian approximation of the messages (Minka, 2001). We plan to further investigate KMM in these general settings. Our key benefit is that we can customize the approximation properties for a particular graphical model.

8.2. Image Retrieval and Categorization

Following the work of (Rubner et al., 2000; Greenspan et al., 2002), we use density estimation as an intermediate step for image retrieval and categorization.

8.2.1. Image Retrieval

Image retrieval is the task of finding from a given database the set of images similar to a given query image. An image is normally characterized by the distribution over features (*e.g.*, color, texture) of pixels, patches, etc. It is thus helpful to compress the distribution by density estimation into more compact forms (*e.g.*, mixtures of Gaussians), on which the query is based. In particular, the advantage is that density estimation can be computed offline before the query takes place, thus offering computational and storage savings.

Method Greenspan et al. (2002) used GMM for density estimation; we propose KMM as an alternative. After density estimation, the dissimilarity between two distributions needs to be measured and the Earth Mover's Distance (EMD) is a state-of-the-art measure. Given two distributions represented by sets of weighted prototypes, EMD regards one collection as mass of earth spread in the feature space, while the other is a collection of holes. The EMD is defined as the least amount of work needed to fill the holes with earth. A unit of work corresponds to the ground distance between two prototypes. If we represent the distributions by mixtures of Gaussians, then a sensible ground distance $D(p_i, p'_j)$ between two Gaussians $p_i = \mathcal{N}(\mu_i, \Sigma_i)$ and $p'_j = \mathcal{N}(\mu'_j, \Sigma'_j)$ is the Fréchet distance used in (Greenspan et al., 2002),

$$D^2(p_i, p'_j) := |\mu_i - \mu'_j|^2 + \mathrm{tr}\left(\Sigma_i + \Sigma'_j - 2(\Sigma_i\Sigma'_j)^{1/2}\right).$$

Based on $D(p_i, p'_j)$, if $p = \sum_i \alpha_i p_i$ where p_i is a Gaussian and α_i is its weight, and similarly $p' = \sum_j \alpha'_j p'_j$, then the EMD between p and p' is

$$\mathrm{EMD}(p, p') := \min_{\gamma_{ij} \text{ feasible}} \sum_i \sum_j \gamma_{ij} D(p_i, p'_j),$$

where $\gamma_{ij} \geq 0$ is the flow between p_i and q_j. Feasibility means $\sum_i \gamma_{ij} \leq \alpha'_j$ and $\sum_j \gamma_{ij} \leq \alpha_i$ for all i and j.

Settings In this experiment, the distance measure is fixed to EMD. We plug the densities estimated by GMM and KMM into EMD[5], and compare the retrieval results. Parameters for KMM and GMM were chosen in the same way as in Section 7.4. Here KMM used POL3. For each image, we sampled 10^3 pixels and each pixel's feature vector was the CIE-Lab value of a 5×5 window centered on it.

Results We collected $L = 10537$ images from various sources including FIRE and CIRES[6]. The dataset included 10 labeled categories like horse, beach, and each category has 100 images. For each image $I_c(i)$ from class c ($c \in \{1, ..., 10\}$, $i \in \{1, ..., 100\}$), we retrieved r ($r \in \{1, ..., L\}$) closest images (in terms of EMD) from the whole database and counted how many among them are also from class c, denoted as $g_c(i, r)$ for GMM and $k_c(i, r)$ for KMM. For each c and r, we performed a paired sign test between $\{g_c(i, r)\}_{i=1}^{100}$ and $\{k_c(i, r)\}_{i=1}^{100}$. Since p-value is always in (0,1], we report in Figure 3 the log p-value if the median of $\{k_c(i, r) - g_c(i, r)\}_{i=1}^{100}$ is higher than 0. Otherwise, we plot the *negative* log p-value. Negative values are in favor of KMM. In Figure 3, performance of KMM is superior to or competitive with GMM in 8 categories and for most values of r (number of retrieved images).

[5]EMD code from http://ai.stanford.edu/∼rubner/emd

[6]FIRE: http://www-i6.informatik.rwth-aachen.de/∼deselaers/fire.html, CIRES: http://cires.matthewriley.com

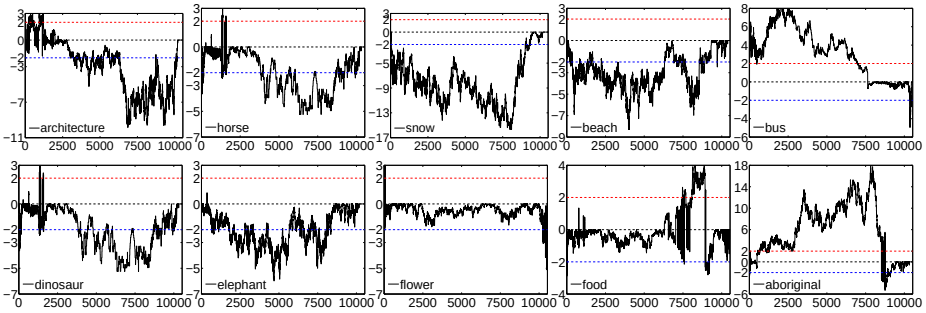


Figure 3. Log sign test p-value (vertical axis) v.s. # retrieved images (horizontal axis). Negative if KMM is better than GMM, and positive otherwise. ± 2 for significance level 0.01.

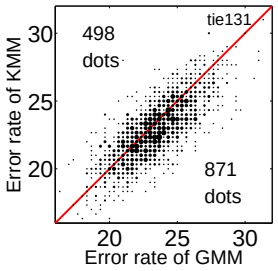


Figure 4. Error rate of image categorization using KMM and GMM.

It is important to note that the Fréchet distance is *not* in the class of functions used in KMM, and KMM still performs reasonably well. In the next section, we learn an image classifier using the *same* kernel as used in KMM.

8.2.2. Image Categorization

A closely related but different task is learning to categorize images using multi-class classification, particularly by SVM. Here all we need is a kernel between pairs of image densities p and q, which is readily given by $\langle \mu[p], \mu[q] \rangle_{\mathcal{H}}$. The SVM classifier takes the form $f(p_j) = \sum_i \gamma_i \langle \mu[p_i], \mu[p_j] \rangle = \mathbb{E}_{x \sim p_j}[\sum_i \gamma_i \mu[p_i](x)]$ for some $\gamma_i \in \mathbb{R}$. Since $\sum_i \gamma_i \mu[p_i] \in \mathcal{H}$, KMM ensures that p_j is estimated such that this expectation is well approximated.

Our 10-class classification used 1000 images from the 10 categories. We randomly divided each category into 70 images for training and 30 images for testing. We used LibSVM to train a multi-class SVM with one-against-one criterion on the combined 700 training images. The loss and regularization tradeoff parameter was determined by an inner loop 10-fold cross validation on the training data. Finally we test the accuracy of the learned model on the 300 test images. The whole process is repeated for 1500 times. We use POL3 for both KMM and SVM, because for both GMM and KMM, POL3 significantly outperforms POL2 and RBF in practice[7]. By using paired sign test, KMM yields lower error rate than GMM at significance level 0.01. Figure 4 shows the scatter plot of the resulting error rates.

Acknowledgements NICTA is funded by the Australian Government's Backing Australia's Ability and the Centre of Excellence programs. This work is also supported by the IST Program of the European Community, under the FP7 Network of Excellence, ICT-216886-NOE.

References

Altun, Y., & Smola, A. (2006). Unifying divergence minimization and statistical inference via convex duality. In *COLT 2006*.

Barndorff-Nielsen, O. E. (1978). *Information and Exponential Families in Statistical Theory*.

Bartlett, P. L., & Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *JMLR*, *3*, 463–482.

de Farias, N., & Roy, B. (2004). On constraint sampling in the linear programming approach to approximate dynamic programming. *Math. Oper. Res.*, *29*(3).

Doucet, A., de Freitas, N., & Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Dudík, M., Phillips, S., & Schapire, R. (2004). Performance guarantees for regularized maximum entropy density estimation. In *COLT 2004*.

DuMouchel, W., Volinsky, C., Cortes, C., Pregibon, D., & Johnson, T. (1999). Squashing flat files flatter. In *KDD 1999*.

Girolami, M., & He, C. (2003). Probability density estimation from optimally condensed data samples. *IEEE TPAMI*, *25*(10), 1253–1264.

Greenspan, H., Dvir, G., & Rubner, Y. (2002). Context-based image modeling. In *ICPR 2002*.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., & Smola, A. (2007). A kernel method for the two-sample-problem. In *NIPS 2007*.

Jebara, T., Kondor, R., & Howard, A. (2004). Probability product kernels. *JMLR*, *5*, 819–844.

McLachlan, G., & Basford, K. (1988). *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker.

Minka, T. (2001). *Expectation Propagation for approximate Bayesian inference*. Ph.D. thesis, MIT.

Parzen, E. (1962). On estimation of a probability density function and mode. *Ann. Math. Stat.*, *33*, 1065–1076.

Rubner, Y., Tomasi, C., & Guibas, L. (2000). The earth mover's distance as a metric for image retrieval. *Intl. J. Computer Vision*, *40*(2), 99–121.

Shawe-Taylor, J., & Dolia, A. (2007). A framework for probability density estimation. In *AISTATS 2007*.

Silverman, B. W. (1986). *Density estimation for statistical and data analysis*. Monographs on statistics and applied probability. Chapman and Hall.

Smola, A., Gretton, A., Song, L., & Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *ALT 2007*.

Steinwart, I. (2002). The influence of the kernel on the consistency of support vector machines. *JMLR*, *2*, 463–482.

Vapnik, V., & Mukherjee, S. (2000). Support vector method for multivariate density estimation. In *NIPS 12*.

Wainwright, M. J., & Jordan, M. I. (2003). Graphical models, exponential families, and variational inference. Tech. Rep. 649, UC Berkeley.

[7] The error rate of GMM using POL3 is $23.3 \pm 2.24\%$, outperforming POL2 ($23.8 \pm 2.14\%$) at significance level 0.01. KMM has $22.9 \pm 2.19\%$ error using POL3, beating POL2 ($24.3 \pm 2.14\%$) at significance level 0.01. Using an RBF kernel where the fixed value of the bandwidth θ was tested over $0.01, 0.1, 1, 10$, and 100 times the median distance between 1000 images, both GMM and KMM incur over 50% error.

Detecting Statistical Interactions with Additive Groves of Trees

Daria Sorokina DARIA@CS.CORNELL.EDU
Rich Caruana CARUANA@CS.CORNELL.EDU
Mirek Riedewald MIREK@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY, USA

Daniel Fink DF36@CORNELL.EDU
Cornell Lab of Ornithology, Ithaca, NY, USA

Abstract

Discovering additive structure is an important step towards understanding a complex multi-dimensional function because it allows the function to be expressed as the sum of lower-dimensional components. When variables interact, however, their effects are not additive and must be modeled and interpreted simultaneously. We present a new approach for the problem of interaction detection. Our method is based on comparing the performance of unrestricted and restricted prediction models, where restricted models are prevented from modeling an interaction in question. We show that an additive model-based regression ensemble, Additive Groves, can be restricted appropriately for use with this framework, and thus has the right properties for accurately detecting variable interactions.

1. Introduction

Many scientific inquiries seek to identify what variables are important and to describe their effects. Discovery of additive structure is an important step towards understanding a complex multi-dimensional function, because it allows for expressing this function as the sum of lower-dimensional components. When variables interact, their effects cannot be decomposed into independent lower-dimensional contributions and hence must be modeled simultaneously. In this paper we develop a methodology to automatically identify additive and interactive structure among large sets of variables.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

The term *statistical interaction* is used to describe the presence of non-additive effects among two or more variables in a function. Two variables are said to interact when the effect of one variable on the response depends on values of the other variable. Precisely, variables x_i and x_j interact in $F(\mathbf{x})$ when partial derivative $\frac{\partial F(\mathbf{x})}{\partial x_i}$ depends on x_j or, more generally, when the *"difference in the value of $F(\boldsymbol{x})$ for different values of x_i depends on the value of x_j"* (Friedman & Popescu, 2005). This is equivalent to the following definition:

Function $F(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, shows no interaction between variables x_i and x_j if it can be expressed as the sum of two functions, $f_{\setminus j}$ and $f_{\setminus i}$, where $f_{\setminus j}$ does not depend on x_j and $f_{\setminus i}$ does not depend on x_i:

$$\begin{aligned} F(\mathbf{x}) &= f_{\setminus j}(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n) \\ &\quad + f_{\setminus i}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \end{aligned} \quad (1)$$

For example, $F(x_1, x_2, x_3) = \sin(x_1 + x_2) + x_1 x_3$ has interactions between x_1 and x_2 and also between x_1 and x_3, but no interaction between x_2 and x_3.[1]

Higher-order interactions between a larger number of variables are defined similarly. There is no K-way interaction between K variables in the function, if it can be represented as a sum of K (or fewer) functions, each of which does not depend on at least one variable in question. If such representation is not possible, we say that there is a K-way interaction. Function $x_1^{x_2+x_3}$ shows a 3-way interaction between x_1, x_2 and x_3, while $x_1x_2 + x_2x_3 + x_1x_3$ has all pairwise interactions, but not a 3-way interaction.

[1] It is important to stress that the concept of statistical interaction is completely unrelated to the dependence and independence of variable distributions. Some authors use "interaction" to refer to different types of dependencies between variables, e.g., correlation (Jakulin & Bratko, 2004). In this paper we discuss statistical (non-additive) interactions only, not correlation or statistical dependence.

Interaction detection has high practical importance because it provides valuable knowledge about a domain. For example, our experiments with bird abundance data (Section 7) demonstrate that detection of spatio-temporal interactions can signal changes in the environment. In this particular case, a fatal eye disease was spreading slowly from the Northeastern US to other regions. This disease affected the annual bird abundance differently depending on location, creating a strong interaction between time and location.

Interactions are also an important part of statistical analysis. Early methods for interaction detection were parametric and required explicit modeling of interactions, most often as multiplicative terms. As a consequence, only limited types of interactions could be detected. More general approaches were introduced recently (Friedman & Popescu, 2005; Hooker, 2007). These methods are based on building a model and detecting interactions in the function learned by the model. A major shortcoming of this approach is that the model may detect spurious interactions over regions of the input space where data is scarce, and known solutions to this problem are either inadequate or computationally expensive. (See (Hooker, 2007) and Section 8 of this paper for more details.)

We introduce a new approach to interaction detection. It is based on comparing the performance of *restricted* and *unrestricted* predictive models. This avoids the drawbacks of previous methods, because it does not require explicit modeling of interacting terms and reports only those interactions that are present in the actual input data. However, the choice of model and the restriction algorithm used are crucial for this framework. We explain why additive models are able to provide the required accurate restrictions and further show that Additive Groves (Sorokina et al., 2007), an additive model-based ensemble of regression trees, works well in this framework. We also investigate how correlations in the data complicate interaction detection and suggest how this problem can be dealt with via feature selection.

The advantage of our new approach for interaction detection, compared with traditional statistical approaches, is that it is more automatic and does not require limiting the functional form that interactions might take. Statistical methods often represent only multiplicative interactions and thus may miss other forms of interactions. When little is known about the system under study, data-driven scientific discovery requires the data to "speak for themselves" with a minimum of analyst input or assumptions. It is possible to conduct a fully nonparametric analysis with the method we propose in this paper, which is particularly valuable for exploratory analysis.

2. Estimating Interactions

Let $F^*(\mathbf{x})$ be an unknown target function and let $F(\mathbf{x})$ be a highly accurate model of F^* that can be learned from a given set of training data. Furthermore, let $R_{ij}(\mathbf{x})$ denote a *restricted* model of F^* that is learned from the same training data. It is restricted in the sense that it is not allowed to contain an interaction between x_i and x_j, but apart from this limitation should be as accurate a model of F^* as possible.

Our interaction estimation technique is based on the following observation. If x_i and x_j interact, then $F(\mathbf{x})$ should have significantly better predictive performance than $R_{ij}(\mathbf{x})$, because the latter cannot accurately capture the true functional dependency between x_i and x_j. On the other hand, if the two variables do not interact, then the absence of the interaction from the model should not hurt its quality. Hence in the absence of an interaction between x_i and x_j the predictive performance of the restricted and the unrestricted model should be comparable. Note that in order to get an adequate estimate of performance, we must measure it on test data not used for training.

Quantifying interaction strength. We can quantify I_{ij}, the degree of interaction between x_i and x_j, by the difference in performance between $F(\mathbf{x})$ and $R_{ij}(\mathbf{x})$. We measure performance as standardized RMSE: root mean squared error (RMSE) scaled by the standard deviation in the response function. Scaling is done to make the results comparable across different data sets; $\mathrm{StD}(F^*(\mathbf{x}))$ is calculated as standard deviation of the response values in the training data.

$$\mathrm{stRMSE}(F(\mathbf{x})) = \frac{\mathrm{RMSE}(F(\mathbf{x}))}{\mathrm{StD}(F^*(\mathbf{x}))} \quad (2)$$

$$I_{ij}(F(\mathbf{x})) = \mathrm{stRMSE}(F(\mathbf{x})) - \mathrm{stRMSE}(R_{ij}(\mathbf{x})) \quad (3)$$

Setting the threshold. To distinguish whether a positive value of I_{ij} indicates presence of an interaction or happened due to random variation, we measure whether the performance of $R_{ij}(\mathbf{x})$ is significantly different from the performance of $F(\mathbf{x})$. We follow common practice and define a difference of three standard deviations of the latter from its mean as significant. The distribution of $\mathrm{stRMSE}(F(\mathbf{x}))$ can come either from different random seeds for bagging or from different data samples (e.g., n-fold cross validation). The threshold for significant interactions then becomes:

$$I_{ij}(F(\mathbf{x})) > 3 \cdot \mathrm{StD}(\mathrm{stRMSE}(F(\mathbf{x}))) \quad (4)$$

Note that everything above naturally generalizes to higher-order interactions as long as there exists a method to restrict the model on a specific type of interaction.

3. Choosing a Prediction Model

To correctly estimate interaction strength with our model comparison technique, we have to make sure that a model has the following key properties:

1. High predictive performance when modeling interactions: if there is an interaction, it should be captured by the unrestricted model.
2. High predictive performance when the model is restricted on non-interacting variables: if there is no interaction, performance of the restricted model should be no worse than the performance of the corresponding unrestricted model.

The first requirement is satisfied by many learning techniques, e.g., bagged decision trees of adequate depth, SVMs, or neural nets. Boosted stumps, on the other hand, do not model interactions. Since they represent functions as the sum of components, each of which depends only on a single variable, boosted 1-level stumps cannot be used in our framework.

While many models satisfy the first requirement, the second requirement — that models perform as well when interaction between non-interacting variables is restricted — is far more challenging. Even when there is a straightforward way of explicitly preventing specific interactions, often the resulting restricted model will not perform as well as the unrestricted model because the restriction may hamper the search in model space compared to the unrestricted model.

Consider a single decision tree. Variables in the tree can interact only if they are used on the same branch of the tree. So the obvious way to restrict interaction between specific variables is to not use one of them if the other already was used earlier on this branch. Now suppose there is no interaction between variables A and B, but they both are important — if the tree does not use one of them, its performance drops. Assume further that A is more important than B. The tree will tend to choose A earlier than B on all branches (in the worst case it will use A at the root) and will then never be able to choose B. Since B is important, the performance of this restricted tree will drop even though there was no interaction between A and B.

One might be tempted to address this problem with an ensemble method like bagging. Unfortunately the situation will not improve much. In bagging, every tree tries to capture the same function from a different sample of the train set. If A is more important, most trees will choose A before B, use of B will be restricted, and performance will drop as before.

Additive models. To detect *absence of interactions* between important variables, we need to build a restricted model that uses these variables in different additive components of the function. There is a class of ensembles that allows us to do this: additive models. Each component in an additive model is trained on the residuals of predictions of all other previous models in the ensemble. The training set for the new model component is created as the difference between true function values and current predictions of the ensemble. This way, when the function has additive structure, different models (or groups of models) are forced to find and model different components of this structure as opposed to each modeling the whole function.

Not all models that fit residuals are suitable for this framework. Linear models do not model interactions, while generalized linear models disguise additive structure with a non-linear transformation. Neural networks pose problems because they either have additive structure (1 internal layer), or the ability to model complex non-linear functions (several layers), while we need an algorithm that combines both. Restricting interactions in a multi-level network splits it into subnets, ultimately leading to "groves of nets".

In this paper we use layered Additive Groves (Sorokina et al., 2007). There exist other methods that might work as well, e.g., gradient boosting trained to minimize least squares loss (Friedman, 2001). However, it is important to understand that the two requirements stated in the beginning of this section are crucial and many (most?) learning algorithms do not satisfy them.

4. Additive Groves of Regression Trees

Additive Groves is an ensemble of trees introduced in (Sorokina et al., 2007). The combination of the ability to model additive structure of the response and to also use large trees that capture complex interactions make Groves suitable for interaction detection.

A single Grove of trees is an additive model where each additive component is represented by a regression tree. Additive Groves use regression trees trained to minimize mean squared error. Tree size is controlled by a parameter α, the minimum fraction of train set cases in a non-leaf node. A single Grove is trained similar to an additive model: each tree is trained on the residuals of the sum of the predictions of the other trees. Trees are discarded and retrained in turn until the overall predictions converge to a stable function. For the pur-

Algorithm 1 Layered training of a single Grove

function Layered(α,N,$TrainSet\{\mathbf{x}, y\}$)
  $\alpha_0 = 0.5, \alpha_1 = 0.2, \alpha_2 = 0.1, \ldots, \alpha_{\max} = \alpha$
  for $i = 1$ **to** N **do**
    $\text{Tree}_i = 0$
  for $j = 0$ **to** max **do**
    repeat
      for $i = 1$ **to** N **do**
        $\text{newTrainSet} = \{\mathbf{x}, y - \sum_{k \neq i} \text{Tree}_k(\mathbf{x})\}$
        $\text{Tree}_i = \text{TrainTree}(\alpha_j, \text{newTrainSet})$
    until (change from the last iteration is small)

pose of interaction detection we use layered training of Additive Groves (Algorithm 1). The main difference between layered training and training classical additive models is the following: Additive Groves begin with an ensemble of very small trees; then during re-training we gradually increase tree size by adding more branches. This layered approach ensures fitting of additive structure of the response function. As with single trees, a single Grove can still overfit to the training data. Hence for the Additive Groves ensemble, we wrap bagging around the layered training algorithm: many single Groves are built on bootstrap samples of the training set and their results are averaged. This procedure reduces variance and yields a very powerful predictive model.

Additive models provide an intuitive and easy way for restricting interactions. Assume we want to restrict a single Grove to not contain interactions between x_i and x_j. Since the modeled function is computed as the sum of the predictions of the individual trees, we only have to enforce that none of the trees uses *both* x_i and x_j. To decide if a tree is not allowed to use x_i (or otherwise x_j), we use a greedy procedure. Each time we train a tree, we first construct two trees: one does not use x_i, the other does not use x_j. The one resulting in better performance is inserted into the model, the other one is discarded. For evaluating performance we use the out-of-bag samples, i.e., that part of the training data that did not get into the current sample and therefore was not used to train the trees.

If we need to restrict on a higher-order interaction (say, k-way interaction between k variables), we need to build k candidate trees instead of 2 every time: each tree is not allowed to use one of the variables. Note that the complexity of testing for a single k-way interaction depends only linearly on k.

(Sorokina et al., 2007) also suggest another, "dynamic programming", style of training for Additive Groves. The method starts with a single small tree. Then on every retraining stage it either increases tree size or adds another tree, which is decided by a heuristic. Although this method provides better performance for unrestricted models, we have encountered problems with it when training restricted models. Therefore we prefer layered Additive Groves for interaction detection. Note that we need to use layered training even for the unrestricted model in order for the performances to be comparable.

5. Feature Selection

Correlations among features are common and complicate the task of detecting interactions. Suppose there exists an interaction between variables x_i and x_j. At the same time, a third variable, x_k, is present in the data. Assume it is highly correlated with x_j, to such an extent that the model can freely use either x_k or x_j with similar results. In this case we will not be able to detect the interaction between x_i and x_j. When we restrict the model to prevent a tree from using x_j, it can use x_k instead and performance will not drop. The same will happen when we try to detect an interaction between x_i and x_k.

Correlation among features is an intrinsic problem of high dimensional data that confronts all methods for interaction detection. For example, methods based on partial dependence functions (Friedman & Popescu, 2005) suffer from a similar problem. The unrestricted prediction model might sometimes use x_j and sometimes x_k. As a result it will find only weak interaction between x_i and x_j and also between x_i and x_k, even though the true interactions are much stronger. If there are more than two correlated variables (again, this is common in high-dimensional datasets), the interaction can be spread out in tiny portions over all of them, making it virtually impossible to detect.

As a consequence, before attempting to detect interactions, we must eliminate correlations. This can be achieved by a feature selection process, which removes some of the variables. The final set of variables should be a compromise between two goals: (1) The performance of the unrestricted model should still be good, ideally at least as good as before feature selection. (2) Each variable should be important, i.e., if we remove it from the set of features, the performance of the unrestricted model should drop significantly. The second criterion also gives us an estimate of the maximum strength of interactions that we can detect: if the performance of the unrestricted model drops by δ when we remove x_i, then we cannot expect the performance of the best model restricted on x_i and x_j to drop by more than δ. The intuition here is that removing an

important variable is a stronger restriction than prohibiting its interactions.

We use a variant of backward elimination (Guyon & Elisseeff, 2003) for the feature selection process. The main idea is to greedily eliminate all features (variables) whose removal either improves performance or reduces performance by at most Δ compared to performance on the full-feature data set. In our experiments we estimated $d = \mathrm{StD}(\mathrm{RMSE}(F(\mathbf{x})))$, where $F(\mathbf{x})$ is the unrestricted model, before running feature selection and used $\Delta = 3d$.

The feature selection procedure is not stable—it depends on the order in which we test each feature. For example, if we consider two completely correlated variables x_j and x_k, we can remove x_j and leave x_k in the set of the features. Or we can do exactly the reverse, depending on which variable we tried to remove first during feature selection. If there is a strong notion of which features should stay in the data set after feature selection, i.e., if we want to test certain features for interactions, the feature selection process should be modified so that features of interest are not removed.

6. Complexity Issues

One concern about interaction detection is the need to conduct a separate test for each interaction. If we want to test for all possible interactions, in theory we need $O(n^k)$ tests, where n is the number of variables and k is the order of the interaction. However, such complexity is unlikely to be required in practice. First, the feature selection process usually leaves a relatively small set of features that makes it feasible to test all pairs for possible interactions. Second, as noted by (Hooker, 2004), interactions possess an important monotonicity property. A k-way interaction can only exist if all its corresponding $(k-1)$-interactions exist. This fact is a straightforward consequence from the definition of a k-way interaction. Hence after we have detected all 2-way interactions, we need to test for 3-way interactions only for those triples of variables that have all 3 pairwise interactions present, and so on. As complex interactions are rare in real datasets, in practice we usually need only few tests for higher-order interactions. Some domains do pose an exception, for example, see our experiments on the *kin8nm* dataset.

7. Experiments

We have applied our approach to both synthetic and real data sets. We can evaluate the performance of our algorithm on synthetic data because we know the true interactions; for real data we try to explain the detected interactions based on the data set description.

In all our experiments we used 100 iterations of bagging. Apart from that, Additive Groves requires two parameters to be set: N (number of trees in a single Grove) and α (fraction of train set cases in the leaf, controls size of a single tree). We determined the best values of α and N on a validation set and reported the performance of Additive Groves with these parameters on a test set. We ran each experiment for the unrestricted model 10 times, using different random seeds and therefore different bootstrap samples for bagging. From these results we estimated the distribution of performance and then calculated the interaction threshold using Equation 4. After that we ran the experiment for each unrestricted model only once. If the resulting estimate of the interaction was above the threshold, we considered it to be evidence of an interaction. Otherwise it was considered insignificantly different from zero, indicating absence of an interaction. Notice that due to variance, in the latter case the estimate could be even negative, but should always be close to zero.

7.1. Synthetic Data.

This data set was generated by a function that was previously used in (Hooker, 2004).

$$F(x) = \pi^{x_1 x_2}\sqrt{2x_3} - sin^{-1}(x_4) + log(x_3 + x_5) - \frac{x_9}{x_{10}}\sqrt{\frac{x_7}{x_8}} - x_2 x_7 \quad (5)$$

Variables x_1, x_2, x_3, x_6, x_7, x_9 are uniformly distributed between 0.0 and 1.0 and variables x_4, x_5, x_8 and x_{10} are uniformly distributed between 0.6 and 1.0. Training, validation and test set contain 1000 points each. Best parameters were detected as $\alpha = 0.02$ and $N = 8$. Feature selection eliminated variables x_6 (not present in the function) and x_8 (virtually no influence on the response). For each of the 28 pairs of remaining variables we constructed a restricted model and compared it to the unrestricted model. Figure 1 shows the interaction value for each variable pair as computed by Equation 2. The dashed line shows the threshold. We can see a group of strong interactions high above the threshold — pairs (x_1, x_2), (x_1, x_3), (x_2, x_3), (x_2, x_7), (x_7, x_9). All cases without interactions fall below the threshold. There are also several weak interactions in the data set: our estimate for (x_9, x_{10}) is barely above the threshold and we failed to detect interactions (x_3, x_5) and (x_7, x_{10}). By construction, x_5 and x_{10} have a small range and their interactions are not significant. There is only one triple of variables with 3 pairwise interactions detected: (x_1, x_2, x_3). A separate test correctly reveals that there is a 3-way interaction

between them. Note that this is the only higher-order interaction that we need to test to conclude the full analysis. The original formula has another 4-way interaction, (x_7, x_8, x_9, x_{10}), but interactions of x_8 and x_{10} turned out to be very weak in the data, so the model did not pick them up.

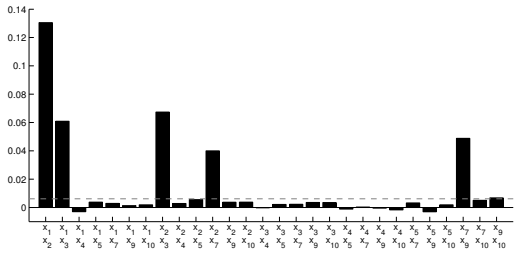


Figure 1. Interaction estimates on synthetic data

For more realistic results, we generated a version of the same data set with a 2 : 1 signal-to-noise ratio. Now feature selection left only 5 variables: x_1, x_2, x_3, x_5, x_7, and results of interaction detection between those variables were qualitatively the same as the correspondent results for the data set without noise.

7.2. Real data sets

We have run experiments on 5 real data sets, 4 of them are regression data sets from Luís Torgo's collection (Torgo, 2007), and the last one is a bird abundance data set from the Cornell Lab of Ornithology (Caruana et al., 2006). We used 4/5 of the data for training, 1/10 for validation and 1/10 for testing.

California Housing. California Housing is a regression data set introduced in (Pace & Barry, 1997). It describes how housing prices depend on different census data variables. Parameters used: $\alpha = 0.0005$, $N = 6$. Feature selection identified six variables as important: longitude, latitude, housingMedianAge, totalRooms, population and medianIncome. (Hooker, 2007) describes the joint effect of latitude and longitude on the response function. Our results confirm that there is a clear strong interaction between these two variables — the location effect on prices cannot be split into the sum of latitude and longitude effects. We have also found an evidence of interaction between population and totalRooms (Figure 2).

Elevators. This data set originates from an aircraft control task (Camacho, 1998). Parameters used: $\alpha = 0.02$ and $N = 18$. Feature selection left six variables: *climbRate*, *p*, *q*, *absRoll*, *diffRollRate*, *Sa*. We detected strong pairwise interactions in the triple (*absRoll*, *diffRollRate*, *Sa*) and a separate test confirmed

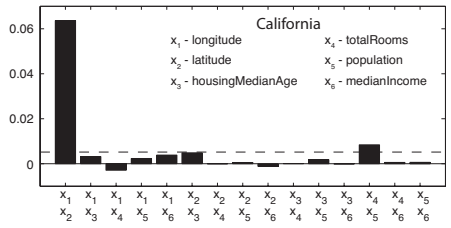


Figure 2. Interaction estimates for California Housing.

that this is indeed a strong 3-way interaction (Figure 3). No other interactions were found.

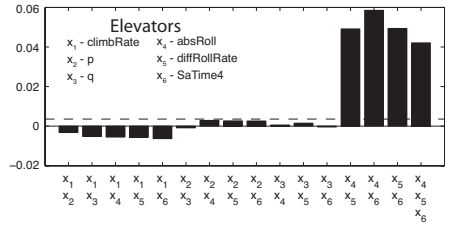


Figure 3. Interaction estimates for Elevators data.

Kinematics (kin8nm). The *kin8nm* dataset from the Delve repository (Rasmussen et al., 2003) describes a simulation of an 8-link robot arm movement. Its input variables correspond to the angular positions of the joints and it is classified as highly non-linear by its creators. Parameters used: $\alpha = 0.005$ and $N = 17$. Our analysis produced symmetrical results that reveal the simulation nature of the dataset: all 8 features turn out to be important, 2 of them do not interact with any other features and the other 6 are connected into a 6-way interaction (Figure 4). For brevity we show only results of tests for 2-way interactions and the final 6-way interaction, but we have also conducted tests for 20 3-way, 15 4-way and 6 5-way interactions between those 6 variables following the procedure described in Section 6. All tests confirmed the presence of interactions. *kin8nm* is the only data set where we had to test for many higher-order interactions. This is a property of the domain: the formula describing the end position of the arm based on joints angles results from interaction between most of the variables.

CompAct. Another dataset from the Delve repository, it describes the level of CPU activity in multiuser computer systems. Parameters used: $\alpha = 0.05$ and $N = 18$. Feature selection left 9 variables: *lread*, *scall*, *sread*, *exec*, *wchar*, *pgout*, *ppgin*, *vflt*, *freeswap*. This data set turns out to be very additive. Although there are many 2-way interactions, they all are relatively small (Figure 5). The largest interactions are (*freeswap*, *wchar*), describing the joint effect of the

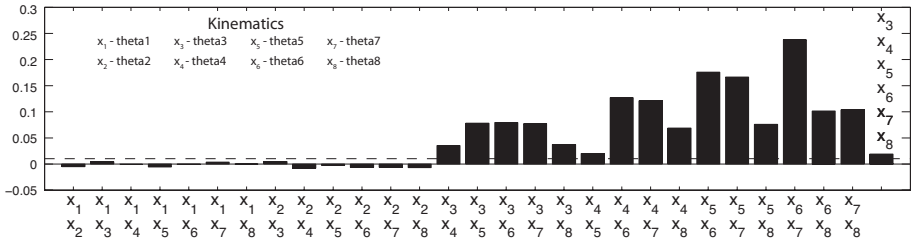


Figure 4. Interaction estimates for Kinematics (kin8nm) data.

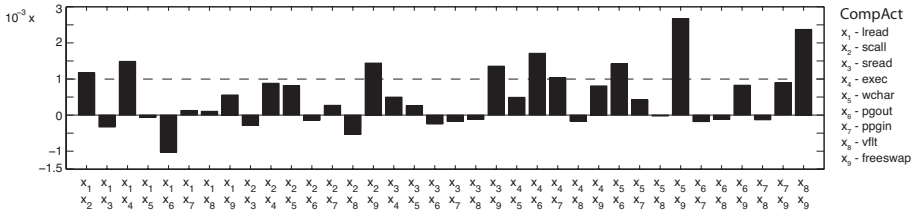


Figure 5. Interaction estimates for CPU Activity (CompAct) data set.

number of blocks available for swapping and system write call speed, and (*freeswap*, *vflt*), describing an interaction between the same available blocks variable and the number of page faults.

House Finch Abundance Data. We tested our approach on a dataset with sightings of House Finches in the North-Eastern US as introduced in (Caruana et al., 2006). The strongest interactions that we detected are between the following variables: (*latitude*, *longitude*, *elevation*) and (*year*, *latitude*, *longitude*). The first 3-way interaction describes the effect of geographical position which is expected to be non-additive. But the interactions between year and location is less trivial. Normally one would not expect that the effect of latitude or longitude on bird abundance would be very different in different years. However, it turns out that during the decade covered by the data set, the population of House Finches was suffering from an eye-disease that was spreading slowly and was responsible for changing the effect of geographical location on bird abundance over time. Our results show that interesting domain information like this can be discovered with the help of interaction detection analysis.

8. Previous Work

Interaction detection is regularly performed as part of statistical analysis (Christensen, 1996). Mostly parametric models are used where the analyst specifies the interaction as a parametric term, or perhaps several terms. In this setting interaction detection becomes a parameter estimation problem. More recently, techniques have been developed to detect interactions within semi-parametric models (Ruppert et al., 2003).

(Friedman & Popescu, 2005) developed tests for interaction detection for a very general class of prediction models, including fully nonparametric models. Their method makes use of the fact that in the absence of an interaction between x_i and x_j the following holds: $\frac{\partial F(\mathbf{X})^2}{\partial x_i \partial x_j} = \frac{\partial F(\mathbf{X})}{\partial x_i} + \frac{\partial F(\mathbf{X})}{\partial x_j}$. They estimate the partial dependence functions (Friedman & Popescu, 2005) of the model and then estimate the strength of an interaction as the difference between the right hand side and the left hand side of the equation above, scaled by variance in the response.

The drawback of that method is that in order to get accurate estimates of the partial dependence function, it relies on predictions for synthetic data points in sparse regions of the input space. As a result, decisions about presence of interactions can be made because of spurious interactions that happen only in those regions (Hooker, 2007). To demonstrate this effect, we generated two simple data sets for the function $F(\mathbf{x}) = x_1^3 + x_2^3$. In the first data set both x_1 and x_2 are distributed uniformly between -10 and 10. For the second data set we took the same points and removed those where both x_1 and x_2 were positive. Neither of the data sets contains interactions, but the estimates produced by Friedman's approach using RuleFit (Friedman, 2005) were 0.0243 for the first and 0.0824 for the second set. The presence of an unpopulated region in the input data increased the estimated strength of the presumed interaction by a

factor of three.

In order to deal with this extrapolation problem, (Friedman & Popescu, 2005) suggest comparing the estimated interaction strength produced by the method described above with a similar estimate on the same data, but for a different response function that does not contain any interactions. However, our experiments with RuleFit revealed several examples of unsatisfactory performance of this technique. For instance, we generated 5 data sets with response function $x_1^2+x_2^2$ without noise and for each of them generated 50 samples from the null distribution. For 3 of those data sets RuleFit produced results that indicated presence of an interaction, i.e., the original estimate was further from the mean of the null distribution than 3 standard deviations. In contrast, our method produced a confident estimation of the absence of interactions in all the cases described above.

(Hooker, 2007; Hooker, 2004) suggests another approach, based on estimating orthogonal components of the ANOVA decomposition. This method has higher computational complexity because it requires generating a full grid of data points with all possible combinations of values for those input variables that are tested for interaction. To overcome the problem of extrapolations over unpopulated regions of the input space, as well as problems caused by correlations, (Hooker, 2007) suggests imposing low weights for points from low-density regions. Unfortunately, this requires the use of external density estimation techniques and further increases complexity of the method.

We take a model comparison approach to interaction detection. In doing so, we do not need to calculate partial dependence functions to estimate predictor effects and we avoid the associated problem of spurious interactions from sparse regions. We believe this is a more direct approach to interaction detection.

9. Discussion

We presented a novel technique for detecting statistical interactions in complex data sets. The main idea is to compare the predictive performance of unrestricted models to restricted models, which do not contain the to-be-tested interaction. Although this idea is quite intuitive, there are significant practical challenges and few algorithms will work in this framework. We demonstrated that layered Additive Groves can be used in this approach due to its high predictive performance for both restricted and unrestricted models. Results on synthetic and real data indicate that we can reliably identify interactions.

Acknowledgements. We would like to thank Wes Hochachka, Giles Hooker, Steve Kelling and Art Munson for insightful discussions. This work was supported by grants ITR EF-0427914, SEI IIS-0612031, IIS-0748626 and NSF-0412930. Daria Sorokina was supported by a fellowship from Leon Levy foundation.

References

Camacho, R. (1998). Inducing models of human control skills. *Proc. ECML'98*.

Caruana, R., Elhawary, M., Fink, D., Hochachka, W. M., Kelling, S., Munson, A., Riedewald, M., & Sorokina, D. (2006). Mining citizen science data to predict prevalence of wild bird species. *KDD'06*.

Christensen, R. (1996). *Plane answers to complex questions, the theory of linear models*. Springer.

Friedman, J. (2005). RuleFit with R. http://www-stat.stanford.edu/~jhf/R-RuleFit.html.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.

Friedman, J. H., & Popescu, B. E. (2005). *Predictive learning via rule ensembles* (Technical Report). Stanford University.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *JMLR*, *3*.

Hooker, G. (2004). Discovering ANOVA structure in black box functions. *Proc. ACM SIGKDD*.

Hooker, G. (2007). Generalized functional ANOVA diagnostics for high dimensional functions of dependent variables. *JCGS*.

Jakulin, A., & Bratko, I. (2004). Testing the significance of attribute interactions. *Proc. ICML'04*.

Pace, R. K., & Barry, R. (1997). Sparse spatial autoregressions. *Statistics and Probability Letters*, *33*.

Rasmussen, C. E., Neal, R. M., Hinton, G., van Camp, D., Revow, M., Ghahramani, Z., Kustra, R., & Tibshirani, R. (2003). Delve. University of Toronto. http://www.cs.toronto.edu/~delve.

Ruppert, D., Wand, M. P., & Carroll, R. J. (2003). *Semiparametric regression*. Cambridge.

Sorokina, D., Caruana, R., & Riedewald, M. (2007). Additive Groves of regression trees. *Proc. ECML*.

Torgo, L. (2007). Regression DataSets. www.liacc.up.pt/~ltorgo/Regression/DataSets.html.

Metric Embedding for Kernel Classification Rules

Bharath K. Sriperumbudur BHARATHSV@UCSD.EDU
Omer A. Lang OLANG@UCSD.EDU
Gert R. G. Lanckriet GERT@ECE.UCSD.EDU
Department of Electrical and Computer Engineering, University of California, San Diego, CA 92093 USA.

Abstract

In this paper, we consider a smoothing kernel based classification rule and propose an algorithm for optimizing the performance of the rule by learning the bandwidth of the smoothing kernel along with a data-dependent distance metric. The data-dependent distance metric is obtained by learning a function that embeds an arbitrary metric space into a Euclidean space while minimizing an upper bound on the resubstitution estimate of the error probability of the kernel classification rule. By restricting this embedding function to a reproducing kernel Hilbert space, we reduce the problem to solving a semidefinite program and show the resulting kernel classification rule to be a variation of the k-nearest neighbor rule. We compare the performance of the kernel rule (using the learned data-dependent distance metric) to state-of-the-art distance metric learning algorithms (designed for k-nearest neighbor classification) on some benchmark datasets. The results show that the proposed rule has either better or as good classification accuracy as the other metric learning algorithms.

1. Introduction

Parzen window methods, also called smoothing kernel rules are widely used in nonparametric density estimation and function estimation, and are popularly known as kernel density and kernel regression estimates, respectively. In this paper, we consider these rules for classification. To this end, let us consider the binary classification problem of classifying $x \in \mathbb{R}^D$, given an i.i.d. training sample $\{(X_i, Y_i)\}_{i=1}^n$ drawn from some unknown distribution $\mathcal{D}$, where $X_i \in \mathbb{R}^D$ and $Y_i \in \{0, 1\}, \forall\, i \in [n] := \{1, \ldots, n\}$. The *kernel classification rule* (Devroye et al., 1996, Chapter 10) is given by

$$g_n(x) = \begin{cases} 0 & \text{if } \sum_{i=1}^n \mathbb{1}_{\{Y_i=0\}} K\left(\frac{x-X_i}{h}\right) \\ & \quad \geq \sum_{i=1}^n \mathbb{1}_{\{Y_i=1\}} K\left(\frac{x-X_i}{h}\right) \\ 1 & \text{otherwise,} \end{cases} \quad (1)$$

where $K : \mathbb{R}^D \to \mathbb{R}$ is a *kernel function*, which is usually nonnegative and monotone decreasing along rays starting from the origin. The number $h > 0$ is called the *smoothing factor*, or *bandwidth*, of the kernel function, which provides some form of distance weighting. We warn the reader not to confuse the kernel function, K, with the reproducing kernel (Schölkopf & Smola, 2002) of a reproducing kernel Hilbert space (RKHS), which we will denote with $\mathfrak{K}$.[1] When $K(x) = \mathbb{1}_{\{\|x\|_2 \leq 1\}}(x)$ (sometimes called the *naïve kernel*), the rule is similar to the k-nearest neighbor (k-NN) rule except that k is different for each X_i in the training set. The k-NN rule classifies each unlabeled example by the majority label among its k-nearest neighbors in the training set, whereas the kernel rule with the naïve kernel classifies each unlabeled example by the majority label among its neighbors that lie within a radius of h. Devroye and Krzyżak (1989) proved that for regular kernels (see Devroye et al., (1996, Definition 10.1)), if the smoothing parameter $h \to 0$ such that $nh^D \to \infty$ as $n \to \infty$, then the kernel classification rule is *universally consistent*. But, for a particular n, asymptotic results provide little guidance in the selection of h. On the other hand, selecting the wrong value of h may lead to very poor error rates. In fact, the crux of every nonparametric estimation problem is the choice of an appropriate smoothing factor. This is one of the questions that we address in this paper by proposing an algorithm to learn an optimal h.

The second question that we address is learning an optimal distance metric. For $x \in \mathbb{R}^D$, K is usually a function of $\|x\|_2$. Some popular kernels include the Gaussian kernel, $K(x) = e^{-\|x\|_2^2}$; the Cauchy kernel, $K(x) =$

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] Unlike $\mathfrak{K}$, K is not required to be a positive definite function. If K is a positive definite function, then it corresponds to a translation-invariant kernel of some RKHS defined on $\mathbb{R}^D$. In such a case, the classification rule in Eq. (1) is similar to the ones that appear in kernel machines literature.

$1/(1+\|x\|_2^{D+1})$; and the Epanechnikov kernel $K(x) = (1-\|x\|_2^2)\mathbb{1}_{\{\|x\|_2\leq 1\}}$.[2] Snapp and Venkatesh (1998) have shown that the finite-sample risk of the k-NN rule may be reduced, for large values of n, by using a weighted Euclidean metric, even though the infinite sample risk is independent of the metric used. This has been experimentally confirmed by Xing et al. (2003); Shalev-Shwartz et al. (2004); Goldberger et al. (2005); Globerson and Roweis (2006); Weinberger et al. (2006). They all assume the metric to be $\rho(x,y) = \sqrt{(x-y)^T\mathbf{\Sigma}(x-y)} = \|\mathbf{L}(x-y)\|_2$ for $x, y \in \mathbb{R}^D$, where $\mathbf{\Sigma} = \mathbf{L}^T\mathbf{L}$ is the weighting matrix, and optimize over $\mathbf{\Sigma}$ to improve the performance of the k-NN rule. Since the kernel rule is similar to the k-NN rule, one would expect that its performance can be improved by making K a function of $\|\mathbf{L}x\|_2$. Another way to interpret this is to find a linear transformation $\mathbf{L} \in \mathbb{R}^{d\times D}$ so that the transformed data lie in a Euclidean metric space.

Some applications call for natural distance measures that reflect the underlying structure of the data at hand. For example, when computing the distance between two images, *tangent distance* would be more appropriate than the Euclidean distance. Similarly, while computing the distance between points that lie on a low-dimensional manifold in $\mathbb{R}^D$, *geodesic distance* is a more natural distance measure than the Euclidean distance. Most of the time, since the true distance metric is either unknown or difficult to compute, Euclidean or weighted Euclidean distance is used as a surrogate. In the absence of prior knowledge, the data may be used to select a suitable metric, which can lead to better classification performance. In addition, instead of $x \in \mathbb{R}^D$, suppose $x \in (\mathcal{X}, \rho)$, where $\mathcal{X}$ is a metric space with ρ as its metric. One would like to extend the kernel classification rule to such $\mathcal{X}$. In this paper, we address these issues by learning a transformation that embeds the data from $\mathcal{X}$ into a Euclidean metric space while improving the performance of the kernel classification rule.

The rest of the paper is organized as follows. In §2, we formulate the multi-class kernel classification rule and propose learning a transformation, φ, (that embeds the training data into a Euclidean space) and the bandwidth parameter, h, by minimizing an upper bound on the resubstitution estimate of the error probability. To achieve this, in §3, we restrict φ to an RKHS and derive a representation for it by invoking the generalized representer theorem. Since the resulting optimization problem is non-convex, in §4, we approximate it with a semidefinite program when K is a naïve kernel. We present experimental results in §5, wherein we show on benchmark datasets that the proposed algorithm performs better than k-NN and state-of-the-art metric learning algorithms developed for the k-NN rule.

[2]The Gaussian kernel is a positive definite function on $\mathbb{R}^D$ while the Epanechnikov and naïve kernels are not.

2. Problem Formulation

Let $\{(X_i, Y_i)\}_{i=1}^n$ denote an i.i.d. training set drawn from some unknown distribution $\mathcal{D}$ where $X_i \in (\mathcal{X}, \rho)$ and $Y_i \in [L]$, with L being the number of classes. The multi-class kernel classification rule is given by

$$g_n(x) = \arg\max_{l\in[L]} \sum_{i=1}^{n} \mathbb{1}_{\{Y_i=l\}} K_{X_i,h}(x), \qquad (2)$$

where $K : \mathcal{X} \to \mathbb{R}_+$ and $K_{x_0,h}(x) = \chi\left(\frac{\rho(x,x_0)}{h}\right)$ for some nonnegative function, χ, with $\chi(0) = 1$. The probability of error associated with the above rule is $L(g_n) := \Pr_{(X,Y)\sim\mathcal{D}}(g_n(X) \neq Y)$ where Y is the true label associated with X. Since $\mathcal{D}$ is unknown, $L(g_n)$ cannot be computed directly but can only be estimated from the training set. The *resubstitution estimate*,[3] $\widehat{L}(g_n)$, which counts the number of errors committed on the training set by the classification rule, is given by $\widehat{L}(g_n) := \frac{1}{n}\sum_{i=1}^n \mathbb{1}_{\{g_n(X_i)\neq Y_i\}}$. As aforementioned, when $\mathcal{X} = \mathbb{R}^D$, ρ is usually chosen to be $\|.\|_2$. Previous works in distance metric learning learn a linear transformation $\mathbf{L} : \mathbb{R}^D \to \mathbb{R}^d$ leading to the distance metric, $\rho_{\mathbf{L}}(X_i, X_j) := \|\mathbf{L}X_i - \mathbf{L}X_j\|_2 = \sqrt{(X_i-X_j)^T\mathbf{\Sigma}(X_i-X_j)}$, where $\mathbf{\Sigma}$ captures the variance-covariance structure of the data. In this work, our goal is to jointly learn h and a measurable function, $\varphi \in \mathcal{C} := \{\varphi : \mathcal{X} \to \mathbb{R}^d\}$, so that the resubstitution estimate of the error probability is minimized with $\rho_\varphi(X_i, X_j) := \|\varphi(X_i) - \varphi(X_j)\|_2$. Once h and φ are known, the kernel classification rule is completely specified by Eq. (2).

Devroye et al., (1996, Section 25.6) show that kernel rules of the form in Eq. (1) picked by minimizing $\widehat{L}(g_n)$ with smoothing factor $h > 0$ are generally inconsistent if X is nonatomic. The same argument can be extended to the multi-class rule given by Eq. (2). To learn φ, simply minimizing $\widehat{L}(g_n)$ without any smoothness conditions on φ can lead to kernel rules that overfit the training set. Such a φ can be constructed as follows. Let n_l be the number of points that belong to l^{th} class. Suppose $n_1 = n_2 = \cdots = n_L$. Then for any $h \geq 1$, choosing $\varphi(X) = Y_i$ when $X = X_i$ and $\varphi(X) = 0$ when $X \notin \{X_i\}_{i=1}^n$ clearly yields zero resubstitution error. However, such a choice of φ leads to a kernel rule that always assigns the unseen data to class 1, leading to very poor performance. Therefore, to avoid overfitting to the training set, the function class $\mathcal{C}$ should satisfy some smoothness properties so that highly non-smooth functions like the one we defined above are not chosen while minimizing $\widehat{L}(g_n)$. To this end, we introduce a penalty functional, $\Omega : \mathcal{C} \to \mathbb{R}_+$, which penalizes

[3]Apart from the resubstitution estimate, holdout and deleted estimates can also be used to estimate the error probability. These estimates are usually more reliable but more involved than the resubstitution estimate.

non-smooth functions in $\mathcal{C}$ so that they are not selected.[4] Therefore, our goal is to learn φ and h by minimizing the regularized error functional given as

$$L_{reg}(\varphi, h) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{g_n(X_i)\neq Y_i\}} + \lambda' \, \Omega[\varphi], \qquad (3)$$

where $\varphi \in \mathcal{C}$, $h > 0$ and the regularization parameter, $\lambda' > 0$. g_n in Eq. (3) is given by Eq. (2), with ρ replaced by ρ_φ. Minimizing $L_{reg}(\varphi, h)$ is equivalent to minimizing the number of training instances for which $g_n(X) \neq Y$, over the function class, $\{\varphi : \Omega[\varphi] \leq s\}$, for some appropriately chosen s.

Consider $g_n(x)$ defined in Eq. (2). Suppose $Y_i = k$ for some X_i. Then $g_n(X_i) = k$ if and only if

$$\sum_{\{j:Y_j=k\}} K^{\varphi}_{X_j,h}(X_i) \geq \max_{\substack{l\in[L]\\ l\neq k}} \sum_{\{j:Y_j=l\}} K^{\varphi}_{X_j,h}(X_i), \qquad (4)$$

where the superscript φ is used to indicate the dependence of K on φ.[5] Since the right hand side of Eq. (4) involves the max function which is not differentiable, we use the inequality $\max\{a_1, \ldots, a_m\} \leq \sum_{i=1}^{m} a_i$ to upper bound[6] it with $\sum_{\substack{l\in[L]\\ l\neq k}} \sum_{j=1}^{n} \mathbb{1}_{\{Y_j=l\}} K_{X_j}(X_i)$. Thus, to maximize $\sum_{i=1}^{n} \mathbb{1}_{\{g_n(X_i)=Y_i\}}$, we maximize its lower bound given by $\sum_{i=1}^{n} \mathbb{1}_{\left\{\sum_{\substack{j=1\\ j\neq i}}^{n} \mathbb{1}_{\{Y_j=Y_i\}} K_{X_j}(X_i) \geq \sum_{j=1}^{n} \mathbb{1}_{\{Y_j\neq Y_i\}} K_{X_j}(X_i)\right\}}$, resulting in a conservative rule.[7] In the above bound, we use $j \neq i$ just to make sure that $\varphi(X_i)$ is not the only point within its neighborhood of radius h. Define $\tau_{ij} := 2\delta_{Y_i,Y_j} - 1$ where δ represents the Kronecker delta. Then, the problem of learning φ and h by minimizing $L_{reg}(\varphi, h)$ in Eq. (3) reduces to solving the following optimization problem,

$$\min_{\varphi,\, h} \Big\{ \sum_{i=1}^{n} \psi_i(\varphi, h) + \lambda \, \Omega[\varphi] \; : \; \varphi \in \mathcal{C},\, h > 0 \Big\}, \qquad (5)$$

where $\lambda = n\lambda'$ and $\psi_i(\varphi, h)$ given by

$$\mathbb{1}_{\left\{\sum_{\substack{j=1\\ j\neq i}}^{n} \mathbb{1}_{\{\tau_{ij}=1\}} K_{X_j}(X_i) < \sum_{j=1}^{n} \mathbb{1}_{\{\tau_{ij}=-1\}} K_{X_j}(X_i)\right\}}$$

is an upper bound on $\mathbb{1}_{\{g_n(X_i)\neq Y_i\}}$ for $i \in [n]$. Solving the above non-convex optimization problem is NP-hard. The gradient optimization is difficult because the gradients are zero almost everywhere. In addition to the computational hardness, the problem in Eq. (5) is not theoretically solvable unless some assumptions about $\mathcal{C}$ are made. In the following section, we assume $\mathcal{C}$ to be an RKHS with the reproducing kernel $\mathfrak{K}$ and provide a representation for the optimum φ that minimizes Eq. (5). We remind the reader that K is a smoothing kernel which is not required to be a positive definite function but takes on positive values, while $\mathfrak{K}$ is a reproducing kernel which is positive definite and can take negative values.

3. Regularization in Reproducing Kernel Hilbert Space

Many machine learning algorithms like SVMs, regularization networks, and logistic regression can be derived within the framework of regularization in RKHS by choosing an appropriate empirical risk functional with the penalizer being the squared RKHS norm (see Evgeniou et al. (2000)). In Eq. (5), we have extended this framework to kernel classification rules, wherein we compute the $\varphi \in \mathcal{C}$ and $h > 0$ that minimize an upper bound on the resubstitution estimate of the error probability. To this end, we choose $\mathcal{C}$ to be an RKHS with the penalty functional being the squared RKHS norm,[8] i.e., $\Omega[\varphi] = \|\varphi\|^2_{\mathcal{C}}$. By fixing h, the objective function $\sum_{i=1}^{n} \psi_i(\varphi, h)$ in Eq. (5) depends on φ only through $\{\|\varphi(X_i) - \varphi(X_j)\|_2\}_{i,j=1}^{n}$. By letting $\sum_{i=1}^{n} \psi_i(\varphi, h) = \theta_h\left(\{\|\varphi(X_i) - \varphi(X_j)\|_2\}_{i,j=1}^{n}\right)$ where $\theta_h : \mathbb{R}^{n^2} \to \mathbb{R}_+$, Eq. (5) can be written as

$$\min_{h>0} \min_{\varphi\in\mathcal{C}} \; \theta_h\left(\{\|\varphi(X_i) - \varphi(X_j)\|_2\}_{i,j=1}^{n}\right) + \lambda \, \|\varphi\|^2_{\mathcal{C}}. \qquad (6)$$

The following result provides a representation for the minimizer of Eq. (6), and is proved in Appendix A. We remind the reader that φ is a vector-valued mapping from $\mathcal{X}$ to $\mathbb{R}^d$.

Theorem 1 (Multi-output regularization)**.** *Suppose $\mathcal{C} = \{\varphi : \mathcal{X} \to \mathbb{R}^d\} = \mathcal{H}_1 \times \ldots \times \mathcal{H}_d$ where $\mathcal{H}_i$ is an RKHS with reproducing kernel $\mathfrak{K}_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $\varphi = (\varphi_1, \ldots, \varphi_d)$ with $\mathcal{H}_i \ni \varphi_i : \mathcal{X} \to \mathbb{R}$. Then each minimizer $\varphi \in \mathcal{C}$ of Eq. (6) admits a representation of the form*

$$\varphi_j = \sum_{i=1}^{n} c_{ij} \mathfrak{K}_j(., X_i), \;\; \forall \, j \in [d] \qquad (7)$$

where $c_{ij} \in \mathbb{R}$ and $\sum_{i=1}^{n} c_{ij} = 0$, $\forall \, i \in [n]$, $\forall \, j \in [d]$.

[4] This is equivalent to restricting the size of the function class $\mathcal{C}$ from which φ has to be selected.

[5] To simplify the notation, from now onwards, we write $K^{\varphi}_{X_j,h}(X_i)$ as $K_{X_j}(X_i)$ where the dependence of K on φ and h is implicit.

[6] Another upper bound that can be used for the max function is $\max\{a_1, \ldots, a_m\} \leq \log\left(\sum_{i=1}^{m} e^{a_i}\right)$.

[7] Using the upper bound of max function in Eq. (4) makes the resulting kernel rule conservative as there can be samples from the training set that do not satisfy this inequality but get correctly classified according to Eq. (2).

[8] Another choice for $\mathcal{C}$ could be the space of bounded Lipschitz functions with the penalty functional, $\Omega[\varphi] = \|\varphi\|_L$, where $\|\varphi\|_L$ is the Lipschitz semi-norm of φ. With this choice of $\mathcal{C}$ and Ω, von Luxburg and Bousquet (2004) studied large margin classification in metric spaces. One more interesting choice for $\mathcal{C}$ could be the space of Mercer kernel maps. It can be shown that solving for φ in Eq. (5) with such a choice for $\mathcal{C}$ is equivalent to learning the kernel matrix associated with φ and $\{X_i\}_{i=1}^{n}$. However, this approach is not useful as it does not allow for an out-of-sample extension.

Remark 2. *(a) By Eq. (7), φ is completely determined by $\{c_{ij} : i \in [n], j \in [d]\}$. Therefore, the problem of learning φ reduces to learning $n \cdot d$ scalars, $\{c_{ij} : \sum_{i=1}^{n} c_{ij} = 0\}$.*

(b) θ_h in Eq. (6) depends on φ through $\|\varphi(.) - \varphi(.)\|_2$. Therefore, for any $z, w \in \mathcal{X}$, we have $\|\varphi(z) - \varphi(w)\|_2^2 = \sum_{m=1}^{d} \left[\mathbf{c}_m^T (\mathbf{k}_m^z - \mathbf{k}_m^w)\right]^2 = \sum_{m=1}^{d} tr(\mathbf{\Sigma}_m (\mathbf{k}_m^z - \mathbf{k}_m^w)(\mathbf{k}_m^z - \mathbf{k}_m^w)^T)$ where $\mathbf{c}_m := (c_{1m}, \ldots, c_{nm})^T$, $\mathbf{k}_m^z := (\mathfrak{K}_m(z, X_1), \ldots, \mathfrak{K}_m(z, X_n))^T$, $\mathbf{\Sigma}_m := \mathbf{c}_m \mathbf{c}_m^T$, $\forall\, m \in [d]$ and tr(.) represents the trace.

(c) The regularizer, $\|\varphi\|_{\mathcal{C}}^2$ in Eq. (6) is given by $\|\varphi\|_{\mathcal{C}}^2 = \sum_{m=1}^{d} \|\varphi_m\|_{\mathcal{H}_m}^2 = \sum_{m=1}^{d} \sum_{i,j=1}^{n} c_{im} c_{jm} \mathfrak{K}_m(X_i, X_j) = \sum_{m=1}^{d} \mathbf{c}_m^T \mathbf{K}_m \mathbf{c}_m = \sum_{m=1}^{d} tr(\mathbf{K}_m \mathbf{\Sigma}_m)$ where $\mathbf{K}_m := (\mathbf{k}_m^{X_1}, \ldots, \mathbf{k}_m^{X_n})$.

(d) Since φ appears in the form of ρ_φ and $\|\varphi\|_{\mathcal{C}}^2$ in Eq. (6), learning φ is equivalent to learning $\{\mathbf{\Sigma}_m \succeq 0 : rank(\mathbf{\Sigma}_m) = 1, \mathbf{1}^T \mathbf{\Sigma}_m \mathbf{1} = 0\}_{m=1}^{d}$.

In the above remark, we have shown that θ_h and $\|\varphi\|_{\mathcal{C}}$ in Eq. (6) depend only on the entries in d kernel matrices (associated with d kernel functions) and $n \cdot d$ scalars, $\{c_{ij}\}$. In addition, we also reduced the representation of φ from $\{\mathbf{c}_m\}_{m=1}^{d}$ to $\{\mathbf{\Sigma}_m\}_{m=1}^{d}$. It can be seen that ρ_φ^2 and $\|\varphi\|_{\mathcal{C}}^2$ are convex quadratic functions of $\{\mathbf{c}_m\}_{m=1}^{d}$, while they are linear functions of $\{\mathbf{\Sigma}_m\}_{m=1}^{d}$. Depending on the nature of K, one representation would be more useful than the other.

Corollary 3. *Suppose $\mathfrak{K}_1 = \ldots = \mathfrak{K}_d = \mathfrak{K}$. Then, for any $z, w \in \mathcal{X}$, $\rho_\varphi^2(z, w)$ is the Mahalanobis distance between $\mathbf{k}^z$ and $\mathbf{k}^w$, with $\sum_{m=1}^{d} \mathbf{\Sigma}_m$ as its metric.*

Proof. By Remark 2, we have $\rho_\varphi^2(z, w) = \|\varphi(z) - \varphi(w)\|_2^2 = \sum_{m=1}^{d} (\mathbf{k}_m^z - \mathbf{k}_m^w)^T \mathbf{\Sigma}_m (\mathbf{k}_m^z - \mathbf{k}_m^w)$. Since $\mathfrak{K}_1 = \ldots = \mathfrak{K}_d = \mathfrak{K}$, we have $\mathbf{k}_1^z = \ldots = \mathbf{k}_d^z = \mathbf{k}^z$. Therefore, $\rho_\varphi^2(z, w) = (\mathbf{k}^z - \mathbf{k}^w)^T \mathbf{\Sigma} (\mathbf{k}^z - \mathbf{k}^w)$ where $\mathbf{\Sigma} := \sum_{m=1}^{d} \mathbf{\Sigma}_m$. □

The above result reduces the problem of learning φ to learning a matrix, $\mathbf{\Sigma} \succeq 0$, such that $\text{rank}(\mathbf{\Sigma}) \leq d$ and $\mathbf{1}^T \mathbf{\Sigma} \mathbf{1} = 0$. We now study the above result for linear kernels. The following corollary shows that applying a linear kernel is equivalent to assuming the underlying distance metric in $\mathcal{X}$ to be the Mahalanobis distance.

Corollary 4 (Linear kernel)**.** *Let $\mathcal{X} = \mathbb{R}^D$ and $z, w \in \mathcal{X}$. If $\mathfrak{K}(z, w) = \langle z, w \rangle_2 = z^T w$, then $\varphi(z) = \mathbf{L} z \in \mathbb{R}^d$ and $\|\varphi(z) - \varphi(w)\|_2^2 = (z - w)^T \mathbf{A} (z - w)$ with $\mathbf{A} := \mathbf{L}^T \mathbf{L}$.*

Proof. By Remark 2 and Corollary 3, we have $\varphi_m(z) = \sum_{i=1}^{n} c_{im} \mathfrak{K}(z, X_i) = \left(\sum_{i=1}^{n} c_{im} X_i\right)^T z =: \boldsymbol{\ell}_m^T z$. Therefore, $\varphi(z) = \mathbf{L} z$, where $\mathbf{L} := (\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_d)^T$. In addition, $\|\varphi(z) - \varphi(w)\|_2^2 = (z - w)^T \mathbf{A} (z - w)$ with $\mathbf{A} := \mathbf{L}^T \mathbf{L}$. □

In the following section, we use these results to derive an algorithm that jointly learns φ and h by solving Eq. (5).

4. Convex Relaxations & Semidefinite Program

Having addressed the theoretical issue of making assumptions about $\mathcal{C}$ to solve Eq. (5), we return to address the computational issue pointed out in §2. The program in Eq. (5) is NP-hard because of the nature of $\{\psi_i\}_{i=1}^{n}$. This issue can be alleviated by minimizing a convex upper bound of ψ_i, instead of ψ_i. Some of the convex upper bounds for the function $\psi(x) = \mathbb{1}_{\{x > 0\}}$ are $\Psi(x) = \max(0, 1 + x) := [1 + x]_+$, $\Psi(x) = \log(1 + e^x)$ etc. Replacing ψ_i by Ψ_i in Eq. (5) results in the following program,

$$\min_{\substack{\varphi \in \mathcal{C} \\ h > 0}} \sum_{i=1}^{n} \Psi_i \left(\gamma_i^-(\varphi, h) - \gamma_i^+(\varphi, h)\right) + \lambda \|\varphi\|_{\mathcal{C}}^2, \qquad (8)$$

where $\gamma_i^+(\varphi, h) := \sum_{\substack{j \neq i \\ \tau_{ij} = 1}} K_{X_j}(X_i)$ and $\gamma_i^-(\varphi, h) := \sum_{\{j : \tau_{ij} = -1\}} K_{X_j}(X_i)$. Eq. (8) can still be computationally hard to solve depending on the choice of the smoothing kernel, K. Even if we choose K such that γ^+ and γ^- are jointly convex in φ and h for some representation of φ (see Remark 2), Eq. (8) is still non-convex as the argument of Ψ_i is a difference of two convex functions.[9] In addition, if $\Psi(x) = [1 + x]_+$, then Eq. (8) is a d.c. (difference of convex functions) program (Horst & Thoai, 1999), which is NP-hard to solve. So, even for the nicest of cases, one has to resort to local optimization methods or computationally intensive global optimization methods. Nevertheless, if one does not worry about this disadvantage, then solving Eq. (8) yields φ (in terms of $\{\mathbf{c}_m\}_{m=1}^{d}$ or $\{\mathbf{\Sigma}_m\}_{m=1}^{d}$, depending on the chosen representation) and h that can be used in Eq. (2) to classify unseen data. However, in the following, we show that Eq. (8) can be turned into a convex program for the naïve kernel. As mentioned in §1, this choice of kernel leads to a classification rule that is similar in principle to the k-NN rule.

4.1. Naïve kernel: Semidefinite relaxation

The naïve kernel, $K_{x_0}(x) = \mathbb{1}_{\{\rho_\varphi(x, x_0) \leq h\}}$, indicates that the points, $\varphi(x)$, that lie within a ball of radius h centered at $\varphi(x_0)$ have a weighting factor of 1, while the remaining points have zero weight. Using this in Eq. (8), we have $\gamma_i^-(\varphi, h) - \gamma_i^+(\varphi, h) = \sum_{\{j : \tau_{ij} = -1\}} \mathbb{1}_{\{\rho_\varphi(X_i, X_j) \leq h\}} - \sum_{\{j : \tau_{ij} = 1\}} \mathbb{1}_{\{\rho_\varphi(X_i, X_j) \leq h\}} + 1$, which represents the difference between number of points with label different from Y_i that lie within the ball of radius of h centered at $\varphi(X_i)$ and the number of points with the same label as X_i (excluding X_i) that lie within the same ball. If this difference is

[9] For example, let K be a Gaussian kernel, $K_y(x) = \exp(-\rho_\varphi^2(x, y)/h)$. Using the $\{\mathbf{\Sigma}_m\}_{m=1}^{d}$ representation for φ, we have $\rho_\varphi^2(x, y)$ is linear in $\{\mathbf{\Sigma}_m\}_{m=1}^{d}$ and therefore, $K_y(x)$ is convex in $\{\mathbf{\Sigma}_m\}_{m=1}^{d}$. Here, we assume h to be fixed. This means γ_i^+ and γ_i^- are convex in φ.

positive, then the classification rule in Eq. (2) makes an error in classifying X_i. Therefore, φ and h should be chosen such that this misclassification rate is minimized. Suppose that $\{\varphi(X_i)\}_{i=1}^n$ is given. Then, h determines the misclassification rate like k in k-NN. It can be seen that the kernel classification rule and k-NN rule are similar when K is a naïve kernel. In the case of k-NN, the number of nearest neighbors are fixed for any point, whereas with the kernel rule, it varies for every point. On the other hand, the radius of the ball containing the nearest neighbors of a point varies with every point in the k-NN setting while it is the same for every point in the kernel rule.

$\gamma_i^-(\varphi,h) - \gamma_i^+(\varphi,h)$ can be further reduced to a more amenable form by the following algebra. Using $\sum_{\{j:\tau_{ij}=1\}} \mathbb{1}_{\{\rho_\varphi(X_i,X_j)\le h\}} = \sum_{j=1}^n \mathbb{1}_{\{\tau_{ij}=1\}} - \sum_{\{j:\tau_{ij}=1\}} \mathbb{1}_{\{\rho_\varphi(X_i,X_j)>h\}}$, we get $\gamma_i^-(\varphi,h) - \gamma_i^+(\varphi,h) = 1 - n_i^+ + \sum_{j=1}^n \mathbb{1}_{\{\tau_{ij}\rho_\varphi^2(X_i,X_j)>\tau_{ij}\tilde{h}\}}$ where $n_i^+ := \sum_{j=1}^n \mathbb{1}_{\{\tau_{ij}=1\}}$ and $\tilde{h} := h^2$. Note that we have neglected the set $\{j : \tau_{ij} = -1; \rho_\varphi(X_i,X_j) = h\}$ in the above calculation for simplicity. Using $\Psi(x) = [1+x]_+$, the first half of the objective function in Eq. (8) reduces to $\sum_{i=1}^n \Big[2 - n_i^+ + \sum_{j=1}^n \mathbb{1}_{\{\tau_{ij}\rho_\varphi^2(X_i,X_j)>\tau_{ij}\tilde{h}\}}\Big]_+$. Applying the convex relaxation one more time to the step function, we get $\sum_{i=1}^n \Big[2 - n_i^+ + \sum_{j=1}^n \Big[1 + \tau_{ij}\rho_\varphi^2(X_i,X_j) - \tau_{ij}\tilde{h}\Big]_+\Big]_+$ as an upper bound on the first half of the objective function in Eq. (8). Since ρ_φ^2 is a quadratic function of $\{\mathbf{c}_m\}_{m=1}^d$, it can be shown that representing φ in terms of $\{\mathbf{c}_m\}_{m=1}^d$ results in a d.c. program, whereas its representation in terms of $\{\mathbf{\Sigma}_m\}_{m=1}^d$ results in a semidefinite program (SDP) (except for the rank constraints), since ρ_φ^2 is linear in $\{\mathbf{\Sigma}_m\}_{m=1}^d$. Assuming for simplicity that $\mathfrak{K}_1 = \ldots = \mathfrak{K}_d = \mathfrak{K}$ and neglecting the constraint $\text{rank}(\mathbf{\Sigma}) \le d$, we obtain the following SDP,

$$\begin{aligned} \min_{\mathbf{\Sigma},\tilde{h}} \quad & \sum_{i=1}^n \Big[2 - n_i^+ + \sum_{j=1}^n \Big[1 + \tau_{ij}\text{tr}(\mathbf{M}_{ij}\mathbf{\Sigma}) - \tau_{ij}\tilde{h}\Big]_+\Big]_+ \\ & + \lambda\,\text{tr}(\mathbf{K}\mathbf{\Sigma}) \\ \text{s.t.} \quad & \mathbf{\Sigma} \succeq 0,\ \mathbf{1}^T\mathbf{\Sigma}\mathbf{1} = 0,\ \tilde{h} > 0, \end{aligned} \tag{9}$$

where $\mathbf{M}_{ij} := (\mathbf{k}^{X_i} - \mathbf{k}^{X_j})(\mathbf{k}^{X_i} - \mathbf{k}^{X_j})^T$. For notational details, refer to Remark 2 and Corollary 3. Since one does not usually know the optimal embedding dimension, d, the $\mathbf{\Sigma}$ representation is advantageous as it is independent of d (as we neglected the rank constraint) and depends only on n. On the other hand, it is a disadvantage as the algorithm does not scale well to large datasets.

Although the program in Eq. (9) is convex, solving it by general purpose solvers that use interior point methods scales as $O(n^6)$, which is prohibitive. Instead, following the ideas of Weinberger et al. (2006), we used a first order gradient method (which scales as $O(n^2)$ per iteration) and an alternating projections method (which scales as $O(n^3)$ per iteration). At each iteration, we take a small step in the direction of the negative gradient of the objective function, followed by a projection onto the set $\mathcal{N} = \{\mathbf{\Sigma} : \mathbf{\Sigma} \succeq 0,\ \mathbf{1}^T\mathbf{\Sigma}\mathbf{1} = 0\}$ and $\{\tilde{h} > 0\}$. The projection onto $\mathcal{N}$ is performed by an alternating projections method which involves projecting a symmetric matrix alternately between the convex sets, $\mathcal{A} = \{\mathbf{\Sigma} : \mathbf{\Sigma} \succeq 0\}$ and $\mathcal{B} = \{\mathbf{\Sigma} : \mathbf{1}^T\mathbf{\Sigma}\mathbf{1} = 0\}$. Since $\mathcal{A} \cap \mathcal{B} \neq \emptyset$, this alternating projections method is guaranteed to find a point in $\mathcal{A} \cap \mathcal{B}$. Given any $\mathbf{A}_0 \in \mathcal{A}$, the alternating projections algorithm computes $\mathbf{B}_m = P_{\mathcal{B}}(\mathbf{A}_m) \in \mathcal{B}$, $\mathbf{A}_{m+1} = P_{\mathcal{A}}(\mathbf{B}_m) \in \mathcal{A}$, $m = 0, 1, 2, \ldots$, where $P_{\mathcal{A}}$ and $P_{\mathcal{B}}$ are the projection on $\mathcal{A}$ and $\mathcal{B}$, respectively. In summary, the update rule can be given as $\mathbf{B}_m = \mathbf{A}_m - \frac{\mathbf{1}^T\mathbf{A}_m\mathbf{1}}{n^2}\mathbf{1}\mathbf{1}^T$ and $\mathbf{A}_{m+1} = \sum_{i=1}^n [\lambda_i]_+ \mathbf{u}_i\mathbf{u}_i^T$ where $\{\mathbf{u}_i\}_{i=1}^n$ and $\{\lambda_i\}_{i=1}^n$ are the eigenvectors and eigenvalues of $\mathbf{B}_m$.[10] A pseudocode of the gradient projection algorithm to solve Eq. (9) is shown in Algorithm 1.

Algorithm 1 Gradient Projection Algorithm

Require: $\{\mathbf{M}_{ij}\}_{i,j=1}^n$, $\mathbf{K}$, $\{\tau_{ij}\}_{i,j=1}^n$, $\{n_i^+\}_{i=1}^n$, $\lambda > 0$, $\epsilon > 0$ and $\{\alpha_i,\ \beta_i\} > 0$ (see Eq. (9))

Set $t = 0$. Choose $\mathbf{\Sigma}_0 \in \mathcal{A}$ and $\tilde{h}_0 > 0$.
repeat
  $A_t = \{i : \sum_{j=1}^n \Big[1 + \tau_{ij}\text{tr}(\mathbf{M}_{ij}\mathbf{\Sigma}_t) - \tau_{ij}\tilde{h}_t\Big]_+ + 2 \le n_i^+\} \times \{j : j \in [n]\}$
  $B_t = \{(i,j) : 1 + \tau_{ij}\text{tr}(\mathbf{M}_{ij}\mathbf{\Sigma}_t) > \tau_{ij}\tilde{h}_t\}$
  $N_t = B_t \backslash A_t$
  $\mathbf{\Sigma}_{t+1} = P_{\mathcal{N}}(\mathbf{\Sigma}_t - \alpha_t \sum_{(i,j)\in N_t} \tau_{ij}\mathbf{M}_{ij} - \alpha_t\lambda\mathbf{K})$
  $\tilde{h}_{t+1} = \max(\epsilon, \tilde{h}_t + \beta_t \sum_{(i,j)\in N_t} \tau_{ij})$
  $t = t + 1$
until convergence
return $\mathbf{\Sigma}_t$, $\tilde{h}_t$

Having computed $\mathbf{\Sigma}$ and $\tilde{h}$ that minimize Eq. (9), a test point, $x \in \mathcal{X}$, can be classified by using the kernel rule in Eq. (2), where $K_{X_i}(x) = \mathbb{1}_{\{\rho_\varphi(x,X_i)\le h\}}$ with $\rho_\varphi^2(x,X_i) = (\mathbf{k}^x - \mathbf{k}^{X_i})^T\mathbf{\Sigma}(\mathbf{k}^x - \mathbf{k}^{X_i})$. Therefore, $\mathbf{\Sigma}$ and h completely specify the classification rule.

5. Experiments & Results

In this section, we compare the performance of our method (referred to as kernel classification rule (KCR)) to several metric learning algorithms on a supervised classification task in terms of the training and test errors. The training phase of KCR involves solving the SDP in Eq. (9) to learn optimal $\mathbf{\Sigma}$ and h from the data, which are then used in Eq. (2) to classify the test data. Note that the SDP in Eq. (9)

[10] Given $\mathbf{A}_m \in \mathcal{A}$, $\mathbf{B}_m$ is obtained by solving $\min\{\|\mathbf{B}_m - \mathbf{A}_m\|_F^2 : \mathbf{1}^T\mathbf{B}_m\mathbf{1} = 0\}$. Similarly, for a given $\mathbf{B}_m \in \mathcal{B}$, $\mathbf{A}_{m+1}$ is obtained by solving $\min\{\|\mathbf{A}_{m+1} - \mathbf{B}_m\|_F^2 : \mathbf{A}_{m+1} \succeq 0\}$.

Table 1. k-NN classification accuracy on UCI datasets. The algorithms compared are k-NN (with Euclidean distance metric), LMNN (large margin NN by Weinberger et al. (2006)), *Kernel*-NN (see footnote 11), KMLCC (kernel version of metric learning by collapsing classes by Globerson and Roweis (2006)), KLMCA (kernel version of LMNN by Torresani and Lee (2007)), and KCR (proposed method). Mean (μ) and standard deviation (σ) of the train and test (generalization) errors (in $\%$) are reported.

Dataset (n, D, l)	Algorithm/ Error	k-NN $\mu \pm \sigma$	LMNN $\mu \pm \sigma$	*Kernel*-NN $\mu \pm \sigma$	KMLCC $\mu \pm \sigma$	KLMCA $\mu \pm \sigma$	KCR $\mu \pm \sigma$
Balance	Train	17.81 ± 1.86	11.40 ± 2.89	10.73 ± 1.32	10.27 ± 2.01	9.93 ± 1.86	10.47 ± 2.11
$(625, 4, 3)$	Test	18.18 ± 1.88	11.49 ± 2.57	17.46 ± 2.13	9.75 ± 1.92	10.54 ± 1.46	$\mathbf{8.94 \pm 3.12}$
Ionosphere	Train	15.89 ± 1.43	3.50 ± 1.18	2.84 ± 0.80	7.05 ± 1.31	3.98 ± 1.94	2.73 ± 1.03
$(351, 34, 2)$	Test	15.95 ± 3.03	12.14 ± 2.92	5.81 ± 2.25	6.54 ± 2.18	$\mathbf{5.19 \pm 2.09}$	5.71 ± 2.60
Iris	Train	4.30 ± 1.55	3.25 ± 1.15	3.60 ± 1.33	3.61 ± 1.59	3.27 ± 1.63	2.29 ± 1.62
$(150, 4, 3)$	Test	4.02 ± 2.22	4.11 ± 2.26	4.83 ± 2.47	3.89 ± 1.55	3.74 ± 2.21	$\mathbf{3.27 \pm 1.87}$
Wine	Train	5.89 ± 1.35	0.90 ± 2.80	4.95 ± 1.35	4.48 ± 1.21	2.18 ± 2.58	1.01 ± 0.73
$(178, 13, 3)$	Test	6.22 ± 2.70	3.41 ± 2.10	7.37 ± 2.82	4.84 ± 2.47	5.17 ± 1.91	$\mathbf{2.13 \pm 1.24}$

is obtained by using the naïve kernel for K in Eq. (2). For other smoothing kernels, one has to solve the program in Eq. (8) to learn optimal $\mathbf{\Sigma}$ and h. Therefore, the results reported in this section under KCR refer to those obtained by using the naïve kernel.

The algorithms used in the comparative evaluation are:

- The k-NN rule with the Euclidean distance metric.
- The LMNN (large margin nearest neighbor) method proposed by Weinberger et al. (2006), which learns a Mahalanobis distance metric by minimizing the distance between predefined target neighbors and separating them by a large margin from the examples with non-matching labels.
- The *Kernel*-NN rule, which uses the empirical kernel maps[11] as training data and performs k-NN classification on this data using the Euclidean distance metric.
- The KMLCC (kernel version of metric learning by collapsing classes) method proposed by Globerson and Roweis (2006), which learns a Mahalanobis distance metric in the kernel space by trying to collapse all examples in the same class to a single point while pushing examples in other classes infinitely far away.
- The KLMCA (kernel version of large margin component analysis) method proposed by Torresani and Lee (2007), which is a non-convex, kernelized version of LMNN.

Four benchmark datasets from the UCI machine learning repository were considered for experimentation. Since the proposed method and KMLCC solve an SDP that scales poorly with n, we did not consider large problem sizes for experimentation.[12] The results shown in Table 1 are the average performance over 20 random splits of the data with 50% for training, 20% for validation and 30% for testing. The Gaussian kernel, $\mathfrak{K}(x, y) = e^{-\upsilon \|x-y\|_2^2}$ was used for the kernel based methods, i.e., *Kernel*-NN, KMLCC, KLMCA and KCR. The parameters υ and λ (only υ for *Kernel*-NN) were set with cross-validation by searching over $\upsilon \in \{2^i\}_{-4}^{4}$ and $\lambda \in \{10^i\}_{-3}^{3}$. While testing, KCR uses the rule in Eq. (2), whereas the k-NN rule was used for all the other methods.[13] It is clear from Table 1 that KCR almost always performs as well as or significantly better than all other methods. However, on the timing front (which we do not report here), KLMCA, which solves a non-convex program for $n \cdot d$ variables, is much faster than KMLCC and KCR, which solve SDPs involving n^2 variables. The role of empirical kernel maps is not clear as there is no consistent behavior between the performance accuracy achieved with k-NN and *Kernel*-NN.

KMLCC, KLMCA, and KCR learn the Mahalanobis distance metric in $\mathbb{R}^n$ which makes it difficult to visualize the class separability achieved by these methods. To visually appreciate their behavior, we generated a synthetic two dimensional dataset of 3 classes with each class being sampled from a Gaussian distribution with different mean and covariance. Figure 1(a) shows this dataset where the three classes are shown in different colors. Using this as training data, distance metrics were learned using KMLCC, KLMCA and KCR. If $\mathbf{\Sigma}$ is the learned metric, then the two dimensional projection of $x \in \mathbb{R}^n$ is obtained as $\widehat{x} = \mathbf{L}x$ where $\mathbf{L} = (\sqrt{\lambda_1}\mathbf{u}_1, \sqrt{\lambda_2}\mathbf{u}_2)^T$, with $\mathbf{\Sigma} = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^T$, and $\lambda_1 \geq \lambda_2 > \cdots \lambda_n$. Figure 1(b-d) show the two di-

[11] *Kernel*-NN is computed as follows. For each training point, X_i, the empirical map w.r.t. $\{X_j\}_{j=1}^{n}$ defined as $\mathbf{k}^{X_i} := (\mathfrak{K}(X_1, X_i), \ldots, \mathfrak{K}(X_n, X_i))^T$ is computed. Then, $\{\mathbf{k}^{X_i}\}_{i=1}^{n}$ is considered to be the training set for the NN classification of empirical maps of the test data using the Euclidean distance metric.

[12] To extend KCR to large datasets, one can represent φ in terms of $\{\mathbf{c}_m\}$, which leads to a non-convex program as in KLMCA.

[13] Although KCR, LMNN, and KMLCC solve SDPs to compute the optimal distance metric, KCR has fewer number of parameters to be tuned compared to these other methods. LMNN requires cross-validation over k (in k-NN) and the regularization parameter along with the knowledge about target neighbors. KMLCC requires cross-validation over k, the kernel parameter, υ and the regularization parameter. In KCR, we only need to cross-validate over υ and λ. In addition, if $\mathcal{X} = \mathbb{R}^D$ and $\mathfrak{K}$ is a linear kernel, then KCR only requires cross-validation over λ while computing the optimal Mahalanobis distance metric.

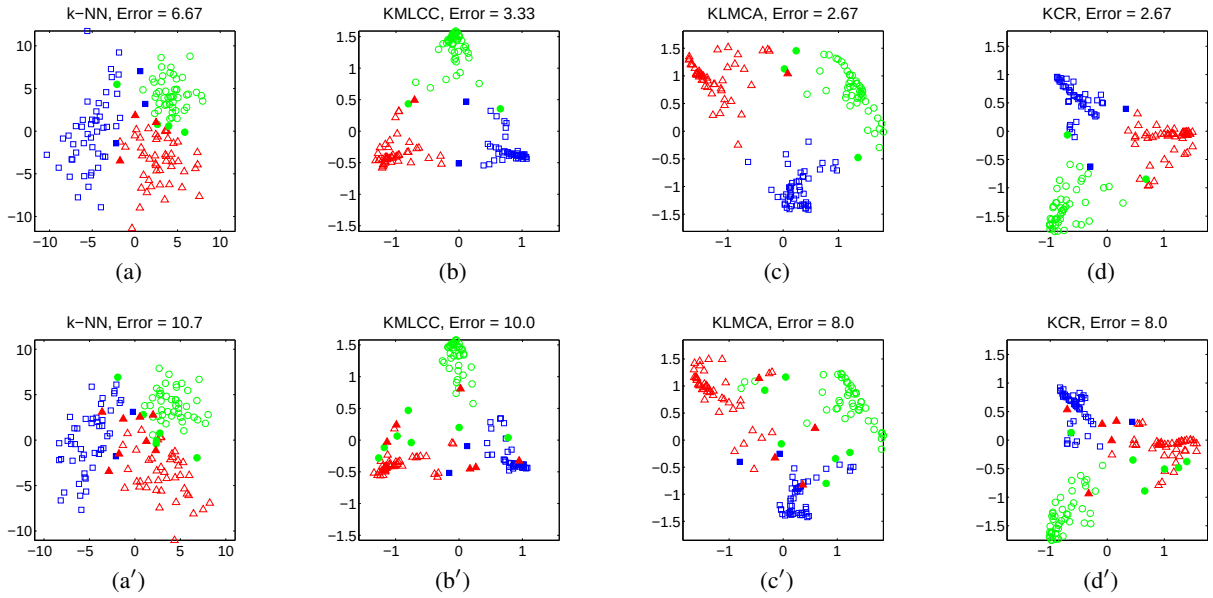


Figure 1. Dataset visualization results of k-NN, KMLCC, KLMCA and KCR applied to a two-dimensional synthetic dataset of three classes with each class being modeled as a Gaussian. (a,a′) denote two independent random draws from this distribution whereas (b-d, b′-d′) represent the two-dimensional projections of these data using the metric learned from KMLCC, KLMCA and KCR. The points in bold represent the misclassified points. It is interesting to note that KLMCA and KCR generate completely different embeddings but have similar error rates. See §5 for more details.

mensional projections of the training set using KMLCC, KLMCA and KCR. The projected points were classified using k-NN if $\mathbf{\Sigma}$ was obtained from KMLCC/KLMCA and using Eq. (2) if $\mathbf{\Sigma}$ was obtained from KCR. The misclassified points are shown in bold. Since the classification is done on the training points, one would expect better error rate and separability between the classes. To understand the generalization performance, a new data sample shown in Figure 1(a′) was generated from the same distribution as the training set. The learned $\mathbf{\Sigma}$ was used to obtain the two dimensional projections of the new data sample which are shown in Figure 1(b′-d′). It is interesting to note that KLMCA and KCR generate completely different projections but have similar error rates.

6. Related Work

We briefly review some relevant work and point out similarities and differences with our method. In our work, we have addressed the problem of extending kernel classification rules to arbitrary metric spaces by learning an embedding function that embeds data into Euclidean space while minimizing an upper bound on the resubstitution estimate of the error probability. The method that is closest in spirit (kernel rules) to ours is the recent work by Weinberger and Tesauro (2007) who learn a Mahalanobis distance metric for kernel regression estimates by minimizing the leave-one-out quadratic regression error of the training set. With the problem being non-convex, they resort to gradient descent techniques. Except for this work, we are not aware of any method related to kernel rules in the context of distance metric learning or learning the bandwidth of the kernel.

There has been lot of work in the area of distance metric learning for k-NN classification, some of which are briefly discussed in §5. The central idea in all these methods is that similarly labeled examples should cluster together and be far away from differently labeled examples. Shalev-Shwartz et al. (2004) proposed an online algorithm for learning a Mahalanobis distance metric with the constraint that any training example is closer to all the examples that share its label than to any other example of different label. In addition, examples from different classes are constrained to be separated by a large margin. Though Shalev-Shwartz et al. (2004) do not solve this as a batch optimization problem, it can be shown that it reduces to an SDP (after rank relaxation) and is in fact the same as Eq. (9) except for the outer $[.]_+$ function and the constraint $\mathbf{1}^T\mathbf{\Sigma}\mathbf{1} = 0$.

7. Concluding Remarks

In this paper, two questions related to the smoothing kernel based classification rule have been addressed. One is related to learning the bandwidth of the smoothing kernel, while the other is to extending the classification rule to arbitrary domains. We jointly addressed them by learning a function in a reproducing kernel Hilbert space while minimizing an upper bound on the resubstitution estimate of the error probability of the kernel rule. For a particular choice

of the smoothing kernel, called the naïve kernel, we showed that the resulting rule is related to the k-NN rule. Because of this relation, the kernel rule was compared to k-NN and its state-of-the-art distance metric learning algorithms on a supervised classification task and was shown to have comparable performance to these methods. In the future, we would like to develop some theoretical guarantees for the proposed method along with extending it to large-scale applications.

Appendix A. Proof of Theorem 1

We need the following result to prove Theorem 1.

Lemma 5. *Let $\mathcal{H} = \{f : \mathcal{X} \to \mathbb{R}\}$ be an RKHS with $\mathfrak{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as its reproducing kernel. Let $\theta : \mathbb{R}^{n^2} \to \mathbb{R}$ be an arbitrary function. Then each minimizer $f \in \mathcal{H}$ of*

$$\theta\left(\{f(x_i) - f(x_j)\}_{i,j=1}^n\right) + \lambda\|f\|_{\mathcal{H}}^2 \tag{10}$$

admits a representation of the form $f = \sum_{i=1}^n c_i \mathfrak{K}(., x_i)$, where $\{c_i\}_{i=1}^n \in \mathbb{R}$ and $\sum_{i=1}^n c_i = 0$.

Proof. The proof follows the generalized representer theorem (Schölkopf et al., 2001, Theorem 4). Since $f \in \mathcal{H}$, $f(x) = \langle f, \mathfrak{K}(., x)\rangle_{\mathcal{H}}$. Therefore, the arguments of θ in Eq. (10) are of the form $\{\langle f, \mathfrak{K}(., x_i) - \mathfrak{K}(., x_j)\rangle_{\mathcal{H}}\}_{i,j=1}^n$. We decompose $f = f_{\|} + f_{\perp}$ so that $f_{\|} \in \text{span}\left(\{\mathfrak{K}(., x_i) - \mathfrak{K}(., x_j)\}_{i,j=1}^n\right)$ and $\langle f_{\perp}, \mathfrak{K}(., x_i) - \mathfrak{K}(., x_j)\rangle_{\mathcal{H}} = 0, \forall\, i, j \in [n]$. So, $f = \sum_{i,j=1}^n \alpha_{ij}(\mathfrak{K}(., x_i) - \mathfrak{K}(., x_j)) + f_{\perp}$ where $\{\alpha_{ij}\}_{i,j=1}^n \in \mathbb{R}$. Therefore, $f(x_i) - f(x_j) = \langle f, \mathfrak{K}(., x_i) - \mathfrak{K}(., x_j)\rangle_{\mathcal{H}} = \langle f_{\|}, \mathfrak{K}(., x_i) - \mathfrak{K}(., x_j)\rangle_{\mathcal{H}} = \sum_{p,m=1}^n \alpha_{pm}(\mathfrak{K}(x_i, x_p) - \mathfrak{K}(x_j, x_p) - \mathfrak{K}(x_i, x_m) + \mathfrak{K}(x_j, x_m))$. Now, consider the penalty functional, $\langle f, f\rangle_{\mathcal{H}}$. For all $f_{\perp}$, $\langle f, f\rangle_{\mathcal{H}} = ||f_{\|}||_{\mathcal{H}}^2 + ||f_{\perp}||_{\mathcal{H}}^2 \geq \|\sum_{i,j=1}^n \alpha_{ij}(\mathfrak{K}(., x_i) - \mathfrak{K}(., x_j))\|_{\mathcal{H}}^2$. Thus for any fixed $\alpha_{ij} \in \mathbb{R}$, Eq. (10) is minimized for $f_{\perp} = 0$. Therefore, the minimizer of Eq. (10) has the form $f = \sum_{i,j=1}^n \alpha_{ij}(\mathfrak{K}(., x_i) - \mathfrak{K}(., x_j))$, which is parameterized by n^2 parameters of $\{\alpha_{ij}\}_{i,j=1}^n$. By simple algebra, f reduces to $f = \sum_{i=1}^n c_i \mathfrak{K}(., x_i)$, where $c_i = \sum_{j=1}^n (\alpha_{ij} - \alpha_{ji})$ satisfies $\sum_{i=1}^n c_i = 0$. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. The arguments of θ_h in Eq. (6) are of the form $\|\varphi(X_i) - \varphi(X_j)\|_2$. Consider $\|\varphi(X_i) - \varphi(X_j)\|_2^2 = \sum_{m=1}^d (\varphi_m(X_i) - \varphi_m(X_j))^2 = \sum_{m=1}^d (\langle \varphi_m, \mathfrak{K}_m(., X_i) - \mathfrak{K}_m(., X_j)\rangle_{\mathcal{H}_m})^2$. The penalizer in Eq. (6) reduces to $\|\varphi\|_{\mathcal{C}}^2 = \sum_{m=1}^d \|\varphi_m\|_{\mathcal{H}_m}^2$. Therefore, applying Lemma 5 to each φ_m, $m \in [d]$ proves the result. □

Acknowledgments

We thank the reviewers for their comments which greatly improved the paper. We wish to acknowledge support from the National Science Foundation (grant DMS-MSPA 0625409), the Fair Isaac Corporation and the University of California MICRO program.

References

Devroye, L., Gyorfi, L., & Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. New York: Springer-Verlag.

Devroye, L., & Krzyżak, A. (1989). An equivalence theorem for L_1 convergence of the kernel regression estimate. *Journal of Statistical Planning and Inference*, *23*, 71–82.

Evgeniou, T., Pontil, M., & Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, *13*, 1–50.

Globerson, A., & Roweis, S. (2006). Metric learning by collapsing classes. *Advances in Neural Information Processing Systems 18* (pp. 451–458). Cambridge, MA: MIT Press.

Goldberger, J., Roweis, S., Hinton, G., & Salakhutdinov, R. (2005). Neighbourhood components analysis. *Advances in Neural Information Processing Systems 17* (pp. 513–520). Cambridge, MA: MIT Press.

Horst, R., & Thoai, N. V. (1999). D.c. programming: Overview. *Journal of Optimization Theory and Applications*, *103*, 1–43.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. *Proc. of the Annual Conference on Computational Learning Theory* (pp. 416–426).

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Shalev-Shwartz, S., Singer, Y., & Ng, A. Y. (2004). Online and batch learning of pseudo-metrics. *Proc. of 21st International Conference on Machine Learning*. Banff, Canada.

Snapp, R. R., & Venkatesh, S. S. (1998). Asymptotic expansions of the k nearest neighbor risk. *The Annals of Statistics*, *26*, 850–878.

Torresani, L., & Lee, K. (2007). Large margin component analysis. *Advances in Neural Information Processing Systems 19* (pp. 1385–1392). Cambridge, MA: MIT Press.

von Luxburg, U., & Bousquet, O. (2004). Distance-based classification with Lipschitz functions. *Journal for Machine Learning Research*, *5*, 669–695.

Weinberger, K. Q., Blitzer, J., & Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. *Advances in Neural Information Processing Systems 18* (pp. 1473–1480). Cambridge, MA: MIT Press.

Weinberger, K. Q., & Tesauro, G. (2007). Metric learning for kernel regression. *Proc. of the 11th International Conference on Artificial Intelligence and Statistics*.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning with application to clustering with side-information. *Advances in Neural Information Processing Systems 15* (pp. 505–512). Cambridge, MA: MIT Press.

Discriminative Parameter Learning for Bayesian Networks

Jiang Su JSU@SITE.UOTTAWA.CA
School of Information Technology and Engineering University of Ottawa, K1N 6N5 Canada

Harry Zhang HZHANG@UNB.CA
Faculty of Computer Science, University of New Brunswick, Fredericton, NB, E3B 5A3, Canada

Charles X. Ling CLING@CSD.UWO.CA
Department of Computer Science, The University of Western Ontario, London, Ontario, N6A 5B7, Canada

Stan Matwin STAN@SITE.UOTTAWA.CA
School of Information Technology and Engineering University of Ottawa, K1N 6N5 Canada

Abstract

Bayesian network classifiers have been widely used for classification problems. Given a fixed Bayesian network structure, parameters learning can take two different approaches: generative and discriminative learning. While generative parameter learning is more efficient, discriminative parameter learning is more effective. In this paper, we propose a simple, efficient, and effective discriminative parameter learning method, called *Discriminative Frequency Estimate* (DFE), which learns parameters by discriminatively computing frequencies from data. Empirical studies show that the DFE algorithm integrates the advantages of both generative and discriminative learning: it performs as well as the state-of-the-art discriminative parameter learning method ELR in accuracy, but is significantly more efficient.

1. Introduction

A Bayesian network (BN) (Pearl, 1988) consists of a directed acyclic graph G and a set P of probability distributions, where nodes and arcs in G represent random variables and direct correlations between variables respectively, and P is the set of local distributions for each node. A local distribution is typically specified by a conditional probability table (CPT). Thus, learning Bayesian networks from data has two elements: structure learning and parameter learning.

Bayesian networks are often used for classification problems, in which a learner attempts to construct a classifier from a given set of training instances with class labels. In learning Bayesian network classifiers, parameter learning often uses *Frequency Estimate* (FE), which determines parameters by computing the appropriate frequencies from data. The major advantage of FE is its efficiency: it only needs to count each data point (training instance) once. It is well-known that FE maximizes likelihood and thus is a typical generative learning method.

In classification, however, the objective is to maximize generalization accuracy, rather than likelihood. Thus, discriminative parameter learning that maximizes generalization accuracy or its alternative objective function, conditional likelihood, are more desirable. Unfortunately, there is no closed form for choosing the optimal parameters, because conditional likelihood does not decompose (Friedman et al., 1997). As a consequence, discriminative parameter learning for Bayesian networks often resorts to search methods, such as gradient descent.

Greiner and Zhou (2002) proposed a gradient descent based parameter learning method, called ELR, to discriminatively learn parameters for Bayesian network classifiers, and showed that ELR significantly outperforms the generative learning method FE. However, the application of ELR is limited due to its high computational cost. For example, Grossman and Domingos (2004) observed that ELR is computationally infeasible in structure learning. In fact, how to find an

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

efficient and effective discriminative parameter learning for Bayesian network classifiers is an open question.

In this paper, we propose a simple, efficient, and effective discriminative parameter learning method, called *Discriminative Frequency Estimate* (DFE). Our motivation is to turn the generative parameter learning method FE into a discriminative one by injecting a discriminative element into it. DFE discriminatively computes frequencies from data, and then estimates parameters based on the appropriate frequencies. Our empirical studies show that DFE inherits the advantages of both generative and discriminative learning.

2. Related Work

Greiner and Zhou (2002) showed that discriminative parameter learning for Bayesian networks is equivalent to a logistic regression problem under certain conditions. For many Bayesian network structures, they indicated that the conditional likelihood function may have only one global maximum, and thus can be maximized by local optimization methods. They also proposed a gradient descent based parameter learning method, called ELR. To make ELR work effectively, they modified the basic gradient descent method using FE to initialize parameters and cross tuning to prevent overfitting. Empirical studies showed that ELR significantly outperforms the generative learning approach.

Grossman and Domingos (2004) proposed a discriminative structure learning method for Bayesian network classifiers, and tried to combine discriminative structure learning with discriminative parameter learning. To overcome the efficiency problem of ELR, they reduced the fold of cross tuning, and used a small sample for parameter learning. They observed that the modified ELR still takes two orders of magnitude of learning time longer than FE in their experiments, and the performance of the combination of discriminative structure and parameter learning does not outperform the discriminative structure learning alone. Therefore, they suggested learning a structure by conditional likelihood, and setting parameters by the FE method.

To our knowledge, ELR is the state-of-the-art algorithm for discriminative parameter learning for Bayesian network classifiers. Unfortunately, its computational cost is quite high. In this paper, we propose a discriminative parameter learning algorithm that is as effective as ELR but much more efficient.

3. Frequency Estimate

We use capital letters X for a discrete random variable. The lower-case letters x is used for the value taken by variable X, and x_{ij} refers to the variable X_i taking on its j_{th} value. We use the boldface capital letters $\mathbf{X}$ for a set of variables, and the boldface lower case letters $\mathbf{x}$ for the values of variables in $\mathbf{X}$. The training data D consists of a set of finite number of training instances, and an instance e is represented by a vector $(\mathbf{x}, c)$, where c is the class label. In general, we use a "hat" to indicate parameter estimates.

A Bayesian network encodes a joint probability distribution $P(\mathbf{X}, C)$ by a set of local distributions P for each variable. By forcing the class variable C to be the parent of each variable X_i, we can compute the posterior probability $P(C|\mathbf{X})$ as follows.

$$P(C|\mathbf{X}) = \alpha P(C) \prod_{i=1}^{n} P(X_i|\mathbf{U}_i), \tag{1}$$

where α is a normalization factor, and $\mathbf{U}_i$ denotes the set of parents of variable X_i. Note that the class variable C is always one parent of X_i. In naive Bayes, $\mathbf{U}_i$ only contains the class variable C. $P(C)$ is called the prior probability and $P(X_i|\mathbf{U}_i)$ is called the local probability distribution of X_i.

The local distribution $P(X_i|\mathbf{U}_i)$ is usually represented by a conditional probability table (CPT), which enumerates all the conditional probabilities for each assignment of values to X_i and its parents $\mathbf{U}_i$. Each conditional probability $P(x_{ij}|\mathbf{u}_{ik})$ in a CPT is often estimated using the corresponding frequencies obtained from the training data as follows.

$$\hat{P}(x_{ij}|\mathbf{u}_{ik}) = \frac{n_{ijk}}{n_{ik}}, \tag{2}$$

where n_{ijk} denotes the number of training instances in which variable X_i takes on the value x_{ij} and its parents $\mathbf{U}_i$ take on the values $\mathbf{u}_{ik}$. n_{ik} is equal to the sum of n_{ijk} over all j. The prior probability $P(C)$ is also estimated in the same way.

For the convenience in implementation, an entry θ_{ijk} in a CPT is the frequency n_{ijk}, instead of $P(x_{ij}|\mathbf{u}_{ik})$, which can be easily converted to $P(x_i|\mathbf{u}_i)$. To compute the frequencies from a given training data set, we go through each training instance, and increase the corresponding entries θ_{ijk} in CPTs by 1. By scanning the training data set once, we can obtain all the required frequencies and then compute the corresponding conditional probabilities. This parameter learning method is called *Frequency Estimate* (FE).

It is well-known that FE is a generative learning approach, because it maximizes likelihood (Friedman

et al., 1997). In classification, however, the parameter setting that maximizes generalization accuracy is desired. Theoretically, if the structure of a Bayesian network is correct, the parameters determined by FE also maximize generalization accuracy. In practice, however, this assumption is rarely true. Therefore, the parameter learning method that directly maximizes generalization accuracy is more desirable in classification.

4. Discriminative Frequency Estimate

We now introduce *Discriminative Frequency Estimate* (DFE), a discriminative parameter learning algorithm for Bayesian network classifiers.

Note that, when counting a training instance in FE, we simply increase the corresponding frequencies by 1. Consequently, we do not directly take the effect on classification into account in computing frequencies. In fact, at any step in this process, we actually have a classifier on hand: the classifier whose local probabilities are computed by Equation 2 using the current entries (frequencies) in CPTs.

Thus, when we count an instance, we can apply the current classifier to it, and then update the corresponding entries based on how well (bad) the current classifier predicts on the instance. Intuitively, if the instance can be classified perfectly, there is no need to change any entries. In general, given an instance e, we can compute the difference between the true probability $P(c|e)$ and the predicted probability $\hat{P}(c|e)$ generated by the current parameters, where c is the true class of e, and then update the corresponding entries based on the difference. Furthermore, the FE process can be generalized such that we can count each instance more than once (as many as needed) until an convergence occurs. This is the basic idea of DFE.

More precisely, the DFE parameter learning algorithm iterates through the training instances. For each instance e, DFE firstly computes the predicted probability $\hat{P}(c|e)$, and then updates the frequencies in corresponding CPTs using the difference between the true $P(c|e)$ and the predicted $\hat{P}(c|e)$. The detail of the algorithm is depicted as follows. Here M is a pre-defined maximum number of steps. $L(e)$ is the prediction loss for training instance e based on the current parameters Θ^t, defined as follows.

$$L(e) = P(c|e) - \hat{P}(c|e). \tag{3}$$

In general, $P(c|e)$ are difficult to know in classification task, because the information we have for c is only the class label. Thus, we assume that $P(c|e) = 1$ when e is in class c in our implementations. Note this assumption may not be held if data can not be separated completely, and thus may introduce bias to our probability estimation.

Algorithm 1 Discriminative Frequency Estimate

1. Initialize each CPT entry θ_{ijk} to 0
2. **For** t from 1 to M **Do**
   - Randomly draw a training instance e from the training data set D.
   - Compute the posterior probability $\hat{P}(c|e)$ using the current parameters Θ^t and Equation 2.
   - Compute the loss $L(e)$ using Equation 3.
   - **For** each corresponding frequency θ_{ijk} in CPTs
     - Let θ^{t+1}_{ijk}=θ^t_{ijk}+$L(e)$.

Note that, in the beginning, each CPT entry θ_{ijk} is 0, and thus the predicted $\hat{P}(c|e)$ is $\frac{1}{|C|}$ after the probability normalization. In each step, if the current parameters Θ^t cannot accurately predict $P(c|e)$ for an instance e, the corresponding entries θ_{ijk} are increased significantly. If the current parameter Θ^t can perfectly predict $P(c|e)$, there will be no change on any entry.

The following summarizes our understanding for DFE:

1. The generative element is Equation 2. If we set the additive updates $L(e)$ in Equation 3 as a constant, DFE will be a maximum likelihood estimator, which is exactly the same as in the traditional naive Bayes. Thus, the parameters learned by DFE are influenced by the likelihood information $P(x_{ij}|\mathbf{u}_{ik})$ through Equation 2.
2. The discriminative element is Equation 3. If we use each entry θ_{ijk} in CPTs as parameters rather than generating the parameters using Equation 2, DFE will be a typical perceptron algorithm in the sense of error-driven learning. Thus, the parameters learned by DFE are also influenced by the prediction error through Equation 3.
3. DFE is different from a perceptron algorithm because of Equation 2. As we explained above, if we set the additive updates $L(e)$ in Equation 3 as a constant, there is no difference between DFE and a traditional naive Bayes. However, if we set the additive updates in a standard perceptron algorithm as a constant, the perceptron algorithm will not learn a traditional naive Bayes.

In summary, DFE learns parameters by considering the likelihood information $P(x_{ij}|\mathbf{u}_{ik})$ and the prediction error $P(c|e) - \hat{P}(c|e)$, and thus can be considered as a combination of generative and discriminative learning. Moreover, the likelihood information $P(x_{ij}|\mathbf{u}_{ik})$ seems to be more important than $P(c|e) - \hat{P}(c|e)$. For example, a DFE algorithm without Equation 2 performs significantly worse than naive Bayes, while a DFE algorithm without Equation 3 can still learn a traditional naive Bayes.

5. An Example

Before presenting our experiments, it could be helpful to get some intuitive feeling on DFE through a simple example.

A1	A2	A3	C
1	1	1	-
1	1	1	-
0	0	0	+
0	0	0	-
0	0	0	-

Figure 1. A data set with duplicate variables

Figure 1 shows a learning problem consisting of 5 instances and 3 variables. The variables A_2 and A_3 are the two duplicates of A_1, and thus all variables are perfectly dependent. For an instance $e = \{A_1 = 0, A_2 = 0, A_3 = 0\}$, the true posterior probability ratio is:

$$\frac{p(C = +|A_1 = 0, A_2 = 0, A_3 = 0)}{p(C = -|A_1 = 0, A_2 = 0, A_3 = 0)} = \frac{1}{2} \qquad (4)$$

However, naive Bayes, which does not consider the dependencies between variables, gives the estimated posterior probability ratio:

$$\frac{\hat{p}(C = +)}{\hat{p}(C = -)}(\frac{\hat{p}(A_1 = 0|+)}{\hat{p}(A_1 = 0|-)})^3 = \frac{2}{1} \qquad (5)$$

Thus, naive Bayes misclassifies e. Moreover, the estimated posterior probability $\hat{p}(C = +|A_1 = 0, A_2 = 0, A_3 = 0)$ from naive Bayes is 0.66, while the true probability $p(C = +|A_1 = 0, A_2 = 0, A_3 = 0) = 0.33$. This mismatch is due to the two duplicates A_2 and A_3. Since $\frac{p(A_i=0|C=+)}{p(A_i=0|C=-)} = 2$, the duplication of A_1 results in overestimating the probability that e belongs to the positive class.

For DFE, the story is different. Figure 2 shows how the estimated probability $\hat{p}(C = +|A_1 = 0, A_2 = 0, A_3 = 0)$ in naive Bayes changes with FE and DFE respectively, as the number of instances used increases. Both algorithms take an instance in the order in Figure 1 at each step, and update the corresponding frequencies. With the increased number of instances used, the estimated probability $\hat{p}(C = +|A_1 = 0, A_2 = 0, A_3 = 0)$ from DFE converges to 0.4 approximately, which is close to the true probability and leads to a correct classification. However, FE converges to 0.66, even using the training instances more than once.

From this example, we can see that computing the frequencies in a discriminative way tends to yield more accurate probability estimation and give more accurate classification consequently. Also, both DFE and FE tend to converge with the increased training effort.

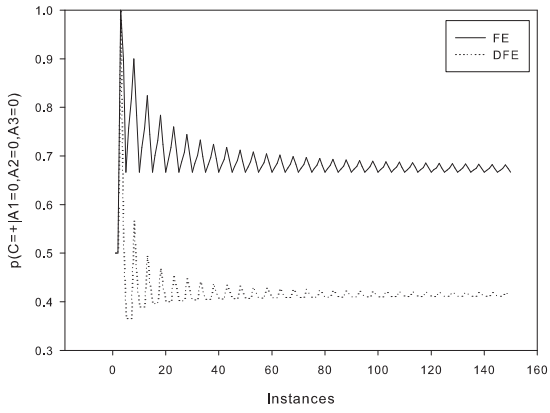


Figure 2. The y-axis is the predicted probability. The x-axis is the t_{th} instance fed into the algorithms.

6. Experiments

6.1. Experimental Setup

We conduct our experiments under the framework of WEKA (Witten & Frank, 2000). All experiments are performed on a Pentium 4 with 2.8GHZ CPU and 1G RAM. In our experiments, we use the 33 UCI data sets, selected by WEKA, which represent a wide range of domains and data characteristics. The smallest training data set "labor" has 51 training instances, and the largest data set "mushroom" has 7311 training instances. Numeric variables are discretized using the unsupervised ten-bin discretization implemented in WEKA. Missing values are replaced with the mean values from the training data. The multi-class data sets are transformed into binary ones by taking the two largest classes. The performance of an algorithm on each data set is observed via 10 runs of 10-fold stratified cross validation.

Table 1. Experimental results on accuracy

Data set	NB+DFE	NB+FE	NB+ELR	NB+Ada	HGC+FE	HGC+DFE
Labor	92.73±12.17	96.27± 7.87	95.53± 9.00	86.53±13.95	89.80±10.80	86.93±12.12
Zoo	100.00± 0.00	100.00± 0.00	100.00± 0.00	100.00± 0.00	100.00± 0.00	100.00± 0.00
Iris	100.00± 0.00	100.00± 0.00	96.20±11.05	100.00± 0.00	100.00± 0.00	100.00± 0.00
Primary-tumor	84.12± 9.17	84.12± 9.48	83.32± 9.99	80.82± 8.77	82.94± 9.07	82.23±10.32
Autos	88.94± 9.81	77.24±12.03 •	90.27± 9.08	88.76± 8.70	83.97±10.75	84.49±11.85
Audiology	100.00± 0.00	99.82± 1.29	97.31± 5.22	99.82± 1.29	99.82± 1.29	100.00± 0.00
Glass	80.45± 9.91	76.37±10.59	81.44±10.04	75.03± 9.12	72.12±11.89 •	71.55±12.32 •
Vowel	95.89± 4.87	83.56± 8.76 •	92.33± 5.71	94.44± 4.63	97.44± 3.22	97.44± 3.41
Soybean	98.58± 3.30	95.52± 4.74	97.29± 4.13	97.38± 3.18	97.49± 3.85	98.05± 3.27
Hepatitis	84.79± 9.11	84.13±10.34	83.49±10.41	81.42± 9.33	83.01± 9.04	83.71± 9.01
Sonar	76.85± 9.30	76.02±10.67	77.36± 9.49	75.23± 9.09	69.16±10.44	68.89±10.49
Lymphography	86.33± 8.95	86.21± 8.12	85.08± 8.84	83.34± 9.56	84.40± 9.10	84.23± 8.49
Heart-statlog	82.89± 5.69	83.70± 5.60	82.96± 5.80	76.44± 7.59 •	83.04± 5.55	82.00± 4.96
Cleveland	83.04± 7.49	83.57± 5.99	82.50± 7.11	79.08± 7.94 •	82.32± 7.46	80.99± 7.43
Breast-cancer	70.36± 8.05	72.87± 7.48	71.34± 8.04	69.73± 7.71	74.26± 5.45	73.49± 5.98
Ionosphere	90.54± 5.32	90.83± 3.99	91.11± 4.82	89.13± 6.14	93.28± 4.53	91.51± 5.29
Horse-colic	82.99± 6.03	78.70± 6.27 •	80.59± 6.71	77.60± 6.30 •	83.70± 5.30	82.01± 6.86
Vehicle	93.74± 3.40	82.82± 6.80 •	92.96± 3.52	93.84± 3.31	95.68± 3.11	96.78± 2.87 ○
Vote	94.80± 2.86	90.29± 4.07 •	95.72± 2.87	94.39± 3.12	94.84± 3.15	95.44± 3.15
Balance	99.48± 0.94	99.24± 1.17	99.83± 0.52	99.27± 1.17	99.27± 1.22	99.69± 0.76
Wisconsin	96.45± 2.05	97.31± 1.70	96.22± 2.10	95.11± 2.40	96.91± 1.51	96.71± 2.11
Segment	100.00± 0.00	100.00± 0.00	100.00± 0.00	100.00± 0.00	100.00± 0.00	100.00± 0.00
Credit-rating	85.51± 3.65	84.75± 3.68	85.25± 4.16	82.96± 4.08 •	85.51± 4.27	86.03± 3.80
Diabetes	75.78± 4.67	75.57± 4.76	76.15± 4.36	74.90± 4.75	75.94± 5.14	75.60± 4.68
Anneal	99.72± 0.74	97.67± 1.98 •	99.87± 0.53	99.87± 0.59	98.95± 1.20	99.92± 0.30
Credit-g	75.98± 4.01	75.90± 3.97	75.64± 3.73	74.22± 4.43	72.88± 3.88 •	72.92± 3.53 •
Letter	99.52± 0.49	96.49± 1.42 •	98.95± 0.74 •	98.94± 0.83 •	98.42± 1.01 •	99.46± 0.52
Splice	97.62± 0.98	97.61± 0.98	97.64± 0.89	95.70± 1.41 •	98.05± 0.91	98.01± 0.85
Kr-vs-kp	94.70± 1.37	87.80± 1.89 •	95.68± 1.21 ○	95.19± 1.19	92.40± 1.61 •	95.54± 1.37 ○
Waveform	91.05± 1.52	87.52± 1.46 •	90.71± 1.16	89.19± 1.63 •	89.66± 1.40 •	89.72± 1.43 •
Hypothyroid	95.95± 0.46	95.49± 0.49 •	95.83± 0.49	95.55± 0.66 •	95.72± 0.52	95.88± 0.51
Sick	97.49± 0.67	96.75± 0.97 •	97.47± 0.74	96.79± 0.83 •	97.82± 0.75	97.91± 0.70
Mushroom	99.96± 0.06	95.53± 0.63 •	100.00± 0.00	100.00± 0.00	99.96± 0.06	100.00± 0.00

• worse, and ○ better, comparing with NB-DFE.

Two Bayesian network classifiers, naive Bayes (NB) and HGC (Heckerman et al., 1995), are used to compare the performance of different parameter learning methods. HGC is a hill-climbing structure search algorithm. In our experiments with HGC, we limit the number of parents of each node to 2.

In general, we use NB+X and HGC+X to indicate that NB and HGC with a specific parameter learning method X respectively: X is one of FE, DFE, ELR and Ada (Freund & Schapire, 1996). Note that, for HGC+DEF, we use HGC to learn the structure first, and then apply DFE to learn parameters. We do not use DFE in the structure learning of HGC. The following summarizes the parameter learning algorithms used in our experiments.

FE: the *generative parameter learning* method. Note the term "one iteration" in this paper indicates that we count all training instances exactly once.

DFE: the *discriminative parameter learning* method, depicted in Section 4. In our implementation, we simply go through the whole training data four times (iterations), instead of randomly choosing instances.

ELR: the *gradient descent based discriminative parameter learning* method, proposed in (Greiner & Zhou, 2002).

Ada: *Adaboost M1* is used as an ensemble method that combines the outputs of base classifiers to produce a better prediction (Freund & Schapire, 1996). The number of classifiers is 20.

In our experiments, we use the implementation of ELR from the authors (Greiner & Zhou, 2002) and the implementation of HGC and Ada in WEKA, and implement DFE in WEKA.

Table 2. Summary of the experimental results on accuracy.

	NB+FE	NB+ELR	NB+Ada	HGC+FE	HGC+DFE
NB+DFE	12/21/0	1/31/1	9/24/0	5/28/0	3/28/2
NB+FE		0/22/11	9/19/5	0/22/11	1/22/10
NB+ELR			4/29/0	4/27/2	2/28/3
NB+Ada				2/26/5	0/28/5
HGC+FE					0/30/3

6.2. Accuracy and Training Time

Table 1 gives the detailed experimental results on accuracy. To better understand the effect of training data size on the algorithm performance, we sort the data sets by their sizes. Table 2 shows the results of the paired t-test with significance level 0.05, in which each entry $w/t/l$ means that the learner in the corresponding row wins in w data sets, ties in t data sets, and loses in l data sets, compared to the learning algorithm in the corresponding column. The following is the highlight of our observations.

1. The two discriminative parameter learning methods ELR and DFE have the similar performance

Table 3. Experimental results on training time

Data set	NB+DFE	NB+FE	NB+ELR	NB+Ada	HGC+FE	HGC+DFE
Labor	0.0009±0.00	0.0002±0.00 •	55.0250± 48.89 ○	0.0066±0.00 ○	0.0367±0.00 ○	0.0416±0.00 ○
Zoo	0.0006±0.00	0.0001±0.00 •	100.1444± 49.28 ○	0.0018±0.00 ○	0.0077±0.00 ○	0.0120±0.00 ○
Iris	0.0005±0.00	0.0001±0.00 •	317.4304± 150.43 ○	0.0019±0.00 ○	0.0030±0.00 ○	0.0058±0.00 ○
Primary-tumor	0.0010±0.00	0.0002±0.00 •	99.7059± 12.72 ○	0.0097±0.00 ○	0.0178±0.00 ○	0.0295±0.00 ○
Autos	0.0017±0.00	0.0002±0.00 •	202.3540± 42.84 ○	0.0213±0.02 ○	0.2843±0.03 ○	0.2956±0.04 ○
Audiology	0.0019±0.00	0.0003±0.00 •	311.3375± 55.36 ○	0.0023±0.00	0.5920±0.05 ○	0.6141±0.05 ○
Glass	0.0008±0.00	0.0002±0.00 •	205.4710± 14.92 ○	0.0117±0.00 ○	0.0363±0.00 ○	0.0464±0.02 ○
Vowel	0.0013±0.00	0.0003±0.00 •	399.6574± 263.88 ○	0.0176±0.00 ○	0.0373±0.00 ○	0.0510±0.00 ○
Soybean	0.0018±0.00	0.0004±0.00 •	507.1840± 55.93 ○	0.0230±0.01 ○	0.0464±0.00 ○	0.0705±0.02 ○
Hepatitis	0.0015±0.00	0.0002±0.00 •	414.9584± 27.77 ○	0.0191±0.00 ○	0.0369±0.00 ○	0.0559±0.02 ○
Sonar	0.0066±0.00	0.0007±0.00 •	932.8643± 106.34 ○	0.0669±0.02 ○	4.8039±0.21 ○	4.8389±0.20 ○
Lymphography	0.0037±0.02	0.0003±0.00	387.2173± 19.10 ○	0.0164±0.00 ○	0.0335±0.00 ○	0.0480±0.00 ○
Heart-statlog	0.0019±0.00	0.0003±0.00 •	579.2737± 74.95 ○	0.0252±0.02 ○	0.0494±0.02 ○	0.0674±0.02 ○
Cleveland	0.0020±0.00	0.0028±0.02	681.2536± 109.79 ○	0.0209±0.01 ○	0.0239±0.00 ○	0.0451±0.00 ○
Breast-cancer	0.0015±0.00	0.0002±0.00 •	541.8432± 56.39 ○	0.0126±0.00 ○	0.0161±0.02 ○	0.0288±0.00 ○
Ionosphere	0.0054±0.00	0.0007±0.00 •	2261.0212± 780.54 ○	0.0629±0.02 ○	0.3492±0.04 ○	0.4219±0.04 ○
Horse-colic	0.0044±0.00	0.0005±0.00 •	1506.9836± 146.88 ○	0.0430±0.01 ○	0.0987±0.02 ○	0.1457±0.02 ○
Vehicle	0.0039±0.00	0.0005±0.00 •	2125.4934± 137.27 ○	0.0480±0.00 ○	0.1531±0.02 ○	0.2009±0.03 ○
Vote	0.0034±0.00	0.0005±0.00 •	1779.7511± 251.58 ○	0.0334±0.02 ○	0.0229±0.02 ○	0.0632±0.02 ○
Balance	0.0017±0.00	0.0005±0.00 •	2710.6686±1280.37 ○	0.0243±0.01 ○	0.0038±0.00 ○	0.0189±0.00 ○
Wisconsin	0.0034±0.00	0.0005±0.00 •	1376.4606± 146.91 ○	0.0559±0.02 ○	0.0243±0.00 ○	0.0624±0.02 ○
Segment	0.0057±0.00	0.0008±0.00 •	3973.2459± 659.38 ○	0.0039±0.00 •	0.1233±0.02 ○	0.1952±0.03 ○
Credit-rating	0.0076±0.02	0.0006±0.00	1316.8793± 68.50 ○	0.0514±0.02 ○	0.0648±0.02 ○	0.1252±0.02 ○
Diabetes	0.0034±0.00	0.0005±0.00 •	1118.3888± 41.75 ○	0.0344±0.01 ○	0.0299±0.00 ○	0.0676±0.02 ○
Anneal	0.0097±0.00	0.0011±0.00 •	4947.6380±1573.55 ○	0.1098±0.03 ○	0.1797±0.03 ○	0.3056±0.03 ○
Credit-g	0.0103±0.00	0.0012±0.00 •	2440.2377± 357.70 ○	0.0745±0.03 ○	0.1473±0.03 ○	0.2611±0.03 ○
Letter	0.0223±0.00	0.0024±0.00 •	262.4565± 142.62 ○	0.3817±0.15 ○	0.2089±0.06 ○	0.5355±0.20 ○
Splice	0.1322±0.04	0.0143±0.02 •	2398.4974± 835.69 ○	1.3441±0.44 ○	8.1985±2.28 ○	9.8085±2.38 ○
Kr-vs-kp	0.1533±0.08	0.0232±0.06 •	1648.1174± 856.53 ○	1.2348±0.16 ○	1.0847±0.17 ○	1.9889±0.11 ○
Waveform	0.1829±0.05	0.0171±0.00 •	2743.9441± 295.50 ○	1.5109±0.16 ○	2.3946±0.26 ○	3.4949±0.28 ○
Hypothyroid	0.1091±0.04	0.0121±0.01 •	1035.1162± 543.62 ○	1.2212±0.63 ○	1.1376±0.34 ○	3.3269±1.24 ○
Sick	0.1246±0.08	0.0095±0.00 •	2662.4956± 379.71 ○	0.6825±0.27 ○	1.6360±0.64 ○	3.4699±0.99 ○
Mushroom	0.2102±0.15	0.0205±0.02 •	11243.5967±3074.40 ○	2.6704±0.95 ○	2.2242±0.86 ○	4.5443±1.32 ○

○ slower, and • faster comparing with NB-DFE The training time unit is second

in terms of accuracy. NB+DFE performs better than NB+ELR in 1 data set and loses in 1 data set.

2. For naive Bayes, the discriminative parameter learning methods significantly improve the performance of the generative parameter learning method FE. NB+ELR and NB+DFE outperform NB+FE in 11 and 12 data sets without a loss respectively. In our experiments, NB+Ada loses to NB in 9 data sets and wins in 5 data sets. This means that using boosting as a discriminative parameter learning method is not effective according to our experiments.

3. NB+DFE outperforms HGC+FE in 5 data sets without a loss. Note that there is no structure learning in NB+DFE at all. Thus, we could expect that discriminative parameter learning can significantly reduce the effort for structure learning.

4. DFE improves the general Bayesian network learning algorithm HGC. HGC+DFE outperforms HGC+FE in 3 data sets without a loss. This improvement is not as significant as in naive Bayes. However, it is consistent with previous research results: while the structure of a Bayesian network is closer to the "true" one, discriminative parameter learning is less helpful (Greiner & Zhou, 2002; Grossman & Domingos, 2004).

5. HGC+FE outperforms NB+FE in 11 data sets without a loss. This results show that many data sets in our experiments contains strong dependencies. The structure learning in HGC relaxes the independence assumption in naive Bayes, and thus improves the performance significantly.

We have also observed the training time for each algorithm. Table 3 shows the average training time of each algorithm from 10 runs of 10-fold stratified cross validation. From Table 3, we can see that DFE is approximately 250,000 times faster than ELR. Recall that their performance in classification accuracy is similar. Certainly, FE is still the most efficient algorithm: 7 times faster than DFE, 70 faster than NB+Ada, and 1,800,000 times faster than ELR approximately.

6.3. Convergence, Overfitting and Learning Curves

In our experiments, we have investigated the convergence of the DFE algorithm. We have observed the relation between the number of iterations and the accuracy of NB+DFE on the 8 largest data sets, shown in Figure 3. Again, an iteration means counting all instances once. Each point in the curves corresponds to the number of iterations that a parameter learning method performs over the training data and the average accuracy from 10-fold cross validation.

Figure 3 shows that NB+DFE converges quickly. We

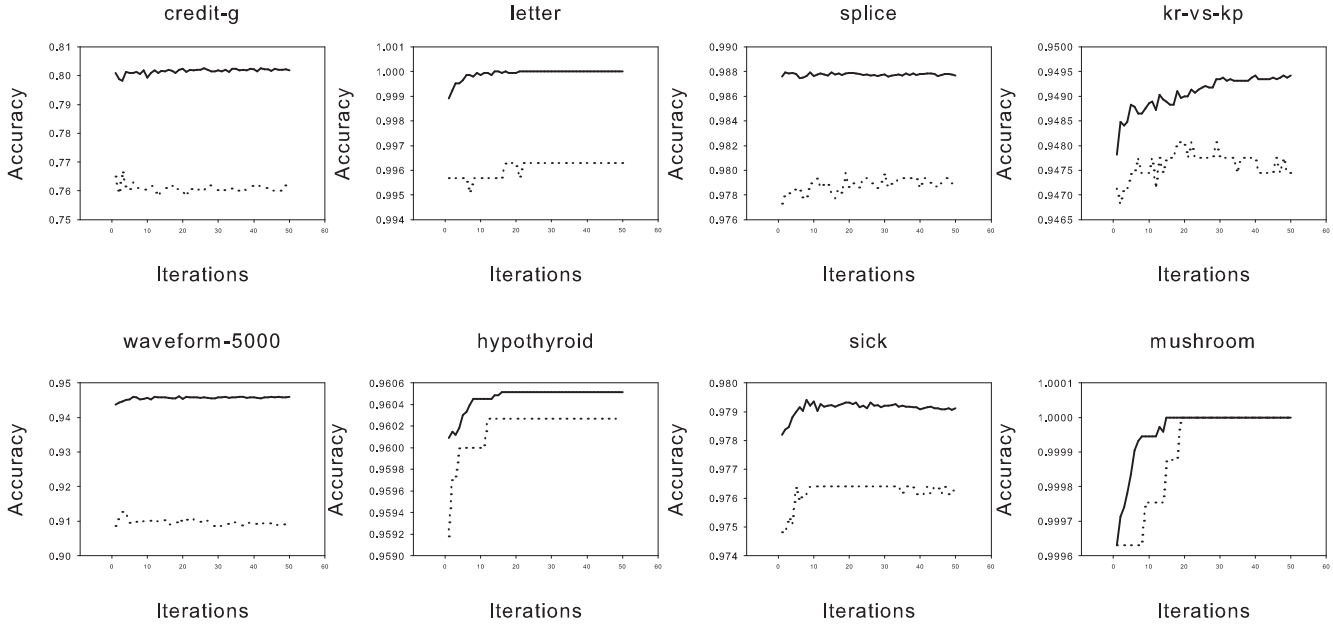


Figure 3. Relation between accuracies and the number of iterations over training and testing data. Solid lines represent training accuracy, and dotted lines represent testing accuracy.

can see that NB+DFE approaches its highest accuracy just after one iteration. As the number of iterations increases after that, there is no significant difference. For example, in all the 8 data sets, the differences between NB+DFE with one iteration and with more iterations are only around 0.005. In our experiments, in fact, we have tried different iteration numbers (1 to 2048) for DFE, and the accuracies of NB+DFE and HGC+DFE do not significantly change.

In the 33 data sets, there is only one data set "Vowel", in which NB+DFE needs more than one iteration to reach the asymptotic accuracy. NB+DFE achieves 90.00% after one iteration, and approaches 95.89% after 4 iterations. The "Vowel" data set has been observed to contain strong variable dependencies (Su & Zhang, 2005), and is small (contains only 180 training instances). However, when the sample size is not small, such as in "Kr-vs-kp" and "Mushroom", one iteration is still enough for DFE to reach its asymptotic accuracy, even though there are strong dependencies in these data sets.

From Figure 3, we can also observe that NB+DFE does not suffer from overfitting. With the increased iterations, the accuracies on test data, shown by the dotted lines, remain the same. That means, once NB+DFE reaches its asymptotic accuracy, the more learning effort does not influence the model. Consequently, no stopping criterion is required for DFE. In contrast, the discriminative learning algorithm ELR requires a stopping criterion to prevent overfitting. Greiner and Zhou (2002) showed that the accuracy of ELR may decrease with an increased training effort.

We have also studied the learning curves of NB+DFE. Ng and Jordan (2001) showed that discriminative learning may have disadvantage comparing to generative learning when sample size is small. Thus, we are interested in how our discriminative parameter learning algorithm DFE performs in this scenario.

Figure 4 shows the learning curves for NB+FE, NB+ELR, and NB+DFE on the same 8 UCI data sets. Since we are interested in the performance in a small sample size, we only observe the performance of each algorithm using up to 50 instances. The accuracy in the learning curves is the average accuracy, obtained on the data that is not used for training with a total of 30 runs. The learning curves show how the accuracy changes as more labeled data are used.

From Figure 4, we can see that NB+FE dominates NB+DFE and NB+ELR only on data sets "Credit-g" and "Hypothyroid" in terms of accuracy. On data sets "Kr-vs-kp" and "Mushroom", however, both discriminative learning algorithms NB+ELR and NB+DFE outperform NB+FE. On all other data sets, the results are mixed. It means that generative learning has actually no obvious advantage over discriminative learning even when the size of training data is small. In fact, our observations agree with the analysis in (Greiner & Zhou, 2002).

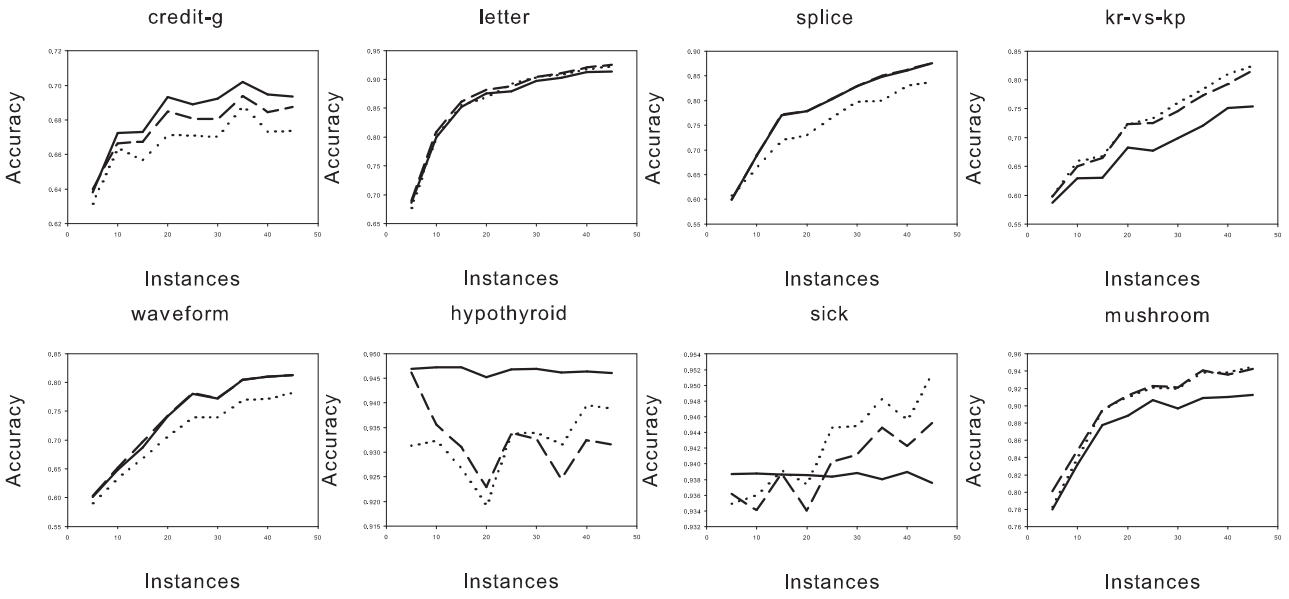


Figure 4. Relation between accuracies and training data sizes. Solid, dotted, and dashed lines correspond to NB+FE, NB+DFE, and NB+ELR respectively.

7. Conclusion

In this paper, we propose a novel discriminative parameter learning method for Bayesian network classifiers. DFE can be viewed as a discriminative version of frequency estimate. Our experiments show that the DFE algorithm combines the advantages of generative and discriminative learning: it is computationally efficient, converges quickly, does not suffer from the overfitting problem, and performs competitively with the state-of-the-art discriminative parameter learning algorithm ELR in accuracy.

This paper mainly studies the empirical side of DFE. Its theoretical nature remains unknown. Moreover, because of the efficiency of DFE, we would expect that DFE could be applied in general structure learning, leading to more accurate Bayesian network classifiers. In our future work, we will study DFE from theoretical perspective and embed DFE into the structure search process of HGC and other structure learning algorithms.

References

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 148 – 156).

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131–163.

Greiner, R., & Zhou, W. (2002). Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *AAAI/IAAI* (pp. 167–173).

Grossman, D., & Domingos, P. (2004). Learning bayesian network classifiers by maximizing conditional likelihood. *ICML '04: Proceedings of the twenty-first international conference on Machine learning* (p. 46). New York, NY, USA: ACM Press.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, *20*, 197–243.

Ng, A. Y., & Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *NIPS* (pp. 841–848).

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems:networks of plausible inference*. Morgan Kauhmann.

Su, J., & Zhang, H. (2005). Representing conditional independence using decision trees. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 874–879. AAAI Press.

Witten, I. H., & Frank, E. (2000). *Data mining – practical machine learning tools and techniques with Java implementation*. Morgan Kaufmann.

A Least Squares Formulation for Canonical Correlation Analysis

Liang Sun SUN.LIANG@ASU.EDU
Shuiwang Ji SHUIWANG.JI@ASU.EDU
Jieping Ye JIEPING.YE@ASU.EDU
Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287 USA

Abstract

Canonical Correlation Analysis (CCA) is a well-known technique for finding the correlations between two sets of multi-dimensional variables. It projects both sets of variables into a lower-dimensional space in which they are maximally correlated. CCA is commonly applied for supervised dimensionality reduction, in which one of the multi-dimensional variables is derived from the class label. It has been shown that CCA can be formulated as a least squares problem in the binary-class case. However, their relationship in the more general setting remains unclear. In this paper, we show that, under a mild condition which tends to hold for high-dimensional data, CCA in multi-label classifications can be formulated as a least squares problem. Based on this equivalence relationship, we propose several CCA extensions including sparse CCA using 1-norm regularization. Experiments on multi-label data sets confirm the established equivalence relationship. Results also demonstrate the effectiveness of the proposed CCA extensions.

1. Introduction

Canonical Correlation Analysis (CCA) (Hotelling, 1936) is commonly used for finding the correlations between two sets of multi-dimensional variables. It makes use of two views of the same set of objects and projects them into a lower-dimensional space in which they are maximally correlated. CCA has been applied successfully in various applications (Hardoon et al., 2004; Vert & Kanehisa, 2003). One popular use of CCA is for supervised learning, in which one view is derived from the data and another view is derived from the class labels. In this setting, the data can be projected into a lower-dimensional space directed by the label information. Such formulation is particularly appealing in the context of dimensionality reduction for multi-label data (Yu et al., 2006).

Multivariate linear regression (MLR) that minimizes the sum-of-squares cost function is a well-studied technique for regression problems. It can also be applied for classification with an appropriate class indicator matrix (Bishop, 2006; Hastie et al., 2001). The solution to least squares problems can be obtained by solving a linear system of equations. A number of algorithms, including the conjugate gradient algorithm, can be applied to solve it efficiently (Golub & Loan, 1996). Furthermore, the least squares formulation can be readily extended using the regularization technique. For example, 1-norm regularization can be incorporated into the least squares formulation to control model complexity and improve sparseness (Tibshirani, 1996). Sparseness often leads to easy interpretation and a good generalization ability. It has been used successfully in PCA (d'Aspremont et al., 2004) and SVM (Zhu et al., 2003).

In contrast to least squares, CCA involves a generalized eigenvalue problem, which is computationally more expensive to solve. Furthermore, it is challenging to derive sparse CCA, as it involves a difficult sparse generalized eigenvalue problem. Convex relaxation of sparse CCA has been studied in (Sriperumbudur et al., 2007), where the exact sparse CCA formulation has been relaxed in several steps. On the other hand, interesting connection between least squares and CCA has been established in the literature. In particular, CCA has been shown to be equivalent to Fisher Linear Discriminant Analysis (LDA) for binary-class problems (Hastie et al., 1995). Meanwhile, it is well-known that LDA is equivalent to least squares in this case (Bishop, 2006; Hastie et al., 2001). Thus CCA can be formulated as a least squares problem for binary-class

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

problems. In practice, the multi-class and multi-label problems are more prevalent. It is therefore tempting to investigate the relationship between least squares and CCA in the more general setting.

In this paper, we study the relationship between CCA and least squares for multi-label problems. We show that, under a mild condition which tends to hold for high-dimensional data, CCA can be formulated as a least squares problem by constructing a specific class indictor matrix. Based on this equivalence relationship, we propose several CCA extensions including sparse CCA using the 1-norm regularization. Furthermore, the entire solution path for sparse CCA can be readily computed by the Least Angle Regression algorithm (LARS) (Efron et al., 2004). We evaluate the established theoretical results using a collection of multi-label data sets. Our experiments confirm the equivalence relationship between these two models under the given assumption. Results also show that, even when the assumption does not hold, they achieve very similar performance. Our experiments also demonstrate the effectiveness of the proposed CCA extensions.

Notations The number of training samples, the data dimensionality, and the number of labels are denoted by n, d, and k, respectively. $x_i \in \mathbb{R}^d$ denotes the ith observation and $y_i \in \mathbb{R}^k$ encodes the corresponding label information. Let $X = [x_1, \cdots, x_n] \in \mathbb{R}^{d\times n}$ be the data matrix and $Y = [y_1, \cdots, y_n] \in \mathbb{R}^{k\times n}$ be the class label matrix. We assume that both $\{x_i\}_1^n$ and $\{y_i\}_1^n$ are centered, i.e., $\sum_{i=1}^n x_i = 0$, and $\sum_{i=1}^n y_i = 0$.

2. Background and Related Work

In this section we give a brief overview of CCA and least squares as well as several other related work.

2.1. Canonical Correlation Analysis

In CCA two different representations of the same set of objects are given, and a projection is computed for each representation such that they are maximally correlated in the dimensionality-reduced space. For $X \in \mathbb{R}^{d\times n}$ and $Y \in \mathbb{R}^{k\times n}$, CCA computes two projection vectors, $w_x \in \mathbb{R}^d$ and $w_y \in \mathbb{R}^k$, such that the following correlation coefficient:

$$\rho = \frac{w_x^T XY^T w_y}{\sqrt{(w_x^T XX^T w_x)(w_y^T YY^T w_y)}} \tag{1}$$

is maximized. Since ρ is invariant to the scaling of w_x and w_y, CCA can be formulated equivalently as

$$\begin{aligned} \max_{w_x, w_y} \quad & w_x^T XY^T w_y \\ \text{subject to} \quad & w_x^T XX^T w_x = 1, \;\; w_y^T YY^T w_y = 1. \end{aligned} \tag{2}$$

We assume that YY^T is nonsingular. It can be shown that w_x can be obtained by solving the following optimization problem:

$$\begin{aligned} \max_{w_x} \quad & w_x^T XY^T \left(YY^T\right)^{-1} YX^T w_x \\ \text{subject to} \quad & w_x^T XX^T w_x = 1. \end{aligned} \tag{3}$$

Both formulations in Eqs. (2) and (3) attempt to find the eigenvectors corresponding to top eigenvalues of the following generalized eigenvalue problem:

$$XY^T(YY^T)^{-1}YX^T w_x = \eta XX^T w_x, \tag{4}$$

where η is the eigenvalue corresponding to the eigenvector w_x. Multiple projection vectors under certain orthonormality constraints can be computed simultaneously by solving the following optimization problem (Hardoon et al., 2004):

$$\begin{aligned} \max_{W} \quad & \mathrm{trace}(W^T XY^T (YY^T)^{-1} YX^T W) \\ \text{subject to} \quad & W^T XX^T W = I, \end{aligned} \tag{5}$$

where $W \in \mathbb{R}^{d\times \ell}$ is the projection matrix, ℓ is the number of projection vectors, and I is the identity matrix. The solution to the optimization problem in Eq. (5) consists of the top ℓ eigenvectors of the generalized eigenvalue problem in Eq. (4).

In regularized CCA (rCCA), a regularization term λI with $\lambda > 0$ is added to XX^T in Eq. (5) to prevent the overfitting and avoid the singularity of XX^T (Bach & Jordan, 2003). Specifically, rCCA solves the following generalized eigenvalue problem:

$$XY^T(YY^T)^{-1}YX^T w_x = \eta(XX^T + \lambda I)w_x. \tag{6}$$

2.2. Least Squares for Regression and Classification

In regression, we are given a training set $\{(x_i, t_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ is the observation and $t_i \in \mathbb{R}^k$ is the corresponding target. We assume that both the observations and the targets are centered. Thus the bias term can be ignored. In this case, the projection matrix W can be computed by minimizing the following sum-of-squares cost function:

$$\min_W \sum_{i=1}^n \|W^T x_i - t_i\|_2^2 = \|W^T X - T\|_F^2, \tag{7}$$

where $T = [t_1, \cdots, t_n]$. It is well known that the optimal projection matrix is given by

$$W_{LS} = (XX^T)^\dagger XT^T, \tag{8}$$

where the pseudo-inverse is used in case XX^T is singular. To improve the generality ability of the model, a penalty term based on 2-norm or 1-norm regularization is commonly applied (Hastie et al., 2001).

Least squares is also commonly applied for classification. In the general multi-class case, we are given a data set consisting of n samples $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$, and $y_i \in \{1, 2, \cdots, k\}$ denotes the class label of the i-th sample, and $k > 2$. To apply the least squares formulation to the multi-class case, the 1-of-k binary coding scheme is usually employed to apply a vector-valued class code to each data point (Bishop, 2006). The solution to the least squares problem depends on the choice of class indicator matrix. Several class indicator matrices have been proposed in the literature (Hastie et al., 2001).

2.3. Related Work

The inherent relationship between least squares and several other models has been established in the past. In particular, LDA for binary-class problems can be formulated as a least squares problem (Duda et al., 2000; Bishop, 2006). Moreover, this equivalence relationship can be extended to the multi-class case using a specific class indicator matrix (Ye, 2007). CCA has been shown to be equivalent to LDA for multi-class problems (Hastie et al., 1995). Thus, CCA is closely related to least squares in the multi-class case. We show in the next section that, under a mild condition, CCA can be formulated as a least squares problem for multi-label classifications when one of the views used in CCA is derived from the labels.

3. Relationship between CCA and Least Squares

In this section we investigate the relationship between CCA and least squares in the multi-label case. We first define four matrices essential for our derivation:

$$\begin{aligned} H &= Y^T(YY^T)^{-\frac{1}{2}} \in \mathbb{R}^{n\times k}, && (9)\\ C_{XX} &= XX^T \in \mathbb{R}^{d\times d}, && (10)\\ C_{HH} &= XHH^TX^T \in \mathbb{R}^{d\times d}, && (11)\\ C_{DD} &= C_{XX} - C_{HH} \in \mathbb{R}^{d\times d}. && (12) \end{aligned}$$

Note that we assume $n \gg k$ and $\mathrm{rank}(Y) = k$ for multi-label problems. Thus, $(YY^T)^{-\frac{1}{2}}$ is well-defined. It follows from the definition above that the solution to CCA can be expressed as the eigenvectors corresponding to top eigenvalues of the matrix $C_{XX}^{\dagger}C_{HH}$.

3.1. Basic Matrix Properties

In this subsection, we study the basic properties of the matrices involved in the following discussion. Following the definition of H in Eq. (9), we have:

Lemma 1. *Let H be defined as in Eq. (9) and let $\{y_i\}_1^n$ be centered, i.e., $\sum_{i=1}^n y_i = 0$. Then we have*

(1). H has orthonormal columns, i.e., $H^TH = I_k$;

(2). $H^Te = 0$.

Given $H \in \mathbb{R}^{n\times k}$ with orthonormal columns, there exists $D \in \mathbb{R}^{n\times(n-k)}$ such that $[H, D] \in \mathbb{R}^{n\times n}$ is an orthogonal matrix (Golub & Loan, 1996), that is

$$I_n = [H, D][H, D]^T = HH^T + DD^T.$$

It follows that

$$C_{DD} = C_{XX} - C_{HH} = XDD^TX^T. \qquad (13)$$

It can be verified from Eqs. (10), (11), and (13) that the matrices C_{XX}, C_{HH}, and C_{DD} are all positive semidefinite.

Let the Singular Value Decomposition (SVD) of X be

$$\begin{aligned} X &= U\Sigma V^T = [U_1, U_2]\ \mathrm{diag}(\Sigma_r, 0)\ [V_1, V_2]^T \\ &= U_1\Sigma_r V_1^T, && (14) \end{aligned}$$

where $r = \mathrm{rank}(X)$, U and V are orthogonal matrices, $\Sigma \in \mathbb{R}^{d\times n}$, $U_1 \in \mathbb{R}^{d\times r}$, $U_2 \in \mathbb{R}^{d\times(d-r)}$, $V_1 \in \mathbb{R}^{n\times r}$, $V_2 \in \mathbb{R}^{n\times(n-r)}$, and $\Sigma_r \in \mathbb{R}^{r\times r}$.

Since U_2 lies in the null space X^T, we have:

Lemma 2. *Let H, X, U_2, and D be defined as above. Then $H^TX^TU_2 = 0$ and $D^TX^TU_2 = 0$.*

3.2. Computing CCA via Eigendecomposition

Recall that the solution to CCA consists of the top ℓ eigenvectors of the matrix $C_{XX}^{\dagger}C_{HH}$. We next show how to compute the eigenvectors of $C_{XX}^{\dagger}C_{HH}$. Define the matrix $A \in \mathbb{R}^{r\times k}$ by

$$A = \Sigma_r^{-1}U_1^TXH. \qquad (15)$$

Let the SVD of A be $A = P\Sigma_A Q^T$, where $P \in \mathbb{R}^{r\times r}$ and $Q \in \mathbb{R}^{k\times k}$ are orthogonal, and $\Sigma_A \in \mathbb{R}^{r\times k}$ is diagonal. Then

$$AA^T = P\Sigma_A\Sigma_A^TP^T. \qquad (16)$$

The matrix $C_{XX}^{\dagger}C_{HH}$ can be diagonalized as follows:

$$\begin{aligned}
C_{XX}^{\dagger}C_{HH} &= U_1\Sigma_r^{-2}U_1^T XHH^TX^T \\
&= U_1\Sigma_r^{-1}AH^TX^TUU^T \\
&= U\begin{bmatrix} I_r \\ 0 \end{bmatrix}\Sigma_r^{-1}A[H^TX^TU_1, H^TX^TU_2]U^T \\
&= U\begin{bmatrix} \Sigma_r^{-1}AA^T\Sigma_r & 0 \\ 0 & 0 \end{bmatrix}U^T \\
&= U\begin{bmatrix} \Sigma_r^{-1}P & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} \Sigma_A\Sigma_A^T & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} P^T\Sigma_r & 0 \\ 0 & I \end{bmatrix}U^T
\end{aligned}$$

where the second equality follows since U is orthogonal, the fourth equality follows since $H^TX^TU_2 = 0$ as shown in Lemma 2, and the last equality follows from Eq. (16). Thus the solution to CCA, which consists of the top ℓ eigenvectors of matrix $C_{XX}^{\dagger}C_{HH}$, is given by

$$W_{CCA} = U_1\Sigma_r^{-1}P_l, \tag{17}$$

where P_ℓ contains the first ℓ columns of P.

3.3. Equivalence of CCA and Least Squares

Recall from Eq. (8) that for a given class indicator matrix T, the solution to the least squares problem is given by

$$(XX^T)^{\dagger}XT^T.$$

We define the class indicator matrix $\tilde{T}$ as follows:

$$\tilde{T} = (YY^T)^{-\frac{1}{2}}Y = H^T. \tag{18}$$

In this case, the solution to the least squares problem is given by

$$\begin{aligned}
W_{LS} &= (XX^T)^{\dagger}XH = U_1\Sigma_r^{-2}U_1^TXH \\
&= U_1\Sigma_r^{-1}A = U_1\Sigma_r^{-1}P\Sigma_AQ^T.
\end{aligned} \tag{19}$$

It is clear from Eqs. (17) and (19) that the difference between CCA and least squares lies in Σ_A and Q^T.

We next show that all diagonal elements of Σ_A are one under a mild condition, that is, $\mathrm{rank}(X) = n-1$ and $\mathrm{rank}(Y) = k$. Note that the first condition is equivalent to requiring that the original data points are linearly independent before centering, which tends to hold for high-dimensional data.

Before presenting the main result summarized in Theorem 1 below, we have the following lemma:

Lemma 3. *Assume*

$$\mathit{rank}(C_{XX}) + s = \mathit{rank}(C_{HH}) + \mathit{rank}(C_{DD}),$$

for some non-negative integer s. Then for the matrix $\hat{\Sigma}_A = \Sigma_A\Sigma_A^T = \mathit{diag}(a_1, a_2, \cdots, a_r) \in \mathbb{R}^{r\times r}$, we have

$$1 = \cdots = a_{f-s} > a_{f-s+1} \geqslant \cdots \geqslant a_f > a_{f+1} = \cdots = 0.$$

where $f = \mathit{rank}(\Sigma_A)$.

Proof. Define the matrix $J \in \mathbb{R}^{d\times d}$ as follows:

$$J = U\begin{bmatrix} \Sigma_r^{-1}P & 0 \\ 0 & I_{d-r} \end{bmatrix}. \tag{20}$$

It follows from the definition of C_{XX}, C_{HH}, and C_{DD} in Eqs. (10)-(12) that

$$\begin{aligned}
J^TC_{XX}J &= \mathrm{diag}(I_r, 0), \\
J^TC_{HH}J &= \mathrm{diag}(\Sigma_A\Sigma_A^T, 0) \\
&= \mathrm{diag}(a_1, \cdots, a_r, 0, \cdots, 0), \\
J^TC_{DD}J &= J^TC_{XX}J - J^TC_{HH}J \\
&= \mathrm{diag}(b_1, \cdots, b_r, 0, \cdots, 0),
\end{aligned} \tag{21}$$

where $b_i = 1 - a_i$, for $i = 1, \cdots, r$. Note that since J is nonsingular, we have

$$\mathrm{rank}(C_{XX}) = \mathrm{rank}(J^TC_{XX}J) = r.$$

It follows from our assumption that

$$\mathrm{rank}(J^TC_{HH}J) + \mathrm{rank}(J^TC_{DD}J) = r + s. \tag{22}$$

Since both $J^TC_{HH}J$ and $J^TC_{DD}J$ are diagonal, there are a total of $r+s$ nonzero elements in $J^TC_{HH}J$ and $J^TC_{DD}J$. Note that $f = \mathrm{rank}(\Sigma_A) = \mathrm{rank}(\hat{\Sigma}_A)$, thus $a_1 \geqslant \cdots \geqslant a_f > 0 = a_{f+1} = \cdots = a_r$. It follows from Eq. (21) that

$$a_i + b_i = 1, \text{ for } 1 \leqslant i \leqslant r, \quad b_r \geqslant \cdots \geqslant b_1 \geqslant 0. \tag{23}$$

This implies that at least one of a_i or b_i is positive for $1 \leqslant i \leqslant r$. To satisfy the rank equality in Eq. (22), we therefore must have

$$\begin{aligned}
1 &= a_1 = a_2 = \cdots = a_{f-s} > a_{f-s+1} \geqslant \cdots \geqslant a_f \\
&> a_{f+1} = \cdots = a_r = 0, \\
0 &= b_1 = b_2 = \cdots = b_{f-s} < b_{f-s+1} \leqslant \cdots \leqslant b_f \\
&< b_{f+1} = \cdots = b_r = 1.
\end{aligned}$$

This completes the proof of the lemma. □

Theorem 1. *Assume that $\mathit{rank}(X) = n-1$ and $\mathit{rank}(Y) = k$ for multi-label problems. Then we have*

$$\mathit{rank}(C_{XX}) = n-1, \tag{24}$$
$$\mathit{rank}(C_{HH}) = k, \tag{25}$$
$$\mathit{rank}(C_{DD}) = n-k-1. \tag{26}$$

Thus $s = 0$, where s is defined in Lemma 3, and

$$1 = a_1 = \cdots = a_k > a_{k+1} = \cdots = a_r = 0.$$

This implies that all diagonal elements of Σ_A are ones.

Proof. Denote $e^T = [1, 1, \cdots, 1] \in \mathbb{R}^{1\times n}$, $H = [h_1, \cdots, h_k]$, and $D = [h_{k+1}, \cdots, h_n]$. Note that X is column centered, i.e., $\sum_{i=1}^n x_i = 0$. It follows from Lemma 1 that $H^T e = 0$, that is,

$$h_i^T e = 0, \text{ for } 1 \leqslant i \leqslant k. \tag{27}$$

Since $[H, D]$ is an orthogonal matrix, $\{h_1, \cdots, h_n\}$ form a basis for $\mathbb{R}^n$. Thus we can represent $e \in \mathbb{R}^n$ as

$$e = \sum_{i=1}^n \alpha_i h_i, \text{ where } \alpha_i \in \mathbb{R}. \tag{28}$$

It follows from the orthogonality of $[H, D]$ and Eq. (27) that e can be expressed as $e = \sum_{i=k+1}^n \alpha_i h_i$, and

$$0 = Xe = X\left(\sum_{i=k+1}^n \alpha_i h_i\right) = \sum_{i=k+1}^n \alpha_i (Xh_i). \tag{29}$$

Since not all α_i's are zero, the $n-k$ columns of XD are linearly dependent, thus $\text{rank}(XD) \leqslant n-k-1$. According to the property of matrix rank, we have

$$\begin{aligned} &\text{rank}(XD) \geqslant \text{rank}(X) + \text{rank}(D) - n \\ &= (n-1) + (n-k) - n = n-k-1. \end{aligned} \tag{30}$$

Thus, $\text{rank}(XD) = n-k-1$ holds.

For matrix XH, we have

$$\begin{aligned} &\text{rank}(X) = \text{rank}(X[H, D]) \leqslant \text{rank}(XH) + \text{rank}(XD) \\ &\Leftrightarrow n-1 \leqslant \text{rank}(XH) + n - k - 1 \\ &\Leftrightarrow \text{rank}(XH) \geqslant k. \end{aligned}$$

On the other hand, since $XH \in \mathbb{R}^{d\times k}$, $\text{rank}(XH) \leqslant k$. Thus we have $\text{rank}(XH) = k$ and

$$\begin{aligned} \text{rank}(C_{XX}) &= \text{rank}(X) = n-1, \\ \text{rank}(C_{HH}) &= \text{rank}(XH) = k, \\ \text{rank}(C_{DD}) &= \text{rank}(XD) = n-k-1. \end{aligned}$$

It follows that $s = 0$. On the other hand,

$$f = \text{rank}(A) = \text{rank}(\Sigma_r^{-1} U_1^T XH) = \text{rank}(XH) = k.$$

Hence,

$$1 = a_1 = a_2 = \cdots = a_k > 0 = a_{k+1} = \cdots = a_r,$$

and all diagonal elements of Σ_A are ones. This completes the proof of the theorem. □

Since $\text{rank}(\Sigma_A) = k$, $C_{XX}^\dagger C_{HH}$ contains k nonzero eigenvalues. If we choose $\ell = k$, then

$$W_{CCA} = U_1 \Sigma_r^{-1} P_k. \tag{31}$$

The only difference between W_{LS} and W_{CCA} lies in the orthogonal matrix Q^T in W_{LS}.

In practice, we can use both W_{CCA} and W_{LS} to project the original data onto a lower-dimensional space before classification. For any classifiers based on Euclidean distance, the orthogonal transformation Q^T will not affect the classification performance since the Euclidean distance is invariant of any orthogonal transformations. Some well-known algorithms with this property include the K-Nearest-Neighbor (KNN) algorithm (Duda et al., 2000) based on the Euclidean distance and the linear Support Vector Machines (SVM) (Schölkopf & Smola, 2002). In the following, the least squares CCA formulation is called "LS-CCA".

4. Extensions of CCA

Based on the equivalence relationship established in the last section, the classical CCA formulation can be extended using the regularization technique.

Regularization is commonly used to control the complexity of the model and improve the generalization performance. Linear regression using the 2-norm regularization, called ridge regression (Hoerl & Kennard, 1970), minimizes the penalized sum-of-squares cost function. By using the class indicator matrix $\tilde{T}$ in Eq. (18), we obtain the 2-norm regularized least squares CCA formulation (called "LS-CCA$_2$") by minimizing the following objective function:

$$L_2(W, \lambda) = \sum_{j=1}^k \left(\sum_{i=1}^n (x_i^T w_j - \tilde{T}_{ij})^2 + \lambda \|w_j\|_2^2 \right),$$

where $W = [w_1, \cdots, w_k]$, and $\lambda > 0$ is the regularization parameter.

In mathematical programming, it is known that sparseness can often be achieved by penalizing the L_1-norm of the variables (Donoho, 2006; Tibshirani, 1996). It has been introduced into the least squares formulation and the resulting model is called lasso (Tibshirani, 1996). Based on the established equivalence relationship between CCA and least squares, we derive the 1-norm least squares CCA formulation (called "LS-CCA$_1$") by minimizing the following objective function:

$$L_1(W, \lambda) = \sum_{j=1}^k \left(\sum_{i=1}^n (x_i^T w_j - \tilde{T}_{ij})^2 + \lambda \|w_j\|_1 \right).$$

The optimal w_j^*, for $1 \leq j \leq k$, is given by

$$w_j^* = \arg\min_{w_j} \left(\sum_{i=1}^n (x_i^T w_j - \tilde{T}_{ij})^2 + \lambda \|w_j\|_1 \right), \tag{32}$$

which can be reformulated as:

$$w_j^* = \arg \min_{\|w_j\|_1 \leq \tau} \sum_{i=1}^{n} (x_i^T w_j - \tilde{T}_{ij})^2, \qquad (33)$$

for some tuning parameter $\tau > 0$ (Tibshirani, 1996). Furthermore, the solution can be readily computed by the Least Angle Regression algorithm (Efron et al., 2004). One key feature of LARS is that it computes the entire solution path for all values of τ, with essentially the same computational cost as fitting the model with a single τ value.

If the value of τ is large enough, the constraint in Eq. (33) is not effective, resulting in an unconstrained optimization problem. We can thus consider τ from a finite range $[0, \hat{\tau}]$, for some $\hat{\tau} > 0$. Define $\gamma = \tau/\hat{\tau}$ so that $\tau = \hat{\tau}\gamma$ with $0 \leq \gamma \leq 1$. The estimation of τ is equivalent to the estimation of γ. Cross-validation is commonly used to estimate the optimal value from a large candidate set $S = \{\gamma_1, \gamma_2, \cdots, \gamma_{|S|}\}$, where $|S|$ denotes the size of S. If the value of γ is sufficiently small, many of the coefficients in W will become exactly zero, which leads to a sparse CCA model. We thus call γ the "sparseness coefficient".

5. Experiments

We use a collection of multi-label data sets to experimentally verify the equivalence relationship established in this paper. We also evaluate the performance of various CCA extensions.

5.1. Experimental Setup

We use two types of data in the experiment. The gene expression pattern image data[1] describe the gene expression patterns of *Drosophila* during development (Tomancak & *et al.*, 2002). Each image is annotated with a variable number of textual terms (labels) from a controlled vocabulary. We apply Gabor filters to extract a 384-dimensional feature vector from each image. We use five data sets with different numbers of terms (class labels). We also use the scene data set (Boutell et al., 2004) which contains 2407 samples of 294-dimension and 6 labels. In all the experiments, ten random splittings of data into training and test sets are generated and the averaged performance is reported.

In the experiment, five methods including CCA and its regularized version rCCA in Eq. (6), as well as LS-CCA and its regularization versions LS-CCA$_2$ and LS-CCA$_1$ are compared. These CCA methods are used to project the data into a lower-dimensional space in which a linear SVM is applied for classification for each label. The Receiver Operating Characteristic (ROC) score is computed for each label and the averaged performance over all labels is reported.

[1]All images were extracted from the FlyExpress database at http://www.flyexpress.net.

5.2. Gene Expression Pattern Image Data

In this experiment we first evaluate the equivalence relationship between CCA and least squares. For all cases, we set the data dimensionality d larger than the sample size n, i.e., $d/n > 1$. The condition in Theorem 1 holds in all cases. We observe that for all splittings of all of the five data sets, $\text{rank}(C_{XX})$ equals $\text{rank}(C_{HH}) + \text{rank}(C_{DD})$), and the ratio of the maximal to the minimal diagonal element of Σ_A is 1, which implies that all diagonal elements of Σ_A are the same, i.e., ones. Our experimental evidences are consistent with the theoretical results presented in Section 3.3.

5.2.1. Performance Comparison

In Table 1, we report the mean ROC scores over all terms and all splittings for each data set. The main observations include: (1) CCA and LS-CCA achieve the same performance for all data sets, which is consistent with our theoretical results; (2) The regularized CCA extensions including rCCA, LS-CCA$_2$, and LS-CCA$_1$ perform much better than their counterparts CCA and LS-CCA without the regularization; and (3) LS-CCA$_2$ is comparable to rCCA in all data sets, while LS-CCA$_1$ achieves the best performance in all cases. These further justify the use of the proposed least squares CCA formulations for multi-label classifications.

Table 1. Comparison of different CCA methods in terms of mean ROC scores. n_{tot} denotes the total number of images in the data set, and k denotes the number of terms (labels). Ten different splittings of the data into training (of size n) and test (of size $n_{tot} - n$) sets are applied for each data set. For the regularized algorithms, the value of the parameter is chosen via cross-validation. The proposed sparse CCA model (LS-CCA$_1$) performs the best for this data set.

n_{tot}	k	CCA	LS-CCA	rCCA	LS-CCA$_2$	LS-CCA$_1$
863	10	0.542	0.542	0.617	0.619	**0.722**
1041	15	0.534	0.534	0.602	0.603	**0.707**
1138	20	0.538	0.538	0.609	0.610	**0.714**
1222	25	0.540	0.540	0.603	0.605	**0.704**
1349	30	0.548	0.548	0.606	0.608	**0.709**

5.2.2. Sensitivity Study

In this experiment, we investigate the performance of LS-CCA and its variants in comparison with CCA when the condition in Theorem 1 does not hold, which

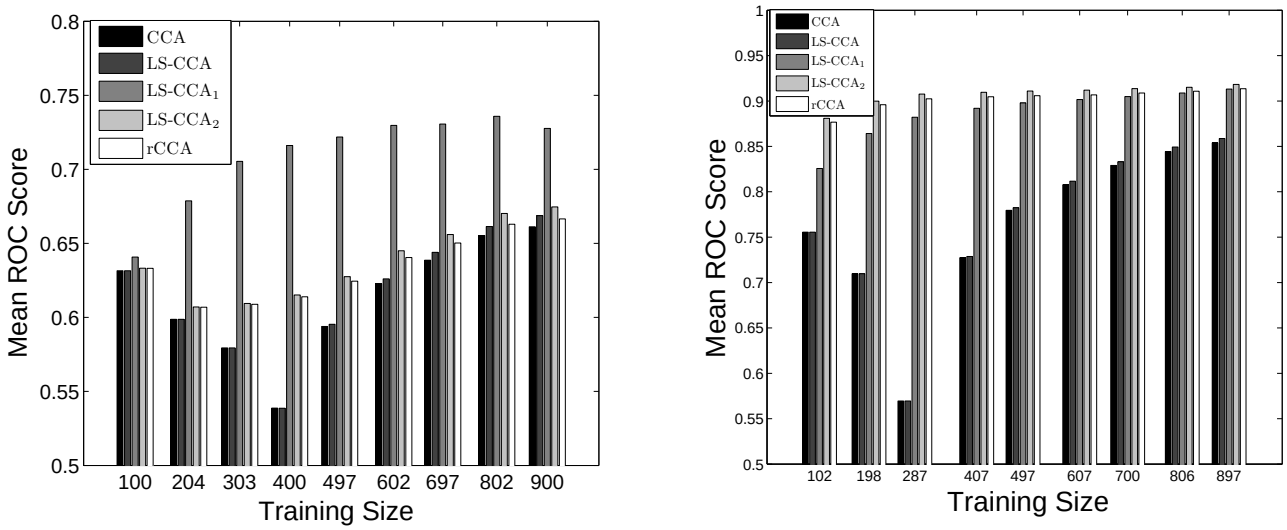


Figure 1. Comparison of all algorithms on gene data set (left) and scene data set (right) in terms of mean ROC scores.

is the case in many applications. Specifically, we use a gene data set with the dimensionality fixed at $d = 384$, while the size of the training set varies from 100 to 900 with a step size about 100.

The performance of different algorithms as the size of training set increases is presented in Figure 1 (left graph). We can observe that in general, the performance of all algorithms increases as the training size increases. When n is small, the condition in Theorem 1 holds, thus CCA and LS-CCA are equivalent, and they achieve the same performance. When n further increases, CCA and LS-CCA achieve different ROC scores, although the difference is always very small in our experiment. Similar to the last experiment, we can observe from the figure that the regularized methods perform much better than CCA and LS-CCA, and LS-CCA$_2$ is comparable to rCCA. The sparse formulation LS-CCA$_1$ performs the best for this data set.

5.3. Scene Data Set

We conduct a similar set of experiments on the scene data. As in the gene data set, the equivalence relationship holds when the condition in Theorem 1 holds.

For the performance comparison and sensitivity study, we generate a sequence of training sets with the size n ranging from 100 to 900 with a step size around 100. The results are summarized in Figure 1 (right graph). Similar to the gene data set, CCA and LS-CCA achieve the same performance when n is small, and they differ slightly when n is large. We can also observe from the figure that the regularized algorithms including rCCA, and LS-CCA$_2$, and LS-CCA$_1$ perform much better than CCA and LS-CCA without regularization, and LS-CCA$_2$ performs slightly better than others in this data set.

5.4. The Entire CCA Solution Path

In this experiment, we investigate the sparse CCA model, i.e., LS-CCA$_1$ using the scene data set. Recall that the sparseness of the weight vectors w_i's depends on the sparseness coefficient γ between 0 and 1.

Figure 2 shows the entire collection of solution paths for a subset of the coefficients from the first weight vector w_1. The x-axis denotes the sparseness coefficient γ, and the y-axis denotes the value of the coefficients. The vertical lines denote (a subset of) the turning point of the path, as the solution path for each of the coefficients is piecewise linear (Efron et al., 2004). We can observe from Figure 2 that when $\gamma = 1$, most of the coefficients are non-zero, i.e., the model is dense. When the value of the sparseness coefficient γ decreases (from the right to the left side along the x-axis), more and more coefficients become exactly zero. All coefficients become zero when $\gamma = 0$.

6. Conclusion and Future Work

In this paper we show that CCA for multi-label classifications can be formulated as a least squares problem under a mild condition, which tends to hold for high-dimensional data. Based on the equivalence relationship established in this paper, we propose several CCA extensions including sparse CCA. We have conducted experiments on a collection of multi-label data

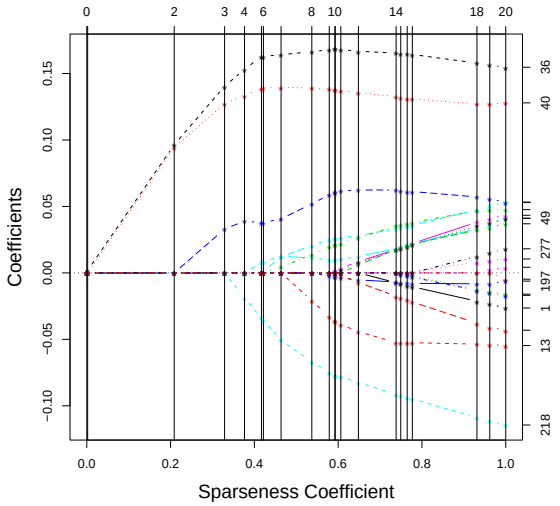


Figure 2. The entire collection of solution paths for a subset of the coefficients from the first weight vector w_1 on the scene data set. The x-axis denotes the sparseness coefficient γ, and the y-axis denotes the value of the coefficients.

sets to validate the proposed equivalence relationship. Our experimental results show that the performance of the proposed least squares formulation and CCA is very close even when the condition does not hold. Results also demonstrate the effectiveness of the proposed CCA extensions.

The proposed least squares formulation facilitates the incorporation of the unlabeled data into the CCA framework through the graph Laplacian, which captures the local geometry of the data (Belkin et al., 2006). We plan to examine the effectiveness of this semi-supervised CCA model for learning from both labeled and unlabeled data. The proposed sparse CCA performs well for the gene data set. We plan to analyze the biological relevance of the features extracted via the sparse CCA model.

Acknowledgments

This research is sponsored in part by funds from the Arizona State University and the National Science Foundation under Grant No. IIS-0612069.

References

Bach, F. R., & Jordan, M. I. (2003). Kernel independent component analysis. *J. Mach. Learn. Res.*, *3*, 1–48.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, *7*, 2399–2434.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*, 1757–1771.

d'Aspremont, A., Ghaoui, L., Jordan, M., & Lanckriet, G. (2004). A direct formulation for sparse PCA using semidefinite programming. *NIPS* (pp. 41–48).

Donoho, D. (2006). For most large underdetermined systems of linear equations, the minimal 11-norm near-solution approximates the sparsest near-solution. *Communications on Pure and Applied Mathematics*, *59*, 907–934.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification*. New York: John Wiley and Sons, Inc.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, *32*, 407.

Golub, G. H., & Loan, C. F. V. (1996). *Matrix computations*. Baltimore, MD: Johns Hopkins Press.

Hardoon, D. R., Szedmak, S. R., & Shawe-taylor, J. R. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Comput.*, *16*, 2639–2664.

Hastie, T., Buja, A., & Tibshirani, R. (1995). Penalized discriminant analysis. *Annals of Statistics*, *23*, 73–102.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: Data mining, inference, and prediction*. New York: Springer.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*, 55–67.

Hotelling, H. (1936). Relations between two sets of variables. *Biometrika*, *28*, 312–377.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Cambridge, MA: MIT Press.

Sriperumbudur, B. K., Torres, D. A., & Lanckriet, G. R. G. (2007). Sparse eigen methods by D.C. programming. *ICML* (pp. 831–838).

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. B*, *58*, 267–288.

Tomancak, P., & *et al.* (2002). Systematic determination of patterns of gene expression during *Drosophila* embryogenesis. *Genome Biology*, *3*.

Vert, J.-P., & Kanehisa, M. (2003). Graph-driven feature extraction from microarray data using diffusion kernels and kernel cca. *NIPS* (pp. 1425–1432).

Ye, J. (2007). Least squares linear discriminant analysis. *ICML* (pp. 1087–1094).

Yu, S., Yu, K., Tresp, V., & Kriegel, H.-P. (2006). Multi-output regularized feature projection. *IEEE Trans. Knowl. Data Eng.*, *18*, 1600–1613.

Zhu, J., Rosset, S., Hastie, T., & Tibshirani, R. (2003). 1-norm support vector machines. *NIPS* (pp. 49–56).

Apprenticeship Learning Using Linear Programming

Umar Syed USYED@CS.PRINCETON.EDU
Princeton University, Department of Computer Science, 35 Olden Street, Princeton, NJ 08540

Michael Bowling BOWLING@CS.UALBERTA.CA
University of Alberta, Department of Computing Science, Edmonton, Alberta, T6G 2E8 Canada

Robert E. Schapire SCHAPIRE@CS.PRINCETON.EDU
Princeton University, Department of Computer Science, 35 Olden Street, Princeton, NJ 08540

Abstract

In apprenticeship learning, the goal is to learn a policy in a Markov decision process that is at least as good as a policy demonstrated by an expert. The difficulty arises in that the MDP's true reward function is assumed to be unknown. We show how to frame apprenticeship learning as a linear programming problem, and show that using an off-the-shelf LP solver to solve this problem results in a substantial improvement in running time over existing methods — up to two orders of magnitude faster in our experiments. Additionally, our approach produces stationary policies, while all existing methods for apprenticeship learning output policies that are "mixed", i.e. randomized combinations of stationary policies. The technique used is general enough to convert any mixed policy to a stationary policy.

1. Introduction

In apprenticeship learning, as with policy learning for Markov decision processes (MDPs), the objective is to find a good policy for an autonomous agent, called the "apprentice", in a stochastic environment. While the setup of an apprenticeship learning problem is almost identical to that of policy learning in an MDP, there are a few key differences. In apprenticeship learning the true reward function is unknown to the apprentice, but is assumed to be a weighted combination of several known functions. The apprentice is also assumed to have access to demonstrations from another agent, called the "expert", executing a policy in the same environment. The goal of the apprentice is to find a policy that is at least as good as the expert's policy with respect to the true reward function. This is distinct from policy learning, where the goal is to find an optimal policy with respect to the true reward function (which cannot be done in this case because it is unknown).

The apprenticeship learning framework, introduced by Abbeel & Ng (2004), is motivated by a couple of observations about real applications. The first is that reward functions are often difficult to describe exactly, and yet at the same time it is usually easy to specify what the rewards must depend on. A typical example, investigated by Abbeel & Ng (2004), is driving a car. When a person drives a car, it is plausible that her behavior can be viewed as maximizing some reward function, and that this reward function depends on just a few key properties of each environment state: the speed of the car, the position of other cars, the terrain, etc. The second observation is that demonstrations of good policies by experts are often plentiful. This is certainly true in the car driving example, as it is in many other applications.

Abbeel & Ng (2004) assumed that the true reward function could be written as a linear combination of k known functions, and described an iterative algorithm that, given a small set of demonstrations of the expert policy, output an apprentice policy within $O(k \log k)$ iterations that was nearly as good as the expert's policy. Syed & Schapire (2008) gave an algorithm that achieved the same guarantee in $O(\log k)$ iterations. They also showed that by assuming that the linear combination is also a convex one, their algorithm can sometimes find an apprentice policy that is substantially better than the expert's policy. Essentially, the assumption of positive weights implies that the apprentice has some prior knowledge about which policies are better than others, and their algorithm leverages this knowledge.

Existing algorithms for apprenticeship learning share a couple of properties. One is that they each use an algorithm for finding an MDP's optimal policy (e.g. value iteration or policy iteration) as a subroutine. Another is that they output apprentice policies that are "mixtures", i.e. randomized combinations of stationary policies. A stationary policy is a function of just the current environment state.

Our first contribution in this paper is to show that, if one uses the linear programming approach for finding an

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

MDP's optimal policy (Puterman, 1994) as a subroutine, then one can modify Syed & Schapire's (2008) algorithm so that it outputs a stationary policy instead of a mixed policy. Stationary policies are desirable for a number of reasons, e.g. they are simpler to describe, and are more natural and intuitive in terms of the behavior that they prescribe. Moreover, this technique can be straightforwardly applied to any mixed policy, such as the ones output by Abbeel & Ng's (2004) algorithms, to convert it to a stationary policy that earns the same expected cumulative reward.

Our technique leads naturally to the second contribution of this paper, which is the formulation of the apprenticeship learning problem as a linear program. We prove that the solution to this LP corresponds to an apprentice policy that has the same performance guarantees as those produced by existing algorithms, and that the efficiency of modern LP solvers results in a very substantial improvement in running time compared to Syed & Schapire's (2008) algorithm — up to two orders of magnitude in our experiments.

In work closely related to apprenticeship learning, Ratliff, Bagnell & Zinkevich (2006) described an algorithm for learning the true reward function by assuming that the expert's policy is not very different from the optimal policy. They took this approach because they wanted to learn policies that were similar to the expert's policy. In apprenticeship learning, by contrast, the learned apprentice policy can be very different from the expert's policy.

2. Preliminaries

Formally, an *apprenticeship learning problem* $\left(\mathcal{S}, \mathcal{A}, \theta, \alpha, \gamma, R^1 \ldots R^k, \mathcal{D}\right)$ closely resembles a Markov decision process. At each time step t, an autonomous agent occupies a state s_t from a finite set $\mathcal{S}$, and can take an action a_t from a finite set $\mathcal{A}$. When the agent is in state s, taking action a leads to state s' with transition probability $\theta_{sas'} \triangleq \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$. Initial state probabilities are given by $\alpha_s \triangleq \Pr(s_0 = s)$. The agent decides which actions to take based on its policy π, where $\pi_{sa} \triangleq \Pr(a_t = a \mid s_t = s)$. The value of a policy π is given by

$$V(\pi) \triangleq E\left[\sum_{t=0}^{\infty} \gamma^t R_{s_t a_t} \;\middle|\; \alpha, \pi, \theta\right]$$

where $R_{sa} \in [-1, 1]$ is the reward associated with the state-action pair (s, a), and $\gamma \in [0, 1)$ is a discount factor. An optimal policy π^* is one that satisfies $\pi^* = \arg\max_\pi V(\pi)$. We say a policy π is *ϵ-optimal* if $V(\pi^*) - V(\pi) \leq \epsilon$.

A policy π has *occupancy measure* x^π if

$$x^\pi_{sa} = E\left[\sum_{t=0}^{\infty} \gamma^t \mathbf{1}_{(s_t = s \wedge a_t = a)} \;\middle|\; \alpha, \pi, \theta\right] \qquad (1)$$

for all s, a. In other words, x^π_{sa} is the expected (discounted) number of visits to state-action pair (s, a) when following policy π.

Unlike an MDP, in apprenticeship learning the true reward function R is unknown. Instead, we are given *basis reward functions*[1] $R^1 \ldots R^k$, where R^i_{sa} is the reward of state-action pair (s, a) with respect to the ith basis reward function. We assume that the true reward function R is an unknown convex combination w^* of the basis reward functions, i.e., for all s, a

$$R_{sa} = \sum_i w^*_i R^i_{sa}$$

where the unknown weights satisfy $w^*_i \geq 0$ and $\sum_i w^*_i = 1$. Each basis reward function R^i has a corresponding *basis value function* $V^i(\pi)$ given by

$$V^i(\pi) \triangleq E\left[\sum_{t=0}^{\infty} \gamma^t R^i_{s_t a_t} \;\middle|\; \alpha, \pi, \theta\right].$$

Given the assumption of positive weights, the value of k can be viewed as a measure of how much the apprentice knows about the true reward function. If $k = 1$, the (only) basis value of a policy is equal to its true value, and the situation reduces to a traditional MDP. At the other extreme, if the ith basis reward function is just an indicator function for the ith state-action pair, then $k = |\mathcal{SA}|$, and the basis values of a policy are equal to its occupancy measure. In this situation, the apprentice knows essentially nothing about which policies are better than others.

The positive weight assumption also implies that if for state-action pairs (s, a) and (s', a') we have $R^i_{sa} \geq R^i_{s'a'}$ for all i, then $R_{sa} \geq R_{s'a'}$. So the basis rewards themselves can encode prior knowledge about the true rewards. If we wish not to assert any such prior knowledge, we can simply add the negative of each basis reward function to the original set, thereby at most doubling the number of basis reward functions.

We also assume that we are given a data set $\mathcal{D}$ of M i.i.d. sample trajectories from an *expert policy* π^E executing in the environment, where the mth trajectory is a sequence of state-action pairs visited by the expert, i.e., $(s^m_0, a^m_0, s^m_1, a^m_1, \ldots, s^m_H, a^m_H)$. For simplicity, we assume that all sample trajectories are truncated to the same length H.

The goal of apprenticeship learning (Abbeel & Ng, 2004) is to find an *apprentice policy* π^A such that

$$V(\pi^A) \geq V(\pi^E) \qquad (2)$$

even though the true value function $V(\pi)$ is unknown (since the true reward function is unknown).

2.1. A More Refined Goal

By our assumptions about the reward functions (and the linearity of expectation), we have

$$V(\pi) = \sum_i w^*_i V^i(\pi).$$

[1] In (Abbeel & Ng, 2004) and (Syed & Schapire, 2008) these functions were called *features*, but we believe that the present terminology is better suited for conveying these functions' role.

Consequently, for any policy π, the smallest possible difference between $V(\pi)$ and $V(\pi^E)$ is $\min_i V^i(\pi) - V^i(\pi^E)$, because in the worst-case, $w_i^* = 1$ for the minimizing i. Based on this observation, Syed & Schapire (2008) proposed finding an apprentice policy π^A that solves the maximin objective

$$v^* = \max_{\pi} \min_i V^i(\pi) - V^i(\pi^E). \tag{3}$$

Note that if π^A is a solution to (3), then $V(\pi^A) \geq V(\pi^E) + v^*$ (because $v^* = \min_i V^i(\pi^A) - V^i(\pi^E) \leq V(\pi^A) - V(\pi^E)$). We also have $v^* \geq 0$ (because $\pi = \pi^E$ is available in (3)). Therefore π^A satisfies the goal of apprenticeship learning given in (2).

Syed & Schapire (2008) showed that in some cases where $V(\pi^E)$ is small, v^* is large, and so adding v^* to the lower bound in (2) serves as a kind of insurance against bad experts. Our algorithms also produce apprentice policies that achieve this more refined goal.

2.2. Estimating the Expert Policy's Values

Our algorithms require knowledge of the basis values of the expert's policy. From the expert's sample trajectories $\mathcal{D}$, we can form an estimate $\widehat{V}^{i,E}$ of $V^i(\pi^E)$ as follows:

$$V^i(\pi^E) \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{t=0}^{H} \gamma^t R^i_{s_t^m a_t^m} \triangleq \widehat{V}^{i,E}.$$

Clearly, as the number of sample trajectories M and the truncation length H increase, the error of this estimate will decrease. Thus the issue of accurately estimating $V^i(\pi^E)$ is related to *sample* complexity, while in this work we are primarily concerned with *computational* complexity. To make our presentation cleaner, we will assume that $\mathcal{D}$ is large enough to yield an estimate $\widehat{V}^{i,E}$ of $V^i(\pi^E)$ such that $|\widehat{V}^{i,E} - V^i(\pi^E)| \leq \epsilon$, for all i. We call such an estimate ϵ-*good*. The sample complexity of apprenticeship learning is treated in (Syed & Schapire, 2008).

2.3. Policy Types

Unless otherwise noted, a policy π is presumed to be stationary, i.e., π_{sa} is the probability of taking action a in state s. One exception is a *mixed policy*. A mixed policy $\tilde{\pi}$ is defined by a set of ordered pairs $\{(\pi^j, \lambda^j)\}_{j=1}^N$. The policy $\tilde{\pi}$ is followed by choosing at time 0 one of the stationary policies π^j, each with probability λ^j, and then following that policy exclusively thereafter. The value of a mixed policy is the expected value of the stationary policies it comprises, i.e.,

$$V(\tilde{\pi}) = E\left[V(\pi^j)\right] = \sum_{j=1}^{N} \lambda^j V(\pi^j), \text{ and}$$

$$V^i(\tilde{\pi}) = E\left[V^i(\pi^j)\right] = \sum_{j=1}^{N} \lambda^j V^i(\pi^j).$$

3. Multiplicative Weights Algorithm for Apprenticeship Learning

Syed & Schapire (2008) observed that solving the objective in (3) is equivalent to finding an optimal strategy in a certain two-player zero-sum game. Because the size of this game's matrix is exponential in the number of states $|\mathcal{S}|$, they adapted a multiplicative weights method for solving extremely large games. The resulting MWAL (Multiplicative Weights Apprenticeship Learning) algorithm is described in Algorithm 1 below.

Algorithm 1 MWAL algorithm

1: **Given:** $\mathcal{S}, \mathcal{A}, \theta, \alpha, \gamma, R^1 \ldots R^k, \mathcal{D}$.
2: Using the expert's sample trajectories $\mathcal{D}$, compute an ϵ-good estimate $\widehat{V}^{i,E}$ of $V^i(\pi^E)$, for all i.
3: Let $\beta = \left(1 + \sqrt{\frac{2 \log k}{T}}\right)^{-1} \in (0, 1]$.
4: Initialize $w_i^1 = \frac{1}{k}$, for $i = 1 \ldots k$.
5: **for** $t = 1 \ldots T$ **do**
6: Compute ϵ-optimal policy π^t for reward function $R_{sa} = \sum_i w_i^t R^i_{sa}$.
7: Compute ϵ-good estimate $\widehat{V}^{i,t}$ of $V^i(\pi^t)$, for $i = 1 \ldots k$.
8: Let $w_i^{t+1} = w_i^t \beta^{\widehat{V}^{i,t} - \widehat{V}^{i,E}}$, for $i = 1 \ldots k$.
9: Renormalize w.
10: **end for**
11: **Return:** Let apprentice policy π^A be the mixed policy defined by $\{(\pi^t, \frac{1}{T})\}_{t=1}^T$.

In each iteration of the MWAL algorithm, an optimal policy π^t is computed with respect to the current weight vector w^t. Then the weights are updated so that w_i is increased/decreased if π^t is a bad/good policy (relative to π^E) with respect to the ith basis reward function.

The next theorem bounds the number of iterations T required for the MWAL algorithm to produce a good apprentice policy. The computational complexity of each iteration is discussed in Section 3.1.

Theorem 1 (Syed & Schapire (2008)). *Let π^A be the mixed policy returned by the MWAL algorithm. If*

$$T \geq O\left(\frac{\log k}{(\epsilon(1-\gamma))^2}\right)$$

then

$$V(\pi^A) \geq V(\pi^E) + v^* - O(\epsilon)$$

where $v^* = \max_\pi \min_i V^i(\pi) - V^i(\pi^E)$.

3.1. MWAL-VI and MWAL-PI

The specification of the MWAL algorithm is somewhat open-ended. Step 6 requires finding an ϵ-optimal policy in an MDP, and Step 7 requires computing ϵ-good estimates of the basis values of that policy. There are several procedures available for accomplishing each of these steps, with each option leading to a different variant of the basic algorithm.

We briefly describe some natural options, and remark on their implications for the overall computational complexity of the MWAL algorithm.

In Step 6, we can find the optimal policy using value iteration (Puterman, 1994), which has a worst-case running time of $O\left(\log_\gamma\left(1/\epsilon(1-\gamma)\right)|\mathcal{S}|^2|\mathcal{A}|\right)$. We can also use value iteration to compute the k basis values in Step 7 (this is sometimes called "policy evaluation"), which implies a worst-case running time of $O\left(k\log_\gamma\left(1/\epsilon(1-\gamma)\right)|\mathcal{S}|^2\right)$. We call this variant the MWAL-VI algorithm.

Another choice for Step 6 is to find the optimal policy using policy iteration (Puterman, 1994). No polynomial time bound for policy iteration is known; however, in practice it has often been observed to be faster than value iteration. We call this variant the MWAL-PI algorithm. In Section 8, we present experiments comparing these algorithms to the ones described later in the paper.

4. Dual Methods for MDPs

As we previously observed, the MWAL algorithm must repeatedly find the optimal policy in an MDP, and this task is usually accomplished via classic iterative techniques such as value iteration and policy iteration. However, there are other techniques available for solving MDPs, and in this work we show that they can lead to better algorithms for apprenticeship learning. Consider the following linear program:

$$\max_x \sum_{s,a} R_{sa}x_{sa} \tag{4}$$

such that

$$\sum_a x_{sa} = \alpha_s + \gamma\sum_{s',a} x_{s'a}\theta_{s'as} \tag{5}$$

$$x_{sa} \geq 0 \tag{6}$$

It is well-known (Puterman, 1994) that if x^* is a solution to (4) - (6), then $\pi^*_{sa} = \frac{x^*_{sa}}{\sum_a x^*_{sa}}$ is an optimal policy, and x^* is the occupancy measure of π^*. Often (5) - (6) are called the *Bellman flow constraints*.

The linear program in (4) - (6) is actually the dual of the linear program that is typically used to find an optimal policy in an MDP. Accordingly, solving (4) - (6) is often called the *Dual LP* method of solving MDPs .

Having found an optimal policy by the Dual LP method, computing its values is straightforward. The next lemma follows immediately from the definitions of the occupancy measure and value of a policy.

Lemma 1. *If policy π has occupancy measure x^π, then $V(\pi) = \sum_{s,a} R_{sa}x^\pi_{sa}$ and $V^i(\pi) = \sum_{s,a} R^i_{sa}x^\pi_{sa}$.*

5. Main Theoretical Tools

Recall that the MWAL algorithm produces mixed policies. In Sections 6 and 7, we will present algorithms that achieve the same theoretical guarantees as the MWAL algorithm, but produce stationary policies (and are also faster). To prove the correctness of these algorithms, we need to show that every mixed policy has an equivalent stationary policy.

In Section 4, we said that the Dual LP method of solving an MDP outputs the occupancy measure of an optimal policy. In fact, *all* x that satisfy the Bellman flow constraints (5) - (6) are the occupancy measure of some stationary policy, as the next theorem shows.

Theorem 2. *Let x satisfy the Bellman flow constraints* (5) - (6)*, and let $\pi_{sa} = \frac{x_{sa}}{\sum_a x_{sa}}$ be a stationary policy. Then x is the occupancy measure for π. Conversely, if π is a stationary policy such that x is its occupancy measure, then $\pi_{sa} = \frac{x_{sa}}{\sum_a x_{sa}}$ and x satisfies the Bellman flow constraints.*

An equivalent result as Theorem 2 is given in (Feinberg & Schwartz, 2002), p. 178. For completeness, a simple and direct proof is contained in the Appendix.

The Bellman flow constraints make it very easy to show that, for every mixed policy, there is a stationary policy that has the same value.

Theorem 3. *Let $\tilde{\pi}$ be a mixed policy defined by $\{(\pi^j, \lambda^j)\}_{j=1}^N$, and let x^j be the occupancy measure of π^j, for all j. Let $\hat{\pi}$ be a stationary policy where*

$$\hat{\pi}_{sa} = \frac{\sum_j \lambda^j x^j_{sa}}{\sum_a \sum_j \lambda^j x^j_{sa}}.$$

Then $V(\hat{\pi}) = V(\tilde{\pi})$.

Proof. By Theorem 2, x^j satisfies the Bellman flow constraints (5) - (6) for all j. Let $\hat{x}_{sa} = \sum_j \lambda^j x^j_{sa}$. By linearity, $\hat{x}$ also satisfies the Bellman flow constraints. Hence, by Theorem 2, the stationary policy $\hat{\pi}$ defined by $\hat{\pi}_{sa} = \frac{\hat{x}_{sa}}{\sum_a \hat{x}_{sa}}$ has occupancy measure $\hat{x}$. Therefore,

$$V(\hat{\pi}) = \sum_{s,a} R_{sa}\hat{x}_{sa} = \sum_j \lambda^j \sum_{s,a} R_{sa}x^j_{sa} = \sum_j \lambda^j V(\pi^j)$$
$$= V(\tilde{\pi}).$$

where these equalities use, in order: Lemma 1; the definition of $\hat{x}$; Lemma 1; the definition of a mixed policy. □

6. MWAL-Dual Algorithm

In this section, we will make a minor modification to the MWAL algorithm so that it outputs a stationary policy instead of a mixed policy.

Recall that the MWAL algorithm requires, in Steps 6 and 7, a way to compute an optimal policy and its basis values, but that no particular methods are prescribed. Our proposal is to use the Dual LP method in Step 6 to find the occupancy measure x_t of a policy π_t that is ϵ-optimal for reward function $R_{sa} = \sum_i w^t_i R^i_{sa}$. Then in Step 7 we let

$\widehat{V}^{i,t} = \sum_{s,a} R^i_{sa} x^t_{sa}$, for $i = 1 \ldots k$. Note that Lemma 1 implies $\widehat{V}^{i,t} = V^i(\pi^t)$.

Now we can apply Theorem 3 to combine all the policies computed during the MWAL algorithm into a single stationary apprentice policy. This amounts to changing Step 11 to the following:

> **Return:** Let apprentice policy π^A be the stationary policy defined by
>
> $$\pi^A_{sa} = \frac{\frac{1}{T}\sum_t x^t_{sa}}{\sum_a \frac{1}{T}\sum_t x^t_{sa}}.$$

We call this modified algorithm the MWAL-Dual algorithm, after the method it uses to compute optimal policies.

It is straightforward to show that these changes to the MWAL algorithm do not affect its performance guarantee.

Theorem 4. *Let π^A be the stationary policy returned by the MWAL-Dual algorithm. If*

$$T \geq O\left(\frac{\log k}{(\epsilon(1-\gamma))^2}\right)$$

then

$$V(\pi^A) \geq V(\pi^E) + v^* - O(\epsilon)$$

where $v^* = \max_\pi \min_i V^i(\pi) - V^i(\pi^E)$.

Proof. By Theorem 3, the stationary policy returned by the MWAL-Dual algorithm has the same value as the mixed policy returned by the original MWAL algorithm. Hence the guarantee in Theorem 1 applies to the MWAL-Dual algorithm as well. □

Of course, the trick used here to convert a mixed policy to a stationary one is completely general, provided that the occupancy measures of the component policies can be computed. For example, this technique could be applied to the mixed policy output by the algorithms due to Abbeel & Ng (2004).

Let $T(n)$ be the worst-case running time of an LP solver on a problem with at most n constraints and variables.[2] For a typical LP solver, $T(n) = O(n^{3.5})$ (Shu-Cherng & Puthenpura, 1993), although they tend to be much faster in practice. Using this notation, we can bound the running time of Steps 6 and 7 in the MWAL-Dual algorithm. Finding an optimal policy using the Dual LP method takes $T(|\mathcal{S}||\mathcal{A}|)$ time. And by Lemma 1, given the occupancy measure of a policy, we can compute its basis values in $O(k|\mathcal{S}||\mathcal{A}|)$ time.

7. LPAL Algorithm

We now describe a way to use the Bellman flow constraints to find a good apprentice policy in a much more direct fashion than the MWAL algorithm. Recall the objective function proposed in (Syed & Schapire, 2008) for solving apprenticeship learning:

$$v^* = \max_\pi \min_i V^i(\pi) - V^i(\pi^E) \tag{7}$$

We observed earlier that, if π^A is a solution to (7), then $V(\pi^A) \geq V(\pi^E) + v^*$, and that $v^* \geq 0$. In this section, we describe a linear program that solves (7). In Section 8, we describe experiments that show that this approach is much faster than the MWAL algorithm, although it does have some disadvantages, which we also illustrate in Section 8.

Our LPAL (Linear Programming Apprenticeship Learning) algorithm is given in Algorithm 2. The basic idea is to use the Bellman flow constraints (5) - (6) and Lemma 1 to define a feasible set containing all (occupancy measures of) stationary policies whose basis values are above a certain lower bound, and then maximize this bound.

Algorithm 2 LPAL algorithm

1: **Given:** $\mathcal{S}, \mathcal{A}, \theta, \alpha, \gamma, R^1 \ldots R^k, \mathcal{D}$.
2: Using the expert's sample trajectories $\mathcal{D}$, compute an ϵ-good estimate $\widehat{V}^{i,E}$ of $V^i(\pi^E)$, for all i.
3: Find a solution (B^*, x^*) to this linear program:

$$\max_{B,x} \quad B \tag{8}$$

such that

$$B \leq \sum_{s,a} R^i_{sa} x_{sa} - \widehat{V}^{i,E} \tag{9}$$

$$\sum_a x_{sa} = \alpha_s + \gamma \sum_{s',a} x_{s'a}\theta_{s'as} \tag{10}$$

$$x_{sa} \geq 0 \tag{11}$$

4: **Return:** Let apprentice policy π^A be the stationary policy defined by

$$\pi^A_{sa} = \frac{x^*_{sa}}{\sum_a x^*_{sa}}.$$

Theorem 5. *Let π^A be the stationary policy returned by the LPAL algorithm. Then*

$$V(\pi^A) \geq V(\pi^E) + v^* - O(\epsilon)$$

where $v^* = \max_\pi \min_i V^i(\pi) - V^i(\pi^E)$.

Proof. By Theorem 2, the Bellman flow constraints (10) - (11) imply that all feasible x correspond to the occupancy measure of some stationary policy π. Using this fact and Lemma 1, we conclude that solving the linear program is equivalent to finding (B^*, π^A) such that

$$B^* = \min_i V^i(\pi^A) - \widehat{V}^{i,E}$$

and B^* is as large as possible. Since $|\widehat{V}^{i,E} - V^i(\pi^E)| \leq \epsilon$ for all i, we know that $B^* \geq v^* - \epsilon$. Together with (9) and

[2] Technically, the time complexity of a typical LP solver also depends on the number of bits in the problem representation.

Lemma 1 this implies

$$V^i(\pi^A) = \sum_{s,a} R^i_{sa} x^*_{sa} \geq \widehat{V}^{i,E} + B^* \geq V^i(\pi^E) + v^* - 2\epsilon.$$

□

Note that the *overall* worst-case running time of the LPAL algorithm is $T(|\mathcal{S}||\mathcal{A}| + k)$, where $T(n)$ is the complexity of an LP solver.

8. Experiments

8.1. Gridworld

We tested each algorithm in gridworld environments that closely resemble those in the experiments of Abbeel & Ng (2004). Each gridworld is an $N \times N$ square of states. Movement is possible in the four compass directions, and each action has a 30% chance of causing a transition to a random state. Each gridworld is partitioned into several square regions, each of size $M \times M$. We always choose M so that it evenly divides N, so that each gridworld has $k = (\frac{N}{M})^2$ regions. Each gridworld also has k basis reward functions, where the ith basis reward function R^i is a 0-1 indicator function for the ith region.

For each gridworld, in each trial, we randomly chose a sparse weight vector w^*. Recall that the true reward function has the form $R(s) = \sum_i w^*_i R^i(s)$, so in these experiments the true reward function just encodes that some regions are more desirable than others. In each trial, we let the expert policy π^E be the optimal policy with respect to R, and then supplied the basis values $V^i(\pi^E)$, for all i, to the MWAL-VI, MWAL-PI, MWAL-Dual and LPAL algorithms.[3]

Our experiments were run on an ordinary desktop computer. We used the Matlab-based `CVX` package (Grant & Boyd, 2008) for our LP solver. Each of the values in the tables below is the time, in seconds, that the algorithm took to find an apprentice policy π^A such that $V(\pi^A) \geq 0.95V(\pi^E)$. Each running time is the average of 10 trials.

Table 1. Time (sec) to find π^A s. t. $V(\pi^A) \geq 0.95V(\pi^E)$

Gridworld Size	MWAL-VI (sec)	MWAL-PI (sec)	MWAL-Dual (sec)	LPAL (sec)
16×16	6.43	5.78	46.99	1.46
24×24	14.45	10.27	90.16	1.55
32×32	27.23	15.04	247.38	2.76
48×48	61.37	35.33	791.61	8.62
64×64	114.12	85.26	3651.70	30.52
128×128	406.24	307.58	4952.74	80.21
256×256	1873.93	1469.56	29988.85	588.60

[3]Typically in practice, π_E will be unknown, and so the basis values would need to be estimated from the data set of expert sample trajectories $\mathcal{D}$. However, since we are primarily concerned with computational complexity in this work, and not sample complexity, we sidestep this issue and just compute each $V^i(\pi^E)$ directly.

Table 2. Time (sec) to find π^A s. t. $V(\pi^A) \geq 0.95V(\pi^E)$

Gridworld Size	Number of Regions	MWAL-VI (sec)	MWAL-PI (sec)	MWAL-Dual (sec)	LPAL (sec)
24×24	64	14.45	10.27	90.16	1.55
	144	32.33	20.06	97.58	2.64
	576	129.87	75.81	120.82	1.86
32×32	64	27.23	15.04	247.38	2.76
	256	107.11	60.24	270.71	8.43
	1024	440.64	267.12	361.36	4.75
48×48	64	61.37	35.33	791.61	8.62
	144	135.83	79.88	800.23	11.42
	256	244.46	150.08	815.66	16.89
	576	575.34	352.15	847.38	16.33
	2304	2320.71	1402.10	1128.32	11.14

In the first set of experiments (Table 1), we tested the algorithms in gridworlds of varying sizes, while keeping the number of regions in each gridworld fixed (64 regions). Recall that the number of regions is equal to the number of basis reward functions. In the next set of experiments (Table 2), we varied the number of regions while keeping the size of the gridworld fixed.

Several remarks about these results are in order. For every gridworld size and every number of regions, the LPAL algorithm is substantially faster than the other algorithms — in some cases two orders of magnitude faster. As we previously noted, LP solvers are often much more efficient than their theoretical guarantees. Interestingly, in Table 2, the running time for LPAL eventually decreases as the number of regions increases. This may be because the number of constraints in the linear program increases with the number of regions, and more constraints often make a linear program problem easier to solve.

Also, the MWAL-Dual algorithm is much slower than the other algorithms. We suspect this is only because the MWAL-Dual algorithm calls the LP solver in each iteration (unlike the LPAL algorithm, which calls it just once), and there is substantial overhead to doing this. Modifying MWAL-Dual so that it uses the LP solver as less of a black-box may be a way to alleviate this problem.

8.2. Car driving

In light of the results from the previous section, one might reasonably wonder whether there is any argument for using an algorithm other than LPAL. Recall that, in those experiments, the expert's policy was an optimal policy for the unknown reward function. In this section we explore the behavior of each algorithm when this is not the case, and find that MWAL produces better apprentice policies than LPAL. Our experiments were run in a car driving simulator modeled after the environments in (Abbeel & Ng, 2004) and (Syed & Schapire, 2008).

The task in our driving simulator is to navigate a car on a busy three-lane highway. The available actions are to move left, move right, drive faster, or drive slower. There are three basis reward functions that map each environment state to a numerical reward: collision (0 if contact with an-

other car, and 1/2 otherwise), off-road (0 if on the grass, and 1/2 otherwise), and speed (1/2, 3/4 and 1 for each of the three possible speeds, with higher values corresponding to higher speeds). The true reward function is assumed to be some unknown weighted combination w^* of the basis reward functions. Since the weights are assumed to be positive, by examining the basis reward functions we see that the true reward function assigns higher reward to states that are intuitively "better".

We designed three experts for these experiments, described in Table 3. Each expert is optimal for one of the basis reward functions, and mediocre for the other two. Therefore each expert policy π^E is an optimal policy if $w^* = w^E$, where w^E is the weight vector that places all weight on the basis reward function for which π^E is optimal. At the same time, each π^E is very likely to be suboptimal for a randomly chosen w^*.

We used the MWAL and LPAL algorithms to learn apprentice policies from each of these experts.[4] The results are presented in Table 4. We let $\gamma = 0.9$, so the maximum value of the basis value function corresponding to speed was 10, and for the others it was 5. Each of the reported policy values for randomly chosen w^* was averaged over 10,000 uniformly sampled w^*'s. Notice that for each expert, when w^* is chosen randomly, MWAL outputs better apprentice policies than LPAL.

Table 3. Expert types

	Speed	Collisions (per sec)	Off-roads (per sec)
"Fast" expert	Fast	1.1	10
"Avoid" expert	Slow	0	10
"Road" expert	Slow	1.1	0

Table 4. Driving simulator experiments.

Expert type	Algorithm used	$w^* = w^E$		w^* chosen randomly	
		$V(\pi^A)$	$V(\pi^E)$	$V(\pi^A)$	$V(\pi^E)$
"Fast"	MWAL	10	10	9.83	8.25
	LPAL	10	10	8.84	8.25
"Avoid"	MWAL	5	5	8.76	6.32
	LPAL	5	5	7.26	6.32
"Road"	MWAL	5	5	9.74	7.49
	LPAL	5	5	8.12	7.49

9. Conclusion and Future Work

Each of the algorithms for apprenticeship learning presented here have advantages and disadvantages that make them each better suited to different situations. As our experiments showed, the LPAL algorithm is much faster than any of the MWAL variants, and so is most appropriate for problems with large state spaces or many basis reward functions. And unlike the original MWAL algorithm, it produces a stationary policy, which make it a good choice whenever a simple and easily interpretable apprentice policy is desired. On the other hand, we also presented evidence that LPAL performs poorly when the expert policy is far from an optimal policy for the true reward function. If one suspects in advance that this may be the case, then one of the MWAL variants would be a better choice for a learning algorithm. Among these variants, only MWAL-Dual produces a stationary policy, although it has the drawback of being the slowest algorithm that we tested.

Although the theoretical performance guarantees of both the MWAL and LPAL algorithm are identical, the results in Table 4 suggest that the two algorithms are not equally effective. It seems possible that the current theoretical guarantees for the MWAL algorithm are not as strong as they could be. Investigation of this possibility is ongoing work.

One way to describe the poor performance of the LPAL algorithm versus MWAL is to say that, when there are several policies that are better than the expert's policy, the LPAL algorithm fails to optimally break these "ties". This characterization suggests that recent techniques for computing robust strategies in games (Johanson et al., 2008) may be an avenue for improving the LPAL algorithm.

It would also be interesting to examine practically and theoretically how apprenticeship learning can be combined with MDP approximation techniques. In particular, the dual linear programming approach in this work might combine nicely with recent work on stable MDP approximation techniques based on the dual form (Wang et al., 2008).

[4] Each of the MWAL variants behaved in exactly the same way in this experiment. The results presented are for the MWAL-PI variant.

Acknowledgements

We would like to thank Michael Littman, Warren Powell, Michele Sebag and the anonymous reviewers for their helpful comments. This work was supported by the NSF under grant IIS-0325500.

References

Abbeel, P., & Ng, A. (2004). Apprenticeship learning via inverse reinforcement learning. *Proceedings of the International Conference on Machine Learning*.

Feinberg, E. A., & Schwartz, A. (2002). *Handbook of Markov Decision Processes: Methods and Applications*. Springer.

Grant, M., & Boyd, S. (2008). CVX: Matlab software for disciplined convex programming (web page and software). http://stanford.edu/∼boyd/cvx.

Horn, R. A., & Johnson, C. R. (1985). *Matrix Analysis*. Cambridge University Press.

Johanson, M., Zinkevich, M., & Bowling, M. (2008). Computing robust counter-strategies. *Advances in Neural Information Processing Systems*.

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley and Sons.

Ratliff, N. D., Bagnell, J. A., & Zinkevich, M. A. (2006). Maximum margin planning. *Proceedings of the International Conference on Machine Learning*.

Shu-Cherng, & Puthenpura, S. (1993). *Linear Optimization and Extensions: Theory and Algorithms*. Prentice Hall.

Syed, U., & Schapire, R. E. (2008). A game-theoretic approach to apprenticeship learning. *Advances in Neural Information Processing Systems*.

Wang, T., Lizotte, D., Bowling, M., & Schuurmans, D. (2008). Stable dual dynamic programming. *Advances in Neural Information Processing Systems*.

10. Appendix

This is a proof of Theorem 2. Before proceeding, we introduce another linear system. For any stationary policy π, the *π-specific Bellman flow constraints* are given by the following linear system in which the x_{sa} variables are unknown:

$$x_{sa} = \pi_{sa}\alpha_s + \pi_{sa}\gamma \sum_{s',a'} x_{s'a'}\theta_{s'a's} \tag{12}$$

$$x_{sa} \geq 0 \tag{13}$$

The next lemma shows that π-specific Bellman flow constraints have a solution.

Lemma 2. *For any stationary policy π, the occupancy measure x^π of π satisfies the π-specific Bellman flow constraints* (12) - (13).

Proof. Clearly, x^π_{sa} is non-negative for all s, a, and so (13) is satisfied. As for (12), we simply plug in the definition of x^π_{sa} from (1). (In the following derivation, all the expectations and probabilities are conditioned on α, θ, and π. They have been omitted from the notation for brevity.)

$$\begin{aligned}
x^\pi_{sa} &= E\left[\sum_{t=0}^{\infty}\gamma^t \mathbf{1}_{(s_t=s\wedge a_t=a)}\right] \\
&= \sum_{t=0}^{\infty}\gamma^t \Pr(s_t=s, a_t=a) \\
&= \pi_{sa}\alpha_s + \sum_{t=0}^{\infty}\gamma^{t+1}\Pr(s_{t+1}=s, a_{t+1}=a) \\
&= \pi_{sa}\alpha_s \\
&\quad + \sum_{t=0}^{\infty}\gamma^{t+1}\sum_{s',a'}\Pr(s_t=s', a_t=a', s_{t+1}=s, a_{t+1}=a) \\
&= \pi_{sa}\alpha_s + \sum_{t=0}^{\infty}\gamma^{t+1}\sum_{s',a'}\Pr(s_t=s', a_t=a')\cdot\theta_{s'a's}\pi_{sa} \\
&= \pi_{sa}\alpha_s + \pi_{sa}\gamma\sum_{s',a'}E\left[\sum_{t=0}^{\infty}\gamma^t\mathbf{1}_{(s_t=s'\wedge a_t=a')}\right]\theta_{s'a's} \\
&= \pi_{sa}\alpha_s + \pi_{sa}\gamma\sum_{s',a'}x^\pi_{s'a'}\theta_{s'a's}
\end{aligned}$$

□

Now we show that the solution to the π-specific Bellman flow constraints given by Lemma 2 is unique.

Lemma 3. *For any stationary policy π, the π-specific Bellman flow constraints* (12) - (13) *have at most one solution.*

Proof. Define the matrix

$$A_{(sa,s'a')} \triangleq \begin{cases} 1-\gamma\theta_{s'a's}\pi_{sa} & \text{if } (s,a)=(s',a') \\ -\gamma\theta_{s'a's}\pi_{sa} & \text{otherwise.}\end{cases}$$

and the vector $b_{sa} \triangleq \pi_{sa}\alpha_s$. (Note that A and b are indexed by state-action pairs.) We can re-write (12) - (13) equivalently as

$$Ax = b \tag{14}$$

$$x \geq 0 \tag{15}$$

The matrix A is column-wise strictly diagonally dominant. This is because $\sum_{s'}\theta_{sas'} = 1$, $\sum_a \pi_{sa} = 1$ and $\gamma < 1$, so for all s', a'

$$\begin{aligned}
&\sum_{s,a}\gamma\theta_{s'a's}\pi_{sa} = \gamma < 1 \\
\Rightarrow\quad & 1-\gamma\theta_{s'a's'}\pi_{s'a'} > \sum_{(s,a)\neq(s',a')}\gamma\theta_{s'a's}\pi_{sa} \\
\Rightarrow\quad & |A_{(s'a',s'a')}| > \sum_{(s,a)\neq(s',a')}|A_{(sa,s'a')}|.
\end{aligned}$$

where the last line is the definition of column-wise strict diagonal dominance. This implies that A is non-singular (Horn & Johnson, 1985), so (14) - (15) has at most one solution. □

We are now ready to prove Theorem 2.

Proof of Theorem 2. For the first direction of the theorem, we assume that x satisfies the Bellman flow constraints (5) - (6), and that $\pi_{sa} = \dfrac{x_{sa}}{\sum_a x_{sa}}$. Therefore,

$$\pi_{sa} = \frac{x_{sa}}{\alpha_s + \gamma\sum_{s',a'}x_{s'a'}\theta_{s'a's}}. \tag{16}$$

Clearly x is a solution to the π-specific Bellman flow constraints (12) - (13), and Lemmas 2 and 3 imply that x is the occupancy measure of π.

For the other direction of the theorem, we assume that x is the occupancy measure of π. Lemmas 2 and 3 imply that x is the unique solution to the π-specific Bellman flow constraints (12) - (13). Therefore, π is given by (16). And since $\sum_a \pi_{sa} = 1$, we have

$$\frac{\sum_a x_{sa}}{\alpha_s + \gamma\sum_{s',a'}x_{s'a'}\theta_{s'a's}} = 1$$

which can be rearranged to show that x satisfies the Bellman flow constraints, and also combined with (16) to show that $\pi_{sa} = \dfrac{x_{sa}}{\sum_a x_{sa}}$. □

Composite Kernel Learning

Marie Szafranski [1] MARIE.SZAFRANSKI@HDS.UTC.FR
Yves Grandvalet [1,2] YVES.GRANDVALET@IDIAP.CH
Alain Rakotomamonjy [3] ALAIN.RAKOTOMAMONJY@INSA-ROUEN.FR

[1] Heudiasyc, UMR CNRS 6599, Université de Technologie de Compiègne, 60205 Compiègne cedex, France
[2] IDIAP Research Institute, Centre du Parc, P.O. Box 592, 1920 Martigny, Switzerland
[3] LITIS EA 4051, UFR de Sciences, Université de Rouen, 76800 Saint Etienne du Rouvray, France

Abstract

The Support Vector Machine (SVM) is an acknowledged powerful tool for building classifiers, but it lacks flexibility, in the sense that the kernel is chosen prior to learning. Multiple Kernel Learning (MKL) enables to learn the kernel, from an ensemble of basis kernels, whose combination is optimized in the learning process. Here, we propose Composite Kernel Learning to address the situation where distinct components give rise to a group structure among kernels. Our formulation of the learning problem encompasses several setups, putting more or less emphasis on the group structure. We characterize the convexity of the learning problem, and provide a general wrapper algorithm for computing solutions. Finally, we illustrate the behavior of our method on multi-channel data where groups correpond to channels.

1. Motivation

Kernel methods have been extensively used in learning problems (Schölkopf & Smola, 2001). In these models, the observations are implicitly mapped in a feature space via a mapping $\Phi : \mathcal{X} \to \mathcal{H}$, where $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) with reproducing kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

We address the problem of learning the kernel in Support Vector Machines (SVM) and related methods. Indeed, the kernel is crucial in many respects, and its relevance is essential to the success of kernel methods. Formally, the primary role of K is to define the evaluation functional in $\mathcal{H}$: $\forall f \in \mathcal{H},\ f(\boldsymbol{x}) = \langle f, K(\boldsymbol{x}, \cdot)\rangle_{\mathcal{H}}$, but K also defines (i) $\mathcal{H}$ itself, since $\forall f \in \mathcal{H},\ f(\boldsymbol{x}) = \sum_{i=1}^{\infty} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x})$; (ii) a metric, and hence a smoothness functional in $\mathcal{H}$: $\|f\|_{\mathcal{H}}^2 = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \alpha_i\alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j)$; (iii) a distance between observations: $\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|^2 = K(\boldsymbol{x}, \boldsymbol{x}) + K(\boldsymbol{x}', \boldsymbol{x}') - 2K(\boldsymbol{x}, \boldsymbol{x}')$.

In this paper, we devise Composite Kernel Learning (CKL), a framework where the kernel is learned in a way to favor the selection of variables or groups of variables. Section 2 motivates our approach while briefly reviewing the different means proposed to extend kernel methods beyond the predefined kernel setup. We then follow in Section 3 by considering some recent developments in variable selection that are relevant for our aims. Section 4 describes the CKL framework; the optimization algorithm is provided in Section 5, and experiments are reported in Section 6.

2. Flexible Kernel Methods

From now on, we restrict our discussion to classification, where, from a learning set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ of pairs of observations and label $(\boldsymbol{x}_i, y_i)$, one aims at building a decision rule that predicts the class label y of any observation $\boldsymbol{x}$. We furthermore focus on the binary case, where $(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \{\pm 1\}$. However, it should be kept in mind that most of our observations carry on to other settings, such as multiclass classification, clustering or regression with kernel methods.

2.1. Support Vector Machines

A SVM builds the decision rule $\operatorname{sign}\left(f^\star(\boldsymbol{x}) + b^\star\right)$, where $f^\star$ and $b^\star$ are defined as the solution of

$$\begin{cases} \min\limits_{f,b,\boldsymbol{\xi}} & \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum\limits_{i=1}^{n}\xi_i \\ \text{s. t.} & y_i\big(f(\boldsymbol{x}_i) + b\big) \geq 1 - \xi_i \quad 1 \leq i \leq n \\ & \xi_i \geq 0 \qquad\qquad\qquad\quad 1 \leq i \leq n\ . \end{cases} \tag{1}$$

The regularization parameter C is the only adjustable parameter in this procedure. This is usually not flexible

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

enough to provide good results when the kernel is chosen prior to seeing data. Hence, most applications of SVM incorporate a mechanism for learning the kernel.

2.2. Learning the Kernel

Cross-validation is the most rudimentary, but also the most common way to learn the kernel. It consists in (i) defining a family of kernels (*e.g.* Gaussian), indexed by one or more parameters (*e.g.* bandwidth), the so-called kernel hyper-parameters, (ii) running the SVM algorithm on each hyper-parameter setting, and (iii) finally choosing the hyper-parameter minimizing a cross-validation score.

A thorough discussion of the pros and cons of cross-validation is out of the scope of this paper, but it is clear that this approach is inherently limited to one or two hyper-parameters and few trial values. This observation led to several proposals allowing for more flexibility.

2.2.1. Filters, Wrappers & Embedded Methods

Learning the kernel amounts to learn the feature mapping. It should thus be of no surprise that the approaches investigated bear some similarities with the ones developed for variable selection, where one encounters filters, wrappers and embedded methods (Guyon & Elisseeff, 2003). Some general frameworks do not belong to a single category but the distinction is appropriate in most cases.

In filter approaches, the kernel is adjusted before building the SVM, with no explicit relationship to the objective value of Problem (1). For example, the kernel target alignment of Cristianini et al. (2002) adapts the kernel to the available data without training any classifier.

In wrapper algorithms, the SVM solver is the inner loop of two nested optimizers, whose outer loop is dedicated to adjust the kernel. This tuning may be guided by various generalization bounds (Cristianini et al., 1999; Weston et al., 2001; Chapelle et al., 2002).

Kernel learning can also be embedded in Problem (1), with the SVM objective value minimized jointly with respect to the SVM parameters and the kernel hyper-parameters (Grandvalet & Canu, 2003). Our approach, which belongs to this family of methods, is based on the Multiple Kernel Learning (MKL) framework (Lanckriet et al., 2004).

2.2.2. Multiple Kernel Learning

MKL is a joint optimization problem of the coefficients of the SVM classifier and a convex combination of kernels that defines the actual SVM kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{m=1}^{M} \sigma_m K_m(\boldsymbol{x}, \boldsymbol{x}') \ , \tag{2}$$

where each kernel K_m is associated to a RKHS $\mathcal{H}_m$ whose elements will be denoted f_m, and $\{\sigma_m\}_{m=1}^M$ are coefficients to be learned under the convex combination constraints

$$\sum_{m=1}^{M} \sigma_m = 1 \ , \ \ \sigma_m \geq 0 \ , 1 \leq m \leq M \ . \tag{3}$$

Bach et al. (2004) proposed the following formulation of MKL [1]:

$$\begin{cases} \min\limits_{\substack{f_1,\dots,f_M,\\ b,\boldsymbol{\xi}}} \frac{1}{2}\big(\sum\limits_m \|f_m\|_{\mathcal{H}_m}\big)^2 + C\sum\limits_i \xi_i \\ \quad \text{s. t.} \quad y_i\big(\sum\limits_m f_m(\boldsymbol{x}_i) + b\big) \geq 1 - \xi_i \quad 1 \leq i \leq n \\ \qquad\quad \xi_i \geq 0 \qquad\qquad\qquad\qquad 1 \leq i \leq n \,, \end{cases} \tag{4}$$

whose solution leads to a decision rule of the form $\operatorname{sign}\left(\sum_m f_m^\star(\boldsymbol{x}) + b^\star\right)$. This expression of the learning problem is remarkable in that it only deviates slightly from the original SVM problem (1). The squared RKHS norm in $\mathcal{H}$ is simply replaced by a mixed-norm, with the standard RKHS norm within each feature space $\mathcal{H}_m$, and an ℓ_1 norm in $\mathbb{R}^M$ on the vector built by concatenating these norms. This ℓ_1 norm encourages sparse solutions, that is, solutions where some functions f_m have zero norm. In this respect, the MKL problem may be seen as the kernelization of the group-LASSO (Yuan & Lin, 2006).

2.2.3. Composite Kernel Learning

When the individual kernels K_m represent a series, such as Gaussian kernels with different scale parameters, MKL may be used as an alternative to cross-validation. When the input data originates from M differents sources, and that each kernel is affiliated to one input variable, MKL can be used to select relevant input variables.

However, MKL is not meant to address problems where several kernels pertain to one input variable. In this situation, the sparseness mechanism of MKL does not favor solutions discarding all the kernels computed from an irrelevant input. Although most of the related coefficients should vanish in combination (2), spurious correlation may cause irrelevant input variables to participate to the solution.

The flat combination of kernels in MKL does not include a mechanism to cluster the kernels related to one input variable. In order to favor the selection of kernels within predefined groups, one has to define a group structure among kernels, which will guide the selection process through a structured kernel combination. This type of hierarchy among

[1] To lighten notations, the range of indexes is often omitted in summations, in which case: indexes i and j refer to examples and go from 1 to n; index m refers to kernels and goes from 1 to M; index ℓ refers to groups of kernels and goes from 1 to L.

variables has been investigated in linear models (Szafranski et al., 2008; Zhao et al., to appear). We briefly recapitulate the general framework in the following section, before discussing its adaptation to kernel learning in Section 4.

3. Grouped and Hierarchical Selection

The introduction of ℓ_1 penalties, with the seminal paper of Tibshirani (1996) on the LASSO, gave rise to many important theoretical and practical advances in the statistics and machine learning fields. As stated in Section 2.2.2, MKL itself belongs to the series of algorithms affiliated to the LASSO, through its relationship with group-LASSO. In this lineage, Zhao et al. (to appear) defined the very general Composite Absolute Penalties (CAP) family.

3.1. Composite Absolute Penalties

Consider a linear model with M parameters, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_M)^\mathsf{t}$, and let $I = \{1, \ldots, M\}$ be a set of index on these parameters. A group structure on the parameters is defined by a series of L subsets $\{G_\ell\}_{\ell=1}^L$, where $G_\ell \subseteq I$. Additionally, let $\{\gamma_\ell\}_{\ell=0}^L$ be $L+1$ norm parameters. Then, the member of the CAP family for the chosen groups and norm parameters is

$$\Omega = \sum_\ell \Big(\sum_{m \in G_\ell} |\beta_m|^{\gamma_\ell} \Big)^{\gamma_0/\gamma_\ell} \quad . \tag{5}$$

Mixed-norms correspond to groups defined as a partition of the set of variables. A CAP may also rely on nested groups, $G_1 \subset G_2 \subset \ldots \subset G_L$, and $\gamma_0 = 1$, in which case it favors what Zhao et al. call hierarchical selection, that is, the selection of groups of variables in the predefined order $\{I \setminus G_L\}, \{G_L \setminus G_{L-1}\}, \ldots, \{G_2 \setminus G_1\}, G_1$. This example is provided here to stress that Zhao et al.'s notion of hierarchy differs from the one that follows.

3.2. Hierarchical Penalization

Hierarchical penalization uses shrinking coefficients to transform a ridge-like penalty into a sparse penalizer (Szafranski et al., 2008). The model parameterized by $\boldsymbol{\beta}$ is fitted by minimizing a differentiable loss function $J(\cdot)$, subject to a ridge penalty with adaptive coefficients that encourages sparseness among and within groups:

$$\begin{cases} \min\limits_{\boldsymbol{\beta}, \boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2} J(\boldsymbol{\beta}) + \lambda \sum\limits_\ell \sum\limits_{m \in G_\ell} \frac{\beta_m^2}{\sqrt{\sigma_{1,\ell}\, \sigma_{2,m}}} \\ \text{s. t. } \sum\limits_\ell d_\ell\, \sigma_{1,\ell} = 1\,, \;\; \sigma_{1,\ell} \geq 0 \qquad 1 \leq \ell \leq L \\ \qquad \sum\limits_m \sigma_{2,m} = 1 \quad , \; \sigma_{2,m} \geq 0 \quad 1 \leq m \leq M\,. \end{cases} \tag{6}$$

The Lagrange parameter λ controls the amount of shrinkage, and d_ℓ is the size of group ℓ. The constraints expressed on the two last lines encourage sparseness in $\sigma_{1,\ell}$ and $\sigma_{2,m}$, which induces sparseness in β_m.

Here, the groups G_ℓ form a partition of I, and the hierarchy refers to the tree-structure of the shrinking coefficients: $\sigma_{2,m}$ shrinks parameter β_m, while $\sigma_{1,\ell}$ shrinks the parameters for group G_ℓ. In the words of Zhao et al., the objective here is grouped variable selection.

The minimizer of Problem (6) is the minimizer of

$$\min_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) + \lambda \Big(\sum_\ell d_\ell^{1/4} \Big(\sum_{m \in G_\ell} |\beta_m|^{4/3} \Big)^{3/4} \Big)^2 \quad ,$$

which is essentially a CAP estimate, where parameter d_ℓ only accounts for the group sizes (Szafranski et al., 2008). The inner $\ell_{4/3}$ norm and the outer ℓ_1 norm form a mixed-norm penalty that will be denoted $\ell_{(4/3,1)}$. The overall penalizer favors sparse solutions at the group level, with few leading coefficients within the selected groups.

4. From Multiple to Composite Kernels

MKL has been formalized as a quadratically constrained program by Lanckriet et al. (2004), then as a second-order cone program by Bach et al. (2004). More recently, other formulations led to wrapper algorithms, where the optimization with respect to kernel hyper-parameters is still based on the SVM objective value, but is performed in an outer loop that wraps a standard SVM solver. The outer loop is cutting planes for Sonnenburg et al. (2006), and gradient descent for Rakotomamonjy et al. (2007). Wrapper algorithms have appealing features: (i) they benefit from the developments of solvers specifically tailored for the SVM problem in the inner loop; (ii) they allow to address large-scale problems; (iii) they are multipurpose, since the SVM inner loop may be replaced by another algorithm with little or no adjustments.

We chose to build on gradient-based MKL. First, it has been shown to be more efficient than the SILP approach of Sonnenburg et al. (2006), thanks to the stability of the updates performed in the outer loop, which induces good initializations for the inner loop solver (Rakotomamonjy et al., 2007). Second, and even more important for our purpose, gradient-based MKL is amenable to the extension to groups of kernels, thanks to the formulation of hierarchical penalization of Section 3.2.

4.1. Variational Multiple Kernel Learning

Problem (4) is not differentiable at $\|f_m\|_{\mathcal{H}_m} = 0$, a difficulty that causes a considerable algorithmic burden. The MKL formulation of Rakotomamonjy et al. (2007) can be viewed as a variational form of Problem (4), where M new variables $\sigma_1, \ldots, \sigma_M$ are introduced in order to avoid

these differentiability issues. The resulting problem, which is equivalent to Problem (4), is stated as:

$$\left\{\begin{array}{ll}\min\limits_{\substack{f_1,\dots,f_M,\\ b,\boldsymbol{\xi},\boldsymbol{\sigma}}} & \frac{1}{2}\sum\limits_m \frac{1}{\sigma_m}\|f_m\|^2_{\mathcal{H}_m} + C\sum\limits_i \xi_i \\ \text{s. t.} & y_i\big(\sum\limits_m f_m(\boldsymbol{x}_i)+b\big) \geq 1-\xi_i \quad 1\leq i\leq n \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad 1\leq i\leq n \\ & \sum\limits_m \sigma_m = 1 \quad ,\ \sigma_m \geq 0 \qquad 1\leq m\leq M\,. \end{array}\right. \tag{7}$$

Here and in what follows, u/v is defined by continuation at zero as $u/0=\infty$ if $u\neq 0$ and $0/0=0$.

The constraints expressed on the last line encourage sparseness in σ_m, which induces sparseness in f_m. As already mentioned in Section 2.2.2, the sparseness applies at the kernel level, ignoring the group structure. The latter is taken into account in the formulation proposed in the following section.

4.2. Variational Composite Kernel Learning

Here, we build on the variational form of the composite absolute penalties presented in Section 3.2 to take into account the group structure. Hierarchical penalization can deal with kernel methods if the ridge penalties are replaced by RKHS norms. We first generalize Problem (6) to obtain smooth variational formulations for arbitrary mixed-norm penalties, so that to address a wide variety of problems including MKL:

$$\left\{\begin{array}{ll}\min\limits_{\substack{f_1,\dots,f_M,\\ b,\boldsymbol{\xi},\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}} & \frac{1}{2}\sum\limits_\ell \sigma_{1,\ell}^{-p}\sum\limits_{m\in G_\ell}\sigma_{2,m}^{-q}\|f_m\|^2_{\mathcal{H}_m} + C\sum\limits_i \xi_i \\ \text{s. t.} & y_i\big(\sum\limits_m f_m(\boldsymbol{x}_i)+b\big) \geq 1-\xi_i \quad 1\leq i\leq n \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad 1\leq i\leq n \\ & \sum\limits_\ell d_\ell\,\sigma_{1,\ell} = 1 \quad ,\ \sigma_{1,\ell}\geq 0 \quad 1\leq \ell\leq L \\ & \sum\limits_m \sigma_{2,m} = 1 \quad ,\ \sigma_{2,m}\geq 0 \quad 1\leq m\leq M, \end{array}\right. \tag{8}$$

where p and q are exponents to be set according to the problem at hand.

This formulation, which is difficult to optimize, is simplified by replacing the two shrinking coefficients $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ by $\boldsymbol{\sigma}$, defined by $\sigma_m = \sigma_{1,\ell}^p\sigma_{2,m}^q$. In a first step, we consider the change of variable that maps $\boldsymbol{\sigma}_2$ to $\boldsymbol{\sigma}$. When $q\neq 0$, this mapping is one-to-one provided $\sigma_{1,\ell}\neq 0$. Furthermore, if $\sigma^\star_{1,\ell}$ and $\sigma^\star_{2,m}$ denote the optimal $\sigma_{1,\ell}$ and $\sigma_{2,m}$ values for Problem (8), we have that $\sigma^\star_{1,\ell}=0 \Rightarrow \sigma^\star_{2,m}=0$, hence Problem (8) is equivalent to

$$\left\{\begin{array}{ll}\min\limits_{\substack{f_1,\dots,f_M,\\ b,\boldsymbol{\xi},\boldsymbol{\sigma}_1,\boldsymbol{\sigma}}} & \frac{1}{2}\sum\limits_m \frac{1}{\sigma_m}\|f_m\|^2_{\mathcal{H}_m} + C\sum\limits_i \xi_i \\ \text{s. t.} & y_i\big(\sum\limits_m f_m(\boldsymbol{x}_i)+b\big) \geq 1-\xi_i \quad 1\leq i\leq n \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad 1\leq i\leq n \\ & \sum\limits_\ell d_\ell\,\sigma_{1,\ell} = 1 \quad ,\ \sigma_{1,\ell}\geq 0 \quad 1\leq \ell\leq L \\ & \sum\limits_\ell \sigma_{1,\ell}^{-p/q}\sum\limits_{m\in G_\ell}\sigma_m^{1/q} \leq 1 \\ & \sigma_m \geq 0 \qquad\qquad\qquad\qquad 1\leq m\leq M\,. \end{array}\right. \tag{9}$$

The new problem is simplified further by showing that $\boldsymbol{\sigma}_1$ can be dropped out from the optimization process, leading to the following formulation of Composite Kernel Learning (CKL):

$$\left\{\begin{array}{ll}\min\limits_{\substack{f_1,\dots,f_M,\\ b,\boldsymbol{\xi},\boldsymbol{\sigma}}} & \frac{1}{2}\sum\limits_m \frac{1}{\sigma_m}\|f_m\|^2_{\mathcal{H}_m} + C\sum\limits_i \xi_i \\ \text{s. t.} & y_i\big(\sum\limits_m f_m(\boldsymbol{x}_i)+b\big) \geq 1-\xi_i \quad 1\leq i\leq n \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad 1\leq i\leq n \\ & \sum\limits_\ell \Big(d_\ell^p\Big(\sum\limits_{m\in G_\ell}\sigma_m^{1/q}\Big)^q\Big)^{1/(p+q)} \leq 1 \\ & \sigma_m \geq 0 \qquad\qquad\qquad\qquad 1\leq m\leq M\,, \end{array}\right. \tag{10}$$

Before considering particular settings of interest, we state below two helpful propositions. The first one gives a more interpretable formulation of Problem (10); the second one presents the conditions for convexity of formulation (10), that will guaranty the convergence towards the global minimum for the algorithm described in Section 5.

Proposition 1. ***CAP Formulation:*** *Problem (10) is equivalent to the following MKL problem with a CAP-like penalty on the RKHS norms:*

$$\left\{\begin{array}{ll}\min\limits_{\substack{f_1,\dots,f_M,\\ b,\boldsymbol{\xi}}} & \frac{1}{2}\Big(\sum\limits_\ell d_\ell^{\gamma^*}\big(\sum\limits_{m\in G_\ell}\|f_m\|^\gamma_{\mathcal{H}_m}\big)^{\gamma_0/\gamma}\Big)^{2/\gamma_0} + C\sum\limits_i \xi_i \\ \text{s. t.} & y_i\big(\sum\limits_m f_m(\boldsymbol{x}_i)+b\big) \geq 1-\xi_i \quad 1\leq i\leq n \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad 1\leq i\leq n, \end{array}\right. \tag{11}$$

with $\gamma=\frac{2}{q+1}$, $\gamma_0=\frac{2}{p+q+1}$ *and* $\gamma^*=1-\frac{\gamma_0}{\gamma}$.

Sketch of proof. Let $\mathcal{L}$ be the Lagrangian of problem (10). The optimality conditions for σ_m are obtained from the first order optimality conditions for σ_m ($\frac{\partial\mathcal{L}}{\partial\sigma_m}=0$):

$$\sigma_m = \Big(\sum_\ell d_\ell^{\gamma^*} s_\ell^{\gamma_0/\gamma}\Big)^{(\gamma_0-2)/\gamma_0} d_\ell^{-\gamma^*} s_\ell^{\gamma^*}\|f_m\|^{2-\gamma}_{\mathcal{H}_m}\,, \tag{12}$$

where $s_\ell = \sum\limits_{m\in G_\ell}\|f_m\|^\gamma_{\mathcal{H}_m}$. Plugging this expression in Problem (10) yields the claimed result. □

Note that the outer exponent $\frac{2}{\gamma_0}$ only influences the strength of the penalty, not its type. Hence, the penalty in the objective function (11) differs from (5) in the RKHS norms $\|\cdot\|_{\mathcal{H}_m}$ and in the parameters d_ℓ that accommodate for group sizes.

Proposition 2. ***Conditions for Convexity:*** *Problem (10) is convex if and only if* $0 \leq q \leq 1$ *and* $0 \leq p+q \leq 1$.

Proof. A problem minimizing a convex criterion on a convex set is convex. The objective function of Problem (10) is convex (Boyd & Vandenberghe, 2004, p. 89). The first, second and fourth constraints define convex sets, and the third one also provided (i) $\left(\sum_{m\in G_\ell} \sigma_m^{1/q}\right)^q$ is a norm, that is $0 \leq q \leq 1$, and (ii) $\sum_\ell t_\ell^{1/(p+q)}$ is convex in t_ℓ, that is $0 \leq p+q \leq 1$. □

Within the values of p and q ensuring convexity, we pick the following particular cases of interest:

- $p = 0$, $q = 1$ yields a LASSO type penalty on the RKHS norms. It results in the generalization of the group-LASSO known as MKL, as formulated in (4);
- $p = 1$, $q = 0$ yields a group-LASSO type penalty on the RKHS norms. It results in another MKL, with L effective kernels $\overline{K}_\ell$, defined as $\overline{K}_\ell = \sum_{m\in G_\ell} K_m$;
- $p = q = \frac{1}{2}$ yields a hierarchical-penalization type penalty on the RKHS norms. It is a true CKL, where there are M effective kernels, and where the penalty favors sparse solutions at the group level, with few leading kernels within the selected groups.

Hence, when p goes from zero to one, with $q = 1-p$, the penalty gives more and more emphasis to the group structure. For most applications where convexity is a key issue, we recommend the balanced setup $p = q = \frac{1}{2}$.

Note however that convex penalties restrict the sparseness of the solution to either the group level or the kernel level. In Section 6, we will illustrate that giving up convexity may turn out to be an interesting option when considering interpretability issues.

5. Algorithm

Our approach to solve Problem (10) draws on the MKL algorithm of Rakotomamonjy et al. (2007). We use the wrapper scheme described below, where the outer loop is carried out by a projected gradient descent update.

5.1. A Gradient-Based Wrapper

The wrapper scheme considers the following constrained optimization problem:

$$\begin{cases} \min\limits_{\boldsymbol{\sigma}} J(\boldsymbol{\sigma}) \\ \text{s. t.} \sum\limits_\ell \left(d_\ell^p \left(\sum\limits_m \sigma_m^{1/q} \right)^q \right)^{1/(p+q)} & \leq 1 \\ \quad \sigma_m \geq 0, & 1 \leq m \leq M\,, \end{cases}$$

where $J(\boldsymbol{\sigma})$ is defined as the objective value of

$$\begin{cases} \min\limits_{f_1,\ldots,f_M,\, b,\boldsymbol{\xi}} \frac{1}{2}\sum\limits_\ell \sum\limits_{m\in G_\ell} \frac{1}{\sigma_m}\|f_m\|_{\mathcal{H}_m}^2 + C\sum\limits_i \xi_i \\ \text{s. t.} \;\; y_i\big(\sum\limits_m f_m(\boldsymbol{x}_i) + b\big) \geq 1-\xi_i\,,\; 1 \leq i \leq n \\ \qquad \xi_i \geq 0\,,\; 1 \leq i \leq n\;. \end{cases} \tag{13}$$

The global optimization problem consists thus of two nested problems. In the inner loop, the criterion is optimized with respect to $f_1, \ldots, f_M$, b and $\boldsymbol{\xi}$, considering that the coefficients $\boldsymbol{\sigma}$ are fixed. In the outer loop, $\boldsymbol{\sigma}$ is updated to decrease the criterion, with f_m, b and $\boldsymbol{\xi}$ being fixed.

Equation (12) may be used to update $\boldsymbol{\sigma}$ in closed form. However, this approach lacks convergence guarantees and may lead to numerical problems, in particular when some elements of $\boldsymbol{\sigma}$ approach zero. Hence, following Rakotomamonjy et al. (2007), we use that the objective function $J(\boldsymbol{\sigma})$ is actually an optimal SVM objective value to update $\boldsymbol{\sigma}$ by an efficient projected gradient descent scheme.

5.2. Computing the Gradient

The dual formulation offers a convenient means to compute the gradient $\nabla J(\boldsymbol{\sigma})$. The derivation of the Lagrangian of Problem (13), which is omitted here for brevity, shows that its dual formulation is identical to the one of a standard SVM using the aggregated kernel $\overline{K}_{\boldsymbol{\sigma}}$ defined in Equation (2). Hence, the dual problem takes the usual form

$$\begin{cases} \max\limits_{\alpha} -\frac{1}{2}\sum\limits_{i,j} \alpha_i\alpha_j y_i y_j \overline{K}_{\boldsymbol{\sigma}}(\boldsymbol{x}_i,\boldsymbol{x}_j) + \sum\limits_i \alpha_i \\ \text{s. t.} \sum\limits_i \alpha_i y_i = 0 \\ \quad C \geq \alpha_i \geq 0 \qquad\qquad 1 \leq i \leq n\;, \end{cases} \tag{14}$$

which can be solved by any SVM solver.

As $J(\boldsymbol{\sigma})$ is defined as the optimal objective value of the convex Problem (13) for which strong duality applies, $J(\boldsymbol{\sigma})$ is also the dual objective value:

$$J(\boldsymbol{\sigma}) = -\frac{1}{2}\sum_{i,j} \alpha_i^\star\alpha_j^\star y_i y_j \overline{K}_{\boldsymbol{\sigma}}(\boldsymbol{x}_i,\boldsymbol{x}_j) + \sum_i \alpha_i^\star\;, \tag{15}$$

where $\alpha^\star$ solves Problem (14).

The existence and computation of the derivatives of $J(\cdot)$ follow from general results on optimal values, such as Theorem 4.1 of Bonnans and Shapiro (1998), which, in a nutshell states that the differentiability of $J(\boldsymbol{\sigma})$ is ensured by the unicity of $\alpha^\star$, and by the differentiability of (15). [2] Furthermore, the derivatives of $J(\boldsymbol{\sigma})$ can be computed as if $\alpha^\star$ were not to depend on $\boldsymbol{\sigma}$. Thus, the gradient $\nabla J(\boldsymbol{\sigma})$ is simply

$$\frac{\partial J}{\partial \sigma_m} = -\frac{1}{2} \sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j \sum_\ell \sum_{m \in G_\ell} K_m(\boldsymbol{x}_i, \boldsymbol{x}_j) \ .$$

5.3. CKL Algorithm

Now, we have all the ingredients to adapt the machinery developed for MKL by Rakotomamonjy et al. (2007). According to the process described in Section 5.1, we propose Algorithm 1.

Algorithm 1 Composite Kernel Learning

initialize $\boldsymbol{\sigma}$
solve the SVM problem $\rightarrow J(\boldsymbol{\sigma})$
repeat
  compute direction $d = -\nabla J(\boldsymbol{\sigma})$
  repeat
    compute d', the projection of d onto the tangent of the surface of the admissible set
    compute the smallest step that nullifies a component of $\boldsymbol{\sigma}$
      $\mathcal{S} = \left\{ j : d'_j < 0 \text{ and } \sigma_j \neq 0 \right\}$
      $\nu = \min_{j \in \mathcal{S}} -\frac{\sigma_j}{d'_j} \quad k = \arg\min_{j \in \mathcal{S}} -\frac{\sigma_j}{d'_j} \quad d_k = 0$
    $\boldsymbol{\sigma}^\dagger = \boldsymbol{\sigma} + \nu \, d'$
    project $\boldsymbol{\sigma}^\dagger$ onto the surface of the admissible set
    solve the SVM problem $\rightarrow J(\boldsymbol{\sigma}^\dagger)$
    if $J(\boldsymbol{\sigma}^\dagger) < J(\boldsymbol{\sigma})$ **then** $\boldsymbol{\sigma} = \boldsymbol{\sigma}^\dagger$
  until $J(\boldsymbol{\sigma}^\dagger) \geq J(\boldsymbol{\sigma})$
  compute $\nu^\star = \arg\min_\nu J(\boldsymbol{\sigma} + \nu \, d)$
  $\boldsymbol{\sigma} = \boldsymbol{\sigma} + \nu^\star \, d$
until convergence

The stopping criterion for assessing the convergence of the outer loop can be based on standard criteria for gradient-based algorithms or on the duality gap. In the following experiments, it is based on the stability of $\boldsymbol{\sigma}$ and $J(\boldsymbol{\sigma})$.

6. Channel Selection for BCI

This experiment deals with single trial classification of EEG signals coming from Brain-Computer Interface (BCI). Depending on each BCI paradigm, these EEG signals are recorded from specific electrode positions. However, as stated by Schröder et al. (2005), automated channel selection should be performed for each single subject since it leads to better performances or a substantial reduction of the number of useful channels. Reducing the number of channels involved in the decision function is of primary importance for BCI real-life applications, since it makes the acquisition system easier to use and to set-up.

We use here the dataset from the BCI 2003 competition for the task of interfacing the P300 Speller (Blankertz et al., 2004). The dataset consists in 7560 EEG signals paired with positive or negative stimuli responses. The signal, processed as in (Rakotomamonjy et al., 2005), leads to 7560 examples of dimension 896 (14 time frames for each of the 64 channels).

The experimental protocol is then the following: we have randomly picked 567 training examples from the datasets and used the remaining as testing examples. For each parameter, C has been selected by retaining a small part of the training set as a validation set, for selecting the parameter which the highest AUC. This overall procedure has been repeated 10 times. Using a small part of the examples for training can be justified by the use of ensemble of SVMs (that we do not consider here) on a latter stage of the EEG classification procedure (Rakotomamonjy et al., 2005), and the AUC performance measure is justified by how the EEG recognition is transformed into selected character in the P300.

The 896 features extracted from the EEG signals are not tranformed before classification: we do not use any kernelization. However, to unify the presentation, we will refer to these features as linear kernels. Hence, in this application where the kernels related to a given channel form a group of kernels, we have to learn $M = 896$ coefficients σ_m, divided into $L = 64$ groups.

CKL is well-suited to the classification objectives, since we aim at classifying the EEG trials with as few channels as possible. Furthermore, it is also likely that some time frames are irrelevant, so that variable selection may be carried out within each channel. To reach a sparse solution at the channel and the time frame levels, we test a non-convex parametrization of CKL that encourages sparseness within and between groups.

In the following, $\text{CKL}_{1/2}$ stands for a convex version of our algorithm, with $p = q = 1/2$ (a $\ell_{(4/3,1)}$ mixed-norm), CKL_1 is a non-convex version, with $p = q = 1$ (a $\ell_{(1,2/3)}$ (pseudo) mixed-norm). Note that MKL is also implemented by our algorithm, with $p = 0$ and $q = 1$.

Table 1 summarizes the average performance of SVM, MKL, and CKL, that is, for 4 different penalization terms: quadratic penalization for the classical SVM (which is

[2] The unicity of $\alpha^\star$ is ensured provided that the Gram matrix built from kernel $K_{\boldsymbol{\sigma}}$ is positive-definite. To enforce this property, a small ridge may be added to the diagonal.

trained with the mean of 896 kernels), ℓ_1 norm for MKL, and mixed-norms for the two versions of CKL: $\text{CKL}_{1/2}$ and CKL_1. The number of channels and kernels selected by these algorithms is also reported.

Table 1. Average Results for SVMs with 4 different penalization terms on the BCI datasets.

Algorithms	AUC	# Channels	# Kernels
SVM	83.87 ± 0.8	64	896
MKL	85.43 ± 0.9	62.2 ± 1	255.8 ± 15
$\text{CKL}_{1/2}$	85.49 ± 1.1	62.9 ± 1	835.7 ± 25
CKL_1	84.15 ± 0.8	24.0 ± 4	60.9 ± 10

The prediction performances of the 4 algorithms are similar, with a slight advantage for sparse methods. $\text{CKL}_{1/2}$ is much less sparse than MKL, which itself keeps about four times as much kernels compared to CKL_1. In the number of groups, MKL and $\text{CKL}_{1/2}$ behave similarly, with only one or two channels removed. CKL_1 is much sparser and removes about two thirds of the channels.

Figure 6 represents the median relevance of the electrodes over the 10 experiments. It displays which electrodes have been selected by the different kernel learning methods. For one experiment, the relevance for channel ℓ is computed by the relative contribution of group ℓ to the norm of the solution, that is

$$\frac{1}{Z} \sum_{m \in G_\ell} \frac{1}{\sigma_m^\star} \|f_m^\star\|_{\mathcal{H}_m}^2 \quad ,$$

where Z is a normalization factor that sets the sum of relevances to one.

The results for CKL_1 are particularly neat, with high relevances for the electrodes in the areas of the visual cortex (especially the lateral electrodes PO_7 and PO_8), and the primary motor and Somatosensory cortex ($\text{C}_\bullet$ and CP_Z). The scalp maps for MKL and $\text{CKL}_{1/2}$ are very similar and show the importance of the same regions. In addition they also highlight numerous frontal electrodes that are not likely to be relevant for the BCI P300 Speller paradigm.

7. Conclusion and Further Works

This paper is at the crossroad of kernel learning and variable selection. From the former viewpoint, we extended the multiple kernel learning problem to take into account the group structure among kernels. From the latter viewpoint, we generalized the hierarchical penalization framework to kernel classifiers by considering penalties in RKHS instead of parametric function spaces.

As a side contribution, we also provide a smooth variational formulation for arbitrary mixed-norm penalties, enabling to tackle a wide variety of problems. This formulation is not restricted to convex mixed-norm, a property that turns out to be of interest for reaching sparser, hence more interpretable solutions.

Our approach is embedded, in the sense that the kernel hyper-parameters are optimized jointly with the parameters of classifier to minimize the soft-margin criterion. It is however implemented by a simple wrapper algorithm, for which the inner and the outer subproblems have the same objective function, and where the inner loop is a standard SVM problem.

In particular, this implementation allows to use available solvers for kernel machines in the inner loop. Hence, although this paper considered binary classification problems, our approach can be readily extended to other learning problems, such as multiclass classification, clustering, regression or ranking.

Acknowledgments

This work was supported in part by the IST Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. The authors would like to thank N. Bourdaud and G. Garipelli for their help in the interpretation of the BCI experiments.

References

Bach, F. R., Lanckriet, G. R. G., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. *Proceedings of the twenty-first international conference on Machine Learning* (pp. 41–48).

Blankertz, B., Müller, K.-R., Curio, G., Vaughan, T. M., Schalk, G., Wolpaw, J. R., Schlögl, A., Neuper, C., Pfurtscheller, G., Hinterberger, T., Schröder, M., & Birbaumer, N. (2004). The BCI competition 2003: progress and perspectives in detection and discrimination of EEG single trials. *IEEE Trans. Biomed. Eng*, *51*, 1044–1051.

Bonnans, J., & Shapiro, A. (1998). Optimization problems with pertubation: A guided tour. *SIAM Review*, *40*, 228–264.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*, 131–159.

Cristianini, N., Campbell, C., & Shawe-Taylor, J. (1999). Dynamically adapting kernels in support vector machines. *Advances in Neural Information Processing Systems 11* (pp. 204–210). MIT Press.

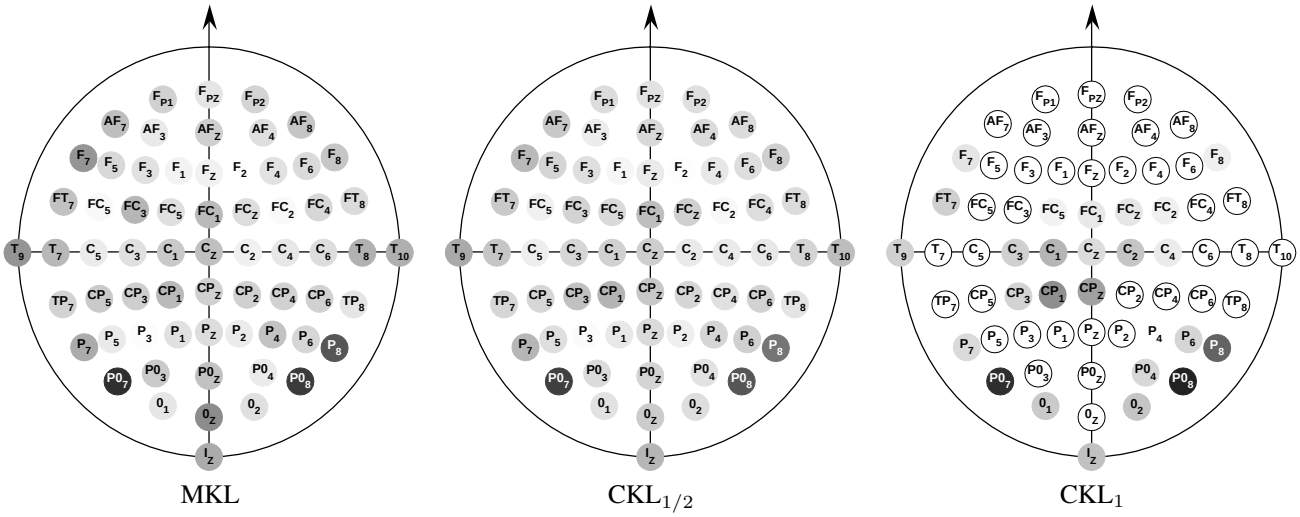


Figure 1. Electrode median relevance for MKL (left), CKL$_{1/2}$ (center) and CKL$_1$ (right). The darker the color, the higher the relevance. Electrodes in white with a black circle are discarded (the relevance is exactly zero). The arrow represents the frontal direction.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, K. (2002). On kernel-target alignment. *Advances in Neural Information Processing Systems 14* (pp. 367–373). MIT Press.

Grandvalet, Y., & Canu, S. (2003). Adaptive scaling for feature selection in SVMs. *Advances in Neural Information Processing Systems 15* (pp. 569–576). MIT Press.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., El Ghaoui, L., & Jordan, M. I. (2004). Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, *5*, 27–72.

Rakotomamonjy, A., Bach, F., Canu, S., & Grandvalet, Y. (2007). More efficiency in multiple kernel learning. *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)* (pp. 775–782). Omnipress.

Rakotomamonjy, A., Guigue, V., Mallet, G., & Alvarado, V. (2005). Ensemble of SVMs for improving brain-computer interface P300 speller performances. *15th International Conference on Artificial Neural Networks* (pp. 45–50). Springer.

Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Schröder, M., Lal, T. N., Hinterberger, T., Bogdan, M., Hill, J., Birbaumer, N., Rosenstiel, W., & Schölkopf, B. (2005). Robust EEG channel selection across subjects for brain computer interfaces. *EURASIP Journal on Applied Signal Processing*, *19*, 3103–3112.

Sonnenburg, S., Rätsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research*, *7*, 1531–1565.

Szafranski, M., Grandvalet, Y., & Morizet-Mahoudeaux, P. (2008). Hierarchical penalization. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*. MIT Press.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*, *58*, 267–288.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2001). Feature selection for SVMs. *Advances in Neural Information Processing Systems 13* (pp. 668–674). MIT Press.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society. Series B*, *68*, 49–67.

Zhao, P., Rocha, G., & Yu, B. (to appear). The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistics*.

The Many Faces of Optimism: a Unifying Approach

István Szita SZITYU@GMAIL.COM
András Lőrincz ANDRAS.LORINCZ@ELTE.HU
Eötvös Loránd University, 1117 Pázmány P. sétány 1/C, Budapest, Hungary

Abstract

The exploration-exploitation dilemma has been an intriguing and unsolved problem within the framework of reinforcement learning. "Optimism in the face of uncertainty" and model building play central roles in advanced exploration methods. Here, we integrate several concepts and obtain a fast and simple algorithm. We show that the proposed algorithm finds a near-optimal policy in polynomial time, and give experimental evidence that it is robust and efficient compared to its ascendants.

1. Introduction

Reinforcement learning (RL) is the art of maximizing long-term rewards in a stochastic, unknown environment. In the construction of RL algorithms, the choice of exploration strategy is of central significance.

We shall examine the problem of exploration in the Markov decision process (MDP) framework. While simple methods like ϵ-greedy and Boltzmann exploration are commonly used, it is known that their behavior can be extremely poor (Koenig & Simmons, 1993). Recently, a number of efficient exploration algorithms have been published, and for some of them, formal proofs of efficiency also exist. We review these methods in Section 2. By combining ideas from several sources, we construct a new algorithm for efficient exploration. The new algorithm, *optimistic initial model* (OIM), is described in Section 3. In Section 4, we show that many of the advanced algorithms, including ours, can be treated in a unified way. We use this fact to sketch a proof that OIM finds a near-optimal policy in polynomial time with high probability. Section 5 provides experimental comparison between OIM and a number of other methods on some benchmark problems. Our results are summarized in Section 6. In the rest of this section, we review the necessary preliminaries, Markov decision processes and the exploration task.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

1.1. Markov Decision Processes (MDPs)

Markov decision processes are the standard framework for RL, and the basis of numerous extensions (like continuous MDPs, partially observable MDPs or factored MDPs). An MDP is characterized by a quintuple $(X, A, \mathcal{R}, P, \gamma)$, where X is a finite set of states; A is a finite set of possible actions; $\mathcal{R} : X \times A \times X \to \mathcal{P}_{\mathbb{R}}$ is the reward distribution, $R(x, a, y)$ denotes the mean value of $\mathcal{R}(x, a, y)$, $P : X \times A \times X \to [0, 1]$ is the transition function; and finally, $\gamma \in [0, 1)$ is the discount rate on future rewards. We shall assume that all rewards are nonnegative and bounded from above by $R^0_{\max}$.

A (stationary) policy of the agent is a mapping $\pi : X \times A \to [0, 1]$. For any $x_0 \in X$, the policy of the agent and the parameters of the MDP determine a stochastic process experienced by the agent through the instantiation $x_0, a_0, r_0, x_1, a_1, r_1, \ldots, x_t, a_t, r_t, \ldots$

The goal is to find a policy that maximizes the expected value of the discounted total reward. Let us define the state-action value function (value function for short) of π as $Q^\pi(x, a) := E\Big(\sum_{t=0}^{\infty} \gamma^t r_t \;\Big|\; x = x_0, a = a_0\Big)$ and the optimal value function as

$$Q^*(x, a) := \max_\pi Q^\pi(x, a)$$

for each $(x, a) \in X \times A$. Let the greedy action at x w.r.t. value function Q be $a^Q_x := \arg\max_a Q(x, a)$. The greedy policy of Q deterministically takes the greedy action in each state. It is well-known that the greedy policy of Q^* is an optimal policy and Q^* satisfies the Bellman equations:

$$Q^*(x, a) = \sum_y P(x, a, y)\Big(R(x, a, y) + \gamma Q^*(y, a^{Q^*}_y)\Big).$$

1.2. The Exploration Problem

In the classical reinforcement learning setting, it is assumed that the environment can be modelled as an MDP, but its parameters (that is, P and R) are unknown to the agent, and she has to collect information by interacting with the environment. If too little time is spent with the exploration of the environment, the agent will get stuck with a suboptimal policy, without knowing that there exists a better one. On the other hand, the agent should not spend too much time visiting areas with low rewards and/or accurately known parameters.

What is the optimal balance between exploring and exploiting the acquired knowledge and how could the agent concentrate her exploration efforts? These questions are central for RL. It is known that the optimal exploration policy in an MDP is non-Markovian, and can be computed only for very simple tasks like k-armed bandit problems.

2. Related Literature

Here we give a short review about some of the most important exploration methods and their properties.

2.1. ϵ-greedy and Boltzmann Exploration

The most popular exploration method is ϵ-greedy action selection. The method works without a model, only an approximation of the action value function $Q(x, a)$ is needed. The agent in state x selects the greedy action a_x^Q or an explorative move with a random action with probabilities $1-\epsilon$ and ϵ, respectively. Sooner or later, all paths with nonzero probability will have been visited many times, so, a suitable learning algorithm can learn to choose the optimal path. It is known, for example, that Q-learning with nonzero exploration converges to the optimal value function with probability 1 (Littman & Szepesvári, 1996), and so does SARSA (Singh et al., 2000), if the exploration rate diminishes according to an appropriate schedule.

Boltzmann exploration selects actions as follows: the probability of choosing action a is $\frac{\exp\left(Q(s,a)/T\right)}{\sum_{a' \in A} \exp\left(Q(s,a')/T\right)}$, where 'temperature' $T\,(>0)$ regulates the amount of explorative actions. Convergence results of the ϵ-greedy method carry through to this case.

Unfortunately, for the ϵ-greedy and the Boltzmann method, exploration time may scale exponentially in the number of states (Koenig & Simmons, 1993).

2.2. Optimistic Initial Values (OIV)

One may boost exploration with a simple trick: the initial value of each state action pair can be set to some overwhelmingly high number. If a state x is visited often, then its estimated value will become more exact, and therefore, lower. Thus, the agent will try to reach the more rarely visited areas, where the estimated state values are still high. This method, called 'exploring starts' or 'optimistic initial values', is a popular exploration heuristic (Sutton & Barto, 1998), sometimes combined with others, e.g., the ϵ-greedy exploration method. Recently, Even-Dar and Mansour (2001) gave theoretical justification for the method: they proved that if the optimistic initial values are sufficiently high, Q-learning converges to a near-optimal solution. One apparent disadvantage of OIV is that if initial estimations are too high, then it takes a long to fix them.

2.3. Bayesian Methods

We may assume that the MDP (with the unknown values of P and R) is drawn from a parameterized distribution $\mathcal{M}_0$. From the collected experience and the prior distribution $\mathcal{M}_0$, we can calculate successive posterior distributions $\mathcal{M}_t, t = 1, 2, \ldots$ by Bayes' rule. Furthermore, we can calculate (at least in principle) the policy that minimizes the uncertainty of the parameters (Strens, 2000). Dearden (2000) approximates the distribution of state values directly. Exact computation of the optimal exploration policy is infeasible and Bayesian methods are computationally demanding even with simplifying assumptions about the distributions, e.g., the independencies of certain parameters.

2.4. Confidence Interval Estimation

Confidence interval estimation algorithms are between Bayesian exploration and OIV. It assumes that each state value is drawn from an independent Gaussian distribution and it computes the confidence interval of the state values. The agent chooses the action with the highest upper confidence bound. Initially, all confidence intervals are very wide, and shrink gradually towards the true state values. Therefore, the behavior of the technique is similar to OIV. The IEQL+ method of Meuleau and Bourgine (1999) directly estimates confidence intervals of Q-values, while Wiering and Schmidhuber (1998) calculate confidence intervals for P and R, and obtain Q-value bounds indirectly. Strehl and Littman (2006) improve the method and prove a polynomial-time convergence bound. Both algorithms are called *model-based interval estimation*. To avoid

confusion, we will refer to them as MBIE(WS) and MBIE(SL).

Auer and Ortner (2006) give a confidence interval-based algorithm, for which the online regret is only logarithmic in the number of steps taken.

2.5. Exploration Bonus Methods

The agent can be directed towards less-known parts of the state space by increasing the value of 'interesting' states artificially with bonuses. States can be interesting given their *frequency*, *recency*, *error*, etc. (Meuleau & Bourgine, 1999; Wiering & Schmidhuber, 1998).

The balance of exploration and exploitation is usually set by a scaling factor κ, so that the total immediate reward of the agent at time t is $r_t + \kappa \cdot b_t(x_t, a_t, x_{t+1})$, where b_t is one of the above listed bonuses. The bonuses are calculated by the agent and act as intrinsic motivating forces. Exploration bonuses for a state can vary swiftly and model-based algorithms (like prioritized sweeping or Dyna) are used for spreading the changes effectively. Alas, the weight of exploration κ needs to be annealed according to a suitable schedule.

Alternatively, the agent may learn *two* value functions separately: a regular one, Q_t^r which is based on the rewards r_t received from the environment, and an exploration value function Q_t^e which is based on the exploration bonuses. The agent's policy will be greedy with respect to their combination $Q_t^r + \kappa Q_t^e$. Then the exploration mechanism may remain the same, but several advantages appear. First of all, the changes in κ take effect *immediately*. As an example, we can immediately switch off exploration by setting κ to 0. Furthermore, Q_t^r may converge even if Q_t^e does not.

Confidence interval estimation can be phrased as an exploration bonus method: see IEQL+ (Meuleau & Bourgine, 1999) or MBIE-EB (Strehl & Littman, 2006). Even-Dar and Mansour (2001) have shown that ϵ-greedy and Boltzmann explorations can be formulated as exploration bonus methods although rewards are not propagated through the Bellman equations.

2.6. E^3 and R-max

The *Explicit explore or exploit* (E^3) algorithm of Kearns and Singh (1998) and its successor, R-max (Brafman & Tennenholtz, 2001) were the first algorithms that have polynomial time bounds for finding near-optimal policies. R-max collects statistics about transitions and rewards. When visits to a state enable high precision estimations of real transition probabilities and rewards then state is declared *known*. R-max also maintains an approximate model of the environment. Initially, the model assumes that all actions in all states lead to a (hypothetical) maximum-reward absorbing state. The model is updated each time when a state becomes known. The optimal policy of the model is either the near-optimal policy in the real environment or enters a not-yet-known state and collects new information.

3. Construction of the Algorithm

Our agent starts with a simple, but overly optimistic model. By collecting new experiences, she updates her model, which becomes more realistic. The value function is computed over the approximate model with (asynchronous) dynamic programming. The agent always chooses her action greedily w.r.t. her value function. Exploration is induced by the optimism of the model: unknown areas are believed to yield large rewards. Algorithmic components are detailed below.

Separate exploration values. Similarly to the approach of Meuleau and Bourgine (1999), we shall separate the 'true' state values from exploration values. Formally, the value function has the form

$$Q(x, a) = Q^r(x, a) + Q^e(x, a)$$

for all $(x, a) \in X \times A$, where Q^r and Q^e will summarize external and exploration rewards, respectively.

'Garden of Eden' state. Similarly to R-max, we introduce a new hypothetical 'garden of Eden' state x_E, and assume an extended state space $X' = X \cup \{x_E\}$. Once there, then, according to the inherited model, the agent remains in x_E indefinitely and receives $R_{\max}$ reward for every step, which may exceed $R^0_{\max} =: \max_{x,a,y} R(x, a, y)$, the maximal reward of the original environment.

Model approximation. The agent builds an approximate model of the environment. For each $x, y \in X$ and $a \in A$, let $N_t(x, a)$, $N_t(x, a, y)$, and $C_t(x, a, y)$ denote the number of times when a was selected in x up to step t, the number of times when transition $x \xrightarrow{a} y$ was experienced, and the sum of external rewards for $x \xrightarrow{a} y$ transitions, respectively. With these notations, the approximate model parameters are

$$\hat{P}_t(x, a, y) = \frac{N_t(x, a, y)}{N_t(x, a)} \text{ and } \hat{R}_t(x, a, y) = \frac{C_t(x, a, y)}{N_t(x, a, y)}.$$

Suitable initializations of $N_t(x, a)$, $N_t(x, a, y)$ and $C_t(x, a, y)$ will ensure that the ratios are well-defined everywhere. The exploration rewards are defined as

$$R^e(x, a, y) := \begin{cases} R_{\max}, & \text{if } y = x_E; \\ 0, & \text{if } y \neq x_E, \end{cases}$$

for each $x, y \in X \cup \{x_E\}$, $a \in A$, and are not modified during the course of learning.

Optimistic initial model. The initial model assumes that x_E has been reached once for each state-action pairs: for each $x \in X \cup \{x_E\}$, $y \in X$ and $a \in A$,

$$\begin{aligned} &N_0(x,a) = 1, \\ &N_0(x,a,y) = 0, \quad C_0(x,a,y) = 0. \\ &N_0(x,a,x_E) = 1, \quad C_0(x,a,x_E) = 0. \end{aligned}$$

Then, the optimal initial value function equals

$$Q_0(x,a) = Q_0^r(x,a) + Q_0^e(x,a) = 0 + \frac{1}{1-\gamma} R_{\max} := V_{\max}$$

for each $(x,a) \in X' \times A$, analogously to OIV.

Dynamic programming. Both value functions can be updated using the approximate model. For each $x \in X$, let a_x be the greedy action according to the combined value function, i.e.,

$$a_x := \arg\max_{a \in A} \big(Q^r(x,a) + Q^e(x,a)\big).$$

The dynamic programming equations for the value function components are

$$Q_{t+1}^r(x,a) := \sum_{y \in X} \hat{P}_t(x,a,y) \left(\hat{R}_t(x,a,y) + \gamma Q_t^r(y,a_y)\right)$$

$$\begin{aligned} Q_{t+1}^e(x,a) := \gamma &\sum_{y \in X} \hat{P}_t(x,a,y) Q_t^e(y,a_y) \\ &+ \hat{P}_t(x,a,x_E) V_{\max}. \end{aligned}$$

Episodic tasks can be handled as usual way; we introduce an absorbing final state with 0 external reward.

Asynchronous update. The algorithm can be online, if instead of full update sweeps over the state space updates are limited to state set L_t in the 'neighborhood' of the agent's current state. Neighborhood is restricted by computation time constraints; any asynchronous dynamic programming algorithm suffices. It is implicitly assumed that the current state is always updated, i.e., $x_t \in L_t$. In this paper, we used the improved prioritized sweeping algorithm of Wiering and Schmidhuber (1998).

Putting it all together. The method is summarized as Algorithm 1.

4. Analysis

In the first part of this section, we analyze the similarities and differences between various exploration methods, with an emphasis on OIM. Based on this analysis, we sketch the proof that OIM finds a near-optimal policy in polynomial time. Details of the proof can be found in (Szita & Lőrincz, 2008).

4.1. Relationship to Other Methods

'Optimism in the face of uncertainty' is a common point in exploration methods: the agent believes that she can obtain extra rewards by reaching the unexplored parts of the state space.

Note that as far as the combined value function Q is concerned, OIM is an asynchronous dynamic programming method augmented with model approximation.

Optimistic initial values. Apparently, OIM is the model-based extension of the OIV heuristic. Note however, that optimistic initialization of Q-values is not effective with a model: the more updates are made, the less effect the initialization has and it fully diminishes if value iteration is run until convergence. Therefore, naive combination of OIV and model construction is contradictory: the number of DP-updates should be kept low in order to save the initial boost, but it should be as high as possible in order to propagate the real rewards quickly.

OIM resolves this paradox by moving the optimism into the model. The optimal value function of the initial model is $Q_0 \equiv V_{\max}$, corresponding to OIV. However, DP updates can not, but only model updates may lower the exploration boost.

Note that we can set the initial model value as high as we like, but we do not have to wait until the initial boost diminishes, because Q^r and Q^e are separated.

R-max. The 'Garden of Eden' state x_E of OIM is identical to the fictitious max-reward absorbing state of R-MAX (and E^3). In both cases, the agent's model tells that all unexplored (x,a) pairs lead to x_E. R-MAX, however, updates the model only when the transition probabilities and rewards are known with high precision, which is only after many visits to (x,a). In contrast, OIM updates the model after each single visit, employing each bit of experience as soon as it is obtained. As a result, the approximate model can be used long before it becomes accurate.

Exploration bonus methods. The extra reward offered by the Garden of Eden state can be understood as an exploration bonus: for each visit of the pair (x,a), the agent gets the bonus $b_t(x,a) = \frac{1}{N_t(x,a)}\big(V_{\max} - Q_t(x,a)\big)$. It is insightful to contrast this formula with those of the other methods like the *frequency-based* bonus $b_t = -\alpha \cdot N_t(x,a)$ or the *error-based* bonus $b_t = \alpha \cdot \big|Q_{t+1}(x,a) - Q_t(x,a)\big|$.

Model-based interval exploration. The exploration bonus form of the MBIE method of Strehl and Littman (2005) sets $b_t = \frac{\alpha}{N_t(x,a)}$. MBIE-EB is not an ad-hoc method: the form of the bonus comes from

Algorithm 1 The Optimistic initial model algorithm

Input: $x_0 \in X$ initial state, $\epsilon > 0$ required precision, optimism parameter $R_{\max}$
Model initialization: $t := 0$; $\forall x, y \in X, \forall a \in A$:
$N(x,a,y) := 0$, $N(x,a,x_E) := 1$, $N(x,a) := 1$, $C(x,a,y) := 0$, $Q^r(x,a) := 0$, $Q^e(x,a) := R_{\max}/(1-\gamma)$;
repeat
  $a_t :=$ greedy action w.r.t. $Q^r + Q^e$; apply a_t and observe r_t and x_{t+1}
  $C(x_t,a_t,x_{t+1}) := C(x_t,a_t,x_{t+1}) + r_t$; $N(x_t,a_t,x_{t+1}) := N(x_t,a_t,x_{t+1}) + 1$; $N(x_t,a_t) := N(x_t,a_t) + 1$
  $L_t :=$ list of states to be updated
  for each $x \in L_t$ **do**
    $Q^r_{t+1}(x,a) := \sum_{y \in X} \hat{P}(x,a,y) \left(\hat{R}(x,a,y) + \gamma Q^r_t(y,a_y) \right)$
    $Q^e_{t+1}(x,a) := \hat{P}(x,a,x_E) R_{\max}/(1-\gamma) + \gamma \sum_{y \in X} \hat{P}(x,a,y) Q^e_t(y,a_y)$.
  end for
  $t := t + 1$
until Bellman-error$> \epsilon$

confidence interval estimations. The comparison to MBIE-EB will be especially valuable, as it converges in polynomial-time and the proof can be transported to OIM with slight modifications.

4.2. Polynomial-time Convergence

Theorem 4.1 *There exists a constant C so that for any $\epsilon > 0$, $\delta > 0$, $\epsilon_1 := \epsilon/4$, $\epsilon_2 := \epsilon_1(1-\gamma)$, $H := \frac{1}{1-\gamma} \ln \frac{R^0_{\max}}{\epsilon_1(1-\gamma)}$, $K := \left(\frac{2C|X|^5|A|H^4}{\delta \epsilon_1^4} \ln \frac{1}{\delta} \right)^{1/6}$, OIM converges almost surely to a near-optimal policy in polynomial time if started with $R_{\max} \geq \frac{3}{\epsilon_2(1-\gamma)} (R^0_{\max})^2 \ln(K R^0_{\max})$, that is, with probability $1-\delta$, the number of timesteps where $Q^{\pi^{OIM}}(x_t,a_t) > Q^*(x_t,a_t) - \epsilon$ does not hold, is $O\Big((\frac{|X|H}{\epsilon_1})^5 \frac{|A|(R^0_{\max})^7}{(1-\gamma)\epsilon_1^5} \ln^2 \frac{1}{\delta} \Big)$.*

For the sketch of the proof, we shall follow the technique of Kearns and Singh (2002) and Strehl and Littman (2006), and will use the shorthands [KS] and [SL] for referring to them. See (**?**) for the detailed proof with a slightly better polynomial bound.

A pair (x,a) is declared known, if it has been visited at least $m = C(\frac{|X|H}{\epsilon_1})^4 (R^0_{\max})^6 \ln \frac{1}{\delta}$ times, with a suitable constant C. OIM preserves the optimism of the value function:

Lemma 4.2 *Let Q_t be the sequence of Q-functions generated by* OIM. *Then, it holds with probability $1 - \delta/2$ that for any t, $Q_t(x,a) \geq Q^*(x,a) - \epsilon_1$.*

Proof: According to [SL], with probability $1 - \delta/2$,

$$\sum_y \hat{P}_t(x,a,y) \big(\hat{R}_t(x,a,y) + \gamma V^*(y) \big) - Q^*(x,a) \geq -\beta/\sqrt{N_t(x,a)}, \quad (1)$$

where $\beta := R^0_{\max}/(1-\gamma)\sqrt{\ln(2\,|X|\,|A|\,m/\delta)/2} = \frac{\sqrt{3}}{1-\gamma} R^0_{\max} \sqrt{\ln(K R^0_{\max})}$.

We will show that

$$R_{\max}/(N_t(x,a)(1-\gamma)) + \epsilon_2 \geq \beta/\sqrt{N_t(x,a)}. \quad (2)$$

For $N_t(x,a) \leq \frac{R_{\max}}{(1-\gamma)\epsilon_2}$, the first term dominates the l.h.s. and we can omit the second term (and prove the stricter inequality). If the relation is reversed, then the first term can be omitted. In both cases, we arrive at the requirement $R_{\max} \geq \frac{3}{\epsilon_2(1-\gamma)} (R^0_{\max})^2 \ln(K R^0_{\max})$, which holds by assumption.

At step t, a number of DP updates are carried out. We proceed by induction on the number of DP-updates. Initially, $Q^{(0)}(x,a) \geq Q^*(x,a) - \epsilon_1$, then
$\sum_y \hat{P}_t(x,a,y) \big(\hat{R}_t(x,a,y) + \gamma V^{(i)}(y) \big) + \frac{V_{\max}}{N_t(x,a)}$
$\geq \sum_y \hat{P}_t(x,a,y) \big(\hat{R}_t(x,a,y) + \gamma (V^*(y) - \epsilon_1) \big) + \frac{V_{\max}}{N_t(x,a)}$
$\geq Q^*(x,a) - \beta/\sqrt{N_t(x,a)} - \gamma\epsilon_1 + \frac{V_{\max}}{N_t(x,a)}$
$\geq Q^*(x,a) - \gamma\epsilon_1 - \epsilon_2 = Q^*(x,a) - \epsilon_1$,
where we applied (1), (2), the induction assumption and the definition of ϵ_2. □

Let M denote the true (and unknown) MDP, let $\hat{M}$ be the approximate model of OIM, and define $\bar{M}$ so that it is identical to M for known pairs, and equals $\hat{M}$ for unknown pairs. The parameters of $\hat{M}$ and $\bar{M}$ are nearly identical: if (x,a) becomes known, then the local values of $\hat{P}$ and $\hat{R}$ are $O(\frac{\epsilon}{|X|H R^0_{\max}})^2$-approximations of P and R with probability $1 - \delta/2$ (Lemma 5 of [KS]). Therefore, by Lemma 4 of [KS],

$$|Q^\pi_{\hat{M}}(x,a) - Q^\pi_{\bar{M}}(x,a)| < \epsilon_1. \quad (3)$$

Define the H-step truncated value function of policy π as $Q^\pi(x,a,H) := E\Big(\sum_{t=0}^H \gamma^t r_t \;\Big|\; x = x_0, a = a_0 \Big)$. According to [KS] Lemma 2, $H = \frac{1}{1-\gamma} \ln \frac{R^0_{\max}}{\epsilon_1(1-\gamma)}$ is an

ϵ_1-horizon time, i.e., $|Q^\pi_M(x,a,H) - Q^\pi_M(x,a)| < \epsilon_1$ for any (x,a), π and any MDP M with discount factor γ.

Consider a state-action pair (x_1,a_1) and a H-step long trajectory generated by π. Let A_M be the event that an unknown pair (x,a) is encountered along the trajectory. Then, by Lemma 3 of [SL],

$$Q^\pi_M(x_1,a_1) \geq Q^\pi_{\hat{M}}(x_1,a_1) - V_{\max} \Pr(A_M). \qquad (4)$$

To conclude the proof, we separate two cases (following the line of thoughts of Theorem 1 in [SL]). In the first case, an exploration step will occur with high probability: Let $V^0_{\max} := R^0_{\max}/(1-\gamma)$. Suppose that $\Pr(A_M) > \epsilon_1/V^0_{\max}$, that is, an unknown pair (x,a) is visited in H steps with high probability. This can happen at most $m|X||A|$ times, so by the Hoeffding-Azuma bound, with probability $1-\delta/2$, all (x,a) will become known after $O(\frac{m|X||A|HV^0_{\max}}{\epsilon_1} \ln \frac{1}{\delta})$ exploration steps.

On the other hand, if $\Pr(A_M) \leq \epsilon_1/V^0_{\max}$, then the policy is near-optimal with probability $1-\delta$:

$$\begin{aligned}
&Q^{\pi^{OIM}}_M(x_1,a_1) \geq Q^{\pi^{OIM}}_M(x_1,a_1,H) \\
&\geq Q^{\pi^{OIM}}_{\hat{M}}(x_1,a_1,H) - V^0_{\max}\Pr(A_M) \\
&\geq Q^{\pi^{OIM}}_{\hat{M}}(x_1,a_1,H) - \epsilon_1 \geq Q^{\pi^{OIM}}_M(x_1,a_1) - 2\epsilon_1 \\
&\geq Q^{\pi^{OIM}}_{\hat{M}}(x_1,a_1) - 3\epsilon_1 \geq Q^*(x_1,a_1) - 4\epsilon_1 \\
&= Q^*(x_1,a_1) - \epsilon,
\end{aligned}$$

where we applied (in this order) the property that truncation decreases the value function; Eq. (4); our assumption; the ϵ_1-horizon property of H; Eq. (3); Lemma 4.2 and the definition of ϵ_1.

5. Experiments

To assess the practical utility of OIM, we compared its performance to other exploration methods. Experiments were run on several small benchmark tasks challenging exploration algorithms.

For fair comparisons, benchmark problems were taken from the literature without changes, nor did we change the experimental settings or the presentation of experimental data. It also means that the presentation format varies for different benchmarks.

5.1. RiverSwim and SixArms

The first two benchmark problems, *RiverSwim* and *SixArms*, were taken from Strehl and Littman (2006).

The *RiverSwim* MDP has 6 states, representing the position of the agent in a river. The agent has two possible actions: she can swim either upstream or downstream. Swimming down is always successful, but swimming up succeeds only with a 30% chance and there is a 10% chance of slipping down. The lowermost position yields +5 reward per step, while the uppermost position yields +10000.

Table 1. Results on the *RiverSwim* task.

Method	Cumulative reward
E^3	$3.020 \cdot 10^6 \pm 0.027 \cdot 10^6$
R-MAX	$3.014 \cdot 10^6 \pm 0.039 \cdot 10^6$
MBIE(SL)	$3.168 \cdot 10^6 \pm 0.023 \cdot 10^6$
MBIE-EB	$3.093 \cdot 10^6 \pm 0.023 \cdot 10^6$
OIM	$3.201 \cdot 10^6 \pm 0.016 \cdot 10^6$

Table 2. Results on the *SixArms* task.

Method	Cumulative reward
E^3	$1.623 \cdot 10^6 \pm 0.244 \cdot 10^6$
R-MAX	$2.819 \cdot 10^6 \pm 0.256 \cdot 10^6$
MBIE(SL)	$9.205 \cdot 10^6 \pm 0.559 \cdot 10^6$
MBIE-EB	$9.486 \cdot 10^6 \pm 0.587 \cdot 10^6$
OIM	$10.007 \cdot 10^6 \pm 0.654 \cdot 10^6$

The *SixArms* MDP consists of a central state and six 'payoff states'. In the central state, the agent can play 6 one-armed bandits. If she pulls arm k and wins, she is transferred to payoff state k. Here, she can get a reward in each step, if she chooses the appropriate action. The winning probabilities range from 1 to 0.01, while the rewards range from 50 to 6000 (for the exact values, see Strehl & Littman, 2006).

Data for E^3, R-MAX, MBIE and MBIE-EB are taken from Strehl and Littman (2006). Parameters of all four algorithms were chosen optimally. Following a coarse search in parameter space, the $R_{\max}$ parameter for OIM was set to 2000 for *RiverSwim* and to 10000 for *SixArms*. State spaces are small and value iteration instead of prioritized sweeping was completed in each step.

On both problems, each algorithm ran for 5000 time steps and the undiscounted total reward was recorded. The averages and 95% confidence intervals are calculated over 1000 test runs (Tables 5.1 and 5.1).

5.2. 50×50 Maze with Subgoals

Another benchmark problem, *MazeWithSubgoals*, was suggested by Wiering and Schmidhuber (1998). The agent has to navigate in a 50×50 maze from the start position at $(2,2)$ to the goal (with +1000 reward) at the opposite corner $(49,49)$. There are sub-

Table 3. Results on the *MazeWithSubgoals* task. The number of steps required to learn p-optimal policies (p=0.95, 0.99, 0.998) on the 50×50 maze task with suboptimal goals. In parentheses: how many runs out of 20 have found the goal. 'k' stands for 1000.

Method	95%	99%	99.8%
ϵ-GREEDY, $\epsilon = 0.2$	– (0)	– (0)	– (0)
ϵ-GREEDY, $\epsilon = 0.4$	43k (4)	52k (4)	68k (4)
RECENCY-BONUS	27k (19)	55k (18)	69k (9)
FREQ.-BONUS	24k (20)	50k (16)	66k (10)
MBIE(WS)	25k (20)	42k (19)	66k (18)
OIM	19k (20)	29k (20)	31k (20)

Table 4. Average accumulated rewards on the *Chain* task. Optimal policy gathers 3677.

METHOD	PHASE 1	PHASE 2	PHASE 8
QL+VAR.-BONUS	–	2570[1]	–
QL+ERR.-BONUS	–	2530[1]	–
QL ϵ-GREEDY	1519	1611	1602
QL BOLTZMANN	1606	1623	–
IEQL+	2344	2557	–
BAYESIAN QL	1697	2417	–
BAYESIAN DP[2]	3158	3611	3643
OIM	3510	3628	3643

optimal goals (with +500 reward) at the other two corners. The maze has blocked places and punishing states (−10 reward), set randomly in 20-20% of the squares. The agent can move in four directions, but with a 10% chance, its action is replaced by a random one. If the agent tries to move to a blocked state, it gets a reward of −2. Reaching any of the goals resets the agent to the start state. In all other cases, the agent gets a −1 reward for each step.

Each algorithm was run on 20 different mazes for 100,000 steps. After every 1000 steps, we tested the learned value functions by averaging 20 test runs, in each one following the greedy policy for 10,000 steps, and averaging cumulated (undiscounted) rewards. We measured the number of test runs needed for the algorithms to learn to collect 95%, 99% and 99.8% of the maximum possible rewards in 100,000 steps, and the number of steps this takes on average, if the algorithms can meet the challenge.

The algorithms that we compared were the recency based and frequency based exploration bonus methods, two versions of ϵ-greedy exploration, MBIE(WS) and OIM. All exploration rules applied the improved prioritized sweeping of Wiering and Schmidhuber (1998). OIM's $R_{\max}$ was set to 1000. The results are summarized in Table 3.

5.3. Chain, Loop and FlagMaze

The next three benchmark MDPs, the *Chain*, *Loop* and *FlagMaze* tasks were investigated, e.g., by Meuleau and Bourgine (1999), Strens (2000) and Dearden (2000). In the *Chain* task, 5 states are lined up along a chain. The agent gets +2 reward for being in state 1 and +10 for being in state 5. One of the actions advances one state ahead, the other one resets the agent to state 1. The *Loop* task has 9 states in two loops (arranged in a 8-shape). Completing the first loop (using any combination of the two actions) yields +1 reward, while the second loop yields +2, but one of the actions resets the agent to the start. The *FlagMaze* task consists of a 6×7 maze with several walls, a start state, a goal state and 3 flags. Whenever the agent reaches the goal, her reward is the number of flags collected.

The following algorithms were compared: Q-learning with variance-based and TD error-based exploration bonus (model-free variants), ϵ-greedy exploration, Boltzmann exploration, IEQL+, Bayesian Q-learning, Bayesian DP and OIM. Data were taken from Meuleau and Bourgine (1999), Strens (2000) and Dearden (2000). According to the sources, parameters for all algorithms were set optimally. OIM's $R_{\max}$ parameter was set to 0.5, 10 and 0.005 for the three tasks, respectively.

Each algorithm ran for 8 learning phases. The total cumulated reward over each learning phase was measured. One phase lasted for 1000 steps for the first two tasks and 20,000 steps for the *FlagMaze* task. We carried out 256 parallel runs for the first 2 tasks and 20 for the third one.

6. Summary of the Results

We proposed a new algorithm for exploration and reinforcement learning in Markov decision processes. The algorithm integrates concepts from other advanced exploration methods. The key component of our algorithm is an optimistic initial model. The optimal policy according to the agent's model will either explore new information that helps to make the model

[1]Results for Phase 5.

[2]Augmented with limited amount of pre-wired knowledge (the list of successor states).

Table 5. Average accumulated rewards on the *Loop* task. Optimal policy gathers 400.

METHOD	PHASE 1	PHASE 2	PHASE 8
QL+VAR.-BONUS	–	179[1]	–
QL+ERR.-BONUS	–	179[1]	–
QL ϵ-GREEDY	337	392	399
QL BOLTZMANN	186	200	–
IEQL+	264	293	–
BAYESIAN QL	326	340	–
BAYESIAN DP[2]	377	397	399
OIM	393	400	400

Table 6. Average accumulated rewards on the *FlagMaze* task. Optimal policy gathers approximately 1890.

METHOD	PHASE 1	PHASE 2	PHASE 8
QL ϵ-GREEDY	655	1135	1147
QL BOLTZMANN	195	1024	–
IEQL+	269	253	–
BAYESIAN QL	818	1100	–
BAYESIAN DP[2]	750	1763	1864
OIM	1133	1169	1171

more accurate, or follows a near-optimal path. The extent of optimism regulates the amount of exploration. We have shown that with a suitably optimistic initialization, our algorithm finds a near-optimal policy in polynomial time. Experiments were conducted on a number of benchmark MDPs. According to the experimental results our novel method is robust and compares favorably to other methods.

Acknowledgments

We are grateful to one of the reviewers for his helpful comments. This research has been supported by the EC FET 'New Ties' Grant FP6-502386 and NEST 'PERCEPT' Grant FP6-043261. Opinions and errors in this manuscript are the author's responsibility, they do not necessarily reflect those of the EC or other project members.

References

Auer, P., & Ortner, R. (2006). Logarithmic online regret bounds for undiscounted reinforcement learning algorithms. *NIPS* (pp. 49–56).

Brafman, R. I., & Tennenholtz, M. (2001). R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *Proc. IJCAI* (pp. 953–958).

Dearden, R. W. (2000). *Learning and planning in structured worlds*. Doctoral dissertation, University of British Columbia.

Even-Dar, E., & Mansour, Y. (2001). Convergence of optimistic and incremental Q-learning. *NIPS* (pp. 1499–1506).

Kearns, M., & Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. *Proc. ICML* (pp. 260–268).

Kearns, M., & Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, *49*, 209–232.

Koenig, S., & Simmons, R. G. (1993). Complexity analysis of real-time reinforcement learning. *Proc. AAAI* (pp. 99–105).

Littman, M. L., & Szepesvári, C. (1996). A generalized reinforcement-learning model: Convergence and applications. *Proc. ICML* (pp. 310–318). Morgan Kaufmann.

Meuleau, N., & Bourgine, P. (1999). Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, *35*, 117–154.

Singh, S. P., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, *38*, 287–308.

Strehl, A. L., & Littman, M. L. (2005). A theoretical analysis of model-based interval estimation. *Proc. ICML* (pp. 856–863).

Strehl, A. L., & Littman, M. L. (2006). An analysis of model-based interval estimation for Markov decision processes. *Submitted.*

Strens, M. (2000). A Bayesian framework for reinforcement learning. *Proc. ICML* (pp. 943–950). Morgan Kaufmann, San Francisco, CA.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge.

Szita, I., & Lőrincz, A. (2008). *The many faces of optimism – extended version* (Technical Report). Eötvös Loránd University, Hungary.

Wiering, M. A., & Schmidhuber, J. (1998). Efficient model-based exploration. *Proc. SAB: From Animals to Animats* (pp. 223–228).

ν-Support Vector Machine as Conditional Value-at-Risk Minimization

Akiko Takeda TAKEDA@AE.KEIO.AC.JP
Keio University, 3-14-1 Hiyoshi, Kouhoku, Yokohama, Kanagawa 223-8522, Japan

Masashi Sugiyama SUGI@CS.TITECH.AC.JP
Tokyo Institute of Technology, 2-12-1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan

Abstract

The ν-support vector classification (ν-SVC) algorithm was shown to work well and provide intuitive interpretations, e.g., the parameter ν roughly specifies the fraction of support vectors. Although ν corresponds to a fraction, it cannot take the entire range between 0 and 1 in its original form. This problem was settled by a non-convex extension of ν-SVC and the extended method was experimentally shown to generalize better than original ν-SVC. However, its good generalization performance and convergence properties of the optimization algorithm have not been studied yet. In this paper, we provide new theoretical insights into these issues and propose a novel ν-SVC algorithm that has guaranteed generalization performance and convergence properties.

1. Introduction

Support vector classification (SVC) is one of the most successful classification algorithms in modern machine learning (Schölkopf & Smola, 2002). SVC finds a hyperplane that separates training samples in different classes with maximum margin (Boser et al., 1992). The maximum margin hyperplane was shown to minimize an upper bound of the generalization error according to the Vapnik-Chervonenkis theory (Vapnik, 1995). Thus the generalization performance of SVC is theoretically guaranteed.

SVC was extended to be able to deal with non-separable data by trading the margin size with the data separation error (Cortes & Vapnik, 1995). This soft-margin formulation is commonly referred to as C-SVC since the trade-off is controlled by the parameter C. C-SVC was shown to work very well in a wide range of real-world applications (Schölkopf & Smola, 2002).

An alternative formulation of the soft-margin idea is ν-SVC (Schölkopf et al., 2000)—instead of the parameter C, ν-SVC involves another trade-off parameter ν that roughly specifies the fraction of support vectors (or sparseness of the solution). Thus, the ν-SVC formulation provides us richer interpretation than the original C-SVC formulation, which would be potentially useful in real applications.

Since the parameter ν corresponds to a fraction, it should be able to be chosen between 0 and 1. However, it was shown that admissible values of ν are actually limited (Crisp & Burges, 2000; Chang & Lin, 2001). To cope with this problem, Perez-Cruz et al. (2003) introduced the notion of negative margins and proposed extended ν-SVC (Eν-SVC) which allows ν to take the entire range between 0 and 1. They also experimentally showed that the generalization performance of Eν-SVC is often better than that of original ν-SVC. Thus the extension contributes not only to elucidating the theoretical property of ν-SVC, but also to improving its generalization performance.

However, there remain two open issues in Eν-SVC. The first issue is that the reason why a high generalization performance can be obtained by Eν-SVC was not completely explained yet. The second issue is that the optimization problem involved in Eν-SVC is non-convex and theoretical convergence properties of the Eν-SVC optimization algorithm have not been studied yet. The purpose of this paper is to provide new theoretical insights into these two issues.

After reviewing existing SVC methods in Section 2, we

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

elucidate the generalization performance of Eν-SVC in Section 3. We first show that the Eν-SVC formulation could be interpreted as minimization of the *conditional value-at-risk* (CVaR), which is often used in finance (Rockafellar & Uryasev, 2002; Gotoh & Takeda, 2005). Then we give new generalization error bounds based on the CVaR risk measure. This theoretical result justifies the use of Eν-SVC.

In Section 4, we address non-convexity of the Eν-SVC optimization problem. We first give a new optimization algorithm that is guaranteed to converge to one of the local optima within a finite number of iterations. Based on this improved algorithm, we further show that the global solution can be actually obtained within finite iterations even though the optimization problem is non-convex.

Finally, in Section 5, we give concluding remarks and future prospects. Proofs of all theorems and lemmas are sketched in Appendix unless mentioned.

2. Support Vector Classification

In this section, we formulate the classification problem and briefly review support vector algorithms.

2.1. Classification Problem

Let us address the classification problem of learning a decision function h from $\mathcal{X}$ ($\subset \mathbb{R}^n$) to $\{\pm 1\}$ based on training samples $(\boldsymbol{x}_i, y_i)$ ($i \in M := \{1, ..., m\}$). We assume that the training samples are i.i.d. following the unknown probability distribution $P(\boldsymbol{x}, y)$ on $\mathcal{X} \times \{\pm 1\}$.

The goal of the classification task is to obtain a classifier h that minimizes the generalization error (or the risk):

$$R[h] := \int \frac{1}{2}|h(\boldsymbol{x}) - y| dP(\boldsymbol{x}, y),$$

which corresponds to the misclassification rate for unseen test samples.

For the sake of simplicity, we generally focus on linear classifiers, i.e.,

$$h(\boldsymbol{x}) = \mathrm{sign}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b), \tag{1}$$

where $\boldsymbol{w}$ ($\in \mathbb{R}^n$) is a non-zero normal vector, b ($\in \mathbb{R}$) is a bias parameter, and $\mathrm{sign}(\xi) = 1$ if $\xi \geq 0$ and -1 otherwise.

Most of the discussions in this paper can be directly applicable to non-linear kernel classifiers (Schölkopf & Smola, 2002). Thus we may not lose generality by restricting ourselves to linear classifiers.

2.2. Support Vector Classification

The Vapnik-Chervonenkis theory (Vapnik, 1995) showed that a large margin classifier has a small generalization error. Motivated by this theoretical result, Boser et al. (1992) developed an algorithm for finding the hyperplane $(\boldsymbol{w}, b)$ with maximum margin:

$$\min_{\boldsymbol{w}, b} \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{s.t. } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1, \quad i \in M. \tag{2}$$

This is called (hard-margin) support vector classification (SVC) and valid when the training samples are linearly separable. In the following, we omit "$i \in M$" in the constraint for brevity.

2.3. C-Support Vector Classification

Cortes and Vapnik (1995) extended the SVC algorithm to non-separable cases and proposed trading the margin size with the data separation error (i.e., "soft-margin"):

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m} \xi_i$$
$$\text{s.t. } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0,$$

where C (> 0) controls the trade-off. This formulation is usually referred to as C-SVC, and was shown to work very well in various real-world applications (Schölkopf & Smola, 2002).

2.4. ν-Support Vector Classification

ν-SVC is another formulation of soft-margin SVC (Schölkopf et al., 2000):

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}, \rho} \frac{1}{2}\|\boldsymbol{w}\|^2 - \nu\rho + \frac{1}{m}\sum_{i=1}^{m} \xi_i$$
$$\text{s.t. } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \rho \geq 0,$$

where ν ($\in \mathbb{R}$) is the trade-off parameter.

Schölkopf et al. (2000) showed that if the ν-SVC solution yields $\rho > 0$, C-SVC with $C = 1/(m\rho)$ produces the same solution. Thus ν-SVC and C-SVC are equivalent. However, ν-SVC has additional intuitive interpretations, e.g., ν is an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors (i.e., sparseness of the solution). Thus, the ν-SVC formulation would be potentially more useful than the C-SVC formulation in real applications.

2.5. Eν-SVC

Although ν has an interpretation as a fraction, it cannot always take its full range between 0 and 1 (Crisp & Burges, 2000; Chang & Lin, 2001).

2.5.1. Admissible Range of ν

For an optimal solution $\{\alpha_i^C\}_{i=1}^m$ of dual C-SVC, let

$$\zeta(C) := \frac{1}{Cm}\sum_{i=1}^m \alpha_i^C,$$
$$\nu_{\min} := \lim_{C\to\infty} \zeta(C) \quad \text{and} \quad \nu_{\max} := \lim_{C\to 0} \zeta(C).$$

Then, Chang and Lin (2001) showed that for $\nu \in (\nu_{\min}, \nu_{\max}]$, the optimal solution set of ν-SVC is the same as that of C-SVC with some C (not necessarily unique). In addition, the optimal objective value of ν-SVC is strictly negative. However, for $\nu \in (\nu_{\max}, 1]$, ν-SVC is unbounded, i.e., there exists no solution; for $\nu \in [0, \nu_{\min}]$, ν-SVC is feasible with zero optimal objective value, i.e., we end up with just having a trivial solution ($\boldsymbol{w} = \boldsymbol{0}$ and $b = 0$).

2.5.2. Increasing Upper Admissible Range

It was shown by Crisp and Burges (2000) that

$$\nu_{\max} = 2\min(m_+, m_-)/m,$$

where m_+ and m_- are the number of positive and negative training samples. Thus, when the training samples are balanced (i.e., $m_+ = m_-$), $\nu_{\max} = 1$ and therefore ν can reach its upper limit 1. When the training samples are imbalanced (i.e., $m_+ \neq m_-$), Perez-Cruz et al. (2003) proposed modifying the optimization problem of ν-SVC as

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi},\rho} \frac{1}{2}\|\boldsymbol{w}\|^2 - \nu\rho + \frac{1}{m_+}\sum_{i:y_i=1}\xi_i + \frac{1}{m_-}\sum_{i:y_i=-1}\xi_i$$
$$\text{s.t. } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \rho \geq 0,$$

i.e., the effect of positive and negative samples are balanced. Under this modified formulation, $\nu_{\max} = 1$ holds even when training samples are imbalanced.

For the sake of simplicity, we assume $m_+ = m_-$ in the rest of this paper; when $m_+ \neq m_-$, all the results can be simply extended in a similar way as above.

2.5.3. Decreasing Lower Admissible Range

When $\nu \in [0, \nu_{\min}]$, ν-SVC produces a trivial solution ($\boldsymbol{w} = \boldsymbol{0}$ and $b = 0$) as shown in Chang and Lin (2001). To prevent this, Perez-Cruz et al. (2003) proposed allowing the margin ρ to be negative and enforcing the norm of $\boldsymbol{w}$ to be unity:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi},\rho} -\nu\rho + \frac{1}{m}\sum_{i=1}^m \xi_i$$
$$\text{s.t. } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) \geq \rho - \xi_i,\ \xi_i \geq 0,\ \|\boldsymbol{w}\|^2 = 1. \quad (3)$$

By this modification, a non-trivial solution can be obtained even for $\nu \in [0, \nu_{\min}]$. This modified formulation is called extended ν-SVC (Eν-SVC).

The Eν-SVC optimization problem is non-convex due to the equality constraint $\|\boldsymbol{w}\|^2 = 1$. Perez-Cruz et al. (2003) proposed the following iterative algorithm for computing a solution. First, for some initial $\widetilde{\boldsymbol{w}}$, solve the problem (3) with $\|\boldsymbol{w}\|^2 = 1$ replaced by $\langle \widetilde{\boldsymbol{w}}, \boldsymbol{w}\rangle = 1$. Then, using the optimal solution $\widehat{\boldsymbol{w}}$, update $\widetilde{\boldsymbol{w}}$ by

$$\widetilde{\boldsymbol{w}} \longleftarrow \gamma\widetilde{\boldsymbol{w}} + (1-\gamma)\widehat{\boldsymbol{w}} \quad (4)$$

for $\gamma = 9/10$, and iterate this procedure until convergence.

Perez-Cruz et al. (2003) experimentally showed that the generalization performance of Eν-SVC with $\nu \in [0, \nu_{\min}]$ is often better than that with $\nu \in (\nu_{\min}, \nu_{\max}]$, implying that E$\nu$-SVC is a promising classification algorithm. However, it is not clear how the notion of negative margins influences on the generalization performance and how fast the above iterative algorithm converges. The goal of this paper is to give new theoretical insights into these issues.

3. Justification of the Eν-SVC Criterion

In this section, we give a new interpretation of Eν-SVC and theoretically explain why it works well.

3.1. New Interpretation of Eν-SVC as CVaR minimization

Let $f(\boldsymbol{w}, b; \boldsymbol{x}, y)$ be the *margin error* for a sample $(\boldsymbol{x}, y)$:

$$f(\boldsymbol{w}, b; \boldsymbol{x}, y) := -\frac{y(\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b)}{\|\boldsymbol{w}\|}.$$

Let us consider the distribution of margin errors over all training samples:

$$\Phi(\alpha|\boldsymbol{w}, b) := P\{(\boldsymbol{x}_i, y_i) \mid f(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i) \leq \alpha\}.$$

For $\beta \in [0, 1)$, let $\alpha_\beta(\boldsymbol{w}, b)$ be the 100β-percentile of the margin error distribution:

$$\alpha_\beta(\boldsymbol{w}, b) := \min\{\alpha \mid \Phi(\alpha|\boldsymbol{w}, b) \geq \beta\}.$$

Thus only the fraction $(1-\beta)$ of the margin error $f(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i)$ exceeds the threshold $\alpha_\beta(\boldsymbol{w}, b)$ (see Figure 1). $\alpha_\beta(\boldsymbol{w}, b)$ is commonly referred to as the *value-at-risk* (VaR) in finance and is often used by security houses or investment banks to measure the market risk of their asset portfolios (Rockafellar & Uryasev, 2002; Gotoh & Takeda, 2005).

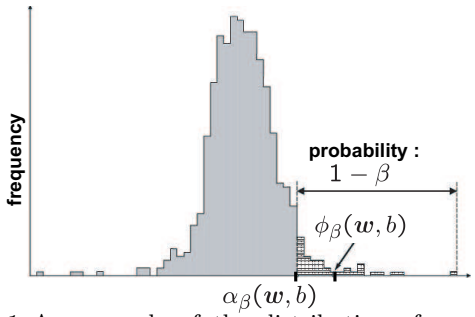


Figure 1. An example of the distribution of margin errors $f(\boldsymbol{w},b;\boldsymbol{x}_i,y_i)$ over all training samples. $\alpha_\beta(\boldsymbol{w},b)$ is the 100β-percentile called the value-at-risk (VaR), and the mean $\phi_\beta(\boldsymbol{w},b)$ of the β-tail distribution is called the conditional VaR (CVaR).

Let us consider the β-tail distribution of $f(\boldsymbol{w},b;\boldsymbol{x}_i,y_i)$:

$$\Phi_\beta(\alpha|\boldsymbol{w},b) := \begin{cases} 0 & \text{for } \alpha < \alpha_\beta(\boldsymbol{w},b), \\ \frac{\Phi(\alpha|\boldsymbol{w},b)-\beta}{1-\beta} & \text{for } \alpha \geq \alpha_\beta(\boldsymbol{w},b). \end{cases}$$

Let $\phi_\beta(\boldsymbol{w},b)$ be the mean of the β-tail distribution of $f(\boldsymbol{w},b;\boldsymbol{x}_i,y_i)$ (see Figure 1 again):

$$\phi_\beta(\boldsymbol{w},b) := \boldsymbol{E}_{\Phi_\beta}[f(\boldsymbol{w},b;\boldsymbol{x}_i,y_i)],$$

where $\boldsymbol{E}_{\Phi_\beta}$ denotes the expectation over the distribution Φ_β. $\phi_\beta(\boldsymbol{w},b)$ is called the *conditional VaR* (CVaR). By definition, the CVaR is always larger than or equal to the VaR:

$$\phi_\beta(\boldsymbol{w},b) \geq \alpha_\beta(\boldsymbol{w},b). \tag{5}$$

Let us consider the problem of minimizing the CVaR $\phi_\beta(\boldsymbol{w},b)$ (which we refer to as minCVaR):

$$\min_{\boldsymbol{w},b} \phi_\beta(\boldsymbol{w},b). \tag{6}$$

Then we have the following theorem.

Theorem 1 *The solution of the minCVaR problem* (6) *is equivalent to the solution of the Eν-SVC problem* (3) *with*

$$\nu = 1-\beta.$$

Theorem 1 shows that Eν-SVC actually minimizes the CVaR $\phi_{1-\nu}(\boldsymbol{w},b)$. Thus, E$\nu$-SVC could be interpreted as minimizing the mean margin error over a set of "bad" training samples. In contrast, the hard-margin SVC problem (2) can be equivalently expressed in terms of the margin error as

$$\min_{\boldsymbol{w},b} \max_{i\in M} f(\boldsymbol{w},b;\boldsymbol{x}_i,y_i).$$

Thus hard-margin SVC minimizes the margin error of the single "worst" training sample. This analysis shows that Eν-SVC can be regarded as an extension of hard-margin SVC to be less sensitive to an outlier (i.e., the single "worst" training sample).

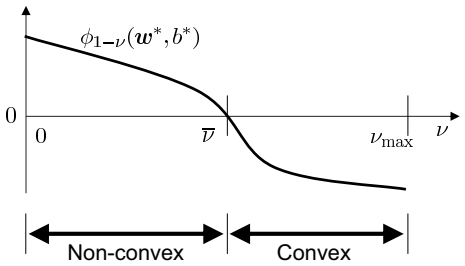


Figure 2. A profile of the CVaR $\phi_{1-\nu}(\boldsymbol{w}^*,b^*)$ as a function of ν. As shown in Section 4, the Eν-SVC optimization problem can be cast as a convex problem if $\nu \in (\overline{\nu},\nu_{\max}]$, while it is essentially non-convex if $\nu \in (0,\overline{\nu})$.

3.2. Justification of Eν-SVC

We have shown the equivalence between Eν-SVC and minCVaR. Here we derive new bounds of the generalization error based on the notion of CVaR and try to justify the use of Eν-SVC.

When training samples are linearly separable, the margin error $f(\boldsymbol{w},b;\boldsymbol{x}_i,y_i)$ is negative for all samples. Then, at the optimal solution $(\boldsymbol{w}^*,b^*)$, the CVaR $\phi_{1-\nu}(\boldsymbol{w}^*,b^*)$ is always negative. However, in non-separable cases, $\phi_{1-\nu}(\boldsymbol{w}^*,b^*)$ could be positive particularly when ν is close to 0. Regarding the CVaR, we have the following lemma.

Lemma 2 *$\phi_{1-\nu}(\boldsymbol{w}^*,b^*)$ is continuous with respect to ν and is strictly decreasing when ν is increased.*

Let $\overline{\nu}$ be such that

$$\phi_{1-\overline{\nu}}(\boldsymbol{w}^*,b^*) = 0$$

if such $\overline{\nu}$ exists; we set $\overline{\nu} = \nu_{\max}$ if $\phi_{1-\nu}(\boldsymbol{w}^*,b^*) > 0$ for all ν and we set $\overline{\nu} = 0$ if $\phi_{1-\nu}(\boldsymbol{w}^*,b^*) < 0$ for all ν. Then we have the following relation (see Figure 2):

$$\phi_{1-\nu}(\boldsymbol{w}^*,b^*) < 0 \text{ for } \nu \in (\overline{\nu},\nu_{\max}],$$
$$\phi_{1-\nu}(\boldsymbol{w}^*,b^*) > 0 \text{ for } \nu \in (0,\overline{\nu}).$$

Below, we analyze the generalization error of Eν-SVC depending on the value of ν.

3.2.1. Justification When $\nu \in (\overline{\nu},\nu_{\max}]$

Theorem 3 *Let $\nu \in (\overline{\nu},\nu_{\max}]$. Suppose that support $\mathcal{X}$ is in a ball of radius R around the origin. Then, for all $(\boldsymbol{w},b)$ such that $\|\boldsymbol{w}\| = 1$ and $\phi_{1-\nu}(\boldsymbol{w},b) < 0$, there exists a positive constant c such that the following bound hold with probability at least $1-$:*

$$R[h] \leq \nu + G(\alpha_{1-\nu}(\boldsymbol{w},b)), \tag{7}$$

where

$$G(\gamma) = \sqrt{\frac{2}{m}\left(\frac{4c^2(R^2+1)^2}{\gamma^2}\log_2(2m) - 1 + \log\frac{2}{}\right)}.$$

The generalization error bound in (7) is furthermore upper-bounded as

$$\nu + G(\alpha_{1-\nu}(\boldsymbol{w}, b)) \leq \nu + G(\phi_{1-\nu}(\boldsymbol{w}, b)).$$

$G(\gamma)$ is monotone decreasing as $|\gamma|$ increases. Thus, the above theorem shows that when $\phi_{1-\nu}(\boldsymbol{w}, b) < 0$, the upper bound $\nu + G(\phi_{1-\nu}(\boldsymbol{w}, b))$ is lowered if the CVaR $\phi_{1-\nu}(\boldsymbol{w}, b)$ is reduced. Since Eν-SVC minimizes $\phi_{1-\nu}(\boldsymbol{w}, b)$ (see Theorem 1), the upper bound of the generalization error is also minimized.

3.2.2. Justification When $\nu \in (0, \overline{\nu}]$

Our discussion below depends on the sign of $\alpha_{1-\nu}(\boldsymbol{w}, b)$. When $\alpha_{1-\nu}(\boldsymbol{w}, b) < 0$, we have the following theorem.

Theorem 4 *Let $\nu \in (0, \overline{\nu}]$. Then, for all $(\boldsymbol{w}, b)$ such that $\|\boldsymbol{w}\| = 1$ and $\alpha_{1-\nu}(\boldsymbol{w}, b) < 0$, there exists a positive constant c such that the following bound holds with probability at least $1 - \ $:*

$$R[h] \leq \nu + G(\alpha_{1-\nu}(\boldsymbol{w}, b)).$$

A proof of the above theorem is omitted since the proof follows a similar line to the proof of Theorem 3. This theorem shows that when $\alpha_{1-\nu}(\boldsymbol{w}, b) < 0$, the upper bound $\nu + G(\alpha_{1-\nu}(\boldsymbol{w}, b))$ is lowered if $\alpha_{1-\nu}(\boldsymbol{w}, b)$ is reduced. On the other hand, Eq.(5) shows that the VaR $\alpha_{1-\nu}(\boldsymbol{w}, b)$ is upper-bounded by the CVaR $\phi_{1-\nu}(\boldsymbol{w}, b)$. Therefore, minimizing $\phi_{1-\nu}(\boldsymbol{w}, b)$ by Eν-SVC may have an effect of lowering the upper bound of the generalization error.

When $\alpha_{1-\nu}(\boldsymbol{w}, b) > 0$, we have the following theorem.

Theorem 5 *Let $\nu \in (0, \overline{\nu}]$. Then, for all $(\boldsymbol{w}, b)$ such that $\|\boldsymbol{w}\| = 1$ and $\alpha_{1-\nu}(\boldsymbol{w}, b) > 0$, there exists a positive constant c such that the following bound hold with probability at least $1 - \ $:*

$$R[h] \geq \nu - G(\alpha_{1-\nu}(\boldsymbol{w}, b)).$$

Moreover, the lower bound of $R[h]$ is bounded from above as

$$\nu - G(\alpha_{1-\nu}(\boldsymbol{w}, b)) \leq \nu - G(\phi_{1-\nu}(\boldsymbol{w}, b)).$$

A proof of the above theorem is also omitted since the proof resembles to Theorem 3. Theorem 5 implies that the lower bound $\nu - G(\alpha_{1-\nu}(\boldsymbol{w}, b))$ of the generalization error is upper-bounded by $\nu - G(\phi_{1-\nu}(\boldsymbol{w}, b))$. On the other hand, Eq.(5) and $\alpha_{1-\nu}(\boldsymbol{w}, b) > 0$ yields $\phi_{1-\nu}(\boldsymbol{w}, b) > 0$. Thus minimizing $\phi_{1-\nu}(\boldsymbol{w}, b)$ by Eν-SVC may contribute to lowering the lower bound $\nu - G(\alpha_{1-\nu}(\boldsymbol{w}, b))$ of the generalization error.

4. New Optimization Algorithm

As reviewed in Section 2.5, Eν-SVC involves a non-convex optimization problem. In this section, we give a new efficient optimization procedure for Eν-SVC. Our proposed procedure involves two optimization algorithms depending on the value of ν. We first describe the two algorithms and then show how these two algorithms are chosen for practical use.

4.1. Optimization When $\nu \in (\overline{\nu}, \nu_{\max}]$

Lemma 6 *When $\nu \in (\overline{\nu}, \nu_{\max}]$, the E$\nu$-SVC problem (3) is equivalent to*

$$\begin{aligned} &\min_{\boldsymbol{w}, b, \boldsymbol{\xi}, \rho} \ -\nu\rho + \frac{1}{m}\sum_{i=1}^{m} \xi_i \\ &\text{s.t. } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq \rho - \xi_i, \ \xi_i \geq 0, \ \|\boldsymbol{w}\|^2 \leq 1. \end{aligned} \quad (8)$$

This lemma shows that the equality constraint $\|\boldsymbol{w}\|^2 = 1$ in the original problem (3) can be replaced by $\|\boldsymbol{w}\|^2 \leq 1$ without changing the solution. Due to the convexity of $\|\boldsymbol{w}\|^2 \leq 1$, the above optimization problem is convex and therefore we can easily obtain the global solution by a standard optimization software.

4.2. Optimization When $\nu \in (0, \overline{\nu}]$

If $\nu \in (0, \overline{\nu}]$, the E$\nu$-SVC optimization problem is essentially non-convex and therefore we need a more elaborate algorithm.

4.2.1. Local Optimum Search

Here, we propose the following iterative algorithm for finding a local optimum.

Algorithm 7 (The Eν-SVC local optimum search algorithm for $\nu \in (0, \overline{\nu}]$)

Step 1: Initialize $\widetilde{\boldsymbol{w}}$.
Step 2: Solve the following linear program:

$$\begin{aligned} &\min_{\boldsymbol{w}, b, \boldsymbol{\xi}, \rho} \ -\nu\rho + \frac{1}{m}\sum_{i=1}^{m} \xi_i \\ &\text{s.t. } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq \rho - \xi_i, \ \xi_i \geq 0, \ \langle \widetilde{\boldsymbol{w}}, \boldsymbol{w} \rangle = 1, \end{aligned} \quad (9)$$

and let the optimal solution be $(\widehat{\boldsymbol{w}}, \widehat{b}, \widehat{\boldsymbol{\xi}}, \widehat{\rho})$.
Step 3: If $\widetilde{\boldsymbol{w}} = \widehat{\boldsymbol{w}}$, terminate and output $\widetilde{\boldsymbol{w}}$. Otherwise, update $\widetilde{\boldsymbol{w}}$ by $\widetilde{\boldsymbol{w}} \longleftarrow \widehat{\boldsymbol{w}}/\|\widehat{\boldsymbol{w}}\|$.
Step 4: Repeat Steps 2–3.

The linear program (9) is the same as the one proposed by Perez-Cruz et al. (2003), i.e., the equality constrained $\|\boldsymbol{w}\|^2 = 1$ of the original problem (3) is

replaced by $\langle \widetilde{\boldsymbol{w}}, \boldsymbol{w} \rangle = 1$. The updating rule of $\widetilde{\boldsymbol{w}}$ in Step 3 is different from the one proposed by Perez-Cruz et al. (2003) (cf. Eq.(4)).

We define a "corner" (or "0-dimensional face") of Eν-SVC (3) as the intersection of an edge of the polyhedral cone formed by linear constraints of (3) and $\|\boldsymbol{w}\|^2 = 1$. Under the new update rule, the algorithm visits a corner of Eν-SVC (3) in each iteration. Since Eν-SVC has finite corners, we can show that Algorithm 7 with the new update rule terminates in a finite number of iterations, i.e., less than or equal to the number of corners of Eν-SVC.

Theorem 8 *Algorithm 7 terminates within a finite number of iterations of Steps 2–3. Furthermore, a solution of the modified Eν-SVC algorithm is a local minimizer if it is unique and non-degenerate.*

4.2.2. Global Optimum Search

Next, we show that the global solution can be actually obtained within finite iterations, despite the non-convexity of the optimization problem.

A naive approach to searching for the global solution is to run the local optimum search algorithm many times with different initial values and choose the best local solution. However, there is no guarantee that this naive approach can find the global solution. Below, we give a more systematic way to find the global solution based on the following lemma.

Lemma 9 *When $\nu \in (0, \overline{\nu}]$, the E$\nu$-SVC problem* (3) *is equivalent to*

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}, \rho} \; -\nu\rho + \frac{1}{m}\sum_{i=1}^{m} \xi_i$$
$$\text{s.t. } y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq \rho - \xi_i,\ \xi_i \geq 0,\ \|\boldsymbol{w}\|^2 \geq 1. \quad (10)$$

Lemma 9 could be proved in a similar way as Lemma 6, so we omit the proof. This lemma shows that the equality constraint $\|\boldsymbol{w}\|^2 = 1$ in the original Eν-SVC problem (3) can be replaced by $\|\boldsymbol{w}\|^2 \geq 1$ without changing the solution if $\nu \in (0, \overline{\nu}]$.

The problem (10) is called a *linear reverse convex program* (LRCP), which is a class of non-convex problems consisting of linear constraints and one concave inequality ($\|\boldsymbol{w}\|^2 \geq 1$ in the current case). The feasible set of the problem (10) consists of a finite number of *faces*. For LRCPs, Horst and Tuy (1995) showed that the local optimal solutions correspond to 0-dimensional faces (or corners). This implies that all the local optimal solutions of the Eν-SVC problem (10) can be traced by checking all the faces.

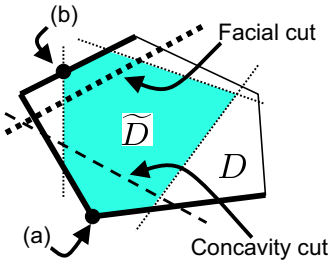


Figure 3. A 0-dimensional face (a) and three proper faces (bold solid lines) of D are identified in $\widetilde{D}$. If the corner (a) is found in Step 2, a concavity cut is constructed. If the corner (b) is found, a facial cut is constructed. If these two cuts are added to $\widetilde{D}$, the remaining area includes no face of D.

Let D be the feasible set of Eν-SVC (3). Below, we summarize the Eν-SVC training algorithm based on the *cutting plane method*, which is an efficient method of tracing faces.

Algorithm 10 (The Eν-SVC global optimum search algorithm for $\nu \in (0, \overline{\nu}]$)

Step 1: $\widetilde{D} \longleftarrow D$.
Step 2: Find a local solution by Algorithm 7.
Step 3: Identify a face of D in $\widetilde{D}$ that corresponds the local solution.
Step 4a: If the face is a corner, construct a "concavity cut".
Step 4b: If the face is a proper face, construct a "facial cut".
Step 5: Add the cut to the problem (9) and $\widetilde{D}$.
Step 6: Repeat Steps 2–5 until $\widetilde{D}$ includes no face of D.
Step 7: Output the best local optimal solution as the global solution.

If the local solution obtained in Step 2 is a corner of D (i.e., the local solution is not on any cutting plane as (a) in Figure 3), a *concavity cut* (Horst & Tuy, 1995) is constructed. The concavity cut has a role of removing the local solution, i.e., a 0-dimensional face of D and its neighborhood. Otherwise, a *facial cut* (Majthay & Whinston, 1974) is constructed to eliminate the proper face (see (b) in Figure 3).

Since the total number of distinct faces of D is finite in the current setting and a facial cut or a concavity cut eliminates at least one face at a time, Algorithm 10 is guaranteed to terminate within finite iterations (precisely, less than or equal to the number of all dimensional faces of Eν-SVC). Furthermore, since the addition of a concavity cut or a facial cut does not remove local solutions which are better than the best local solution found so far, Algorithm 10 is guaranteed to

trace all *sufficient* local solutions. Thus we can always find a global solution within finite iterations by Algorithm 10. A more detailed discussion on the concavity cut and the facial cut is shown in Horst and Tuy (1995) and Majthay and Whinston (1974), respectively.

4.3. Choice of Two Algorithms

We have two convergent algorithms when $\nu \in (\overline{\nu}, \nu_{\max}]$ and $\nu \in (0, \overline{\nu}]$. Thus, choosing a suitable algorithm depending on the value of ν would be an ideal procedure. However, the value of the threshold $\overline{\nu}$ is difficult to explicitly compute since it is defined via the optimal value $\phi_{1-\overline{\nu}}(\boldsymbol{w}^*, b^*)$ (see Figure 2). Therefore, it is not straightforward to choose a suitable algorithm for a given ν.

When we use Eν-SVC in practice, we usually compute the solutions for several different values of ν and choose the most promising one based on, e.g., cross-validation. In such scenarios, we can properly switch two algorithms without explicitly knowing the value of $\overline{\nu}$—our key idea is that the solution of the problem (8) is non-trivial (i.e., $\boldsymbol{w} \neq \boldsymbol{0}$) if and only if $\nu \in (\overline{\nu}, \nu_{\max}]$. Thus if the solutions are computed from large ν to small ν, the switching point can be identified by checking the triviality of the solution. The proposed algorithm is summarized as follows.

Algorithm 11 (The Eν-SVC algorithm for ($\nu_{\max} \geq$) $\nu_1 > \nu_2 > \cdots > \nu_k > 0$)

Step 1: $i \longleftarrow 1$.
Step 2: Compute $(\boldsymbol{w}^*, b^*)$ for ν_i by solving (8).
Step 3a: If $\boldsymbol{w}^* \neq \boldsymbol{0}$, accept $(\boldsymbol{w}^*, b^*)$ as the solution for ν_i, increment i, and go to Step 2.
Step 3b: If $\boldsymbol{w}^* = \boldsymbol{0}$, reject $(\boldsymbol{w}^*, b^*)$.
Step 4: Compute $(\boldsymbol{w}^*, b^*)$ for ν_i by Algorithm 10.
Step 5: Accept $(\boldsymbol{w}^*, b^*)$ as the solution for ν_i, increment i, and go to Step 4 unless $i > k$.

5. Conclusions

We characterized the generalization error of Eν-SVC in terms of the *conditional value-at-risk* (CVaR, see Figure 1) and showed that a good generalization performance is expected by Eν-SVC. We then derived a globally convergent optimization algorithm even though the optimization problem involved in Eν-SVC is non-convex.

We introduced the threshold $\overline{\nu}$ based on the sign of the CVaR (see Figure 2). We can check that the problem (8) is equivalent to ν-SVC in the sense that they share the same negative optimal value in $(\overline{\nu}, \nu_{\max}]$ and $(\nu_{\min}, \nu_{\max}]$, respectively (Gotoh & Takeda, 2005). On the other hand, the problem (8) and ν-SVC have the zero optimal value in $(0, \overline{\nu}]$ and $[0, \nu_{\min}]$, respectively. Thus, although the definitions of $\overline{\nu}$ and $\nu_{\min}$ are different, they would be essentially the same. We will study the relation between $\overline{\nu}$ and $\nu_{\min}$ in more detail in the future work.

References

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *COLT* (pp. 144–152). ACM Press.

Chang, C.-C., & Lin, C.-J. (2001). Training ν-support vector classifiers: Theory and algorithms. *Neural Computation*, *13*, 2119–2147.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297.

Crisp, D. J., & Burges, C. J. C. (2000). A geometric interpretation of ν-SVM classifiers. *NIPS 12* (pp. 244–250). MIT Press.

Gotoh, J., & Takeda, A. (2005). A linear classification model based on conditional geometric score. *Pacific Journal of Optimization*, *1*, 277–296.

Horst, R., & Tuy, H. (1995). *Global optimization: Deterministic approaches*. Berlin: Springer-Verlag.

Majthay, A., & Whinston, A. (1974). Quasi-concave minimization subject to linear constraints. *Discrete Mathematics*, *9*, 35–59.

Perez-Cruz, F., Weston, J., Hermann, D. J. L., & Schölkopf, B. (2003). Extension of the ν-SVM range for classification. *Advances in Learning Theory: Methods, Models and Applications 190* (pp. 179–196). Amsterdam: IOS Press.

Rockafellar, R. T., & Uryasev, S. (2002). Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, *26*, 1443–1472.

Schölkopf, B., Smola, A., Williamson, R., & Bartlett, P. (2000). New support vector algorithms. *Neural Computation*, *12*, 1207–1245.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Berlin: Springer-Verlag.

A. Sketch of Proof of Theorem 1

Let $(\boldsymbol{w}^*, b^*, \alpha^*)$ be the optimal solution of

$$\min_{\boldsymbol{w}, b, \alpha} F_\beta(\boldsymbol{w}, b, \alpha), \tag{11}$$

where, for $[X]^+ := \max\{X, 0\}$,

$$F_\beta(\boldsymbol{w}, b, \alpha) := \alpha + \frac{\sum_{i \in M} [f(\boldsymbol{w}, b; \boldsymbol{x}_i, y_i) - \alpha]^+}{(1-\beta)m}. \tag{12}$$

Then Rockafellar and Uryasev (2002) showed that

$$F_\beta(\boldsymbol{w}^*, b^*, \alpha^*) = \phi_\beta(\boldsymbol{w}^*, b^*) = \min_{\boldsymbol{w}, b} \phi_\beta(\boldsymbol{w}, b), \qquad (13)$$

i.e, the problems (6) and (11) are equivalent.

Introducing slack variables ξ_i, imposing $\|\boldsymbol{w}\|^2 = 1$ (which does not change the solution essentially; only the scale is changed), and letting $\nu = 1 - \beta$ and $\rho = -\alpha$ in Eq.(11), we establish the theorem.

B. Sketch of Proof of Lemma 2

Since Eq.(11) only involves continuous functions, continuity of $F_\beta(\boldsymbol{w}^*, b^*, \alpha^*)$ with respect to β is clear. From Eq.(13), $\phi_\beta(\boldsymbol{w}^*, b^*)$ is also continuous. Let $(\boldsymbol{w}^*_{\beta_i}, b^*_{\beta_i}, \alpha^*_{\beta_i})$ be the optimal solutions of Eq.(11) for $0 < \beta_1 < \beta_2 < 1$. Then we have

$$\begin{aligned}\phi_{\beta_1}(\boldsymbol{w}^*_{\beta_1}, b^*_{\beta_1}) &= F_{\beta_1}(\boldsymbol{w}^*_{\beta_1}, b^*_{\beta_1}, \alpha^*_{\beta_1}) \leq F_{\beta_1}(\boldsymbol{w}^*_{\beta_2}, b^*_{\beta_2}, \alpha^*_{\beta_2}) \\ &< F_{\beta_2}(\boldsymbol{w}^*_{\beta_2}, b^*_{\beta_2}, \alpha^*_{\beta_2}) = \phi_{\beta_2}(\boldsymbol{w}^*_{\beta_2}, b^*_{\beta_2}),\end{aligned}$$

where the first inequality is due to optimality of $(\boldsymbol{w}^*_{\beta_1}, b^*_{\beta_1}, \alpha^*_{\beta_1})$ and the second strict inequality is clear from Eq.(12). Thus $\phi_\beta(\boldsymbol{w}^*, b^*)$ is strictly increasing with respect to β, implying that $\phi_{1-\nu}(\boldsymbol{w}^*, b^*)$ is strictly decreasing with respect to ν.

C. Sketch of Proof of Theorem 3

For a homogeneous classifier $h(\widetilde{\boldsymbol{x}}) = \mathrm{sign}(\langle \widetilde{\boldsymbol{w}}, \widetilde{\boldsymbol{x}} \rangle)$, the following lemma holds:

Lemma 12 *(Schölkopf et al., 2000) Suppose that support $\mathcal{X}$ of $\widetilde{\boldsymbol{x}}$ is in a ball of radius $\widetilde{R}$ around the origin. Then, for all $\widetilde{\boldsymbol{w}}$ such that $\|\widetilde{\boldsymbol{w}}\| = 1$, there exists a positive constant c such that the following bound holds with probability at least $1 - \delta$:*

$$\begin{aligned}R[h] \leq & \frac{|\{i \mid y_i \langle \widetilde{\boldsymbol{w}}, \widetilde{\boldsymbol{x}}_i \rangle < \widetilde{\gamma}\}|}{m} \\ & + \sqrt{\frac{2}{m}\left(\frac{4c^2\widetilde{R}^2}{\widetilde{\gamma}^2}\log_2(2m) - 1 + \log\frac{2}{\delta}\right)}.\end{aligned}$$

Let $\widetilde{\boldsymbol{w}} = \frac{(\boldsymbol{w}^\top, b)^\top}{\sqrt{1+b^2}}$ and $\widetilde{\boldsymbol{x}} = (\boldsymbol{x}^\top, 1)^\top$. Then our classifier (1) can be regarded as homogeneous. The assumption that all the data points $\boldsymbol{x}$ live in a centered ball of radius R implies that all the data points $\widetilde{\boldsymbol{x}}$ live in a centered ball of radius

$$\widetilde{R} = \sqrt{R^2 + 1}.$$

The assumption $\|\boldsymbol{w}\| = 1$ implies $\|\widetilde{\boldsymbol{w}}\| = 1$. Then we can apply Lemma 12 to the current setting. The condition $y_i \langle \widetilde{\boldsymbol{w}}, \widetilde{\boldsymbol{x}}_i \rangle < \widetilde{\gamma}$ results in

$$y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) < \widetilde{\gamma}\sqrt{1+b^2} := \gamma.$$

When all the data points $\boldsymbol{x}$ live in a centered ball of radius R, we can assume without loss of generality that $|b| \leq R$. Then we have

$$\frac{1}{\widetilde{\gamma}^2} = \frac{1+b^2}{\gamma^2} \leq \frac{1+R^2}{\gamma^2}.$$

Now let us set

$$\gamma = -\alpha_{1-\nu}(\boldsymbol{w}, b).$$

Then we can show that

$$\tfrac{1}{m}|\{i \mid y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) < -\alpha_{1-\nu}(\boldsymbol{w}, b)\}| \leq \nu.$$

We omit its proof due to lack of space. Then we obtain the upper bound $\nu + G(\alpha_{1-\nu}(\boldsymbol{w}, b))$; the upper bound $\nu + G(\phi_{1-\nu}(\boldsymbol{w}, b))$ is clear from Eq.(5).

D. Sketch of Proof of Lemma 6

Since the difference between the problems (3) and (8) is only the norm constraint of $\boldsymbol{w}$, it is enough to show that for $\nu \in (\overline{\nu}, \nu_{\max}]$, $\|\boldsymbol{w}^*\|^2 = 1$ holds at the optimal solution $(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*, \rho^*)$ of the problem (8). For such ν, $\phi_{1-\nu}(\boldsymbol{w}^*, b^*) < 0$ holds, i.e., the optimal value of Eν-SVC is negative. If we suppose $\|\boldsymbol{w}^*\|^2 < 1$, another feasible solution $(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*, \rho^*)/\|\boldsymbol{w}^*\|$ achieves a smaller optimal value than $(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*, \rho^*)$. This contradicts to the optimality of (8), and hence $\|\boldsymbol{w}^*\|^2 = 1$ is proved.

E. Sketch of Proof of Theorem 8

Let $(\widehat{\boldsymbol{w}}_k, \widehat{b}_k, \widehat{\boldsymbol{\xi}}_k, \widehat{\rho}_k)$ be an optimal solution of the linear program (9) in the k-th iteration. Then, a feasible solution of Eν-SVC (3) is given by

$$(\widetilde{\boldsymbol{w}}_k, \widetilde{b}_k, \widetilde{\boldsymbol{\xi}}_k, \widetilde{\rho}_k) = (\widehat{\boldsymbol{w}}_k, \widehat{b}_k, \widehat{\boldsymbol{\xi}}_k, \widehat{\rho}_k)/\|\widehat{\boldsymbol{w}}_k\|.$$

Since $(\widehat{\boldsymbol{w}}_k, \widehat{b}_k, \widehat{\boldsymbol{\xi}}_k, \widehat{\rho}_k)$ is at a corner of the feasible set of the linear program (9), $(\widetilde{\boldsymbol{w}}_k, \widetilde{b}_k, \widetilde{\boldsymbol{\xi}}_k, \widetilde{\rho}_k)$ is also a corner of the feasible set of Eν-SVC (3).

Let $q(\cdot)$ be the objective function of Eν-SVC (3), which is also the objective function of the linear program (9). Then we have

$$q(\widetilde{\boldsymbol{\xi}}_{k-1}, \widetilde{\rho}_{k-1}) > q(\widehat{\boldsymbol{\xi}}_k, \widehat{\rho}_k) \geq q(\widetilde{\boldsymbol{\xi}}_k, \widetilde{\rho}_k) = q(\widehat{\boldsymbol{\xi}}_k, \widehat{\rho}_k)/\|\widehat{\boldsymbol{w}}_k\|,$$

where the first inequality comes from the optimality of $(\widehat{\boldsymbol{\xi}}_k, \widehat{\rho}_k)$ of the linear program (9). The second inequality comes from $\|\widehat{\boldsymbol{w}}_k\| > 1$, which is ensured by $\langle \widetilde{\boldsymbol{w}}_{k-1}, \widehat{\boldsymbol{w}}_k \rangle = 1$. Thus the algorithm finds a distinct corner of Eν-SVC (3) in each iteration. Since the number of corners of Eν-SVC (3) is finite, the algorithm terminates within finite iterations.

Let $\Delta\boldsymbol{d} = (\Delta\boldsymbol{w}^\top\ \Delta b^\top\ \Delta\rho^\top\ \Delta\boldsymbol{\xi}^\top)^\top$ be a perturbation from the solution $\boldsymbol{d}^* = (\boldsymbol{w}^*, b^*, \rho^*, \boldsymbol{\xi}^*)$ of Algorithm 7. Note that $\boldsymbol{d}^*$ is an optimal solution of the linear program (9) with $\widetilde{\boldsymbol{w}} = \boldsymbol{w}^*$. Using the Karush-Kuhn-Tucker (KKT) optimality conditions, we can express the increase Δq of the objective value as

$$\begin{aligned}\Delta q &:= -\nu\Delta\rho + \frac{1}{m}\sum_{i \in M}\Delta\xi_i \\ &= \Delta\boldsymbol{d}^\top \left(\begin{array}{ccc|c} y_1\boldsymbol{x}_1 & \dots & y_m\boldsymbol{x}_m & \boldsymbol{O} \\ y_1 & \dots & y_m & \boldsymbol{0} \\ 1 & \dots & 1 & \boldsymbol{0} \\ & \boldsymbol{I} & & \boldsymbol{I} \end{array}\right) \begin{pmatrix} \boldsymbol{\lambda}^* \\ \boldsymbol{\mu}^* \end{pmatrix} - \delta^*\Delta\boldsymbol{w}^\top\boldsymbol{w}^*,\end{aligned}$$

where $\boldsymbol{\lambda}^* \in \mathbb{R}^m_+$, $\boldsymbol{\mu}^* \in \mathbb{R}^m_+$, and $\delta^* \leq 0$ are KKT multipliers. If $\Delta\boldsymbol{d}$ is a feasible perturbation (i.e., $\boldsymbol{d}^* + \Delta\boldsymbol{d}$ is feasible), we can show that $\Delta q > 0$ (we omit its proof due to lack of space), which implies that $\boldsymbol{d}^*$ is locally optimal.

Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient

Tijmen Tieleman TIJMEN@CS.TORONTO.EDU
Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4, Canada

Abstract

A new algorithm for training Restricted Boltzmann Machines is introduced. The algorithm, named Persistent Contrastive Divergence, is different from the standard Contrastive Divergence algorithms in that it aims to draw samples from almost exactly the model distribution. It is compared to some standard Contrastive Divergence and Pseudo-Likelihood algorithms on the tasks of modeling and classifying various types of data. The Persistent Contrastive Divergence algorithm outperforms the other algorithms, and is equally fast and simple.

1. Introduction

Restricted Boltzmann Machines (RBMs) (Hinton et al., 2006; Smolensky, 1986) are neural network models for unsupervised learning, but have recently seen a lot of application as feature extraction methods for supervised learning algorithms (Salakhutdinov et al., 2007; Larochelle et al., 2007; Bengio et al., 2007; Gehler et al., 2006; Hinton et al., 2006; Hinton & Salakhutdinov, 2006). The success of these models raises the issue of how best to train them.

Most training algorithms are based on gradient descent, but the standard objective function (training data likelihood) is intractable, so the algorithms differ in their choice of approximation to the gradient of the objective function. At present, the most popular gradient approximation is the Contrastive Divergence (CD) approximation (Hinton et al., 2006; Hinton, 2002; Bengio & Delalleau, 2007); more specifically the CD-1 approximation. However, it is not obvious whether it is the best. For example, the CD algorithm has a parameter specifying the number of Markov Chain transitions performed, and although the most commonly chosen value is 1, other choices are possible and reasonable, too (Carreira-Perpinan & Hinton, 2005).

In this paper, a new gradient approximation algorithm is presented and compared to a variety of CD-based algorithms. The quantitative measures of test data likelihood (for unsupervised learning) and classification error rate (for supervised learning) are investigated, and the type of feature detectors that are developed are also shown. We find that the new algorithm produces more meaningful feature detectors, and outperforms the other algorithms.

The RBMs on which these experiments were done all had binary units. However, this special case can easily be generalized to other *harmoniums* (Smolensky, 1986; Welling et al., 2005) in which the units have Gaussian, Poisson, multinomial, or other distributions in the exponential family, and the training algorithms described here require only minor modifications to work in most of those models.

In Section 2, the RBM model and CD gradient estimator are discussed. In Section 3, the Persistent Contrastive Divergence algorithm is introduced. In Sections 4 and 5, the experiments and results are described, and Section 6 concludes with a discussion and some plans for future work.

2. RBMs and the CD Gradient Approximation

2.1. Restricted Boltzmann Machines

An RBM is an energy-based model for unsupervised learning (Hinton, 2002; Smolensky, 1986). It consists of two layers of binary units: one *visible*, to represent the data, and one *hidden*, to increase learning capacity. Standard notation is to use i for indices of visible units, j for indices of hidden units, and w_{ij} for the strength of the connection between the i^{th} visible unit and the j^{th} hidden unit. If v_i denotes the state of the

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

i^{th} visible unit, and h_j denotes the state of the j^{th} hidden unit, an *energy* function is defined on states: $E(v,h) = -\sum_{i,j} v_i h_j w_{ij} - \sum_i v_i b_i - \sum_j h_j b_j$, where b stands for the biases. Through these energies, probabilities are defined as $P(v,h) = \frac{e^{-E(v,h)}}{Z}$ where Z is the normalizing constant $Z = \sum_{x,y} e^{-E(x,y)}$. The probability of a data point (represented by the state v of the visible layer) is defined as the marginal: $P(v) = \sum_h P(v,h) = \frac{\sum_h e^{-E(v,h)}}{Z}$. Thus, the training data likelihood, using just one training point for simplicity, is $\phi = \log P(v) = \phi^+ - \phi^-$ where $\phi^+ = \log \sum_h e^{-E(v,h)}$ and $\phi^- = \log Z = \log \sum_{x,y} e^{-E(x,y)}$. The *positive* gradient $\frac{\partial \phi^+}{\partial w_{ij}}$ is simple: $\frac{\partial \phi^+}{\partial w_{ij}} = v_i \cdot P(h_j = 1|v)$. The *negative* gradient $\frac{\partial \phi^-}{\partial w_{ij}} = P(v_i = 1, h_j = 1)$, however, is intractable. If we could get samples from the model, we could Monte Carlo approximate it, but even getting those samples is intractable.

2.2. The Contrastive Divergence Gradient Approximation

To get a tractable approximation of $\frac{\partial \phi^-}{\partial w_{ij}}$, one uses some algorithm to *approximately* sample from the model. The Contrastive Divergence (CD) algorithm is one way to do this. It is designed in such a way that at least the *direction* of the gradient estimate is somewhat accurate, even when the size is not. CD-1 is, at present, the most commonly used algorithm for training RBMs. One of the algorithms we compare is regular CD-1; another is CD-10, which is generally considered to be better if the required computer time is available.

A variation on CD is *mean field* CD (Welling & Hinton, 2002), abbreviated MF CD. This has the advantage of being a deterministic gradient estimate, which means that larger learning rates can be used. We include mean field CD-1 in the comparison.

3. The Persistent Contrastive Divergence Algorithm

CD-1 is fast, has low variance, and is a reasonable approximation to the likelihood gradient, but it is still significantly different from the likelihood gradient when the mixing rate is low. This can be seen by drawing samples from the distribution that it learns (see Figure 4). Generally speaking, CD-n for greater n is preferred over CD-1, if enough running time is available. In Neal's 1992 paper about Sigmoid Belief Networks (1992), a solution is suggested for such situations. In the context of RBMs, the idea is as follows (see also (Yuille, 2004)).

What we need for approximating $\frac{\partial \phi^-}{\partial w_{ij}}$ is a sample from the model distribution. The standard way to get it is by using a Markov Chain, but running a chain for many steps is too time-consuming. However, between parameter updates, the model changes only slightly. We can take advantage of that by initializing a Markov Chain at the state in which it ended for the previous model. This initialization is often fairly close to the model distribution, even though the model has changed a bit in the parameter update. Neal uses this approach with Sigmoid Belief Networks to approximately sample from the posterior distribution over hidden layer states given the visible layer state. For RBMs, the situation is a bit simpler: there is only one distribution from which we need samples, as opposed to one distribution per training data point. Thus, the algorithm can be used to produce gradient estimates *online* or using mini-batches, using only a few training data points for the positive part of each gradient estimate, and only a few 'fantasy' points for the negative part. The fantasy points are updated by one full step of the Markov Chain each time a mini-batch is processed.

Of course this still is an approximation, because the model does change slightly with each parameter update. With infinitesimally small learning rate it becomes exact, and in general it seems to work best with small learning rates.

We call this algorithm *Persistent Contrastive Divergence* (PCD), to emphasize that the Markov Chain is not reset between parameter updates.

4. Experiments

We did a variety of experiments, using different data sets (digit images, emails, artificial data, horse image segmentations, digit image patches), different models (RBMs, classification RBMs, fully visible Markov Random Fields), different training procedures (PCD, CD-1, CD-10, MF CD, pseudo likelihood), and different tasks (unsupervised vs. supervised learning).

4.1. Data Sets

The first data set that we used was the MNIST dataset of handwritten digit images (LeCun & Cortes,). The images are 28 by 28 pixels, and the data set consists of 60,000 training cases and 10,000 test cases. To have a validation set, we split the official training set of 60,000 cases into a training set of 50,000 cases and a validation set of 10,000 cases. To have binary data, we treat the pixel intensities as probabilities. Each time a binary data point is required, a real-valued MNIST im-

age is binarized by sampling from the given Bernoulli distribution for each pixel. Thus, in effect, our data set is a mixture of 70,000 factorial distributions: one for each of the data points in the MNIST data set.

Another data set was obtained by taking small patches of 5 by 5 pixels, from the MNIST images. To have somewhat smooth-looking data, we binarized by thresholding at 1/2. The 70,000 MNIST data points were thus turned into 70,000 times $(28 - 5 + 1)^2$ is 4,032,000 patches. This data set was split into training (60%), validation (20%), and test (20%) sets.

A data set consisting of descriptions of e-mails was made available by Sam Roweis. It describes 5,000 e-mails using a variety of binary features - mostly word presence vs. absence features. The e-mails are labeled as spam or non-spam.

An artificial data set was created by combining the outlines of rectangles and triangles. Because this data set is artificially generated, there is an infinite amount of it, which helps shed some light on the reasons for using weight decay regularization.

Lastly, we used a data set of image segmentations: in pictures of horses, the segmentation indicates which pixels are part of the horse and which are background (Borenstein et al., 2004). By using only the segmentation, we have a binary data set.

4.2. Models

The first model we used is an RBM, exactly as described above. For the MNIST and horse segmentation data sets, we used 500 hidden units; for the artificial data set we used 100.

One of the evaluations is how well the learned RBM models the test data, i.e. log likelihood. This is intractable for regular size RBMs, because the time complexity of that computation is exponential in the size of the smallest layer (visible or hidden). One experiment, therefore, was done using only 25 hidden units, so that log likelihood could be calculated exactly in about two hours. Another experiment uses an approximate assessment of the normalization constant Z, that was developed recently in our group (Salakhutdinov & Murray, 2008). This algorithm works for any number of hidden units, but its reliability has not been researched extensively. Nonetheless, it seems to give a reasonable indication, and can be used to complement other results.

RBMs, however, are models for unsupervised learning, so for classification we used a slightly different model, described in more detail in (Hinton et al., 2006). We used an RBM with one added visible unit, which represented the label. The training data points are then combinations of inputs with their labels, and testing is done by choosing the most likely label given the input, under the learned model. This model we call a 'classification RBM'. Note that the label unit is not necessarily binary (although in the spam classification task it is). In the MNIST classification task it is multinomial: it can have 10 different values. This, however, does not significantly change the algorithms (Hinton, 2002). For MNIST classification we used 500 hidden units; for spam classification we used 100.

The third model we tested is significantly different: a fully visible, fully connected Markov Random Field (MRF) (see for example (Wainwright & Jordan, 2003)). One can use the PCD algorithm on it, although it looks a bit different in this case. We compared its performance to the more commonly used Pseudo-Likelihood optimization algorithm (Besag, 1986). To have exact test data log likelihood measurements, we used small models, with only 25 units.

4.3. The Mini-batch Optimization Procedure

We used the *mini-batch* learning procedure: we only used a small number of training points for each gradient estimate. We used 100 training points in each mini-batch for most data sets.

4.4. Algorithm Details

The PCD algorithm can be implemented in various ways. One could, for example, choose to randomly reset some of the Markov Chains at regular intervals. Initial tests showed that the best implementation is as follows: no Markov Chains get reset; one full Gibbs update is done on each of the Markov Chains for each gradient estimate; and the number of Markov Chains is equal to the number of training data points in a mini-batch.

PCD for fully visible MRFs is a bit different from PCD for RBMs. A pleasant difference is that $\frac{\partial \phi^+}{\partial \theta}$ is constant, so it can be precomputed for the entire training set. Thus, no variance results from the use of mini-batches, and the training set can be discarded after $\frac{\partial \phi^+}{\partial \theta}$ is computed over it. An unpleasant difference is that the Markov Chain defined by Gibbs sampling has slower mixing: MRFs with connections between the visible units lack the pleasant property of RBMs that all visible units can be updated at the same time.

A Pseudo-Likelihood (PL) gradient computation requires more work than a PCD gradient computation, because it requires a logistic regression gradient esti-

mate for each of the units. As a result, we found that using mini-batches of 50 training points instead of 100 took only a little bit more time per training point, and did allow updating the model parameters almost twice as often, which is preferable in the mini-batch optimization procedure.

4.5. Other Technical Details

The learning rates used in the experiments are not constant. In practice, decaying learning rates often work better. In these experiments, the learning rate was linearly decayed from some initial learning rate to zero, over the duration of the learning. Preliminary experiments showed that this works better than the $\frac{1}{t}$ schedule suggested in theoretical work by (Robbins & Monro, 1951), which is preferable when infinitely much time is available for the optimization.

Some experiment parameters, such as the number of hidden units, and the size of the mini-batches, were fixed. However, the initial learning rate was chosen using a validation set, as was weight decay for the (shorter) experiments on the spam, horses, MNIST patches, and artificial data sets. For each algorithm, each task, and each training duration, 30 runs were performed with evaluation on validation data, trying to find the settings that worked best. Then a choice of initial learning rate and, for the shorter experiments, weight decay, were made, and with those chosen settings, 10 more runs were performed, evaluating on test data. This provided 10 test performance numbers, which were summarized by their average and standard deviation (shown as error bars).

5. Results

5.1. The three MNIST Tasks

The results on the three MNIST tasks are shown in Figures 1, 2, and 3.

It is clear that PCD outperforms the other algorithms. PCD, CD-1, and MF CD all take approximately the same amount of time per gradient estimate, with MF CD being a little bit faster because it does not have to create random numbers. CD-10 takes about four times as long as PCD, CD-1, and MF CD, but it is indeed better than CD-1.

While CD-1 is good for some purposes, it is substantially different from the true likelihood gradient. This can be seen by drawing samples from an RBM that was trained with CD-1. Figure 4 shows those next to samples drawn from an RBM that was trained using PCD. It is clear that PCD is a better approximation

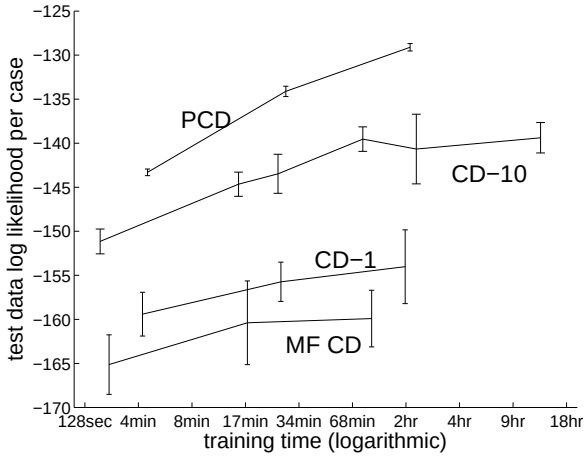


Figure 1. Modeling MNIST data with 25 hidden units (exact log likelihood)

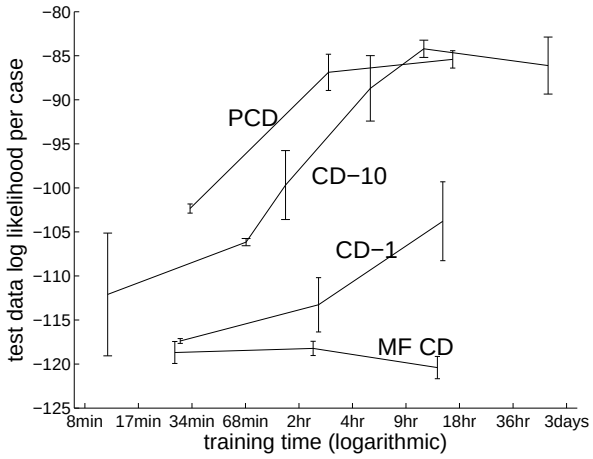


Figure 2. Modeling MNIST data with 500 hidden units (approximate log likelihood)

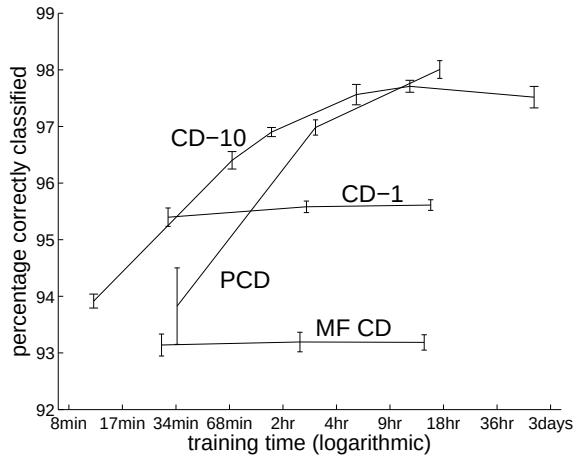


Figure 3. Classification of MNIST data

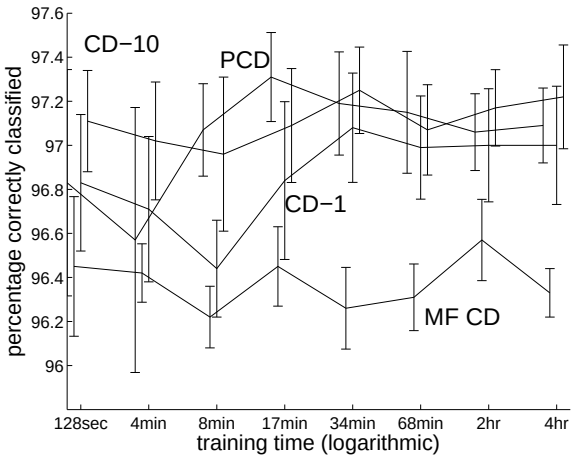


Figure 6. Classifying e-mail as spam versus non-spam

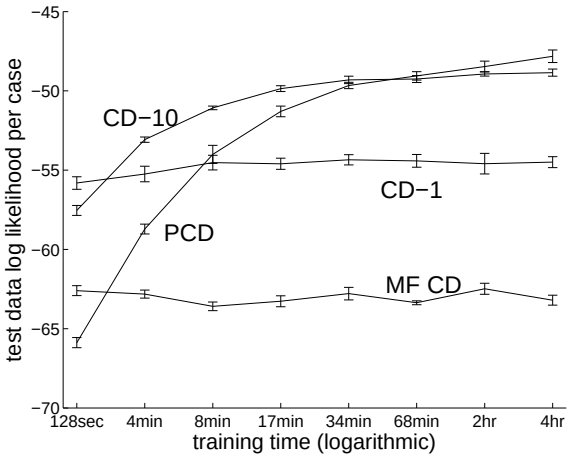


Figure 5. Modeling artificial data

to the likelihood gradient.

Classification is a particularly interesting task because it gives an indication of how well the model can extract relevant features from the input. RBMs are most often used as feature detectors, and this finding suggests that PCD creates feature detectors that give better classification than CD-1.

5.2. Modeling Artificial Data

In Figure 5 we see essentially the same as what happened on the MNIST tasks. MF CD is clearly the worst of the algorithms, CD-1 works better, and CD-10 and PCD work best, with CD-10 being preferable when little time is available and PCD being better if more time is available.

This data set was artificially generated, so there was an infinite amount of data available. Thus, one might think that the use of weight decay serves no purpose. However, all four algorithms did work best with some weight decay. The explanation for this is that CD algorithms are quite dependent on the mixing rate of the Markov Chain defined by the Gibbs sampler, and that mixing rate is higher when the parameters of the model are smaller. Thus, weight decay keeps the model mixing reasonably well, and makes CD algorithms work better. The effect is strongest for MF CD, which performs only one Gibbs update and does so without introducing noise. MF CD worked best with a weight decay strength of 10^{-3}. CD-1 does introduce some noise in the update procedure, and required less weight decay: $3 \cdot 10^{-4}$. CD-10 performs more updates, and is less dependent on the mixing rate. The best weight decay value for CD-10 turned out to be approximately $1.3 \cdot 10^{-4}$. Finally, the mixing mechanism used by PCD is even better, but it is still based on the Gibbs sampler, so it, too, works better with some weight decay. The best weight decay strength for PCD was approximately $2.5 \cdot 10^{-5}$.

5.3. Classifying E-mail Data

In Figure 6 the results on the e-mail classification task are shown. Because this is a small data set (5,000 data points in total, i.e. only 1000 test data points), we see that the error bars on the performace are quite large. Thus, we cannot carefully compare the performance of CD-1, CD-10, and PCD. We only see that MF CD is, again, not the best method.

However, we can conclude that RBMs can be used for this task, too, with acceptable performance, and that

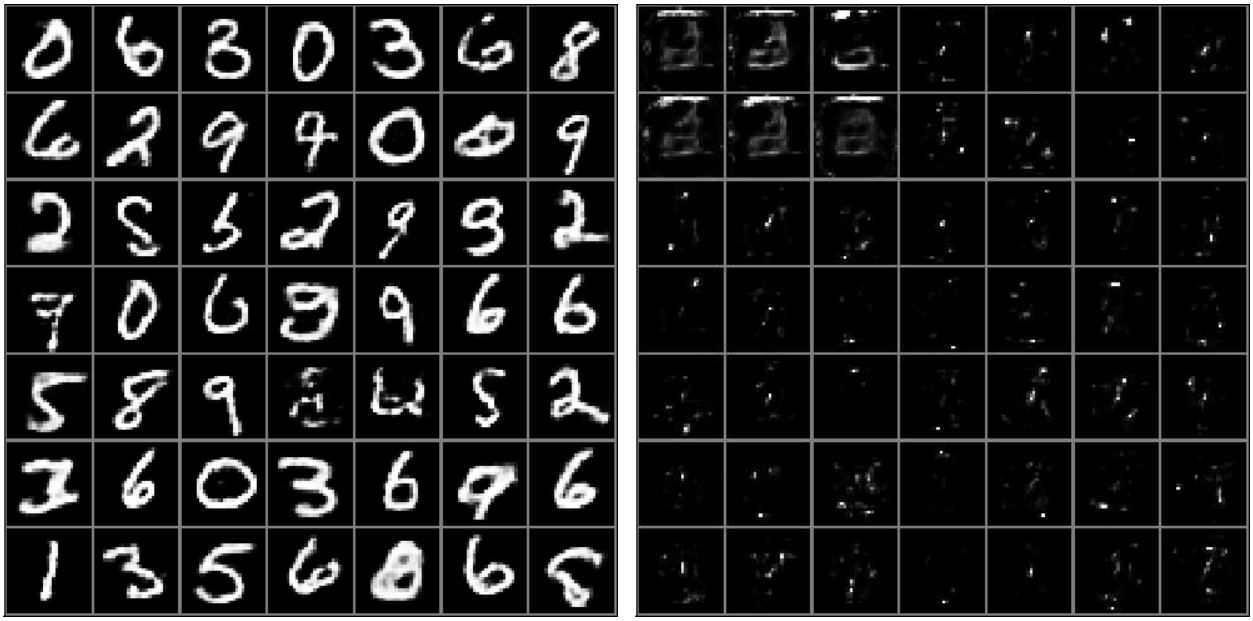


Figure 4. Samples from an RBM that was trained using PCD (left) and an RBM that was trained using CD-1 (right). Clearly, CD-1 did not produce an accurate model of the MNIST digits. Notice, however, that some of the CD-1 samples vaguely resemble a three.

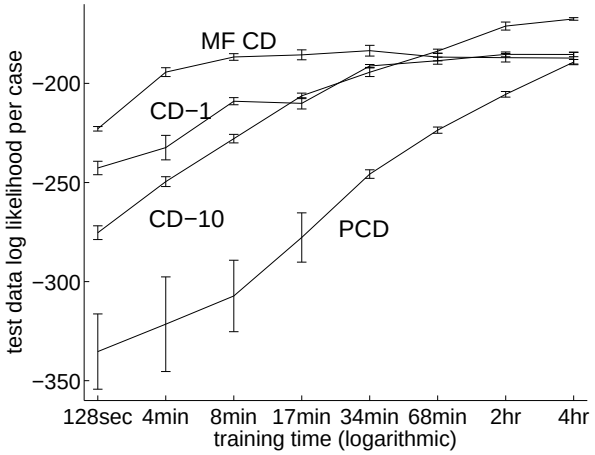


Figure 7. Modeling horse segmentation data

PCD is a reasonable choice of training algorithm.

5.4. Modeling Horse Contours

In Figure 7 we see a different picture: PCD is not the best algorithm here. The most plausible explanation is that although the same amount of training time was used, the data is much bigger: 1024 visible units, and 500 hidden units. Thus, there were 20 times as many connections in the RBM to be learned, which also means processing one mini-batch took more than 10 times as long as for the artificial data. Thus, we are essentially looking at a short optimization. Above, we already saw that CD-10 is better than PCD when little time is available, and that is confirmed here. We conjecture that, given significantly more training time, PCD would perform better than the other algorithms.

5.5. PCD on Fully Visible MRFs

To verify that PCD also works well with other models, we did some experiments with fully visible, fully connected MRFs. To be able to have exact test data likelihood evaluation, we made the MRFs small, and modeled 5 by 5 pixel patches from the MNIST digit images.

Pseudo-Likelihood (PL) training works reasonably well on this data set, but it does not produce the best probability models. Presumably this is simply because PL optimizes a different objective function. As a result, PL needed early stopping to prevent diverging too much from the data likelihood objective function, and the optimal learning rates are more or less inversely proportional to the duration of the optimization. Even with only a few seconds training time, the best test data likelihood is already achieved: -5.35.

PCD training does go more in the direction of the data likelihood function - asymptotically it gives its exact gradient. Thus, PCD did profit from having more time to run. Figure 8 shows the performance. The asymptotic value of approximately -5.15 does seem to be the best possible model: we also used exact gradient

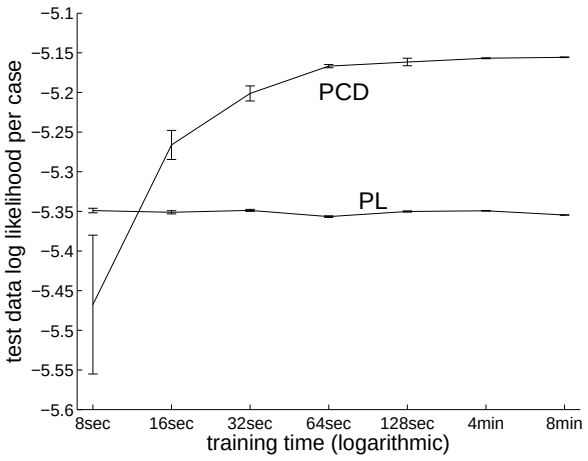


Figure 8. Training a fully visible MRF

optimization (which is slow, but possible), and this equally ended up with test data log likelihood of −5.15. However, the entropy of the training data distribution is significantly less than 5.15 'nats': it is 4.78 nats. This difference is probably due to the fact that the model has insufficient complexity to completely learn the training data distribution.

Incidentally, the training data log likelihood is only 0.004 better than the test data log likelihood - presumably because this data set is quite large and the model is quite small.

6. Discussion and Future Work

One issue not investigated is the use of weight decay. It is quite possible that the more approximate algorithms (such as CD-1 and MF CD) would benefit more from weight decay than CD-10 and PCD. In an RBM with zero weights, CD-1 and MF CD give exactly the likelihood gradient, and in general, in RBMs with small weights those algorithms give better approximations to the likelihood gradient than in RBMs with large weights. Weight decay keeps the weights small, and thus enables gradient estimates that approximate the likelihood gradient more closely. For many tasks, however, large weights may be required for good performance, so strong weight decay is undesirable if it can be avoided.

Also, the amount of training time used in these experiments is insufficient to find the asymptotic performance. In Figure 3 one can see, for example, that PCD clearly profits from more training time. To find out what its performance would be with more training time is future work, but we have seen runs (with more training time and more hidden units) where as few as 104 out of the 10,000 test cases were misclassified. Clearly, this is worth investigating further.

Another issue suggesting future work is that the classification RBMs in these experiments were not trained to maximize classification performance. They were trained to accurately model the joint distribution over images and labels. It is possible to train classification RBMs directly for classification performance; the gradient is fairly simple and certainly tractable. A natural way to use this classification error gradient is *after* training the RBM for joint density modeling. However, in preliminary experiments we found that this procedure begins to overfit very quickly (often after improving performance by less than 0.1%), so we did not include it in this paper. It is, however, still possible that *combining* the classification gradient with the density modeling gradient is a method that could yield more improvements. This is future work.

The main limitation of PCD is that it appears to require a low learning rate in order to allow the "fantasy" points to be sampled from a distribution that is close to the stationary distribution for the current weights. A theoretical analysis of this requirement can be found in (Yuille, 2004) and (Younes, 1999). Some preliminary experiments, however, suggest that PCD can be made to work well even when the learning rate is much larger than the one suggested by the asymptotic justification of PCD and we are currently exploring variations that allow much larger learning rates.

Acknowledgements

I thank Geoffrey Hinton and Ruslan Salakhutdinov for many useful discussions and helpful suggestions. Nikola Karamanov and Alex Levinshtein helped by providing data sets. The anonymous reviewers also provided many useful suggestions. This research was supported by NSERC and Microsoft.

References

Bengio, Y., & Delalleau, O. (2007). *Justifying and generalizing contrastive divergence* (Technical Report 1311). Université de Montréal.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., & Montreal, Q. (2007). Greedy Layer-Wise Training of Deep Networks. *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference.*

Besag, J. (1986). On the statistical analysis of dirty

pictures. *Journal of the Royal Statistical Society B*, *48*, 259–302.

Borenstein, E., Sharon, E., & Ullman, S. (2004). Combining Top-Down and Bottom-Up Segmentation. *Computer Vision and Pattern Recognition Workshop, 2004 Conference on*, 46–46.

Carreira-Perpinan, M., & Hinton, G. (2005). On contrastive divergence learning. *Artificial Intelligence and Statistics, 2005*.

Gehler, P., Holub, A., & Welling, M. (2006). The rate adapting poisson model for information retrieval and object recognition. *Proceedings of the 23rd international conference on Machine learning*, 337–344.

Hinton, G. (2002). Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, *14*, 1771–1800.

Hinton, G., & Salakhutdinov, R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*, 504–507.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. *Proceedings of the 24th international conference on Machine learning*, 473–480.

LeCun, Y., & Cortes, C. The MNIST database of handwritten digits.

Neal, R. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, *56*, 71–113.

Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, *22*, 400–407.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. *Proceedings of the 24th international conference on Machine learning*, 791–798.

Salakhutdinov, R., & Murray, I. (2008). On the quantitative analysis of deep belief networks. *Proceedings of the International Conference on Machine Learning*.

Smolensky, P. (1986). *Information processing in dynamical systems: foundations of harmony theory*. MIT Press Cambridge, MA, USA.

Wainwright, M., & Jordan, M. (2003). Graphical models, exponential families, and variational inference. *UC Berkeley, Dept. of Statistics, Technical Report*, *649*.

Welling, M., & Hinton, G. (2002). A New Learning Algorithm for Mean Field Boltzmann Machines. *Artificial Neural Networks-Icann 2002: International Conference, Madrid, Spain, August 28-30, 2002: Proceedings*.

Welling, M., Rosen-Zvi, M., & Hinton, G. (2005). Exponential family harmoniums with an application to information retrieval. *Advances in Neural Information Processing Systems*, *17*, 1481–1488.

Younes, L. (1999). On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics An International Journal of Probability and Stochastic Processes*, *65*, 177–228.

Yuille, A. (2004). The Convergence of Contrastive Divergences. *Advances in Neural Information Processing Systems*, *3*, 4.

A Semiparametric Statistical Approach to Model-Free Policy Evaluation

Tsuyoshi Ueno[†] TSUYOS-U@SYS.I.KYOTO-U.AC.JP
Motoaki Kawanabe[‡] MOTOAKI.KAWANABE@FIRST.FRAUNHOFER.DE
Takeshi Mori[†] TAK-MORI@SYS.I.KYOTO-U.AC.JP
Shin-ichi Maeda[†] ICHI@SYS.I.KYOTO-U.AC.JP
Shin Ishii[†] ISHII@I.KYOTO-U.AC.JP

[†]Graduate School of Informatics, Kyoto University, Gokasho, Uji, Kyoto 611-0011, Japan
[‡]Fraunhofer FIRST, IDA, Kekuléstr. 7, 12489 Berlin, Germany

Abstract

Reinforcement learning (RL) methods based on least-squares temporal difference (LSTD) have been developed recently and have shown good practical performance. However, the quality of their estimation has not been well elucidated. In this article, we discuss LSTD-based policy evaluation from the new viewpoint of semiparametric statistical inference. In fact, the estimator can be obtained from a particular estimating function which guarantees its convergence to the true value asymptotically, without specifying a model of the environment. Based on these observations, we 1) analyze the asymptotic variance of an LSTD-based estimator, 2) derive the optimal estimating function with the minimum asymptotic estimation variance, and 3) derive a suboptimal estimator to reduce the computational burden in obtaining the optimal estimating function.

1. Introduction

Reinforcement learning (RL) is a machine learning framework based on reward-related interactions with environments (Sutton & Barto, 1998). In many RL methods, policy evaluation, in which a value function is estimated from sample trajectories, is an important step for improving a current policy. Since RL problems often involve high-dimensional state spaces, the value functions are often approximated by low-dimensional parametric models. Linear function approximation has mostly been used due to their simplicity and computational convenience.

To estimate the value function with a linear model, an online procedure called temporal difference (TD) learning (Sutton & Barto, 1998) and a batch procedure called least-squares temporal difference (LSTD) learning are widely used (Bradtke & Barto, 1996). LSTD can achieve fast learning, because it uses entire sample trajectories simultaneously. Recently, efficient procedures for policy improvement combined with policy evaluation by LSTD have been developed, and have shown good performance in realistic problems. For example, the least squares policy iteration (LSPI) method maximizes the Q-function estimated by LSTD (Lagoudakis & Parr, 2003), and the natural actor-critic (NAC) algorithm uses the natural policy gradient obtained by LSTD (Peters et al., 2005). Although variance reduction techniques have been proposed for other RL algorithms (Greensmith et al., 2004; Mannor et al., 2007), the important issue of how to evaluate and reduce the estimation variance of LSTD learning remains unresolved.

In this article, we discuss LSTD-based policy evaluation in the framework of semiparmetric statistical inference, which is new to the RL field. Estimation of linearly-represented value functions can be formulated as a semiparametric inference problem, where the statistical model includes not only the parameters of interest but also additional nuisance parameters with innumerable degrees of freedom (Godambe, 1991; Amari & Kawanabe, 1997; Bickel et al., 1998). We approach this problem by using estimating functions, which provide a well-established method for semiparametric estimation (Godambe, 1991). We then show that the instrumental variable method, a technique used in LSTD

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

learning, can be constructed from an estimating function which guarantees its consistency (asymptotic lack of bias) by definition.

As the main results, we show the asymptotic estimation variance in a general instrumental variable method (Lemma 2) and the optimal estimating function that yields the minimum asymptotic variance of the estimation (Theorem 1). We also derive a suboptimal instrumental variable, based on the idea of the c-estimator (Amari & Kawanabe, 1997), to reduce the computational difficulty of estimating the optimal instrumental variable (Theorem 2). As a proof of concept, we compare the mean squared error (MSE) of our new estimators with that of LSTD on a simple example of the Markov decision process (MDP).

2. Background

2.1. MDPs and Policy Evaluation

RL is an approach to finding an optimal policy for sequential decision-making in an unknown environment. We consider a finite MDP, which is defined as a quadruple $(\mathcal{S}, \mathcal{A}, p, r)$: $\mathcal{S}$ is a finite set of states; $\mathcal{A}$ is a finite set of actions; $p(s_{t+1}|s_t, a_t)$ is the transition probability to a next state s_{t+1} when taking an action a_t at state s_t; and $r(s_t, a_t, s_{t+1})$ is a reward received with the state transition. Let $\pi(s_t, a_t) = p(a_t|s_t)$ be a stochastic policy that the agent follows. We introduce the following assumption concerning the MDP.

Assumption 1. *An MDP has a stationary state distribution $d^\pi(s) = p(s)$ under the policy $\pi(s_t, a_t)$.*

There are two major choices in definition of the state value function: discounted reward accumulation and average reward (Bertsekas & Tsitsiklis, 1996). With the former choice, the value function is defined as

$$V^\pi(s) := \sum_{t=0}^{\infty} \mathbb{E}^\pi \left[\gamma^t r_{t+1} | s_0 = s\right], \tag{1}$$

where $\mathbb{E}^\pi[\cdot|s_0 = s]$ is the expectation with respect to the sample trajectory conditioned on $s_0 = s$ and $r_{t+1} := r(s_t, a_t, s_{t+1})$. $\gamma \in [0, 1)$ is the discount factor. With the latter choice, on the other hand, the value function is defined as

$$V^\pi(s) := \sum_{t=0}^{\infty} \mathbb{E}^\pi \left[r_{t+1} - \bar{r} | s_0 = s\right], \tag{2}$$

where $\bar{r} := \sum_{s\in\mathcal{S}} \sum_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} d^\pi(s)\pi(s, a)p(s'|s, a)r(s, a, s')$ denotes the average reward over the stationary distribution.

According to the Bellman equation, eq. (2) can be rewritten as

$$V^\pi(s_t) = \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)\bar{r}(s_t, s_{t+1}) - \bar{r} + \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)V^\pi(s_{t+1}), \tag{3}$$

where $p(s_{t+1}|s_t) := \sum_{a_t\in\mathcal{A}} \pi(s_t, a_t)p(s_{t+1}|s_t, a_t)$ and $\bar{r}(s_t, s_{t+1}) := \frac{\sum_{a_t\in\mathcal{A}} \pi(s_t,a_t)p(s_{t+1}|s_t,a_t)r(s_t,a_t,s_{t+1})}{p(s_{t+1}|s_t)}$. Throughout this article, we assume that the linear function approximation is faithful, and discuss only asymptotic estimation variance. (In general cases, bias becomes non-negligible and selection of basis functions is more important.)

Assumption 2. *The value function can be represented as a linear function of some features:*

$$V^\pi(s_t) = \boldsymbol{\phi}(s_t)^\top \boldsymbol{\theta} = \boldsymbol{\phi}_t^\top \boldsymbol{\theta}, \tag{4}$$

where $\boldsymbol{\phi}(s) : \mathcal{S} \to \mathcal{R}^m$ is a feature vector and $\boldsymbol{\theta} \in \mathcal{R}^m$ is a parameter vector.

Here, the symbol $\top$ denotes a transpose and the dimensionality of the feature vector m is smaller than the number of states $|\mathcal{S}|$. Substituting eq. (4) for eq. (3), we obtain the following equation

$$\left\{\boldsymbol{\phi}_t - \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)\boldsymbol{\phi}_{t+1}\right\}^\top \boldsymbol{\theta} = \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)\bar{r}(s_t, s_{t+1}) - \bar{r}. \tag{5}$$

When the matrix $\mathbb{E}^\pi\left[\left(\boldsymbol{\phi}_t - \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)\boldsymbol{\phi}_{t+1}\right)\left(\boldsymbol{\phi}_t - \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)\boldsymbol{\phi}_{t+1}\right)^\top\right]$ is non-singular and $p(s_{t+1}|s_t)$ is known, we can easily obtain the parameter $\boldsymbol{\theta}$. However, since $p(s_{t+1}|s_t)$ is unknown in normal RL settings, we have to estimate this parameter from the sample trajectory $\{s_0, a_0, r_1, \cdots, s_{N-1}, a_{N-1}, r_N\}$ alone, instead of using it directly.

Eq. (5) can be rewritten as

$$y_t = \boldsymbol{x}_t^\top \boldsymbol{\theta} + \epsilon_t, \tag{6}$$

where y_t, $\boldsymbol{x}_t$ and ϵ_t are defined as

$$y_t := r_{t+1} - \bar{r}, \quad \boldsymbol{x}_t := \boldsymbol{\phi}_t - \boldsymbol{\phi}_{t+1}$$
$$\epsilon_t := \left\{\boldsymbol{\phi}_{t+1} - \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)\boldsymbol{\phi}_{t+1}\right\}^\top \boldsymbol{\theta} + r_{t+1} - \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)\bar{r}(s_t, s_{t+1}). \tag{7}$$

When we use the discounted reward accumulation for the value function, eq. (6) also holds with

$$y_t := r_{t+1}, \quad \boldsymbol{x}_t := \boldsymbol{\phi}_t - \gamma\boldsymbol{\phi}_{t+1}$$
$$\epsilon_t := \gamma \left\{ \boldsymbol{\phi}_{t+1} - \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)\boldsymbol{\phi}_{t+1} \right\}^\top \boldsymbol{\theta} + r_{t+1} - \sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)\bar{r}(s_t, s_{t+1}). \tag{8}$$

Because $\mathbb{E}^\pi[\epsilon_t] = 0$, eq. (6) can be seen as a linear regression problem, where $\boldsymbol{x}$, y and ϵ are an input, an output and observation noise, respectively (Bradtke & Barto, 1996). Note that

$$\mathbb{E}^\pi[\epsilon_t g(s_t, s_{t-1}, \cdots, s_0)] = 0 \tag{9}$$

holds for any function $g(s_t, s_{t-1}, \cdots, s_0)$ because of the Markov property. The regression problem (6) has an undesirable property, however, which is known as an "error-in-variable problem" (Young, 1984): the input $\boldsymbol{x}_t$ and observation noise variables ϵ_t are mutually dependent.

It is not easy to solve such an error-in-variable problem in a rigorous manner; the simple least-squares method lacks consistency. Therefore, LSTD learning has used the instrumental variable method (Bradtke & Barto, 1996), a standard method to solve the error-in-variable problem that employs an "instrumental variable" to remove the effects of correlation between the input and the observation noise. When $\boldsymbol{X} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{N-1}]$ and $\boldsymbol{y} = [y_0, y_1, \cdots, y_{N-1}]^\top$, the estimator of the instrumental variable method is given by

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{Z}\boldsymbol{X}^\top]^{-1}[\boldsymbol{Z}\boldsymbol{y}], \tag{10}$$

where $\boldsymbol{Z} = [\boldsymbol{z}_0, \boldsymbol{z}_1, \cdots, \boldsymbol{z}_{N-1}]$, and $\boldsymbol{z}_t$ is an instrumental variable that is assumed to be correlated with the input $\boldsymbol{x}_t$ but uncorrelated with the observation noise ϵ_t.

2.2. Semiparametric Model and Estimating Functions

In the error-in-variable problem, if it is possible to assume a reasonable model with a small number of parameters on the joint input-output probability $p(\boldsymbol{x}, y)$, a proper estimator with consistency can be obtained by the maximum likelihood method. Since the transition probability $p(s_{t+1}|s_t)$ is unknown and usually difficult to estimate, it is practically impossible to construct such a parametric model. Let $\boldsymbol{k}_x$ and $\boldsymbol{k}_\epsilon$ be parameters which characterize the input distribution $p(\boldsymbol{x})$ and the conditional distribution $p(y|\boldsymbol{x})$ of output y given $\boldsymbol{x}$, respectively. Then, the joint distribution becomes

$$p(\boldsymbol{x}, y; \boldsymbol{\theta}, \boldsymbol{k}_x, \boldsymbol{k}_\epsilon) = p(y|\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{k}_\epsilon)p(\boldsymbol{x}; \boldsymbol{k}_x). \tag{11}$$

We would like to estimate the parameter $\boldsymbol{\theta}$ representing the value function in the presence of the extra unknowns $\boldsymbol{k}_x$ and $\boldsymbol{k}_\epsilon$, which can have innumerable degrees of freedom. Statistical models which contain such (possibly infinite-dimensional) nuisance parameters in addition to parameters of interest are called semiparametric (Bickel et al., 1998). In semiparametric inference, one established way of estimating parameters is to employ an estimating function (Godambe, 1991), which can give a consistent estimator of $\boldsymbol{\theta}$ without estimation of the nuisance parameters $\boldsymbol{k}_x$ and $\boldsymbol{k}_\epsilon$. Now we begin with a short overview of the estimating function in the simple i.i.d. case, and then discuss the Markov chain case.

We consider a general semiparametric model $p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{\kappa})$, where $\boldsymbol{\theta}$ is an m-dimensional parameter and $\boldsymbol{\kappa}$ is a nuisance parameter. An m-dimensional vector function $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})$ is called an estimating function when it satisfies the following conditions for any $\boldsymbol{\theta}$, $\boldsymbol{\kappa}$;

$$\mathbb{E}[\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})|\boldsymbol{\theta}, \boldsymbol{\kappa}] = \boldsymbol{0} \tag{12}$$
$$\det \left| \mathbb{E}\left[\frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) \middle| \boldsymbol{\theta}, \boldsymbol{\kappa} \right] \right| \neq 0 \tag{13}$$
$$\mathbb{E}\left[||\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})||^2 | \boldsymbol{\theta}, \boldsymbol{\kappa} \right] < \infty, \tag{14}$$

where $\mathbb{E}[\cdot|\boldsymbol{\theta}, \boldsymbol{\kappa}]$ denotes the expectation with respect to $\boldsymbol{x}$, which obeys the distribution $p(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{\kappa})$. The notations $\det|\cdot|$ and $||\cdot||$ denote the determinant and the Euclidean norm, respectively. Consider that i.i.d. samples $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{N-1}\}$ are obtained from the true model $p(\boldsymbol{x}; \boldsymbol{\theta} = \boldsymbol{\theta}^*, \boldsymbol{\kappa} = \boldsymbol{\kappa}^*) = p(\boldsymbol{x}; \boldsymbol{\theta}^*, \boldsymbol{\kappa}^*)$ for the observed trajectory. If there is an estimating function $\boldsymbol{f}$, by solving the estimating equation

$$\sum_{t=0}^{N-1} \boldsymbol{f}(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}) = \boldsymbol{0}, \tag{15}$$

we can obtain an estimator $\hat{\boldsymbol{\theta}}$ with good asymptotic properties. A solution of eq. (15) is called an "M-estimator" in statistics; the M-estimator is consistent, i.e., converges to the true parameter $\boldsymbol{\theta}^*$ regardless of the true nuisance parameter $\boldsymbol{\kappa}^*$ when the sample size N reaches infinity. In addition, the asymptotic variance $\mathrm{AV}[\hat{\boldsymbol{\theta}}]$ is given by

$$\mathrm{AV}[\hat{\boldsymbol{\theta}}] = \mathbb{E}[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)^\top] = \frac{1}{N}\boldsymbol{A}^{-1}\boldsymbol{M}\boldsymbol{A}^{-\top}, \tag{16}$$

where $\boldsymbol{A} = \mathbb{E}\left[\frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta})|\boldsymbol{\theta}^*, \boldsymbol{\kappa}^*\right]$ and $\boldsymbol{M} = \mathbb{E}\left[\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{f}^\top(\boldsymbol{x};\boldsymbol{\theta})|\boldsymbol{\theta}^*, \boldsymbol{\kappa}^*\right]$. The symbol $-\top$ denotes transpose of the inverse matrix. We omit the time index t, unless it is necessary to clarify. Note that the asymptotic variance AV depends on the true parameters, $\boldsymbol{\theta}^*$ and $\boldsymbol{\kappa}^*$, not on the samples $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_{N-1}\}$.

The notion of the estimating function can be extended to cases in which samples are given by a certain stochastic process (Godambe, 1985). In the semiparametric model for policy evaluation, under Assumption 1, there exist sufficient conditions of estimating functions which are almost the same as eqs. (12) - (14). The instrumental variable method is a type of estimating function method for semiparametric problems where the unknown distribution is given by eq. (11).

Lemma 1. *Suppose $\{\boldsymbol{x}_t, y_t\}$ is given by eq. (7) or (8), and $\boldsymbol{z}_t$ is given by a function of $\{s_t, \cdots, s_{t-T}\}$. If $\mathbb{E}^\pi[\boldsymbol{z}_t\boldsymbol{x}_t^\top]$ is nonsingular and $\mathbb{E}^\pi\left[||\boldsymbol{z}_t(\boldsymbol{x}_t^\top\boldsymbol{\theta} - y_t)||^2\right]$ is finite, then*

$$\boldsymbol{z}_t(\boldsymbol{x}_t^\top\boldsymbol{\theta} - y_t) \tag{17}$$

is an estimating function for the parameter $\boldsymbol{\theta}$. Therefore, the estimating equation is given by

$$\sum_{t=0}^{N-1} \boldsymbol{z}_t(\boldsymbol{x}_t^\top\boldsymbol{\theta} - y_t) = \boldsymbol{0}. \tag{18}$$

Proof For all t, the conditions corresponding to (13) and (14) are satisfied by the assumptions, and the condition (12) is satisfied as $\mathbb{E}^\pi[\boldsymbol{z}_t(\boldsymbol{x}_t^\top\boldsymbol{\theta} - y_t)] = \mathbb{E}^\pi[\boldsymbol{z}_t\epsilon_t] = \boldsymbol{0}$ from the property in eq. (9). (Q.E.D.)

LSTD is specifically an instrumental variable method in which the feature vector $\boldsymbol{z}_t = \boldsymbol{\phi}(s_t) = \boldsymbol{\phi}_t$ is used as an instrumental variable:

$$\boldsymbol{f}_{\mathrm{LSTD}} = \boldsymbol{\phi}_t(\boldsymbol{x}_t^\top\boldsymbol{\theta} - y_t). \tag{19}$$

The solution of the estimating equation is an M-estimator, and its asymptotic variance is given as follows.

Lemma 2. *Let $\boldsymbol{z}_t$ be a function of $\{s_t, ..., s_{t-T}\}$ satisfying the two conditions in Lemma 1 and $\epsilon_t^* = \boldsymbol{x}_t^\top\boldsymbol{\theta}^* - y_t$ be the residual for the true parameter $\boldsymbol{\theta}^*$. Then, the solution $\hat{\boldsymbol{\theta}}$ of the estimating equation (18) has the asymptotic variance*

$$\mathrm{AV}[\hat{\boldsymbol{\theta}}] = \frac{1}{N}\boldsymbol{A}_{\mathrm{IV}}^{-1}\boldsymbol{M}_{\mathrm{IV}}\boldsymbol{A}_{\mathrm{IV}}^{-\top}, \tag{20}$$

where $\boldsymbol{A}_{\mathrm{IV}} = \mathbb{E}_d[\boldsymbol{z}_t\boldsymbol{x}_t^\top]$, $\boldsymbol{M}_{\mathrm{IV}} = \mathbb{E}_d[(\epsilon_t^*)^2\boldsymbol{z}_t\boldsymbol{z}_t^\top]$. $E_d[\cdot]$ denotes the expectation when the sample trajectory starts from the stationary distribution $d^\pi(s_0)$. [1]

Proof The estimating equation (18) can be expressed as

$$\boldsymbol{Z}\boldsymbol{y} = \boldsymbol{Z}\boldsymbol{X}^\top\hat{\boldsymbol{\theta}} \tag{21}$$

where $\boldsymbol{Z} = [\boldsymbol{z}_0, \cdots, \boldsymbol{z}_{N-1}]$, $\boldsymbol{X} = [\boldsymbol{x}_0, \cdots, \boldsymbol{x}_{N-1}]$, $\boldsymbol{y} = [y_0, \cdots, y_{N-1}]^\top$. On the other hand, from eq. (6), the left hand side of eq. (21) is equal to $\boldsymbol{Z}\boldsymbol{X}^\top\boldsymbol{\theta}^* + \boldsymbol{Z}\boldsymbol{X}^\top\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = [\epsilon_0^*, \cdots, \epsilon_{N-1}^*]^\top$. Thus the asymptotic variance of the estimator $\hat{\boldsymbol{\theta}}$ is obtained as

$$\mathbb{E}^\pi[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)^\top] = \mathbb{E}^\pi[(\boldsymbol{Z}\boldsymbol{X}^\top)^{-1}\boldsymbol{Z}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top\boldsymbol{Z}^\top(\boldsymbol{X}\boldsymbol{Z}^\top)^{-1}] \overset{N\to\infty}{\longrightarrow} \boldsymbol{A}_{\mathrm{IV}}^{-1}\frac{1}{N^2}\mathbb{E}^\pi[\boldsymbol{Z}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top\boldsymbol{Z}^\top]\boldsymbol{A}_{\mathrm{IV}}^{-\top}$$

where we used the fact that the matrix $\boldsymbol{Z}\boldsymbol{X}^\top$ has the limit

$$\frac{1}{N}(\boldsymbol{Z}\boldsymbol{X}^\top) = \frac{1}{N}\sum_{t=0}^{N-1}\boldsymbol{z}_t\boldsymbol{x}_t^\top \overset{N\to\infty}{\longrightarrow} \boldsymbol{A}_{\mathrm{IV}}.$$

Also, the matrix $\mathbb{E}^\pi[\boldsymbol{Z}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top\boldsymbol{Z}^\top]$ has the following limit:

$$\frac{1}{N}\mathbb{E}^\pi[\boldsymbol{Z}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top\boldsymbol{Z}^\top] = \frac{1}{N}\sum_{t=0}^{N-1}\mathbb{E}^\pi[(\epsilon_t^*)^2\boldsymbol{z}_t\boldsymbol{z}_t^\top] \overset{N\to\infty}{\longrightarrow} \boldsymbol{M}_{\mathrm{IV}},$$

where we used the property in eq. (9). Therefore, we have

$$\mathbb{E}^\pi[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*)^\top] \overset{N\to\infty}{\longrightarrow} \frac{1}{N}\boldsymbol{A}_{\mathrm{IV}}^{-1}\boldsymbol{M}_{\mathrm{IV}}\boldsymbol{A}_{\mathrm{IV}}^{-\top}.$$

(Q.E.D.)

To summarize, if we have an instrumental variable which satisfies the assumptions in Lemmas 1 and 2, we can obtain an M-estimator from the estimating equation (18) with the asymptotic variance eq. (20). When more than one instrumental variable exists, it is appropriate to choose the one whose estimator has the minimum asymptotic variance.

3. Main Results

In this section, we show that estimating functions for the semiparametric model of policy evaluation are limited to the type of equation used in the instrumental variable method. Furthermore, we derive the optimal

[1] We remark that the definitions of $\boldsymbol{A}_{\mathrm{IV}}$ and $\boldsymbol{M}_{\mathrm{IV}}$ do not depend on the t. Nevertheless, we keep the time index t for clarification.

instrumental variable having the minimum asymptotic variance of the estimation.

We first remark on the invariance property of the instrumental variable method.

Lemma 3. *The value function estimation* $\hat{V}(s_t) = \boldsymbol{\phi}_t^\top \hat{\boldsymbol{\theta}} = \boldsymbol{\phi}_t^\top [\boldsymbol{Z}\boldsymbol{X}^\top]^{-1}[\boldsymbol{Z}\boldsymbol{y}]$ *is invariant with respect to the application of any regular linear transformation to either the instrumental variable* $\boldsymbol{z}_t$ *or the basis functions* $\boldsymbol{\phi}_t$.

Proof Assume that the instrumental variable and the basis functions are both transformed by any regular matrices $\boldsymbol{W}_z$ and $\boldsymbol{W}_\phi$ as $\boldsymbol{z}'_t = \boldsymbol{W}_z \boldsymbol{z}_t$ and $\boldsymbol{\phi}'_t = \boldsymbol{W}_\phi \boldsymbol{\phi}_t$. Noting that the linear transformation of $\boldsymbol{\phi}_t$ yields the linear transformation of the input $\boldsymbol{x}'_t = \boldsymbol{W}_\phi \boldsymbol{x}_t$, the estimator of the instrumental variable method given by eq. (10) becomes $\hat{\boldsymbol{\theta}}' = [\boldsymbol{Z}'(\boldsymbol{X}')^\top]^{-1}[\boldsymbol{Z}'\boldsymbol{y}] = \boldsymbol{W}_\phi^{-1}\hat{\boldsymbol{\theta}}$. This means that the estimated value function is invariant as $(\boldsymbol{\phi}'_t)^\top \hat{\boldsymbol{\theta}}' = \boldsymbol{\phi}_t^\top \hat{\boldsymbol{\theta}}$. (Q.E.D.)

When the basis functions span over the whole space of functions of the state, any set of basis functions can be represented by applying a linear transformation to another set of basis functions. This observation leads to the following Corollary.

Corollary 1. *When the basis functions* $\boldsymbol{\phi}_t$ *span the whole space of functions of the state, the value function estimation is invariant with respect to the choice of basis functions and of the instrumental variable.*

An instrumental variable may depend not only on the current state s_t, but also on the previous states $\{s_{t-1}, \cdots, s_{t-T}\}$, because such an instrumental variable does not violate the condition, $\mathrm{cov}[\boldsymbol{z}_t, \epsilon_t] = \boldsymbol{0}$. However, we do not need to consider such instrumental variables, as the following Lemma shows.

Lemma 4. *Let* $\boldsymbol{z}_t(s_t, \cdots, s_{t-T})$ *be any instrumental variable depending on the current and previous states which satisfies the conditions in Lemmas 1 and 2. Then, there is necessarily an instrumental variable depending only on the current state whose corresponding estimator has equal or minimum asymptotic variance.*

Proof We show that the conditional expectation $\tilde{\boldsymbol{z}}_t = \mathbb{E}^\pi[\boldsymbol{z}_t | s_t]$ which depends only on the current state s_t, gives an equally good or better estimator. The matrices in the asymptotic variance, eq. (20), can be calculated as

$$\begin{aligned}\boldsymbol{A}_{\boldsymbol{z}} &= \mathbb{E}_d\left[\tilde{\boldsymbol{z}}_t \boldsymbol{x}_t^\top\right] + \mathbb{E}_d\left[(\tilde{\boldsymbol{z}}_t - \boldsymbol{z}_t)\boldsymbol{x}_t^\top\right] = \boldsymbol{A}_{\tilde{\boldsymbol{z}}} \\ \boldsymbol{M}_{\boldsymbol{z}} &= \mathbb{E}_d[(\epsilon_t^*)^2(\tilde{\boldsymbol{z}}_t + \boldsymbol{z}_t - \tilde{\boldsymbol{z}}_t)(\tilde{\boldsymbol{z}}_t + \boldsymbol{z}_t - \tilde{\boldsymbol{z}}_t)^\top] \\ &= \boldsymbol{M}_{\tilde{\boldsymbol{z}}} + \mathbb{E}_d[(\epsilon_t^*)^2(\boldsymbol{z}_t - \tilde{\boldsymbol{z}}_t)(\boldsymbol{z}_t - \tilde{\boldsymbol{z}}_t)^\top],\end{aligned}$$

where we have used eq. (9). This implies that

$$\mathrm{AV}[\hat{\boldsymbol{\theta}}_{\boldsymbol{z}}] = \frac{1}{N}\boldsymbol{A}_{\boldsymbol{z}}^{-1}\boldsymbol{M}_{\boldsymbol{z}}\boldsymbol{A}_{\boldsymbol{z}}^{-\top} \succeq \frac{1}{N}\boldsymbol{A}_{\tilde{\boldsymbol{z}}}^{-1}\boldsymbol{M}_{\tilde{\boldsymbol{z}}}\boldsymbol{A}_{\tilde{\boldsymbol{z}}}^{-\top} = \mathrm{AV}[\hat{\boldsymbol{\theta}}_{\tilde{\boldsymbol{z}}}].$$

(Q.E.D.)

Here, the inequality $\succeq$ denotes the semipositive definiteness of the subtraction. Now, we consider the general form of estimating functions for inference of the value function. In the following, we consider only 'admissible' estimating functions. More precisely, we discard 'inadmissible' estimating functions whose estimators are always inferior to those of other estimating functions in the sense of asymptotic variance. To simplify analysis, we only consider the limited set of estimating functions which are defined on a one-step sample $\{s, a, s'\}$.

Proposition 1. *For the semiparametric model of eqs. (6), (7) or eqs. (6), (8), all admissible estimating functions of only the one-step sample* $\{s, a, s'\}$ *must have the form of* $\boldsymbol{f} = \boldsymbol{z}(y - \boldsymbol{x}^\top\boldsymbol{\theta})$, *where* $\boldsymbol{z}$ *is any function which does not depend on* s' *and satisfies the assumption in Lemma 1.*

Proof Due to space limitation, we will just outline the proof. To be an estimating function, the function $\boldsymbol{f}$ must satisfy $\mathbb{E}_d[\boldsymbol{f}] = \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} p(s'|s,a)\pi(s,a)\boldsymbol{f}(s,a,s') = \boldsymbol{0}$. Because we can prove that the stationary distribution $d^\pi(s)$ takes any probability vector, $\sum_{s\in\mathcal{S}} d^\pi(s)\boldsymbol{v}(s) = \boldsymbol{0}$ implies that $\boldsymbol{v}(s) = \boldsymbol{0}$ for any state s, where $\boldsymbol{v}(s) := \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} p(s'|s,a)\pi(s,a)\boldsymbol{f}(s,a,s')$. Furthermore, the Bellman equation (6) holds, whatever the $p(s'|s,a)$ is. To fulfil $\boldsymbol{v} = \boldsymbol{0}$, $\boldsymbol{f}$ must have the form of $\boldsymbol{f} = \boldsymbol{z}(y - \boldsymbol{x}^\top\boldsymbol{\theta}) + \boldsymbol{h}$, where $\boldsymbol{z}$ does not depend on s' or a, and $\boldsymbol{h}$ is any function that satisfies $\sum_{a\in\mathcal{A}} \pi(s,a)\boldsymbol{h}(s,a,s') = \boldsymbol{0}$. However, the addition of such a function $\boldsymbol{h}$ necessarily enlarges the asymptotic variance of the estimation. Therefore, the admissible estimating function is restricted to the form of $\boldsymbol{f} = \boldsymbol{z}(y - \boldsymbol{x}^\top\boldsymbol{\theta})$. (Q.E.D)

We are currently working on the conjecture that whether Proposition 1 can be extended to general estimating functions depend on all previous states and actions. If this is true, from Lemma 4, it is sufficient to consider the instrumental variable method with $\boldsymbol{z}_t$ depending only on the current state s_t for the semiparametric inference problem. Therefore, we next discuss the optimal instrument variable of this type in terms of asymptotic variance, which corresponds to the optimal estimating function.

Algorithm 1 The pseudo code of gLSTD.

gLSTD($\mathcal{D}$, ϕ)
// $D = \{s_0, r_1, \cdots, s_{N-1}, r_N\}$: Sample sequence
// ϕ: Basis functions
// Calculate the initial parameter and its residual
$\hat{\boldsymbol{\theta}}_0 \leftarrow \left[\sum_{t=0}^{N-1} \boldsymbol{\phi}_t \boldsymbol{x}_t^\top\right]^{-1} \left[\sum_{t=0}^{N-1} \boldsymbol{\phi}_t y_t\right]$
$\hat{\epsilon}_t \leftarrow \boldsymbol{x}_t^\top \hat{\boldsymbol{\theta}}_0 - y_t$
// Calculate the estimator $\widetilde{\mathbb{E}^\pi[(\hat{\epsilon}_t)^2|s_t]}, \widetilde{\mathbb{E}^\pi[\boldsymbol{x}_t|s_t]}$
// of the conditional expectations
// and construct the instrumental variable
$\hat{\boldsymbol{z}}_t \leftarrow \widetilde{\mathbb{E}^\pi[(\hat{\epsilon}_t)^2|s_t]}^{-1} \widetilde{\mathbb{E}^\pi[\boldsymbol{x}_t|s_t]}$
// Calculate the parameter
$\hat{\boldsymbol{\theta}}_g \leftarrow \left[\sum_{t=0}^{N-1} \hat{\boldsymbol{z}}_t \boldsymbol{x}_t^\top\right]^{-1} \left[\sum_{t=0}^{N-1} \hat{\boldsymbol{z}}_t y_t\right]$
Return $\hat{\boldsymbol{\theta}}_g$

Theorem 1. *The optimal instrumental variable gives the minimum asymptotic variance*

$$\boldsymbol{z}_t^* = \begin{cases} \mathbb{E}^\pi\left[(\epsilon_t^*)^2|s_t\right]^{-1}\left(\boldsymbol{\phi}_t - \gamma\mathbb{E}^\pi[\boldsymbol{\phi}_{t+1}|s_t]\right) \\ \quad (\textit{discounted reward accumulation}) \\ \mathbb{E}^\pi\left[(\epsilon_t^*)^2|s_t\right]^{-1}\left(\boldsymbol{\phi}_t - \mathbb{E}^\pi[\boldsymbol{\phi}_{t+1}|s_t]\right) \\ \quad (\textit{average reward}). \end{cases} \tag{22}$$

The proof is given in Appendix A. Note that the definition of the optimal instrumental variable includes both the residual ϵ_t^* and the conditional expectations $\mathbb{E}^\pi[\boldsymbol{\phi}_{t+1}|s_t]$ and $\mathbb{E}^\pi[(\epsilon_t^*)^2|s_t]$. To make this estimator practical, we replace the residual ϵ_t^* with that of the LSTD estimator, and approximate the expectation, $\mathbb{E}^\pi[\boldsymbol{\phi}_{t+1}|s_t]$ and $\mathbb{E}^\pi[(\epsilon_t^*)^2|s_t]$, by using function approximation. We call this procedure "gLSTD learning" (see Algorithm 1 for its pseudo code).

To avoid estimating the functions depending on the current state, $\mathbb{E}^\pi[\boldsymbol{\phi}_{t+1}|s_t]$ and $\mathbb{E}^\pi[(\epsilon_t^*)^2|s_t]$, which appear in the instrumental variable, we simply replace them by constants. When $\boldsymbol{z}$ is an instrumental variable, addition of any constant value to $\boldsymbol{z}$, $\boldsymbol{z}' = \boldsymbol{z} + \boldsymbol{c}$, leads to another valid instrumental variable; because of Lemma 1, it is easily confirmed that $\boldsymbol{f}_{\mathrm{c}} = (\boldsymbol{z}_t + \boldsymbol{c})(\boldsymbol{x}_t^\top\boldsymbol{\theta} - y_t)$ is an estimating function. Therefore, obtaining the optimal constant $\boldsymbol{c}$ yields a suboptimal instrumental variable within instrumental variables produced by constant shifts.

Theorem 2. *The optimal shift is given by*

$$\boldsymbol{c}^* := -\frac{\mathbb{E}_d[(\epsilon_t^*)^2\boldsymbol{z}_t] - \mathbb{E}_d[(\epsilon_t^*)^2\boldsymbol{z}_t\boldsymbol{z}_t^\top]\mathbb{E}_d[\boldsymbol{x}_t\boldsymbol{z}_t^\top]^{-1}\mathbb{E}_d[\boldsymbol{x}_t]}{\mathbb{E}_d[(\epsilon_t^*)^2] - \mathbb{E}_d[(\epsilon_t^*)^2\boldsymbol{z}_t^\top]\mathbb{E}_d[\boldsymbol{x}_t\boldsymbol{z}_t^\top]^{-1}\mathbb{E}_d[\boldsymbol{x}_t]}. \tag{23}$$

Algorithm 2 The pseudo code of LSTDc

LSTDc($\mathcal{D}$, ϕ)
// $D = \{s_0, r_1, \cdots, s_{N-1}, r_N\}$: Sample sequence
// ϕ: Basis functions
// Calculate the initial parameter and its residual
$\hat{\boldsymbol{\theta}}_0 \leftarrow \left[\sum_{t=0}^{N-1} \boldsymbol{\phi}_t \boldsymbol{x}_t^\top\right]^{-1} \left[\sum_{t=0}^{N-1} \boldsymbol{\phi}_t y_t\right]$
$\hat{\epsilon}_t \leftarrow \boldsymbol{x}_t^\top \hat{\boldsymbol{\theta}}_0 - y_t$
// Construct the suboptimal
// instrumental variable with optimal shift
$$\hat{\boldsymbol{c}} \leftarrow -\frac{\left[\sum_{t=0}^{N-1} \hat{\epsilon}_t^2\boldsymbol{\phi}_t\right] - \left[\sum_{t=0}^{N-1} \hat{\epsilon}_t^2\boldsymbol{\phi}_t\boldsymbol{\phi}_t^\top\right]\left[\sum_{t=0}^{N-1} \boldsymbol{x}_t\boldsymbol{\phi}_t^\top\right]^{-1}\left[\sum_{t=0}^{N-1} \boldsymbol{x}_t\right]}{\left[\sum_{t=0}^{N-1} \hat{\epsilon}_t^2\right] - \left[\sum_{t=0}^{N-1} \hat{\epsilon}_t^2\boldsymbol{\phi}_t^\top\right]\left[\sum_{t=0}^{N-1} \boldsymbol{x}_t\boldsymbol{\phi}_t^\top\right]^{-1}\left[\sum_{t=0}^{N-1} \boldsymbol{x}_t\right]}$$
$\hat{\boldsymbol{z}}_t = \boldsymbol{\phi}_t + \hat{\boldsymbol{c}}$
//Calculate the parameter
$\hat{\boldsymbol{\theta}}_c \leftarrow \left[\sum_{t=0}^{N-1} \hat{\boldsymbol{z}}_t \boldsymbol{x}_t^\top\right]^{-1} \left[\sum_{t=0}^{N-1} \hat{\boldsymbol{z}}_t y_t\right]$
Return $\hat{\boldsymbol{\theta}}_c$

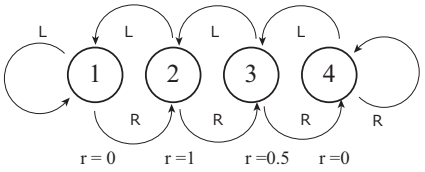


Figure 1. A four-state MDP.

The proof is given in Appendix B. In eq. (23), however, the residual ϵ_t^* is again unknown; hence, we need to approximate this, too, as in the gLSTD learning. We call this procedure "LSTDc learning" (see Algorithm 2 for its pseudo code).

4. Simulation Experiments

So far, we have discussed the asymptotic variance under the assumption that we have an infinite number of samples. In this section, we evaluate the performance of the proposed estimator in a practical situation with a finite number of samples. We use an MDP defined on a simple Markov random walk, which was also used in a previous study (Lagoudakis & Parr, 2003). This MDP incorporates a one-dimensional chain walk with four states (Figure 1). Two actions, "left"(L) and "right"(R), are available at every state. Rewards 1 and 0.5 are given when states '2' and '3' are visited, respectively.

We adopt the simplest direct representation of states; the state variable took $s = 1$, $s = 2$, $s = 3$ or $s = 4$,

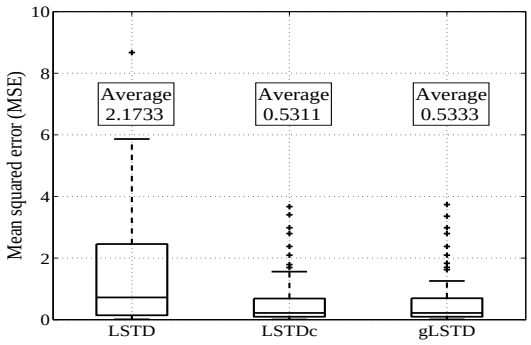


Figure 2. Simulation result.

when the corresponding state was visited. The value function was defined as the average reward, eq. (2), and was approximated by a linear function with a three-dimensional basis function: $\boldsymbol{\phi}(s) = [s, s^2, s^3]^\top$. The policy was set at random, and at the beginning of each episode an initial state was randomly selected according to the stationary distribution of this Markov chain.

Under these conditions, we performed 100 episodes each of which consisted of 100 random walk steps. We evaluated the "mean squared error" (MSE) of the value function, i.e., $\sum_{i\in\{1,2,3,4\}} d^\pi(i)|\hat{V}(i) - V^*(i)|^2$; where $\hat{V}$ and V^* denote $\hat{V}(i) = \boldsymbol{\phi}(s=i)^\top\hat{\boldsymbol{\theta}}$ and $V^*(i) = \boldsymbol{\phi}(s=i)^\top\boldsymbol{\theta}^*$, respectively.

Figure 2 shows box-plots of the MSEs of LSTD, LSTDc, and gLSTD. For this example, estimators of the conditional expectations in gLSTD can be calculated by sample average in each state, because there were only four discrete states. In continuous state problems, however, estimation of such conditional expectations would become much harder.

In Figure 2, the y-axis denotes the MSE of the value function. The center line and the upper and lower sides of each box denote the median of MSEs and the upper and lower quartiles, respectively. The number above each box represents the average MSE. There is significant difference between the MSE of LSTD and those of LSTDc and gLSTD. The estimators for LSTDc and for gLSTD both achieved a much smaller MSE than that for the ordinary LSTD.

5. Conclusion

In this study, we have discussed LSTD-based policy evaluation in the framework of semiparametric statistical inference. We showed that the standard LSTD algorithm is indeed an estimating function method which is guaranteed to be consistent regardless of the stochastic properties of the environments. Based on the optimal estimating functions in the two classes of estimating functions, we constructed two new policy evaluation methods called gLSTD and LSTDc. We also evaluated the asymptotic variance of the general instrumental variable methods for MDP. Moreover, we showed that the form of possible estimating functions for the value function estimation is restricted to be the same as those used in the instrumental variable methods. We then demonstrated, through an experiment using a simple MDP problem, that the gLSTD and LSTDc estimators reduce substantially the asymptotic variance of the LSTD estimator.

Further work is necessary to construct procedures for policy updating based on evaluation by gLSTD and LSTDc. It should be possible to incorporate our proposed ideas into the least-squares policy iteration (Lagoudakis & Parr, 2003) and the natural actor-critic method (Peters et al., 2005).

A. Proof of Theorem 1: The Optimal Instrumental Variable

As shown in eq. (20), the asymptotic variance of the estimator $\hat{\boldsymbol{\theta}}_{\boldsymbol{z}}$ is given by

$$\mathrm{AV}[\hat{\boldsymbol{\theta}}_{\boldsymbol{z}}] = \frac{1}{N}\boldsymbol{A}_{\boldsymbol{z}}^{-1}\boldsymbol{M}_{\boldsymbol{z}}\boldsymbol{A}_{\boldsymbol{z}}^{-\top},$$

where $\boldsymbol{A}_{\boldsymbol{z}} := \mathbb{E}_d[\boldsymbol{z}_t\boldsymbol{x}_t^\top]$ and $\boldsymbol{M}_{\boldsymbol{z}} := \mathbb{E}_d[(\epsilon_t^*)^2\boldsymbol{z}_t\boldsymbol{z}_t^\top]$. If we add a small change $\boldsymbol{\delta}_t(s_t, \cdots, s_{t-T})$ to the instrumental variable $\boldsymbol{z}_t$, the matrices become

$$\begin{aligned}\boldsymbol{A}_{\boldsymbol{z}+\boldsymbol{\delta}} &= \boldsymbol{A}_{\boldsymbol{z}} + \mathbb{E}_d[\boldsymbol{\delta}_t\boldsymbol{x}_t^\top],\\ \boldsymbol{M}_{\boldsymbol{z}+\boldsymbol{\delta}} &= \boldsymbol{M}_{\boldsymbol{z}} + \mathbb{E}_d[(\epsilon_t^*)^2(\boldsymbol{\delta}_t\boldsymbol{z}_t^\top + \boldsymbol{z}_t\boldsymbol{\delta}_t^\top)].\end{aligned}$$

Therefore, the deviation of the trace of asymptotic variance can be calculated as

$$\begin{aligned}&\mathrm{Tr}\left[\boldsymbol{A}_{\boldsymbol{z}+\boldsymbol{\delta}}^{-1}\boldsymbol{M}_{\boldsymbol{z}+\boldsymbol{\delta}}\boldsymbol{A}_{\boldsymbol{z}+\boldsymbol{\delta}}^{-\top}\right] - \mathrm{Tr}\left[\boldsymbol{A}_{\boldsymbol{z}}^{-1}\boldsymbol{M}_{\boldsymbol{z}}\boldsymbol{A}_{\boldsymbol{z}}^{-\top}\right]\\ =&-\mathrm{Tr}\left\{\boldsymbol{A}_{\boldsymbol{z}}^{-1}\mathbb{E}_d\left[\boldsymbol{\delta}_t\boldsymbol{x}_t^\top\right]\boldsymbol{A}_{\boldsymbol{z}}^{-1}\boldsymbol{M}_{\boldsymbol{z}}\boldsymbol{A}_{\boldsymbol{z}}^{-\top}\right\}\\ &-\mathrm{Tr}\left\{\boldsymbol{A}_{\boldsymbol{z}}^{-1}\mathbb{E}_d\left[\boldsymbol{x}_t\boldsymbol{\delta}_t^\top\right]\boldsymbol{A}_{\boldsymbol{z}}^{-1}\boldsymbol{M}_{\boldsymbol{z}}\boldsymbol{A}_{\boldsymbol{z}}^{-\top}\right\}\\ &+\mathrm{Tr}\left\{\boldsymbol{A}_{\boldsymbol{z}}^{-1}\mathbb{E}_d\left[(\epsilon_t^*)^2\left(\boldsymbol{z}_t\boldsymbol{\delta}_t^\top + \boldsymbol{\delta}\boldsymbol{z}_t^\top\right)\right]\boldsymbol{A}_{\boldsymbol{z}}^{-\top}\right\}\\ =&2\mathbb{E}_d\left[\boldsymbol{\delta}_t^\top\boldsymbol{A}_{\boldsymbol{z}}^{-\top}\boldsymbol{A}_{\boldsymbol{z}}^{-1}\mathbb{E}^\pi\left[(\epsilon_t^*)^2|s_t,\cdots,s_{t-T}\right]\boldsymbol{z}_t\right]\\ &-2\mathbb{E}_d\left[\boldsymbol{\delta}_t^\top\boldsymbol{A}_{\boldsymbol{z}}^{-\top}\boldsymbol{A}_{\boldsymbol{z}}^{-1}\boldsymbol{M}_{\boldsymbol{z}}\boldsymbol{A}_{\boldsymbol{z}}^{-\top}\mathbb{E}^\pi[\boldsymbol{x}_t|s_t,\cdots,s_{t-T}]\right].\end{aligned}$$

By using the condition that the deviation becomes $\mathbf{0}$ for any small change $\boldsymbol{\delta}_t(s_t, \cdots, s_{t-T})$, the optimal instrumental variable can be obtained as

$$\boldsymbol{z}_t^* = \mathbb{E}^\pi\left[(\epsilon_t^*)^2|s_t\right]^{-1}\boldsymbol{M}_{\boldsymbol{z}}\boldsymbol{A}_{\boldsymbol{z}}^{-\top}\mathbb{E}^\pi[\boldsymbol{x}_t|s_t].$$

Considering Lemma 3, the optimal instrumental variable is restricted as

$$\boldsymbol{z}_t^* = \mathbb{E}^\pi\left[(\epsilon_t^*)^2|s_t\right]^{-1}\mathbb{E}^\pi[\boldsymbol{x}_t|s_t],$$

or as its transformation by any regular matrix.

Now, we show that eq. (22) also satisfies the global optimality. Substituting $\boldsymbol{z}_t^*$ to the matrix $\boldsymbol{A}_{\boldsymbol{z}^*}$, we obtain

$$\boldsymbol{A}_{\boldsymbol{z}^*} = \boldsymbol{M}_{\boldsymbol{z}^*}\boldsymbol{A}_{\boldsymbol{z}^*}^{-\top}\boldsymbol{F},$$

where

$$\boldsymbol{F} := \mathbb{E}_d\left[\mathbb{E}^\pi[(\epsilon_t^*)^2|s_t]^{-1}\mathbb{E}^\pi[\boldsymbol{x}_t|s_t]\mathbb{E}^\pi[\boldsymbol{x}_t^\top|s_t]\right].$$

Furthermore, the matrices at $\boldsymbol{z}_t^* + \boldsymbol{\delta}_t$ become

$$\begin{aligned}
\boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}} &= \boldsymbol{M}_{\boldsymbol{z}^*}\boldsymbol{A}_{\boldsymbol{z}^*}^{-\top}\boldsymbol{F} + \mathbb{E}_d[\boldsymbol{x}_t\boldsymbol{\delta}_t^\top],\\
\boldsymbol{M}_{\boldsymbol{z}^*+\boldsymbol{\delta}} &= \boldsymbol{M}_{\boldsymbol{z}^*}\boldsymbol{A}_{\boldsymbol{z}^*}^{-\top}\boldsymbol{F}\boldsymbol{A}_{\boldsymbol{z}^*}^{-1}\boldsymbol{M}_{\boldsymbol{z}^*} + \boldsymbol{M}_{\boldsymbol{z}^*}\boldsymbol{A}_{\boldsymbol{z}^*}^{-\top}\mathbb{E}_d[\boldsymbol{x}_t\boldsymbol{\delta}_t^\top]\\
&\quad + \mathbb{E}_d[\boldsymbol{\delta}_t\boldsymbol{x}_t^\top]\boldsymbol{A}_{\boldsymbol{z}^*}^{-1}\boldsymbol{M}_{\boldsymbol{z}^*} + \mathbb{E}_d[(\epsilon_t^*)^2\boldsymbol{\delta}_t\boldsymbol{\delta}_t^\top].
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}^{-1}\boldsymbol{M}_{\boldsymbol{z}^*+\boldsymbol{\delta}}\boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}^{-\top} - \boldsymbol{A}_{\boldsymbol{z}^*}^{-1}\boldsymbol{M}_{\boldsymbol{z}^*}\boldsymbol{A}_{\boldsymbol{z}^*}^{-\top}\\
&= \boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}^{-1}(\boldsymbol{M}_{\boldsymbol{z}^*+\boldsymbol{\delta}} - \boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}\boldsymbol{A}_{\boldsymbol{z}^*}^{-1}\boldsymbol{M}_{\boldsymbol{z}^*}\boldsymbol{A}_{\boldsymbol{z}^*}^{-\top}\boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}^{\top})\boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}^{-\top}\\
&= \boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}^{-1}(\mathbb{E}_d[(\epsilon_t^*)^2\boldsymbol{\delta}_t\boldsymbol{\delta}_t^\top] - \mathbb{E}_d[\boldsymbol{\delta}_t\boldsymbol{x}_t^\top]\boldsymbol{F}^{-1}\mathbb{E}_d[\boldsymbol{x}_t\boldsymbol{\delta}_t^\top])\boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}^{-\top}\\
&= \boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}^{-1}\Big\{\mathbb{E}_d\Big[\mathbb{E}^\pi[(\epsilon_t^*)^2|s_t]\\
&\quad\times\left(\boldsymbol{\delta}_t - \mathbb{E}^\pi[(\epsilon_t^*)^2|s_t]^{-1}\mathbb{E}_d[\boldsymbol{\delta}_t\boldsymbol{x}_t^\top]\boldsymbol{F}^{-1}\mathbb{E}^\pi[\boldsymbol{x}_t|s_t]\right)\\
&\quad\times\left(\boldsymbol{\delta}_t - \mathbb{E}^\pi[(\epsilon_t^*)^2|s_t]^{-1}\mathbb{E}_d[\boldsymbol{\delta}_t\boldsymbol{x}_t^\top]\boldsymbol{F}^{-1}\mathbb{E}^\pi[\boldsymbol{x}_t|s_t]\right)^\top\Big]\\
&\quad\Big\}\boldsymbol{A}_{\boldsymbol{z}^*+\boldsymbol{\delta}}^{-\top} \succeq \boldsymbol{0}.
\end{aligned}$$

The equality holds only when $\boldsymbol{\delta}_t(s_t) \propto \boldsymbol{z}_t^*$. (Q.E.D.)

B. Proof of Theorem 2: The Optimal $\boldsymbol{c}$

By differentiating the trace of eq. (20), the optimal constant $\boldsymbol{c}$ must satisfy

$$\mathbb{E}_d[(\epsilon_t^*)^2]\boldsymbol{c} + \mathbb{E}_d[(\epsilon_t^*)^2\boldsymbol{\phi}_t] = \boldsymbol{M}_{\boldsymbol{c}}\boldsymbol{A}_{\boldsymbol{c}}^{-\top}\mathbb{E}_d[\boldsymbol{x}_t].$$

Using the well-known matrix inversion lemma (Horn & Johnson, 1985), the solution can be obtained as eq. (23).

In addition, the global optimality among those applied by constant shifts can be proved using a similar argument to that in Appendix A. (Q.E.D.)

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments and suggestions.

References

Amari, S., & Kawanabe, M. (1997). Information geometry of estimating functions in semi-parametric statistical models. *Bernoulli*, *3*, 29–54.

Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bickel, D., Ritov, D., Klaassen, C., & Wellner, J. (1998). *Efficient and Adaptive Estimation for Semiparametric Models*. Springer.

Bradtke, S., & Barto, A. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, *22*, 33–57.

Godambe, V. (1985). The foundations of finite sample estimation in stochastic processes. *Biometrika*, *72*, 419–428.

Godambe, V. (Ed.). (1991). *Estimating Functions*. Oxford Science.

Greensmith, E., Bartlett, P., & Baxter, J. (2004). Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, *5*, 1471–1530.

Horn, R., & Johnson, C. (1985). *Matrix analysis*. Cambridge University Press.

Lagoudakis, M., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, *4*, 1107–1149.

Mannor, S., Simester, D., Sun, P., & Tsitsiklis, J. N. (2007). Bias and variance approximation in value function estimates. *Management Science*, *53*, 308–322.

Peters, J., Vijayakumar, S., & Schaal, S. (2005). Natural actor-critic. *Proceedings of the 16th European Conference on Machine Learning* (pp. 280–291).

Sutton, R., & Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Young, P. (1984). *Recursive Estimation and Time-series Analysis*. Springer-Verlag.

Topologically-Constrained Latent Variable Models

Raquel Urtasun RURTASUN@CSAIL.MIT.EDU
UC Berkeley EECS & ICSI; CSAIL MIT

David J. Fleet FLEET@CS.TORONTO.EDU
University of Toronto

Andreas Geiger GEIGER@MRT.UKA.DE
Karlsruhe Institute of Technology

Jovan Popović JOVAN@CSAIL.MIT.EDU
CSAIL MIT

Trevor J. Darrell TREVOR@EECS.BERKELEY.EDU
UC Berkeley EECS & ICSI; CSAIL MIT

Neil D. Lawrence NEIL.LAWRENCE@MANCHESTER.AC.UK
University of Manchester

Abstract

In dimensionality reduction approaches, the data are typically embedded in a Euclidean latent space. However for some data sets this is inappropriate. For example, in human motion data we expect latent spaces that are cylindrical or a toroidal, that are poorly captured with a Euclidean space. In this paper, we present a range of approaches for embedding data in a non-Euclidean latent space. Our focus is the Gaussian Process latent variable model. In the context of human motion modeling this allows us to (a) learn models with interpretable latent directions enabling, for example, style/content separation, and (b) generalise beyond the data set enabling us to learn transitions between motion styles even though such transitions are not present in the data.

1. Introduction

Dimensionality reduction is a popular approach to dealing with high dimensional data sets. It is often the case that linear dimensionality reduction, such as principal component analysis (PCA) does not adequately capture the structure of the data. For this reason there has been considerable interest in the machine learning community in non-linear dimensionality reduction. Approaches such as locally linear embedding (LLE), Isomap and maximum variance unfolding (MVU) (Roweis & Saul, 2000; Tenenbaum et al., 2000; Weinberger et al., 2004) all define a topology through interconnections between points in the data space. However, if a given data set is relatively sparse or particularly noisy, these interconnections can stray beyond the 'true' local neighbourhood and the resulting embedding can be poor.

Probabilistic formulations of latent variable models do not usually include explicit constraints on the embedding and therefore the natural topology of the data manifold is not always respected [1]. Even with the correct topology and dimension of the latent space, the learning might get stuck in local minima if the initialization of the model is poor. Moreover, the maximum likelihood solution may not be a good model, due e.g., to the sparseness of the data. To get better models in such cases, more constraints on the model are needed.

This paper shows how explicit topological constraints can be imposed within the context of probabilistic latent variable models. We describe two approaches, both within the context of the Gaussian process latent variable model (GP-LVM) (Lawrence, 2005). The

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] An exception is the back-constrained GP-LVM (Lawrence & Quiñonero-Candela, 2006) where a constrained maximum likelihood algorithm is used to enforce these constraints.

first uses prior distributions on the latent space that encourage a given topology. The second influences the latent space and optimisation through constrained maximum likelihood.

Our approach is motivated by the problem of modeling human pose and motion for character animation. Human motion is an interesting domain because, while there is an increasing amount of motion capture data available, the diversity of human motion means that we will necessarily have to incorporate a large amount of prior knowledge to learn probabilistic models that can accurately reconstruct a wide range of motions. Despite this, most existing methods for learning pose and motion models (Elgammal & Lee, 2004; Grochow et al., 2004; Urtasun et al., 2006) do not fully exploit useful prior information, and many are limited to modeling a single human activity (e.g., walking with a particular style).

This paper describes how prior information can be used effectively to learn models with specific topologies that reflect the nature of human motion. Importantly, with this information we can also model multiple activities, including transitions between them (e.g. from walking to running), even when such transitions are not present in the training data. As a consequence, we can now learn latent variable models with training motions comprising multiple subjects with stylistic diversity, as well as multiple activities, such as running and walking. We demonstrate the effectiveness of our approach in a character animation application, where the user specifies a set of constraints (e.g., foot locations), and the remaining kinematic degrees of freedom are infered.

2. Gaussian Process Latent Variable Models (GP-LVM)

We begin with a brief review of the GP-LVM (Lawrence, 2005). The GP-LVM represents a high-dimensional data set, $\mathbf{Y}$, through a low dimensional latent space, $\mathbf{X}$, and a Gaussian process mapping from the latent space to the data space. Let $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_N]^T$ be a matrix in which each row is a single training datum, $\mathbf{y}_i \in \Re^D$. Let $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$ denote the matrix whose rows represent the corresponding positions in latent space, $\mathbf{x}_i \in \Re^d$. Given a covariance function for the Gaussian process, $k_Y(\mathbf{x}, \mathbf{x}')$, the likelihood of the data given the latent positions is,

$$p(\mathbf{Y} \mid \mathbf{X}, \bar{\beta}) = \frac{1}{Z_1} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_Y^{-1}\mathbf{Y}\mathbf{Y}^T\right)\right), \qquad (1)$$

where Z_1 is a normalization factor, $\mathbf{K}_Y$ is known as the kernel matrix, and $\bar{\beta}$ denotes the kernel hyperparameters. The elements of the kernel matrix are defined by the covariance function, $(\mathbf{K}_Y)_{i,j} = k_Y(\mathbf{x}_i, \mathbf{x}_j)$. A common choice is the radial basis function (RBF), $k_Y(\mathbf{x}, \mathbf{x}') = \beta_1 \exp(-\frac{\beta_2}{2}||\mathbf{x} - \mathbf{x}'||^2) + \frac{\delta_{\mathbf{x},\mathbf{x}'}}{\beta_3}$, where the kernel hyperparameters $\bar{\beta} = \{\beta_1, \beta_2, \beta_3\}$ determine the output variance, the RBF support width, and the variance of the additive noise. Learning in the GP-LVM consists of maximizing (1) with respect to the latent positions, $\mathbf{X}$, and the hyperparameters, $\bar{\beta}$.

When one has time-series data, $\mathbf{Y}$ represents a sequence of observations, and it is natural to augment the GP-LVM with an explicit dynamical model. For example, the Gaussian Process Dynamical Model (GPDM) models the sequence as a latent stochastic process with a Gaussian process prior (Wang et al., 2008) , i.e.,

$$p(\mathbf{X} \mid \bar{\alpha}) = \frac{p(\mathbf{x}_1)}{Z_2} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{K}_X^{-1}\mathbf{X}_{out}\mathbf{X}_{out}^T\right)\right) \qquad (2)$$

where Z_2 is a normalization factor, $\mathbf{X}_{out} = [\mathbf{x}_2, ..., \mathbf{x}_N]^T$, $\mathbf{K}_X \in \Re^{(N-1)\times(N-1)}$ is the kernel matrix constructed from $\mathbf{X}_{in} = [\mathbf{x}_1, ..., \mathbf{x}_{N-1}]$, $\mathbf{x}_1$ is given an isotropic Gaussian prior and $\bar{\alpha}$ are the kernel hyperparameters for $\mathbf{K}_X$; below we use an RBF kernel for $\mathbf{K}_X$. Like the GP-LVM the GPDM provides a generative model for the data, but additionally it provides one for the dynamics. One can therefore predict future observation sequences given past observations, and simulate new sequences.

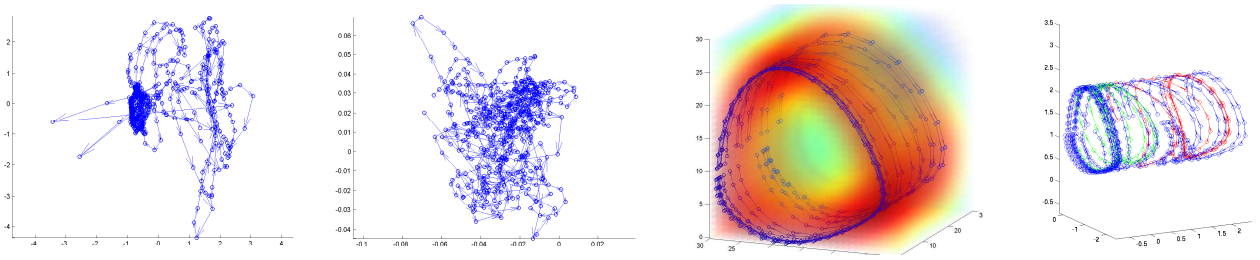
3. Top Down Imposition of Topology

The smooth mapping in the GP-LVM ensures that distant points in data space remain distant in latent space. However, as discussed in (Lawrence & Quiñonero-Candela, 2006), the mapping in the opposite direction is not required to be smooth. While the GPDM may mitigate this effect, it often produces models that are neither smooth nor generalize well (Urtasun et al., 2006; Wang et al., 2008).

To help ensure smoother, well-behaved models, (Lawrence & Quiñonero-Candela, 2006) suggested the use of *back-constraints*, where each point in the latent space is a smooth function of its corresponding point in data space, $x_{ij} = g_j(\mathbf{y}_i; \mathbf{a}_j)$, where $\{\mathbf{a}_j\}_{1\le j\le d}$ is the set of parameters of the mappings. One possible mapping is a kernel-based regression model, where regression on a kernel induced feature space provides the mapping,

$$x_{ij} = \sum_{m=1}^{N} a_{jm} k(\mathbf{y}_i, \mathbf{y}_m) \; . \qquad (3)$$

This approach is known as the back-constrained GP-LVM. When learning the back-constrained GP-LVM,



(a) (b) (c) (d)

Figure 1. When training data contain large stylistic variations and multiple motions, the generic GPDM (a) and the back-constrained GPDM (b) do not produce useful models. Simulations of both models here do not look realistic. (c,d) Hybrid model learned using local linearities for smoothness (i.e., style) and backconstraints for topologies (i.e., content). The training data is composed of 9 walks and 10 runs performed by different subjects and speeds. (c) Likelihood for the reconstruction of the latent points (d) 3D view of the latent trajectories for the training data in blue, and the automatically generated motions of Figs. 3 and 4 in green and red respectively.

one needs to determine the hyperparameters of the kernel matrices (for the back-constraints and the covariance of the GP), as well as the mapping weights, $\{\mathbf{a}_j\}$. (Lawrence & Quiñonero-Candela, 2006) fixed the hyperparameters of the back-constraint's kernel matrix, optimizing over the remaining parameters.

Nevertheless, when learning human motion data with large stylistic variations or different motions, neither GPDM nor back-constrained GP-LVM produce smooth models that generalize well. Fig. 1 depicts three 3–D models learned from 9 walks and 10 runs. The GPDM (Fig. 1(a)) and the back-constrainted GPDM[2] (Fig. 1 (b)) do not generalize well to new runs and walks, nor do they produce realistic animations.

In this paper we show that with a well designed set of back-constraints good models can be learned (Fig. 1(c)). We also consider an alternative approach to the hard constraints on the latent space arising from $g_j(\mathbf{y}_i; \mathbf{a}_j)$. We introduce topological constraints through a prior distribution in the latent space, based on a neighborhood structure learned through a generalized local linear embedding (LLE) (Roweis & Saul, 2000). We then show how to incorporate domain-specific prior knowledge, which allows us to develop motion models with specific topologies that incorporate different activities within a single latent space and transitions between them.

3.1. Locally Linear GP-LVM

The locally linear embedding (LLE) (Roweis & Saul, 2000) preserves topological constraints by finding a representation based on reconstruction in a low dimensional space with an optimized set of local weightings. Here we show how the LLE objective can be combined with the GP-LVM, yielding a *locally linear GP-LVM* (LL-GPLVM).

[2]We use an RBF kernel for the inverse mapping in (3).

The locally linear embedding assumes that each data point and its neighbors lie on, or close to, a locally linear patch on the data manifold. The local geometry of these patches can then be characterized by linear coefficients that reconstruct each data point from its neighbors. This is done in a three step procedure: (1) the K nearest neighbors, $\{\mathbf{y}_j\}_{j\in\eta_i}$, of each point, $\mathbf{y}_i$, are computed using Euclidean distance in the input space, $d_{ij} = ||\mathbf{y}_i - \mathbf{y}_j||^2$; (2) the weights $\mathbf{w} = \{w_{ij}\}$ that best reconstruct each data point from its neighbors are obtained by minimizing $\Phi(\mathbf{w}) = \sum_{i=1}^{N} ||\mathbf{y}_i - \sum_{j\in\eta_i} w_{ij}\mathbf{y}_j||^2$; and (3) the latent positions $\mathbf{x}_i$ best reconstructed by the weights w_{ij} are computed by minimizing $\Phi(\mathbf{X}) = \sum_{i=1}^{N} ||\mathbf{x}_i - \sum_{j\in\eta_i} w_{ij}\mathbf{x}_j||^2$.

In the LLE, the weight matrix $\mathbf{w}$ is sparse (only a small number of neighbors is used), and the two minimizations can be computed in closed form. In particular, computing the weights can be done by solving, $\forall j \in \eta_i$, the following system,

$$\sum_k C_{kj}^{sim} w_{ij}^{sim} = 1 \ , \tag{4}$$

where $C_{kj}^{sim} = (\mathbf{y}_i - \mathbf{y}_k)^T(\mathbf{y}_i - \mathbf{y}_j)$ if $j,k \in \eta_i$, and 0 otherwise. Once the weights are computed, they are rescaled so that $\sum_j w_{ij} = 1$.

The LLE energy function can be interpreted, for a given set of weights $\mathbf{w}$, as a prior that forces each latent point to be locally reconstructed by its neighbors,i.e., $p(\mathbf{X}|\mathbf{w}) = \frac{1}{Z}\exp\left\{-\frac{1}{\sigma^2}\Phi(\mathbf{X})\right\}$, where Z is a normalization constant, and σ^2 represents a global scaling of the prior. Note that strictly speaking this is not a proper prior as it is conditioned on the weights which depend on the training data. Following (Roweis & Saul, 2000), we first compute the neighbors based on the Euclidean distance. For each training point $\mathbf{y}_i$, we then compute the weights solving Eq. (4).

Learning the LL-GPLVM is then equivalent to mini-

mizing the negative log posterior of the model, [3] i.e.,

$$\begin{aligned}\mathcal{L}_{\mathcal{S}} &= \log\ p(\mathbf{Y}|\mathbf{X},\bar{\beta})\ \ p(\bar{\beta})\ \ p(\mathbf{X}|\mathbf{w}) \\ &= \frac{D}{2}\ln|\mathbf{K}_Y| + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_Y^{-1}\mathbf{Y}\mathbf{Y}^T\right) + \sum_i \ln\beta_i \\ &\quad + \frac{1}{\sigma^2}\sum_{k=1}^{d}\sum_{i=1}^{N}\|x_i^k - \sum_{j=1}^{N} w_{ij}^k x_j^k\|^2 + C\ , \quad (5)\end{aligned}$$

where C is a constant, and x_i^k is the k-th component of $\mathbf{x}_i$. Note that we have extended the LLE to have a different prior for each dimension. This will be useful below as we incorporate different sources of prior knowledge. Fig. 2 (a) shows a model of 2 walks and 2 runs learned with the locally linear GPDM. Note how smooth the latent trajectories are.

We now have general tools to influence the structure of the models. In what follows we generalize the top-down imposition of topology strategies (i.e. back-constraints and locally linear GP-LVM) to incorporate domain specific prior knowledge.

4. Reflecting Knowledge in Latent Space Structure

A problem for modeling human motion data is the sparsity of the data relative to the diversity of naturally plausible motions. For example, while we might have a data set comprising different motions, such as runs, walks *etc.*, the data may not contain transitions between motions. In practice however, we know that these motions will be approximately cyclic and that transitions can only physically occur at specific points in the cycle. How can we encourage a model to respect such topological constraints which arise from prior knowledge?

We consider two alternatives to solve this problem. First, we show how one can adjust the distance metric used in the locally linear embedding to better reflect different types of prior knowledge. We then show how one can define similarity measures for use with the back-constrained GP-LVM. Both these approaches encourage the latent space to construct a representation that reflects our prior knowledge. They are complementary and can be combined to learn better models.

[3] When learning a locally linear GPDM, the dynamics and the locally linear prior are combined as a product of potentials. The objective function becomes $\mathcal{L}_{\mathcal{S}} + \frac{d}{2}\ln|\mathbf{K}_X| + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_X^{-1}\mathbf{X}_{out}\mathbf{X}_{out}^T\right) + \sum_i \ln\alpha_i$, with $\mathcal{L}_{\mathcal{S}}$ defined as in (5).

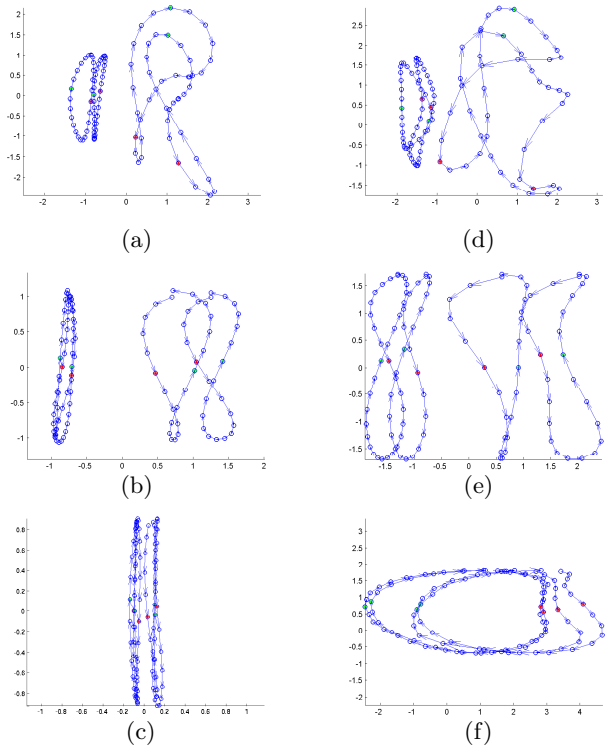


Figure 2. First two dimensions of 3–D models learned using (a) LL-GPDM (b) LL-GPDM with topology (c) LL-GPDM with topology and transitions. (d) Back-constrained GPDM with an RBF mapping. (e) GPDM with topology through backconstraints. (f) GPDM with backconstraints for the topology and transitions. For the models using topology, the cyclic structure is imposed in the last 2 dimensions. The two types of transition points (left and right leg contact points) are shown in red and green, and are used as prior knowledge in (c,f).

4.1. Prior Knowledge through Local Linearities

We now turn to consider how one might incorporate prior knowledge in the LL-GPLVM framework. This is accomplished by replacing the local Euclidean distance measures used in Section 3.1 with other similarity measures. That is, we can modify the covariance used to compute the weights in Eq. (4) to reflect our prior knowledge in the latent space. We consider two examples: the first involves transitions between activities; with the second we show how topological constraints can be placed on the form of the latent space.

Covariance for Transitions Modeling transitions between motions is important in character animation. Transitions can be infered automatically based on similarity between poses (Kovar et al., 2002) or at points of non-linearity of the dynamics (Bissacco, 2005), and they can be used for learning. For example, for motions as walking or running, two types of transitions can be identified: left and right foot ground contacts.

To model such transitions, we define an index on the frames of the motion sequence, $\{t_i\}_{i=1}^N$. We then define subsets of this set, $\{\hat{t}_i\}_{i=1}^M$, which represents frames where transitions are possible. To capture transitions in the latent model we define the elements for the covariance matrix as follows,

$$C_{kj}^{trans} = 1 - \left[\delta_{kj} \exp(-\zeta(t_k - t_j)^2)\right] \quad (6)$$

with ζ a constant, and $\delta_{ij} = 1$ if t_i and t_j are in the same set $\{\hat{t}_k\}_{k=1}^M$, and otherwise $\delta_{ij} = 0$. This covariance encourages the latent points at which transitions are physically possible to be close together.

Covariance for Topologies We now consider covariances that encourage the latent space to have a particular topology. Specifically we are interested in suitable topologies for walking and running data. Because the data are approximately periodic, it seems appropriate to have a non-Cartesian topology. To this end one can extract the phase of the motion[4], ϕ, and use it with a covariance to encourage the latent points to exhibit a periodic topological structure within a Cartesian space. As an example we consider a cylindrical topology within a 3–D latent space by constraining two of the latent dimensions with the phase. In particular, to represent the cyclic motion we construct a distance function on the unit circle, where a latent point corresponding to phase ϕ is represented with coordinates $(\cos(\phi), \sin(\phi))$. To force a cylindrical topology on the latent space, we specify different covariances for each latent dimension

$$C_{k,j}^{cos} = (\cos(\phi_i) - \cos(\phi_k))\,(\cos(\phi_i) - \cos(\phi_j)) \quad (7)$$

$$C_{k,j}^{sin} = (\sin(\phi_i) - \sin(\phi_k))\,(\sin(\phi_i) - \sin(\phi_j))\,, \quad (8)$$

with $k, j \in \eta_i$. The covariance for the remaining dimension is constructed as usual, based on Euclidean distance in the data space. Fig. 2 (b) shows a GPDM constrained in this way, and in Fig. 2 (c) the covariance is augmented with transitions.

Note that the use of different distance measures for each dimension of the latent space implies that the neighborhood and the weights in the locally linear prior will also be different for each dimension. Here, three different locally linear embeddings form the prior distribution.

4.2. Prior Knowledge with Back Constraints

As explained above, we can also design back-constraints to influence the topology and learn useful transitions. This can be done by replacing the kernel of Eq. (3). Many kernels have interpretations as similarity measures. In particular, any similarity measure that leads to a positive semi-definite matrix can be interpreted as a kernel. Here, just as we define covariance matrices above, we extend the original formulation of back constraints by constructing similarity measures (i.e., kernels) to reflect prior knowledge.

Similarity for Transitions To capture transitions between two motions, we wish to design a kernel that expresses strong similarity between points in the respective motions where transitions may occur. We can encourage transition points of different sequences to be proximal with the following kernel matrix for the back-constraint mapping:

$$k^{trans}(t_i, t_j) = \sum_m \sum_l \delta_{ml} k(t_i, \hat{t}_m) k(t_j, \hat{t}_l) \quad (9)$$

where $k(t_i, \hat{t}_l)$ is an RBF centered at $\hat{t}_l$, and $\delta_{ml} = 1$ if $\hat{t}_m$ and $\hat{t}_l$ are in the same set. The influence of the back-constraints is controlled by the support width of the RBF kernel.

Topologically Constrained Latent Spaces We now consider kernels that force the latent space to have a particular topology. To force a cylindrical topology on the latent space, we can introduce similarity measures based on the phase, specifying different similarity measures for each latent dimension. As before we construct a distance function in the unit circle, that takes into account the phase. A periodic mapping can be constructed from a kernel matrix as follows,

$$x_{n,1} = \sum_{m=1}^{N} a_m^{cos} k(\cos(\phi_n), \cos(\phi_m)) + a_0^{cos} \delta_{n,m},$$

$$x_{n,2} = \sum_{m=1}^{N} a_m^{sin} k(\sin(\phi_n), \sin(\phi_m)) + a_0^{sin} \delta_{n,m},$$

where k is an RBF kernel function, and $x_{n,i}$ is the i^{th} coordinate of the n^{th} latent point. These two mappings project onto two dimensions of the latent space, forcing them to have a periodic structure (which comes about through the sinusoidal dependence of the kernel on phase). Fig. 2 (e) shows a model learned using GPDM with the last two dimensions constrained in this way (the third dimension is out of plane). The first dimension is constrained by an RBF mapping on the input space. Each dimension's kernel matrix can then be augmented by adding the transition similarity of Eq.(9), resulting in the model shown in Fig. 2 (f).

4.3. Model Combination

One advantage of our framework is that covariance matrices can be combined in a principled manner to form

[4]The phase can be easily extracted from the data by Fourier analysis or by detecting key postures and interpolating the phases between them. Another idea, not further explored here, would be to optimize the GP-LVM with respect to the phase.

new covariance matrices. Covariances can be multiplied (on an element by element basis) or added together. Similarly, similarities can be combined. Multiplication has, loosely speaking, an 'AND gate effect', i.e. both similarity measures must agree that an object is similar for their product to express similarity. Adding them produces more of an 'OR gate effect', i.e. if either representation expresses similarity the resulting measure will also express similarity.

The two sections above have shown how to incorporate prior knowledge in the GP-LVM by means of 1) local linearities and 2) back-constraints. In general, the latter should be used when the manifold has a well-defined topology, since it has more influence on the learning. When the topology is not so well defined (e.g., due to noise) one should use local linearities. Both techniques are complementary and can be combined straightforwardly by including priors over some dimensions, and constraining the others through back-constraint mappings. Fig. 1 shows a model learned with LL-GPDM for smoothness and back-constraints for topology.

4.4. Multiple Activities and Transitions

Once we know how to ensure that transition points are close together and that the latent structure has the desired topology, we still need to address two issues. How do we learn models that have very different dynamics? How can we simulate dynamical models that lie somewhere between the different training motions? Our goal in this section is to show how latent models for different motions can be learned independently, but in a shared latent space that facilitates transitions between activities with different dynamics.

Let $\mathbf{Y} = [\mathbf{Y}_1^T, ..., \mathbf{Y}_M^T]^T$ denote training data for M different activities. Each $\mathbf{Y}_m$ comprises several different motions. Let $\mathbf{X} = [\mathbf{X}_1^T, ..., \mathbf{X}_M^T]^T$ denote the corresponding latent positions. When dealing with multiple activities, a single dynamical model cannot cope with the complexity of the different dynamics. Instead, we consider a model where the dynamics of each activity are modeled independently[5]. This has the advantage that a different kernel can be used for each activity.

To enable interpolation between motions with different dynamics, we combined these independent dynamical models in the form of a mixture model. This allows us to produce motions that gracefully transition between different styles and motion types (Figs. 3 and 4).

[5] Another interpretation is that we have a block diagonal kernel matrix for the GP that governs the dynamics.

5. Results

We demonstrate the effectiveness of our approach with two applications. First we show how models of multiple activities can be learned, and realistic animations can be produced by drawing samples from the model. We then show an interactive character animation application, where the user specifies a set of sparse constraints and the remaining kinematic degrees of freedom are infered.

5.1. Learning multiple activities

We first considered a small training set comprised of 4 gait cycles (2 walks and 2 runs) performed by one subject at different speeds. Fig. 2 shows the latent spaces learned under different prior constraints. All the models are learned using two independent dynamical models, one for walking and one for running. Note how the phases are aligned when imposing a cylindrical topology, and how the LL-GPDM is smooth. Notice the difference between the LL-GPDM (Fig. 2 (c)) and the backconstrained GPDM (Fig. 2 (f)) when transition constraints are included. Neverthess, both models ensure that the transition points (shown in red and green) are proximal.

Fig. 1 (c) shows a hybrid model learned using LL-GPDM for smoothness, and back-constraints for topology. The larger training set comprises approximately one gait cycle from each of 9 walking and 10 running motions performed by different subjects at different speeds (3 km/h for walking, 6–12 km/h for running). Colors in Fig. 1 (a) represent the variance of the GP as a function of latent position. Only points close to the surface of the cylinder produce poses with high certainty.

We now illustrate the model's ability to simulate different motions and transitions. Given an initial latent position $\mathbf{x}_0$, we generate new motions by sampling the mixture model, and using mean prediction for the reconstruction. Choosing different initial conditions results in very different simulations (Fig. 1 (d)). The training data are shown in blue. For the first simulation (depicted in green), the model is initialized to a running pose with a latent position not far from walking data. The system transitions to walking quite naturally. The resulting animation is depicted in Fig. 3. For the second example (in red), we initialize the simulation to a latent position far from walking data. The system evolves to different running styles and speeds (Fig. 4). Note how the dynamics, and the strike length, change considerably during simulation.

Figure 3. **Transition from running to walking**: The system transitions from running to walking in a smooth and realistic way. The transition is encouraged by incorporating prior knowledge in the model. The latent trajectories are shown in green in Fig. 1 (d).

Figure 4. **Different running styles and speeds**: The system is able to simulate a motion with considerably changes in speed and style. The latent trajectories are shown in red in Fig. 1 (d).

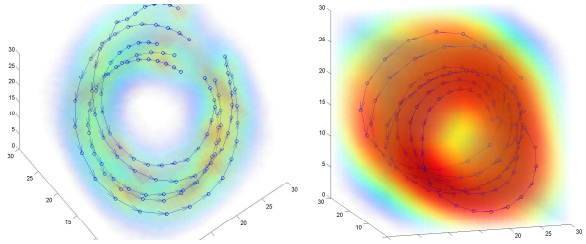


Figure 5. Single activity 3D latent models learned from (left) 5 jumps of 2 different subjects using local linearities, (b) 7 walking cycles of one subject using back-constraints.

5.2. Character animation from constraints

A key problem in the film and game industry is the lack of tools to allow designers to easily generate animations. Traditional techniques such as keyframing are time consuming; an expert can expend days in generating a few seconds of animation. A very useful tool would provide the user with a simple way of generating motions from constraints that she/he defined. Typical constraints are keyframes (i.e., specification of the position of the full body in a particular time instant), or joint trajectories. Here we use the topologically constrained motion models as priors over the space of possible motions.

Our motion estimation formulation is based on a state-space model with a GPDM prior over pose and motion. Given the state, $\phi_t = (\mathbf{y}_t, \mathbf{x}_t)$, the goal is to estimate the state sequence $\phi_{1:T} = (\phi_1, \cdots, \phi_T)$ that satisfies the user constraints $\mathbf{u}_{1:J}$. Inference is performed in a *Batch* mode, so that the state is infered all at once. The posterior can be expressed as

$$p(\phi_{1:T}|\mathbf{u}_{1:J}, \mathbf{M}) \propto p(\mathbf{u}_{1:J}|\phi_{1:T})p(\phi_{1:T}|\mathbf{M}) \qquad (10)$$

where we assumed that $p(\mathbf{u}_{1:J})$ is uniformily distributed; all the user constraints are equally probable. The prediction distribution $p(\phi_{1:T}|\mathbf{M})$ can be further factored as follows

$$p(\phi_{1:T}|\mathbf{M}) = p(\mathbf{x}_{1:T}|\mathbf{M}) \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{M}) \qquad (11)$$

Rather than approximating the entire posterior, we use hill-climbing to find MAP estimates. Assuming that the user constraints are noise-free, the minimization can be expressed as

$$\begin{aligned} \min_{\phi_{1:T}} \quad & \mathcal{L}_{pose} + \mathcal{L}_{dyn} + \mathcal{L}_{smooth} \\ \text{subject to} \quad & ||\mathbf{u} - \mathbf{f}(\mathbf{y}_{\psi(\mathbf{u})})|| = \mathbf{0} \end{aligned} \qquad (12a)$$

where $\mathbf{f}$ is a forward kinematics function (i.e., a function that maps joint angles to positions in the 3D world), $\psi(\mathbf{u})$ is a function that outputs the frame where each constraint $\mathbf{u}_j$ is defined, $\mathcal{L}_{pose} = -\sum_i \ln p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{M})$ and $\mathcal{L}_{dyn} = -\ln p(\mathbf{x}_{1:T}|M)$ are the pose and dynamics likelihood from the GPDM prior (Urtasun et al., 2006), and $\mathcal{L}_{smooth} = \frac{1}{\sigma_s^2}\sum_{t=1}^{T-1}\sum_{j=1}^{P}\frac{1}{\sigma_j^2}(y_{t+1}^j - y_t^j)^2$ is a term that encourage smooth motions, where y_t^j is the j-th component of $\mathbf{y}_t$, and σ_j^2 is a constant that encounters from the fact that each degree of freedom has a different variance.

Initialization is important since a large number of variables need to be optimised and our objective function is non-convex. In particular, we sample the model starting at each training point and use as initialization the sample that is closest to the user constraints.

To demonstrate the effectiveness of our approach we learned models of two different motions, walking and jumping (Fig. 5). We impose smoothness and cyclic topologies using back-constraints for the walking and local linearities for the jumping. We demonstrate the ability of the model to generalize to unseen styles.

We first show how the model can produce realistic animations from a very small set of user defined constraints. The user specifies the contact points of the foot with the ground (first row of Fig. 6) for walking and the foot trajectories for the jumping (third row of Fig. 6), and the rest of the degrees of freedom are infered producing very realistic animations.

The model can also generalize to styles very different

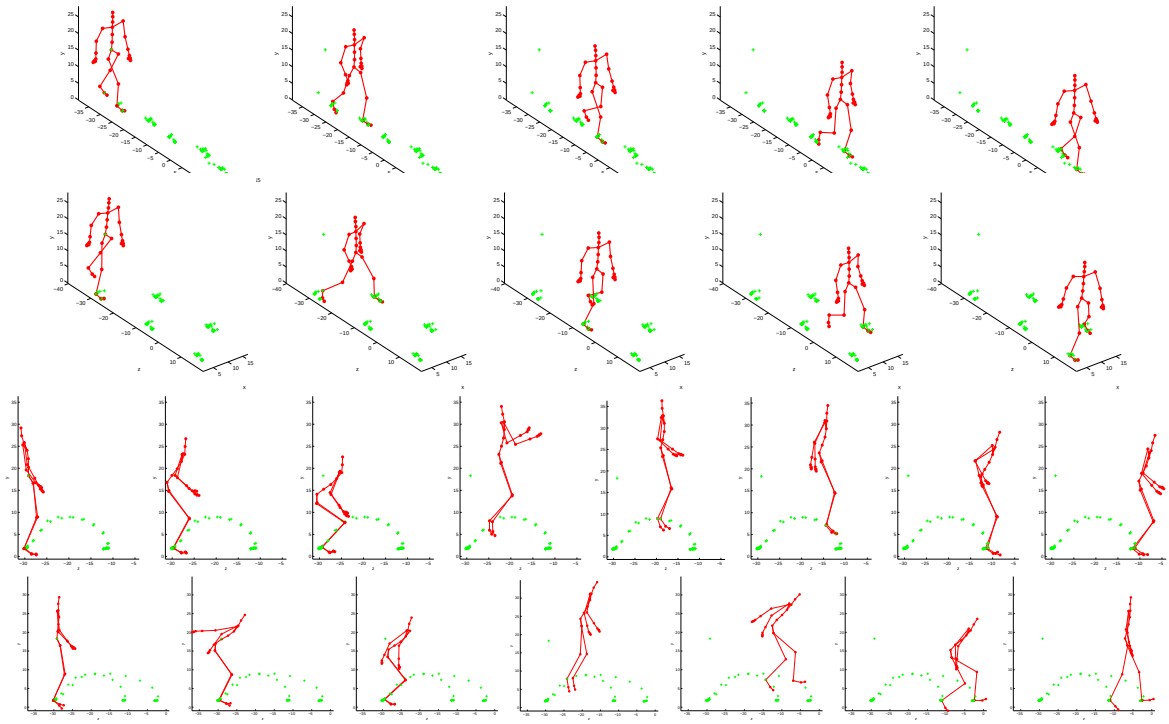


Figure 6. Animations generated from a set of foot constraints (green). **First row**: Normal walk. **Second row**: Generalization a different style by changing the user constraints to be separate in the coronal plane. **Third row**: Short jump. **Last row**: Longer stylistic jump. *See video at http://people.csail.mit.edu/rurtasun*

from the ones in the training set, by imposing constraints that can be satisfied only by motions very different from the training data. In particular, the user placed the foot constraints far in the coronal plane for walking. Consequently the character opens the legs to satisfy the constraints (second row of Fig. 6). In the last row of Fig. 6 the user places the foot trajectories to create a jump with a style very different from the traning data (the character opens his legs and bends his body and arms in an exaggerated way).

6. Conclusions

In this paper we have proposed a general framework of probabilistic models that learn smooth latent variable models of different activities within a shared latent space. We have introduced a principled way to include prior knowledge, that allow us to learn specific topologies and transitions between the different motions. Although we have learned models composed of walking, running and jumping, our framework is general, being applicable in any data sets where there is a large degree of prior knowledge for the problem domain, but the data availability is relatively sparse compared to its complexity.

References

Bissacco, A. (2005). Modeling and Learning Contact Dynamics in Human Motion In *CVPR*. (pp. 421–428).

Elgammal, A., & Lee, C. (2004). Inferring 3D Body Pose from Silhouettes using Activity Manifold Learning. In *CVPR* (pp. 681–688).

Grochow, K., Martin, S., Hertzmann, A., & Popovic, Z. (2004). Style-based inverse kinematics In *SIGGRAPH*.

Kovar, L., Gleicher, M., & Pighin, F. (2002). Motion Graphs In *SIGGRAPH*, (pp. 473–482).

Lawrence, N. (2005). Probabilistic non-linear principal component analysis with gaussian process latent variable models. *JMLR*, *6*, 1783–1816.

Lawrence, N. D., & Quiñonero-Candela, J. (2006). Local distance preservation in the GP-LVM through back constraints In *ICML* (pp. 96–103).

Rasmussen, C. E., & Williams, C. K. (2006). *Gaussian process for machine learning*. MIT Press.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*.

Tenenbaum, J., de Silva, V., & Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Urtasun, R., Fleet, D. J., & Fua, P. (2006). 3d people tracking with gaussian process dynamical models. In *CVPR*, (pp. 932–938).

Wang, J., Fleet, D. J., & Hertzman, A. (2008). Gaussian process dynamical models . In *PAMI 30*(2), 283–298 .

Weinberger, K. Q., Sha, F., & Saul, L. K. (2004). Learning a kernel matrix for nonlinear dimensionality reduction. In *ICML*, (pp. 106–113).

Beam Sampling for the Infinite Hidden Markov Model

Jurgen Van Gael JV279@CAM.AC.UK
Yunus Saatci YS267@CAM.AC.UK
Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK

Yee Whye Teh YWTEH@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, University College London, WC1N 3AR, UK

Zoubin Ghahramani ZOUBIN@ENG.CAM.AC.UK
Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK

Abstract

The infinite hidden Markov model is a nonparametric extension of the widely used hidden Markov model. Our paper introduces a new inference algorithm for the infinite Hidden Markov model called *beam sampling*. Beam sampling combines slice sampling, which limits the number of states considered at each time step to a finite number, with dynamic programming, which samples whole state trajectories efficiently. Our algorithm typically outperforms the Gibbs sampler and is more robust. We present applications of iHMM inference using the beam sampler on changepoint detection and text prediction problems.

1. Introduction

The hidden Markov model (HMM) (Rabiner, 1989) is one of the most widely used models in machine learning and statistics for sequential or time series data. The HMM consists of a hidden state sequence with Markov dynamics, and independent observations at each time given the corresponding state. There are three learning related tasks associated with the HMM: inference of the hidden state sequence, learning of the parameters, and selection of the right model size.

Inference for the hidden state trajectory can be performed exactly using the forward-backward algorithm (Rabiner, 1989), a dynamic programming algorithm with $O(TK^2)$ computational costs where T is the number of time steps and K number of states.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

The standard approach to learning uses the Baum-Welch algorithm, a special instance of the EM algorithm (Dempster et al., 1977) which produces (locally) maximum likelihood (ML) parameters. Such ML learning of parameters can potentially lead to overfitting if the model size is inappropriate for the amount of data available. This can be partially mitigated using a more fully Bayesian learning procedure, e.g. using variational approximations (MacKay, 1997) or Markov chain Monte Carlo (MCMC) sampling (Scott, 2002). Such Bayesian approaches also produce estimates of the marginal probability of data, which can be used to select for the appropriate model size (or to average over model sizes if ones desires a more Bayesian analysis). Such model selection procedures can be computationally expensive since multiple HMMs of different sizes need to be explored.

A new twist on the problem of model selection has emerged in recent years with the increasing popularity of nonparametric Bayesian models. These are models of infinite capacity, a finite portion of which will be used to model a finite amount of observed data. The idea of searching/averaging over the space of finite models is replaced with Bayesian inference over the size of submodel used to explain data. Examples of successful applications of nonparametric Bayesian methods include Gaussian Processes (Rasmussen & Williams, 2005) for regression and classification, Dirichlet Process (DP) mixture models (Escobar & West, 1995; Rasmussen, 2000) for clustering heterogeneous data and density estimation, Indian Buffet Processes for latent factor analysis (Griffiths & Ghahramani, 2006), and defining distributions over non-trivial combinatorial objects such as trees (Teh et al., 2008).

The Infinite Hidden Markov Model (iHMM), otherwise known as the HDP-HMM, (Beal et al., 2002) is a non-

parametric Bayesian extension of the HMM with an infinite number of hidden states. Exact Bayesian inference for the iHMM is intractable. Specifically, given a particular setting of the parameters the forward-backward algorithm cannot be applied since the number of states K is infinite, while with the parameters marginalized out all hidden state variables will be coupled and the forward-backward algorithm cannot be applied either. Currently the only approximate inference algorithm available is Gibbs sampling, where individual hidden state variables are resampled conditioned on all other variables (Teh et al., 2006). Unfortunately convergence of Gibbs sampling is notoriously slow in the HMM setting due to the strong dependencies between consecutive time steps often exhibited by time series data (Scott, 2002).

In this paper we propose a new sampler for the iHMM called *beam sampling*. Beam sampling combines two ideas—slice sampling and dynamic programming—to sample whole state trajectories efficiently. Our application of slice sampling (Neal, 2003) is inspired by (Walker, 2007), who used it to limit the number of clusters considered when sampling assignment variables in DP mixtures to a finite number. We apply slice sampling to limit to a finite number the states considered in each time step of the iHMM, so that dynamic programming can be used to sample whole state trajectories efficiently. We call our proposal beam sampling due to its similarity to beam search, a heuristic procedure for finding the maximum a posteriori trajectory given observations in non-linear dynamical systems. The underlying idea in both is to limit the search to a small number of states so that a good trajectory can be found using reasonable computational resources. However, ours is a MCMC sampling method with guaranteed convergence to the true posterior.

We first present a self-contained description of the iHMM using the Hierarchical Dirichlet process (HDP) formalism (Teh et al., 2006) in Section 2, followed by a discussion of Gibbs sampling in Section 3. We introduce beam sampling in Section 4 and compare it against Gibbs sampling on both artificial and real datasets in Section 5. We find that beam sampling is (1) at least as fast if not faster than Gibbs sampling; (2) more robust than Gibbs sampling as its performance is not as dependent on initialization and hyperparameter choice; (3) handles non-conjugacy in the model more naturally; (4) straightforward to implement. We conclude in Section 6 with a discussion and suggestions for other cases in which beam sampling might prove useful. All software is available from *http://mlg.eng.cam.ac.uk/jurgen* to encourage more widespread adoption of the iHMM and the beam sampler.

2. The Infinite Hidden Markov Model

We start this section by describing the finite HMM, then taking the infinite limit to obtain an intuition for the infinite HMM, followed by a more precise definition. A finite HMM consists of a hidden state sequence $\boldsymbol{s} = (s_1, s_2, \ldots, s_T)$ and a corresponding observation sequence $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$. Each state variable s_t can take on a finite number of states, say $1 \ldots K$. Transitions between states are governed by Markov dynamics parameterized by the transition matrix $\boldsymbol{\pi}$, where $\pi_{ij} = p(s_t = j | s_{t-1} = i)$, while the initial state probabilities are $\pi_{0i} = p(s_1 = i)$. For each state $s_t \in \{1 \ldots K\}$ there is a parameter ϕ_{s_t} which parametrizes the observation likelihood for that state: $y_t | s_t \sim F(\phi_{s_t})$. Given the parameters $\{\pi_0, \boldsymbol{\pi}, \boldsymbol{\phi}, K\}$ of the HMM, the joint distribution over hidden states $\boldsymbol{s}$ and observations $\boldsymbol{y}$ can be written (with $s_0 = 0$):

$$p(\boldsymbol{s}, \boldsymbol{y} | \pi_0, \boldsymbol{\pi}, \boldsymbol{\phi}, K) = \prod_{t=1}^{T} p(s_t | s_{t-1}) p(y_t | s_t)$$

We complete the Bayesian description by specifying the priors. Let the observation parameters $\boldsymbol{\phi}$ be iid drawn from a prior distribution H. With no further prior knowledge on the state sequence, the typical prior for the transition (and initial) probabilities are symmetric Dirichlet distributions.

A naïve way to obtain a nonparametric HMM with an infinite number of states might be to use symmetric Dirichlet priors over the transition probabilities with parameter α/K and take $K \to \infty$. Such an approach has been successfully used to derive DP mixture models (Rasmussen, 2000) but unfortunately does not work in the HMM context. The subtle reason is that there is no coupling across transitions out of different states since the transition probabilities are given independent priors (Beal et al., 2002). To introduce coupling across transitions, one may use a hierarchical Bayesian formalism where the Dirichlet priors have shared parameters and given a higher level prior, e.g.

$$\begin{aligned} \boldsymbol{\pi}_k &\sim \mathsf{Dirichlet}\left(\alpha \boldsymbol{\beta}\right), \\ \boldsymbol{\beta} &\sim \mathsf{Dirichlet}\left(\gamma/K \ldots \gamma/K\right) \end{aligned} \tag{1}$$

where $\boldsymbol{\pi}_k$ are transition probabilities out of state k and $\boldsymbol{\beta}$ are the shared prior parameters. As $K \to \infty$, the hierarchical prior (1) approaches (with some alterations) a *hierarchical Dirichlet process* (Teh et al., 2006).

A hierarchical Dirichlet process (HDP) is a set of Dirichlet processes (DPs) coupled through a shared random base measure which is itself drawn from a DP (Teh et al., 2006). Specifically, each $G_k \sim \mathsf{DP}(\alpha, G_0)$ with shared base measure G_0, which can be understood as the mean of G_k, and concentration parameter $\alpha > 0$, which governs variability around G_0,

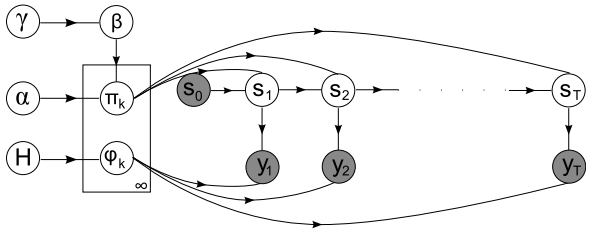


Figure 1. iHMM Graphical Model

with small α implying greater variability. The shared base measure is itself given a DP prior: $G_0 \sim \mathsf{DP}(\gamma, H)$ with H a global base measure. The stick-breaking construction for HDPs shows that the random measures can be expressed as follows: $G_0 = \sum_{k'=1}^{\infty} \beta_{k'} \delta_{\phi_{k'}}$ and $G_k = \sum_{k'=1}^{\infty} \pi_{kk'} \delta_{\phi_{k'}}$, where $\boldsymbol{\beta} \sim \mathsf{GEM}(\gamma)$ is the stick-breaking construction for DPs (Sethuraman, 1994), $\boldsymbol{\pi}_k \sim \mathsf{DP}(\alpha, \boldsymbol{\beta})$, and each $\phi_{k'} \sim H$ independently.

Identifying each G_k as describing both the transition probabilities $\pi_{kk'}$ from state k to k' and the emission distributions parametrized by $\phi_{k'}$, we can now formally define the iHMM as follows:

$$\boldsymbol{\beta} \sim \mathsf{GEM}(\gamma), \qquad \boldsymbol{\pi}_k | \boldsymbol{\beta} \sim \mathsf{DP}(\alpha, \boldsymbol{\beta}), \qquad \phi_k \sim H, \quad (2)$$

$$s_t | s_{t-1} \sim \mathsf{Multinomial}(\boldsymbol{\pi}_{s_{t-1}}), \qquad y_t | s_t \sim F(\phi_{s_t}). \quad (3)$$

The graphical model corresponding to this hierarchical model is shown in figure 1. Thus $\beta_{k'}$ is the prior mean for transition probabilities leading into state k', and α governs the variability around the prior mean. If we fix $\boldsymbol{\beta} = (\frac{1}{K} \ldots \frac{1}{K}, 0, 0 \ldots)$ where the first K entries are $\frac{1}{K}$ and the remaining are 0, then transition probabilities into state k' will be non-zero only if $k' \in \{1 \ldots K\}$, and we recover the Bayesian HMM of (MacKay, 1997).

Finally we place priors over the hyperparameters α and γ. A common solution, when we do not have strong beliefs about the hyperparameters, is to use gamma hyperpriors: $\alpha \sim \mathsf{Gamma}(a_\alpha, b_\alpha)$ and $\gamma \sim \mathsf{Gamma}(a_\gamma, b_\gamma)$. (Teh et al., 2006) describe how these hyperparameters can be sampled efficiently, and we will use this in the experiments to follow.

3. The Gibbs Sampler

The Gibbs sampler was the first sampling algorithm for the iHMM that converges to the true posterior. One proposal builds on the direct assignment sampling scheme for the HDP in (Teh et al., 2006) by marginalizing out the hidden variables $\boldsymbol{\pi}, \boldsymbol{\phi}$ from (2), (3) and ignoring the ordering of states implicit in $\boldsymbol{\beta}$. Thus we only need to sample the hidden trajectory $\boldsymbol{s}$, the base DP parameters $\boldsymbol{\beta}$ and the hyperparameters α, γ. Sampling $\boldsymbol{\beta}, \alpha, \gamma$ is exactly the same as for the HDP so we refer to (Teh et al., 2006) for details.

In order to resample s_t, we need to compute the probability $p(s_t | s_{-t}, \boldsymbol{\beta}, \boldsymbol{y}, \alpha, H) \propto p(y_t | s_t, \boldsymbol{s}_{-t}, \boldsymbol{y}_{-t}, H) \cdot p(s_t | \boldsymbol{s}_{-t}, \boldsymbol{\beta}, \alpha)$. The first factor is the conditional likelihood of y_t given $\boldsymbol{s}$, $\boldsymbol{y}$ and H: $\int p(y_t | s_t, \boldsymbol{\phi}_{s_t}) p(\boldsymbol{\phi}_{s_t} | \boldsymbol{s}_{-t}, \boldsymbol{y}_{-t}, H) d\boldsymbol{\phi}_{s_t}$. This is easy to compute when the base distribution H and likelihood F from equations (2) and (3) are conjugate. For the second factor we can use the fact that the hidden state sequence is Markov. Let n_{ij} be the number of transitions from state i to state j excluding time steps $t-1$ and t. Let $n_{\cdot i}, n_{i \cdot}$ be the number of transitions in and out of state i. Finally, let K be the number of distinct states in $\boldsymbol{s}_{-t}$. Then we have that[1] $p(s_t = k | \boldsymbol{s}_{-t}, \boldsymbol{\beta}, \alpha) \propto$

$$\begin{cases} (n_{s_{t-1},k} + \alpha\beta_k) \frac{n_{k,s_{t+1}} + \alpha\beta_{s_{t+1}}}{n_{k\cdot} + \alpha} & \text{if } k \le K,\ k \ne s_{t-1} \\ (n_{s_{t-1},k} + \alpha\beta_k) \frac{n_{k,s_{t+1}} + 1 + \alpha\beta_{s_{t+1}}}{n_{k\cdot} + 1 + \alpha} & \text{if } k = s_{t-1} = s_{t+1} \\ (n_{s_{t-1},k} + \alpha\beta_k) \frac{n_{k,s_{t+1}} + \alpha\beta_{s_{t+1}}}{n_{k\cdot} + 1 + \alpha} & \text{if } k = s_{t-1} \ne s_{t+1} \\ \alpha\beta_k \beta_{s_{t+1}} & \text{if } k = K+1. \end{cases}$$

For each $1 \le t \le T$ we need to compute $\mathcal{O}(K)$ probabilities, hence the Gibbs sampler has an $\mathcal{O}(TK)$ computational complexity. Non-conjugate models can be handled using more sophisticated sampling techniques. In our experiments below, we used algorithm 8 from (Neal, 2000).

The Gibbs sampler's success is due to its straightforward implementation. However, it suffers from one major drawback: sequential and time series data are likely to be strongly correlated. For example, if we know the value of a stock at time t then we can be reasonably sure that it will be similar at time $t+1$. As is well known, this is a situation which is far from ideal for the Gibbs sampler: strong correlations in the hidden states will make it unlikely that individual updates to s_t can cause large blocks within $\boldsymbol{s}$ to be changed. We will now introduce the beam sampler which does not suffer from this slow mixing behavior by sampling the whole sequence $\boldsymbol{s}$ in one go.

4. The Beam Sampler

The forward-backward algorithm does not apply to the iHMM because the number of states, and hence the number of potential state trajectories, are infinite. The idea of beam sampling is to introduce auxiliary variables $\boldsymbol{u}$ such that conditioned on $\boldsymbol{u}$ the number of trajectories with positive probability is finite. Now dynamic programming can be used to compute the conditional probabilities of each of these trajectories and thus sample *whole* trajectories efficiently. These

[1] Recall that we ignored the ordering of states in $\boldsymbol{\beta}$. In this representation the K distinct states in $\boldsymbol{s}$ are labeled $1 \ldots K$ and $K+1$ denotes a new state.

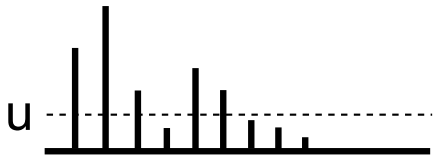


Figure 2. The auxiliary variable u partitions the probability distribution π (vertical bars) into a set of entries less than u and a set of entries larger than u.

auxiliary variables do not change the marginal distribution over other variables hence MCMC sampling will converge to the true posterior. This idea of using auxiliary variables to limit computation costs is inspired by (Walker, 2007), who applied it to limit the number of components in a DP mixture model that need be considered during sampling.

As opposed to the sampler in the previous section, the beam sampler does not marginalize out $\boldsymbol{\pi}$ nor $\boldsymbol{\phi}$. Specifically, the beam sampler iteratively samples the auxiliary variables $\boldsymbol{u}$, the trajectory $\boldsymbol{s}$, the transition probabilities $\boldsymbol{\pi}$, the shared DP parameters $\boldsymbol{\beta}$ and the hyperparameters α and γ conditioned on all other variables. In the following, we shall describe in more detail how to sample each set of variables, as well as how the auxiliary variables allow dynamic programming to be carried out over a finite number of trajectories without approximations.

Sampling $\boldsymbol{u}$: for each t we introduce an auxiliary variable u_t with conditional distribution $u_t \sim \mathsf{Uniform}(0, \pi_{s_{t-1}s_t})$ depending on $\boldsymbol{\pi}$, s_{t-1} and s_t.

Sampling $\boldsymbol{s}$: we sample the whole trajectory $\boldsymbol{s}$ given the auxiliary variables $\boldsymbol{u}$ and other variables using a form of forward filtering-backward sampling. The important observation here is that only trajectories $\boldsymbol{s}$ with $\pi_{s_{t-1}s_t} \geq u_t$ for all t will have non-zero probability given $\boldsymbol{u}$. There are only finitely many such trajectories[2] and as a result we can compute the conditional distribution over all such trajectories efficiently using dynamic programming.

First note that the probability density for u_t is $p(u_t|s_{t-1}, s_t, \boldsymbol{\pi}) = \frac{\mathbb{I}(0<u_t<\pi_{s_{t-1},s_t})}{\pi_{s_{t-1},s_t}}$, where $\mathbb{I}(C) = 1$ if condition C is true and 0 otherwise. We compute $p(s_t|y_{1:t}, u_{1:t})$ for all t as follows (we omitted the additional conditioning variables $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ for clarity):

$$\begin{aligned}
&p(s_t|y_{1:t}, u_{1:t}) \\
\propto &p(s_t, u_t, y_t|y_{1:t-1}, u_{1:t-1}), \\
= &\sum_{s_{t-1}} p(y_t|s_t)p(u_t|s_t, s_{t-1})p(s_t|s_{t-1}) \\
&\qquad p(s_{t-1}|y_{1:t-1}, u_{1:t-1}), \\
= &p(y_t|s_t) \sum_{s_{t-1}} \mathbb{I}(u_t < \pi_{s_{t-1},s_t}) p(s_{t-1}|y_{1:t-1}, u_{1:t-1}), \\
= &p(y_t|s_t) \sum_{s_{t-1}:u_t<\pi_{s_{t-1},s_t}} p(s_{t-1}|y_{1:t-1}, u_{1:t-1}). \qquad (4)
\end{aligned}$$

Note that we only need to compute (4) for the finitely many s_t values belonging to some trajectory with positive probability. Further, although the sum over s_{t-1} is technically a sum over an infinite number of terms, the auxiliary variable u_t truncates this summation to the finitely many s_{t-1}'s that satisfy both constraints $\pi_{s_{t-1},s_t} > u_t$ and $p(s_{t-1}|y_{1:t-1}, u_{1:t-1}) > 0$. Finally, to sample the whole trajectory $\boldsymbol{s}$, we sample s_T from $p(s_T|y_{1:T}, u_{1:T})$ and perform a backward pass where we sample s_t given the sample for s_{t+1}: $p(s_t|s_{t+1}, y_{1:T}, u_{1:T}) \propto p(s_t|y_{1:t}, u_{1:t})p(s_{t+1}|s_t, u_{t+1})$.

Sampling $\boldsymbol{\pi}$, $\boldsymbol{\phi}$, $\boldsymbol{\beta}$: these follow directly from the theory of HDPs (Teh et al., 2006), but we briefly describe these for completeness.

Let n_{ij} be the number of times state i transitions to state j in the trajectory $\boldsymbol{s}$, where $i, j \in \{1 \dots K\}$, K is the number of distinct states in $\boldsymbol{s}$, and these states have been relabeled $1 \dots K$. Merging the infinitely many states not represented in $\boldsymbol{s}$ into one state, the conditional distribution of $(\pi_{k1} \dots \pi_{kK}, \sum_{k'=K+1}^{\infty} \pi_{kk'})$ given its Markov blanket $\boldsymbol{s}, \boldsymbol{\beta}, \alpha$ is

$$\mathsf{Dirichlet}\left(n_{k1} + \alpha\beta_1 \dots n_{kK} + \alpha\beta_K, \alpha \textstyle\sum_{i=K+1}^{\infty} \beta_i\right),$$

To sample $\boldsymbol{\beta}$ we introduce a further set of auxiliary variables m_{ij} which are independent with conditional distributions

$$p(m_{ij} = m|\boldsymbol{s}, \boldsymbol{\beta}, \alpha) \propto S(n_{ij}, m)(\alpha\beta_j)^m,$$

where $S(\cdot, \cdot)$ denotes Stirling numbers of the first kind. The shared DP parameter $(\beta_1 \dots, \beta_K, \sum_{k'=K+1}^{\infty} \beta_{k'})$ has conditional distribution

$$\mathsf{Dirichlet}\left(m_{\cdot 1} \dots m_{\cdot K}, \gamma\right),$$

where $m_{\cdot k} = \sum_{k'=1}^{K} m_{k'k}$. (Teh et al., 2006; Antoniak, 1974) gives more details.

Finally, each ϕ_k is independent of others conditional on $\boldsymbol{s}$, $\boldsymbol{y}$ and their prior distribution H, i.e. $p(\boldsymbol{\phi}|\boldsymbol{s}, \boldsymbol{y}, H) =$

[2] To see this, note that $u_t > 0$ with probability 1 for each t, since each $\pi_{kk'} > 0$ with probability 1. Given the auxiliary variable u_t, note further that for each possible value of s_{t-1}, u_t partitions the set of transition probabilities out of state s_{t-1} into two sets: a finite set with $\pi_{s_{t-1}k} > u_t$ and an infinite set with $\pi_{s_{t-1}k} < u_t$, as illustrated in figure 2. Thus we can recursively show that for $t = 1, 2 \dots T$ the set of trajectories $s_{1:t}$ with all $\pi_{s_{t'-1}s_{t'}} > u_t$ is finite.

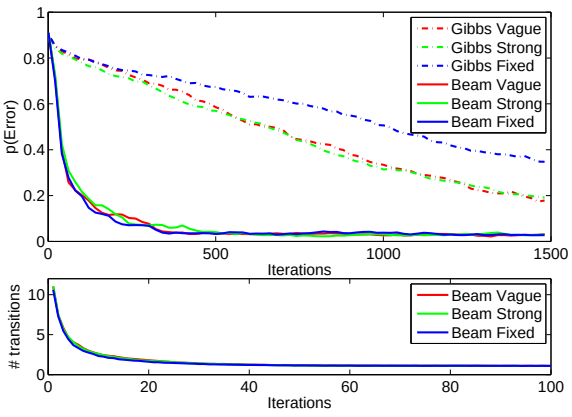


Figure 3. iHMM performance on strong negatively correlated data. The top plot shows the error of the Gibbs and beam sampler for the first 1500 iterations averaged over 20 runs. The bottom plot shows the average number of previous states considered in equation (4) for the first 100 iterations of the beam sampler.

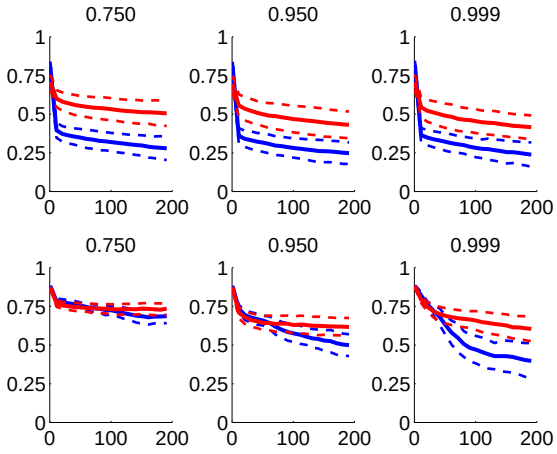


Figure 4. iHMM error on increasing positively correlated data. The blue curve shows the beam sampler while the red curve shows the Gibbs sampler performance. The dotted line show the one standard deviation error bars.

$\prod_k p(\phi_k|\boldsymbol{s},\boldsymbol{y},H)$. When the base distribution H is conjugate to the data distribution F each ϕ_k can be sampled efficiently. Otherwise we may resort to Metropolis-Hastings or other approaches. Note that in the non-conjugate case this is simpler than for Gibbs sampling. In the experimental section, we describe an application where the base distribution and likelihood are non-conjugate.

To conclude our discussion of the beam sampler, it is useful to point out that there is nothing special about sampling u_t from the uniform distribution on $[0, \pi_{s_{t-1},s_t}]$: by choosing a distribution over $[0, \pi_{s_t,s_{t-1}}]$ with higher mass near smaller values of u_t, we will allow more trajectories to have positive probability and hence considered by the forward filtering-backward sampling algorithm. Although this will typically improve mixing time, it also comes at additional computational cost. This brings us to the issue of the computational cost of the beam sampler: since for each timestep and each state assignment we need to sum over all represented previous states, the worst case complexity is $\mathcal{O}(TK^2)$. However, the sum in (4) is only over previous states for which the transition probability is larger than u_t; this means that in practice we might only need to sum over a few previous states. In our experiments below, we will give some empirical evidence for this "average case" behavior. Further, we have found that the drastically improved mixing of the beam sampler more than made up for the additional cost over Gibbs sampling. Finally, although we did not find any advantage doing so, it is certainly possible to interleave the beam sampler and the Gibbs sampler.

5. Experiments

We evaluate the beam sampler on two artificial and two real datasets to illustrate the following properties: (1) the beam sampler mixes in much fewer iterations than the Gibbs sampler; (2) the actual complexity per iteration of the beam sampler is only marginally more than the Gibbs sampler; (3) the beam sampler mixes well regardless of strong correlations in the data; (4) the beam sampler is more robust with respect to varying initialization and prior distribution; (5) the beam sampler handles non conjugate models naturally; (6) the iHMM is a viable alternative to the finite HMM. All datasets and a Matlab version of our software are available at *http://mlg.eng.cam.ac.uk/jurgen*.

5.1. Artificial Data

Our first experiment compares the performance of the iHMM on a sequence of length 800 generated by a 4 state HMM. The hidden state sequence was almost cyclic (1-2-3-4-1-2-3-...) with a 1% probability of self transition: i.o.w the true distribution of hidden states is strong negatively correlated. We use a multinomial output distribution with the following emission matrix

$$\begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.6666 & 0.1666 & 0.1666 \\ 0.5 & 0.0 & 0.5 \\ 0.3333 & 0.3333 & 0.3333 \end{bmatrix}.$$

Next we run the Gibbs and beam sampler 20 times from a random initialization with every state randomly chosen between 1 and 20. We test the performance of both samplers using three different hyperparameter settings: (1) vague gamma hyperpriors for α and

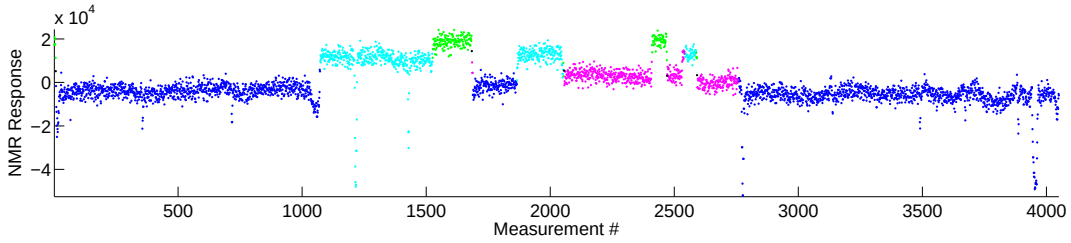


Figure 5. The 40'th sample of the beam sampler with every state represented by a different color on the well-log dataset.

γ (Gamma$(1,1)$ and Gamma$(2,1)$ respectively); (2) strong gamma hyperpriors for α and γ (Gamma$(6,15)$ and Gamma$(16,4)$ respectively); (3) fixed hyperparameters $\alpha = 0.4, \gamma = 3.8$. The latter were chosen using the values the beam and Gibbs samplers converged to. At every iteration, we greedily compute an assignment of sample states to true states to maximize overlap and use the resulting Hamming distance as our error measure. The top plot in figure 3 clearly shows that the beam sampler discovers the underlying structure much faster than the Gibbs sampler. Also, the beam sampler is insensitive to the prior while the performance of the Gibbs sampler becomes worse as we strengthen our prior beliefs. The bottom plot of figure 3 shows how many states are summed over in equation (4) averaged per timestep, per state. We find that after only about 20 iterations, the beam sampler on average considers a little more than one state. This implies that the actual complexity of the beam sampler is closer to $\mathcal{O}(TK)$ rather than the worst case complexity of $\mathcal{O}(TK^2)$. Although this behavior is dependent on the choice of distribution for the auxiliary variable u_t and the sparsity of the transition matrix, we have verified that this behavior is consistent also for larger iHMM's.

Our second experiment illustrates the performance of the beam sampler on data generated from HMM's with increasing positive correlation between the hidden states. We generated sequences of length 4000 from a 4 state HMM with self-transition probabilities increasing from 0.75 to 0.95 and finally 0.999. In one experiment (top plot of figure 4) we generated normal distributed observation from an informative output model with means $-2.0, 4.0, 1.0, -0.5$ and standard deviation 0.5, in another experiment (bottom plot of figure 4) we generated normal distributed observations from a less informative output model with means $-1.0, 0.5, -0.5, 0.0$ and standard deviation 0.5. We initialize the experiment as above and set the base distribution for the state means to be a 0 mean normal with 2.0 standard deviation. Then, we greedily compute the error compared to ground truth and average the results over 60 different random starting positions. The top row shows that with an informative prior, both the Gibbs and beam sampler can reduce the initial error by at least 50% independent of the correlation between hidden states. When the output model is less informative however and there is little correlation between the hidden states, the learning problem is hardest: the lower left plot shows that both the beam and Gibbs sampler discover structure only slowly. When the correlation increases, the learning problem should become easier. However, as the lower right plot shows, although the beam sampler mixes increasingly well, the Gibbs sampler suffers from slow random walk behavior.

5.2. Well Data

The next experiment illustrates the performance of the iHMM on a changepoint detection problem. The data consists of 4050 noisy measurements of nuclear-response of rock strata obtained via lowering a probe through a bore-hole. Figure 5 illustrates this datasets. The data has been previously analyzed in (Ruanaidh & Fitzgerald, 1996) by eliminating the forty greatest outliers and running a changepoint detection algorithm with a fixed number of changepoints. This approach works well as this one-dimensional dataset can be inspected visually to make a decision on whether to throw away datapoints and get a rough idea for the number of changepoints. However, we believe that with a nonparametric model, we can automatically adapt the number of changepoints. Moreover, by setting up a noise model with fat tails, we hope to automatically handle the outlier problem.

We model the mean of the nuclear-response for every segment. First we normalize the data to have zero mean; then we specify a zero mean normal distribution for the base distribution H. We choose the variance of this normal to be the empirical variance of the dataset. For the output model, we let F correspond to a Student-t distribution with $\nu = 1$, also known as the Cauchy distribution. We set the scale parameter for the Cauchy distribution to twice the empirical standard deviation for the dataset. Since the Cauchy likelihood is not conjugate with respect to the normal base distribution, we modified the Gibbs sampler based on algorithm 8 in (Neal, 2000). We use the aux-

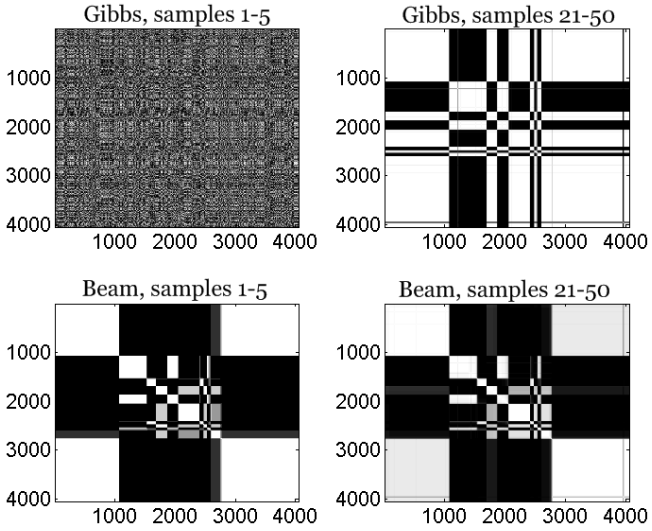


Figure 6. The left plots show how frequent two datapoints were in the same cluster averaged over the first 5 samples. The right plots show how frequently two datapoints were in the same cluster averaged over the last 30 samples.

iliary variable sampling scheme discussed in (Gelman et al., 2004) to resample the segment means.

Figure 5 shows the results of one sample from the beam sampler: the iHMM segments the dataset reasonably well and robustly handles the outliers. To compare the Gibbs and beam samplers, we compute 50 samples after a burnin of 5000 iterations with 1000 iterations in between each sample. For every pair of datapoints we compute the probability that they are in the same segment, averaged over the first five samples (left plots in figure 6) and the last thirty samples (right plots in figure 6). First, note that after the first 10000 iterations, the Gibbs sampler hasn't discovered any structure while the beam sampler has. This supports our claim that the beam sampler mixes faster than the Gibbs sampler. Moreover, we expect that the Gibbs sampler will have trouble to reassign the state assignment for whole segments because of slow random walk behavior. The beam sampler on the other hand resamples whole hidden state sequences and should be able to reassign whole segments more easily. The right plots of figure 6 confirm our expectation: a careful inspection of both plots shows that the Gibbs sampler is visually more black-white indicating that either two datapoints are always in the same cluster or never in the same cluster; the beam sampler, on the other hand, has gray areas which indicate that it averages over different assignments of the segments: e.g. the Gibbs plot (upper right) suggests that the leftmost segment and rightmost segment are *always* in the same state, while the beam sampler plot (bottom right) indicates that only part of the time, the left and rightmost segments are in the same state (90% of the time).

5.3. Alice in Wonderland

Another application domain for HMMs is the area of text prediction. One such task is that of predicting sequences of letters in text taken from *Alice's Adventures in Wonderland*. We compare the performance of a finite HMM trained using variational Bayes (as described in (MacKay, 1997)) with two iHMMs trained using beam sampling and Gibbs sampling. Both samplers had a burn-in of 1000 iterations and an additional 10000 iterations to collect 50 samples of hidden state sequences from the posterior (i.e. we sample every 200 iterations).

The training data for each HMM (whether finite or infinite) was taken to be a single sequence of 1000 characters from the first chapter of the book. There were 31 different observation symbols (26 letters ignoring case plus space and basic punctuation characters). The test data was taken to be the subsequent 4000 characters from the same chapter. For all finite HMMs we analyzed performance on models with the number of hidden states ranging from 1 to 50. For VB, we note that the true predictive distribution is intractable to compute. Therefore, we used the posterior parameter distributions to sample 50 candidate parameter settings, and used these to compute an approximate predictive log-likelihood. For the iHMMs, we sampled 50 hidden state sequences from the stationary distribution after convergence and used these samples to compute an approximate predictive log-likelihood. For the VB-HMM we set the prior pseudo-counts for the transition matrix to $4/K$ across all states and the prior pseudo-counts for the emission matrix to 0.3 across all symbols. Accordingly, we set the hyperprior for the iHMMs such that $a_\alpha = 4$ and $b_\alpha = 1$ and $H \sim \text{Dirichlet}\,(()\,0.3, \cdots 0.3)$. The results for VB and the iHMMs were averaged over 50 and 20 independent

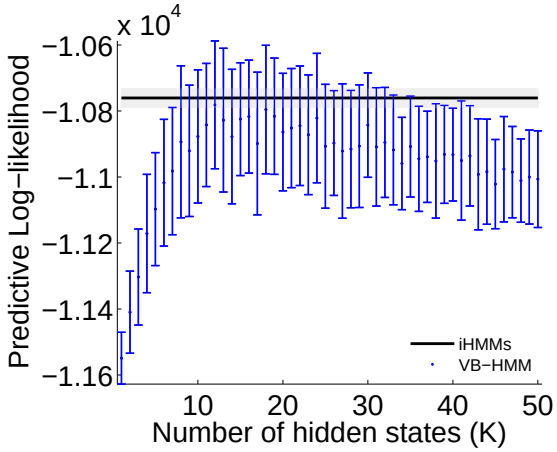


Figure 7. Comparing VB-HMM with the iHMM.

runs respectively. The plot includes error bars corresponding to 2 standard deviations.

Figure 7 illustrates the estimated predictive log-likelihoods for the finite VB-HMM and the two iHMMs trained using beam and Gibbs sampling. We find that the iHMMs have superior predictive power when compared to the VB-HMM, even when we select the best number of hidden states (around $K = 16$). Both the iHMMs converged to a posterior distribution over hidden state sequences with around 16 states, showing that nonparametric Bayesian techniques are an effective way to handle model selection. The final performance of the Gibbs and beam sampler were not found to be significantly different as we set the number of iterations high enough to ensure that both algorithms converge. Indeed, the aim of this experiment is not to compare the performance of individuals iHMM sampling schemes, rather, it is to further illustrate the relative effectiveness of using models of infinite capacity.

6. Conclusion

In this paper we introduced the beam sampler, a new inference algorithm for the iHMM that draws inspiration from slice sampling and dynamic programming to sample whole hidden state trajectories efficiently. We showed that the beam sampler is a more robust sampling algorithm than the Gibbs sampler. We believe that the beam sampler is the algorithm of choice for iHMM inference because it converges faster than the Gibbs sampler and is straightforward to implement. Moreover, it conveniently allows us to learn non-conjugate models. To encourage adoption of the iHMM as an alternative to HMM learning, we have made the software and datasets used in this paper available at *http://mlg.eng.cam.ac.uk/jurgen*.

The beam sampler idea is flexible enough to do inference for various extensions of the iHMM: our current work involves an adaptation of the beam sampler to an extension of the iHMM that handles inputs, effectively resulting in a nonparametric generalization of the input-output HMM (Bengio & Frasconi, 1995). We believe this is a promising model for nonparametric Bayesian learning of POMDPs. Another project currently underway is to use the beam sampler for efficiently learning finite, but very large hidden Markov models. Finally, we are exploring the possibilities of using the embedded HMM construction (Neal et al., 2004) as an alternative for the beam sampler for efficient inference in the iHMM.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. JVG is supported by a Microsoft Research PhD scholarship; ZG is also in the Machine Learning Department, CMU.

References

Antoniak, C. E. (1974). Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The Annals of Statistics*, *2*, 1152–1174.

Beal, M. J., Ghahramani, Z., & Rasmussen, C. E. (2002). The infinite hidden markov model. *NIPS*, *14*.

Bengio, Y., & Frasconi, P. (1995). An input output hmm architecture. *NIPS*, *7*.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*, 1–38.

Escobar, M. D., & West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, *90*, 577–588.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis*. CRC Press. 2rev ed edition.

Griffiths, T. L., & Ghahramani, Z. (2006). Infinite latent feature models and the indian buffet process. *NIPS*, *18*.

MacKay, D. J. C. (1997). *Ensemble learning for hidden markov models*. Technical report, Cavendish Laboratory, University of Cambridge, 1997.

Neal, R. M. (2000). Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, *9*, 249–265.

Neal, R. M. (2003). Slice sampling. *The Annals of Statistics*, *31*, 705–741.

Neal, R. M., Beal, M. J., & Roweis, S. T. (2004). Inferring state sequences for non-linear systems with embedded hidden markov models. *NIPS*, *16*.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications inspeech recognition. *Proceedings of the IEEE*, *77*, 257–286.

Rasmussen, C. E. (2000). The infinite gaussian mixture model. *NIPS*, *12*.

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian processes for machine learning*. The MIT Press.

Ruanaidh, J., & Fitzgerald, W. J. (1996). *Numerical bayesian methods applied to signal processing*. Springer-Verlag New York Inc.

Scott, S. L. (2002). Bayesian methods for hidden Markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, *97*, 337–351.

Sethuraman, J. (1994). A constructive definition of dirichlet priors. *Statistica Sinica*, *4*, 639–650.

Teh, Y. W., III, H. D., & Roy, D. (2008). Bayesian agglomerative clustering with coalescents. *NIPS*, *20*.

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical dirichlet processes. *Journal of the American Statistical Association*, *101*, 1566–1581.

Walker, S. G. (2007). Sampling the dirichlet mixture model with slices. *Communications in Statistics - Simulation and Computation*, *36*, 45.

Extracting and Composing Robust Features with Denoising Autoencoders

Pascal Vincent VINCENTP@IRO.UMONTREAL.CA
Hugo Larochelle LAROCHEH@IRO.UMONTREAL.CA
Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Pierre-Antoine Manzagol MANZAGOP@IRO.UMONTREAL.CA
Université de Montréal, Dept. IRO, CP 6128, Succ. Centre-Ville, Montral, Qubec, H3C 3J7, Canada

Abstract

Previous work has shown that the difficulties in learning deep generative or discriminative models can be overcome by an initial unsupervised learning step that maps inputs to useful intermediate representations. We introduce and motivate a new training principle for unsupervised learning of a representation based on the idea of making the learned representations robust to partial corruption of the input pattern. This approach can be used to train autoencoders, and these denoising autoencoders can be stacked to initialize deep architectures. The algorithm can be motivated from a manifold learning and information theoretic perspective or from a generative model perspective. Comparative experiments clearly show the surprising advantage of corrupting the input of autoencoders on a pattern classification benchmark suite.

1. Introduction

Recent theoretical studies indicate that deep architectures (Bengio & Le Cun, 2007; Bengio, 2007) may be needed to *efficiently* model complex distributions and achieve better generalization performance on challenging recognition tasks. The belief that additional levels of functional composition will yield increased representational and modeling power is not new (McClelland et al., 1986; Hinton, 1989; Utgoff & Stracuzzi, 2002). However, in practice, learning in deep architectures has proven to be difficult. One needs only to ponder the difficult problem of inference in deep directed graphical models, due to "explaining away". Also looking back at the history of multi-layer neural networks, their difficult optimization (Bengio et al., 2007; Bengio, 2007) has long prevented reaping the expected benefits of going beyond one or two hidden layers. However this situation has recently changed with the successful approach of (Hinton et al., 2006; Hinton & Salakhutdinov, 2006; Bengio et al., 2007; Ranzato et al., 2007; Lee et al., 2008) for training Deep Belief Networks and stacked autoencoders.

One key ingredient to this success appears to be the use of an unsupervised training criterion to perform a layer-by-layer initialization: each layer is at first trained to produce a higher level (hidden) representation of the observed patterns, based on the representation it receives as input from the layer below, by optimizing a local unsupervised criterion. Each level produces a representation of the input pattern that is more abstract than the previous level's, because it is obtained by composing more operations. This initialization yields a starting point, from which a global fine-tuning of the model's parameters is then performed using another training criterion appropriate for the task at hand. This technique has been shown empirically to avoid getting stuck in the kind of poor solutions one typically reaches with random initializations. While unsupervised learning of a mapping that produces "good" intermediate representations of the input pattern seems to be key, little is understood regarding what constitutes "good" representations for initializing deep architectures, or what explicit criteria may guide learning such representations. We know of only a few algorithms that seem to work well for this purpose: Restricted Boltzmann Machines (RBMs) trained with contrastive divergence on one hand, and various types of autoencoders on the other.

The present research begins with the question of what

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

explicit criteria a good intermediate representation should satisfy. Obviously, it should at a minimum retain a certain amount of "information" about its input, while at the same time being constrained to a given form (e.g. a real-valued vector of a given size in the case of an autoencoder). A supplemental criterion that has been proposed for such models is sparsity of the representation (Ranzato et al., 2008; Lee et al., 2008). Here we hypothesize and investigate an additional specific criterion: **robustness to partial destruction of the input**, i.e., partially destroyed inputs should yield almost the same representation. It is motivated by the following informal reasoning: a good representation is expected to capture stable structures in the form of dependencies and regularities characteristic of the (unknown) distribution of its observed input. For high dimensional redundant input (such as images) at least, such structures are likely to depend on evidence gathered from a combination of many input dimensions. They should thus be recoverable from partial observation only. A hallmark of this is our human ability to recognize partially occluded or corrupted images. Further evidence is our ability to form a high level concept associated to multiple modalities (such as image and sound) and recall it even when some of the modalities are missing.

To validate our hypothesis and assess its usefulness as one of the guiding principles in learning deep architectures, we propose a modification to the autoencoder framework to explicitly integrate robustness to partially destroyed inputs. Section 2 describes the algorithm in details. Section 3 discusses links with other approaches in the literature. Section 4 is devoted to a closer inspection of the model from different theoretical standpoints. In section 5 we verify empirically if the algorithm leads to a difference in performance. Section 6 concludes the study.

2. Description of the Algorithm

2.1. Notation and Setup

Let X and Y be two random variables with joint probability density $p(X,Y)$, with marginal distributions $p(X)$ and $p(Y)$. Throughout the text, we will use the following notation: Expectation: $\mathbb{E}_{p(X)}[f(X)] = \int p(\mathbf{x})f(\mathbf{x})\mathrm{d}\mathbf{x}$. Entropy: $\mathbb{H}(X) = \mathbb{H}(p) = \mathbb{E}_{p(X)}[-\log p(X)]$. Conditional entropy: $\mathbb{H}(X|Y) = \mathbb{E}_{p(X,Y)}[-\log p(X|Y)]$. Kullback-Leibler divergence: $\mathbb{D}_{\mathsf{KL}}(p\|q) = \mathbb{E}_{p(X)}[\log \frac{p(X)}{q(X)}]$. Cross-entropy: $\mathbb{H}(p\|q) = \mathbb{E}_{p(X)}[-\log q(X)] = \mathbb{H}(p) + \mathbb{D}_{\mathsf{KL}}(p\|q)$. Mutual information: $\mathbf{I}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y)$. Sigmoid: $s(x) = \frac{1}{1+e^{-x}}$ and $s(\mathbf{x}) = (s(\mathbf{x}_1), \ldots, s(\mathbf{x}_d))^T$. Bernoulli distribution with mean μ: $\mathcal{B}_\mu(x)$. and by extension $\mathcal{B}_\mu(\mathbf{x}) = (\mathcal{B}_{\mu_1}(\mathbf{x}_1), \ldots, \mathcal{B}_{\mu_d}(\mathbf{x}_d))$.

The setup we consider is the typical supervised learning setup with a training set of n (input, target) pairs $D_n = \{(\mathbf{x}^{(1)}, t^{(1)}) \ldots, (\mathbf{x}^{(n)}, t^{(n)})\}$, that we suppose to be an i.i.d. sample from an unknown distribution $q(X,T)$ with corresponding marginals $q(X)$ and $q(T)$.

2.2. The Basic Autoencoder

We begin by recalling the traditional autoencoder model such as the one used in (Bengio et al., 2007) to build deep networks. An autoencoder takes an input vector $\mathbf{x} \in [0,1]^d$, and first maps it to a hidden representation $\mathbf{y} \in [0,1]^{d'}$ through a deterministic mapping $\mathbf{y} = f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b})$, parameterized by $\theta = \{\mathbf{W}, \mathbf{b}\}$. $\mathbf{W}$ is a $d' \times d$ weight matrix and $\mathbf{b}$ is a bias vector. The resulting latent representation $\mathbf{y}$ is then mapped back to a "reconstructed" vector $\mathbf{z} \in [0,1]^d$ in input space $\mathbf{z} = g_{\theta'}(\mathbf{y}) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$ with $\theta' = \{\mathbf{W}', \mathbf{b}'\}$. The weight matrix $\mathbf{W}'$ of the reverse mapping may optionally be constrained by $\mathbf{W}' = \mathbf{W}^T$, in which case the autoencoder is said to have *tied weights*. Each training $\mathbf{x}^{(i)}$ is thus mapped to a corresponding $\mathbf{y}^{(i)}$ and a reconstruction $\mathbf{z}^{(i)}$. The parameters of this model are optimized to minimize the *average reconstruction error*:

$$\begin{aligned}\theta^\star, \theta'^\star = &\quad \arg\min_{\theta,\theta'} \frac{1}{n}\sum_{i=1}^{n} L\left(\mathbf{x}^{(i)}, \mathbf{z}^{(i)}\right) \\ = &\quad \arg\min_{\theta,\theta'} \frac{1}{n}\sum_{i=1}^{n} L\left(\mathbf{x}^{(i)}, g_{\theta'}(f_\theta(\mathbf{x}^{(i)}))\right) \quad (1)\end{aligned}$$

where L is a loss function such as the traditional *squared error* $L(\mathbf{x},\mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|^2$. An alternative loss, suggested by the interpretation of $\mathbf{x}$ and $\mathbf{z}$ as either bit vectors or vectors of bit probabilities (Bernoullis) is the *reconstruction cross-entropy*:

$$\begin{aligned}L_{\mathbf{H}}(\mathbf{x},\mathbf{z}) = &\quad \mathbb{H}(\mathcal{B}_{\mathbf{x}}\|\mathcal{B}_{\mathbf{z}}) \\ = &\quad -\sum_{k=1}^{d}[\mathbf{x}_k \log \mathbf{z}_k + (1-\mathbf{x}_k)\log(1-\mathbf{z}_k)] \quad (2)\end{aligned}$$

Note that if $\mathbf{x}$ is a binary vector, $L_{\mathbf{H}}(\mathbf{x},\mathbf{z})$ is a negative log-likelihood for the example $\mathbf{x}$, given the Bernoulli parameters $\mathbf{z}$. Equation 1 with $L = L_{\mathbf{H}}$ can be written

$$\theta^\star, \theta'^\star = \arg\min_{\theta,\theta'} \mathbb{E}_{q^0(X)}\left[L_{\mathbf{H}}\left(X, g_{\theta'}(f_\theta(X))\right)\right] \quad (3)$$

where $q^0(X)$ denotes the empirical distribution associated to our n training inputs. This optimization will typically be carried out by stochastic gradient descent.

2.3. The Denoising Autoencoder

To test our hypothesis and enforce robustness to partially destroyed inputs we modify the basic autoencoder we just described. We will now train it to reconstruct a clean "repaired" input from a *corrupted*, partially destroyed one. This is done by first corrupting the initial input $\mathbf{x}$ to get a partially destroyed version $\tilde{\mathbf{x}}$ by means of a stochastic mapping $\tilde{\mathbf{x}} \sim q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x})$. In our experiments, we considered the following corrupting process, parameterized by the desired proportion ν of "destruction": for each input $\mathbf{x}$, a fixed number νd of components are chosen at random, and their value is forced to 0, while the others are left untouched. All information about the chosen components is thus removed from that particuler input pattern, and the autoencoder will be trained to "fill-in" these artificially introduced "blanks". Note that alternative corrupting noises could be considered[1]. The corrupted input $\tilde{\mathbf{x}}$ is then mapped, as with the basic autoencoder, to a hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}}) = s(\mathbf{W}\tilde{\mathbf{x}}+\mathbf{b})$ from which we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y}) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$ (see figure 1 for a schematic representation of the process). As before the parameters are trained to minimize the average reconstruction error $L_{\mathsf{H}}(\mathbf{x}, \mathbf{z}) = \mathsf{H}(\mathcal{B}_{\mathbf{x}} \| \mathcal{B}_{\mathbf{z}})$ over a training set, i.e. to have $\mathbf{z}$ as close as possible to the uncorrupted input $\mathbf{x}$. But the key difference is that $\mathbf{z}$ is now a deterministic function of $\tilde{\mathbf{x}}$ rather than $\mathbf{x}$ and thus the result of a stochastic mapping of $\mathbf{x}$.

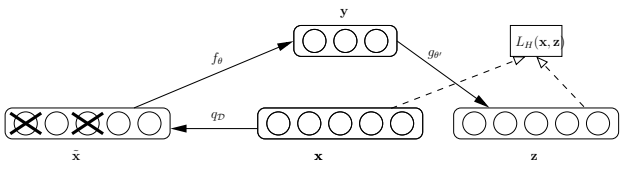


Figure 1. An example $\mathbf{x}$ is corrupted to $\tilde{\mathbf{x}}$. The autoencoder then maps it to $\mathbf{y}$ and attempts to reconstruct $\mathbf{x}$.

Let us define the joint distribution

$$q^0(X, \widetilde{X}, Y) = q^0(X) q_{\mathcal{D}}(\widetilde{X}|X) \delta_{f_\theta(\widetilde{X})}(Y) \qquad (4)$$

where $\delta_u(v)$ puts mass 0 when $u \neq v$. Thus Y is a deterministic function of $\widetilde{X}$. $q^0(X, \widetilde{X}, Y)$ is parameterized by θ. The objective function minimized by stochastic gradient descent becomes:

$$\arg\min_{\theta,\theta'} \mathbb{E}_{q^0(X,\widetilde{X})} \left[L_{\mathsf{H}} \left(X, g_{\theta'}(f_\theta(\widetilde{X})) \right) \right]. \qquad (5)$$

So from the point of view of the stochastic gradient descent algorithm, in addition to picking an input sample from the training set, we will also produce a random corrupted version of it, and take a gradient step towards reconstructing the uncorrupted version from the corrupted version. Note that in this way, the autoencoder cannot learn the identity, unlike the basic autoencoder, thus removing the constraint that $d' < d$ or the need to regularize specifically to avoid such a trivial solution.

[1]The approach we describe and our analysis is not specific to a particular kind of corrupting noise.

2.4. Layer-wise Initialization and Fine Tuning

The basic autoencoder has been used as a building block to train deep networks (Bengio et al., 2007), with the representation of the k-th layer used as input for the $(k+1)$-th, and the $(k+1)$-th layer trained after the k-th has been trained. After a few layers have been trained, the parameters are used as initialization for a network optimized with respect to a supervised training criterion. This greedy layer-wise procedure has been shown to yield significantly better local minima than random initialization of deep networks , achieving better generalization on a number of tasks (Larochelle et al., 2007).

The procedure to train a deep network using the denoising autoencoder is similar. The only difference is how each layer is trained, i.e., to minimize the criterion in eq. 5 instead of eq. 3. Note that the corruption process $q_{\mathcal{D}}$ is only used during training, but not for propagating representations from the raw input to higher-level representations. Note also that when layer k is trained, it receives as input the uncorrupted output of the previous layers.

3. Relationship to Other Approaches

Our training procedure for the denoising autoencoder involves learning to recover a clean input from a corrupted version, a task known as *denoising*. The problem of image denoising, in particular, has been extensively studied in the image processing community and many recent developments rely on machine learning approaches (see e.g. Roth and Black (2005); Elad and Aharon (2006); Hammond and Simoncelli (2007)). A particular form of gated autoencoders has also been used for denoising in Memisevic (2007). Denoising using autoencoders was actually introduced much earlier (LeCun, 1987; Gallinari et al., 1987), as an alternative to Hopfield models (Hopfield, 1982). Our objective however is fundamentally different from that of developing a competitive image denoising algorithm. We investigate explicit robustness to corrupting noise as a novel criterion guiding the learning of suitable intermediate representations to initialize a deep network. Thus our corruption+denoising procedure is applied not only on the input, but also recursively to intermediate representations.

The approach also bears some resemblance to the well known technique of augmenting the training data with stochastically "transformed" patterns. E.g. augmenting a training set by transforming original bitmaps through small rotations, translations, and scalings is known to improve final classification performance. In contrast to this technique our approach does not use any prior knowledge of image topology, nor does it produce extra labeled examples for supervised training. We use corrupted patterns in a generic (i.e. *not* specific to images) *unsupervised* initialization step, while the supervised training phase uses the unmodified original data.

There is a well known link between "training with noise" and regularization: they are equivalent for small additive noise (Bishop, 1995). By contrast, our corruption process is a large, non-additive, destruction of information. We train autoencoders to "fill in the blanks", not merely be smooth functions (regularization). Also in our experience, regularized autoencoders (i.e. with weight decay) do not yield the quantitative jump in performance and the striking qualitative difference observed in the filters that we get with denoising autoencoders.

There are also similarities with the work of (Doi et al., 2006) on robust coding over noisy channels. In their framework, a linear encoder is to encode a clean input for optimal transmission over a noisy channel to a decoder that reconstructs the input. This work was later extended to robustness to noise in the input, in a proposal for a model of retinal coding (Doi & Lewicki, 2007). Though some of the inspiration behind our work comes from neural coding and computation, our goal is not to account for experimental data of neuronal activity as in (Doi & Lewicki, 2007). Also, the non-linearity of our denoising autoencoder is crucial for its use in initializing a deep neural network.

It may be objected that, if our goal is to handle *missing values* correctly, we could have more naturally defined a proper latent variable generative model, and infer the posterior over the latent (hidden) representation in the presence of missing inputs. But this usually requires a costly marginalization[2] which has to be carried out for each new example. By contrast, our approach tries to learn a fast and robust deterministic mapping f_θ from examples of *already corrupted* inputs. The burden is on learning such a constrained mapping during training, rather than on unconstrained inference at use time. We expect this may force the model to capture implicit invariances in the data, and result in interesting features. Also note that in section 4.2 we will see how our learning algorithm for the denoising autoencoder can be viewed as a form of variational inference in a particular generative model.

[2] as for RBMs, where it is exponential in the number of missing values

4. Analysis of Denoising Autoencoders

The above intuitive motivation for the denoising autoencoder was given with the perspective of discovering robust representations. In the following, which can be skipped without hurting the remainder of the paper, we propose alternative perspectives on the algorithm.

4.1. Manifold Learning Perspective

The process of mapping a corrupted example to an uncorrupted one can be visualized in Figure 2, with a low-dimensional manifold near which the data concentrate. We learn a stochastic operator $p(X|\widetilde{X})$ that maps an $\widetilde{X}$ to an X, $p(X|\widetilde{X}) = \mathcal{B}_{g_{\theta'}(f_\theta(\widetilde{X}))}(X)$. The corrupted examples will be much more likely to be outside and farther from the manifold than the uncorrupted ones. Hence the stochastic operator $p(X|\widetilde{X})$ learns a map that tends to go from lower probability points $\widetilde{X}$ to high probability points X, generally on or near the manifold. Note that when $\widetilde{X}$ is farther from the manifold, $p(X|\widetilde{X})$ should learn to make bigger steps, to reach the manifold. At the limit we see that the operator should map even far away points to a small volume near the manifold.

The denoising autoencoder can thus be seen as a way to define and learn a manifold. The intermediate representation $Y = f(X)$ can be interpreted as a coordinate system for points on the manifold (this is most clear if we force the dimension of Y to be smaller than the dimension of X). More generally, one can think of $Y = f(X)$ as a representation of X which is well suited to capture the main variations in the data, i.e., on the manifold. When additional criteria (such as sparsity) are introduced in the learning model, one can no longer directly view $Y = f(X)$ as an explicit low-dimensional coordinate system for points on the manifold, but it retains the property of capturing the main factors of variation in the data.

4.2. Top-down, Generative Model Perspective

In this section we recover the training criterion for our denoising autoencoder (eq. 5) from a generative model perspective. Specifically we show that training the denoising autoencoder as described in section 2.3 is equivalent to maximizing a variational bound on a particular generative model.

Consider the generative model $p(X, \widetilde{X}, Y) = p(Y)p(X|Y)p(\widetilde{X}|X)$ where $p(X|Y) = \mathcal{B}_{s(\mathbf{W}'Y+\mathbf{b}')}$ and

$p(\widetilde{X}|X) = q_{\mathcal{D}}(\widetilde{X}|X)$. $p(Y)$ is a uniform prior over $Y \in [0,1]^{d'}$. This defines a generative model with parameter set $\theta' = \{\mathbf{W}', \mathbf{b}'\}$. We will use the previously defined $q^0(X, \widetilde{X}, Y) = q^0(X) q_{\mathcal{D}}(\widetilde{X}|X) \delta_{f_\theta(\widetilde{X})}(Y)$ (equation 4) as an auxiliary model in the context of a variational approximation of the log-likelihood of $p(\widetilde{X})$. Note that we abuse notation to make it lighter, and use the same letters X, $\widetilde{X}$ and Y for different sets of random variables representing the same quantity under different distributions: p or q^0. Keep in mind that whereas we had the dependency structure $X \to \widetilde{X} \to Y$ for q or q^0, we have $Y \to X \to \widetilde{X}$ for p.

Since p contains a corruption operation at the last generative stage, we propose to fit $p(\widetilde{X})$ to corrupted training samples. Performing maximum likelihood fitting for samples drawn from $q^0(\widetilde{X})$ corresponds to minimizing the cross-entropy, or maximizing

$$\begin{aligned}\mathcal{H} &= \max_{\theta'}\{-\mathbb{H}(q^0(\widetilde{X}) \| p(\widetilde{X}))\} \\ &= \max_{\theta'}\{\mathbb{E}_{q^0(\widetilde{X})}[\log p(\widetilde{X})]\}. \end{aligned} \tag{6}$$

Let $q^\star(X, Y|\widetilde{X})$ be a conditional density, the quantity $\mathcal{L}(q^\star, \widetilde{X}) = \mathbb{E}_{q^\star(X,Y|\widetilde{X})}\left[\log \frac{p(X,\widetilde{X},Y)}{q^\star(X,Y|\widetilde{X})}\right]$ is a lower bound on $\log p(\widetilde{X})$ since the following can be shown to be true for any $q^\star$:

$$\log p(\widetilde{X}) = \mathcal{L}(q^\star, \widetilde{X}) + \mathbb{D}_{\mathsf{KL}}(q^\star(X, Y|\widetilde{X}) \| p(X, Y|\widetilde{X}))$$

Also it is easy to verify that the bound is tight when $q^\star(X, Y|\widetilde{X}) = p(X, Y|\widetilde{X})$, where the $\mathbb{D}_{\mathsf{KL}}$ becomes 0. We can thus write $\log p(\widetilde{X}) = \max_{q^\star} \mathcal{L}(q^\star, \widetilde{X})$, and consequently rewrite equation 6 as

$$\begin{aligned}\mathcal{H} &= \max_{\theta'}\{\mathbb{E}_{q^0(\widetilde{X})}[\max_{q^\star} \mathcal{L}(q^\star, \widetilde{X})]\} \\ &= \max_{\theta', q^\star}\{\mathbb{E}_{q^0(\widetilde{X})}[\mathcal{L}(q^\star, \widetilde{X})]\} \end{aligned} \tag{7}$$

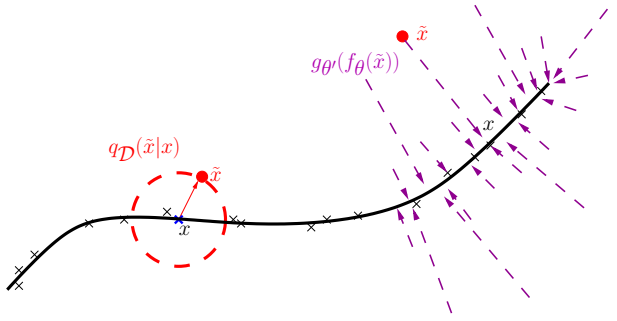


Figure 2. **Manifold learning perspective.** Suppose training data (×) concentrate near a low-dimensional manifold. Corrupted examples (•) obtained by applying corruption process $q_{\mathcal{D}}(\widetilde{X}|X)$ will lie farther from the manifold. The model learns with $p(X|\widetilde{X})$ to "project them back" onto the manifold. Intermediate representation Y can be interpreted as a coordinate system for points on the manifold.

where we moved the maximization outside of the expectation because an unconstrained $q^\star(X, Y|\widetilde{X})$ can in principle perfectly model the conditional distribution needed to maximize $\mathcal{L}(q^\star, \widetilde{X})$ for any $\widetilde{X}$. Now if we replace the maximization over an unconstrained $q^\star$ by the maximization over the parameters θ of our q^0 (appearing in f_θ that maps an $\mathbf{x}$ to a $\mathbf{y}$), we get a lower bound on $\mathcal{H}$: $\mathcal{H} \geq \max_{\theta',\theta}\{\mathbb{E}_{q^0(\widetilde{X})}[\mathcal{L}(q^0, \widetilde{X})]\}$ Maximizing this lower bound, we find

$$\begin{aligned}&\arg\max_{\theta,\theta'}\{\mathbb{E}_{q^0(\widetilde{X})}[\mathcal{L}(q^0, \widetilde{X})]\} \\ &= \arg\max_{\theta,\theta'} \mathbb{E}_{q^0(X,\widetilde{X},Y)}\left[\log \frac{p(X,\widetilde{X},Y)}{q^0(X,Y|\widetilde{X})}\right] \\ &= \arg\max_{\theta,\theta'} \mathbb{E}_{q^0(X,\widetilde{X},Y)}\Big[\log p(X,\widetilde{X},Y)\Big] \\ &\qquad + \mathbb{E}_{q^0(\widetilde{X})}\left[\mathbb{H}[q^0(X,Y|\widetilde{X})]\right] \\ &= \arg\max_{\theta,\theta'} \mathbb{E}_{q^0(X,\widetilde{X},Y)}\left[\log p(X,\widetilde{X},Y)\right]. \end{aligned}$$

Note that θ only occurs in $Y = f_\theta(\widetilde{X})$, and θ' only occurs in $p(X|Y)$. The last line is therefore obtained because $q^0(X|\widetilde{X}) \propto q_{\mathcal{D}}(\widetilde{X}|X) q^0(X)$ (none of which depends on (θ, θ')), and $q^0(Y|\widetilde{X})$ is deterministic, i.e., its entropy is constant, irrespective of (θ, θ'). Hence the entropy of $q^0(X, Y|\widetilde{X}) = q^0(Y|\widetilde{X}) q^0(X|\widetilde{X})$, does not vary with (θ, θ'). Finally, following from above, we obtain our training criterion (eq. 5):

$$\begin{aligned}&\arg\max_{\theta,\theta'} \mathbb{E}_{q^0(\widetilde{X})}[\mathcal{L}(q^0, \widetilde{X})] \\ &= \arg\max_{\theta,\theta'} \mathbb{E}_{q^0(X,\widetilde{X},Y)}[\log[p(Y)p(X|Y)p(\widetilde{X}|X)]] \\ &= \arg\max_{\theta,\theta'} \mathbb{E}_{q^0(X,\widetilde{X},Y)}[\log p(X|Y)] \\ &= \arg\max_{\theta,\theta'} \mathbb{E}_{q^0(X,\widetilde{X})}[\log p(X|Y = f_\theta(\widetilde{X}))] \\ &= \arg\min_{\theta,\theta'} \mathbb{E}_{q^0(X,\widetilde{X})}\left[L_{\mathbf{H}}\left(X, g_{\theta'}(f_\theta(\widetilde{X}))\right)\right] \end{aligned}$$

where the third line is obtained because (θ, θ') have no influence on $\mathbb{E}_{q^0(X,\widetilde{X},Y)}[\log p(Y)]$ because we chose $p(Y)$ uniform, i.e. constant, nor on $\mathbb{E}_{q^0(X,\widetilde{X})}[\log p(\widetilde{X}|X)]$, and the last line is obtained by inspection of the definition of $L_{\mathbf{H}}$ in eq. 2, when $p(X|Y = f_\theta(\widetilde{X}))$ is a $\mathcal{B}_{g_{\theta'}(f_\theta(\widetilde{X}))}$.

4.3. Other Theoretical Perspectives

Information Theoretic Perspective: Consider $X \sim q(X)$, q unknown, $Y = f_\theta(\widetilde{X})$. It can easily be shown (Vincent et al., 2008) that minimizing the expected reconstruction error amounts to *maximizing*

a lower bound on mutual information $\mathbf{I}(X;Y)$. Denoising autoencoders can thus be justified by the objective that Y captures as much information as possible about X *even as* Y *is a function of corrupted input.*

Stochastic Operator Perspective: Extending the manifold perspective, the denoising autoencoder can also be seen as corresponding to a semi-parametric model from which we can sample (Vincent et al., 2008):

$$p(X) = \frac{1}{n}\sum_{i=1}^{n}\sum_{\tilde{\mathbf{x}}} p(X|\widetilde{X}=\tilde{\mathbf{x}})q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x}_i),$$

where $\mathbf{x}_i$ is one of the n training examples.

5. Experiments

We performed experiments with the proposed algorithm on the same benchmark of classification problems used in (Larochelle et al., 2007)[3]. It contains different variations of the MNIST digit classification problem (input dimensionality $d = 28 \times 28 = 784$), with added factors of variation such as rotation (*rot*), addition of a background composed of random pixels (*bg-rand*) or made from patches extracted from a set of images (*bg-img*), or combinations of these factors (*rot-bg-img*). These variations render the problems particularly challenging for current generic learning algorithms. Each problem is divided into a training, validation, and test set (10000, 2000, 50000 examples respectively). A subset of the original MNIST problem is also included with the same example set sizes (problem *basic*). The benchmark also contains additional binary classification problems: discriminating between convex and non-convex shapes (*convex*), and between wide and long rectangles (*rect*, *rect-img*).

Neural networks with 3 hidden layers initialized by stacking *denoising autoencoders* (SdA-3), and fine tuned on the classification tasks, were evaluated on all the problems in this benchmark. Model selection was conducted following a similar procedure as Larochelle et al. (2007). Several values of hyper parameters (destruction fraction ν, layer sizes, number of unsupervised training epochs) were tried, combined with early stopping in the fine tuning phase. For each task, the best model was selected based on its classification performance on the validation set.

Table 1 reports the resulting classification error on the test set for the new model (SdA-3), together with the performance reported in Larochelle et al. (2007)[4] for SVMs with Gaussian and polynomial kernels, 1 and 3 hidden layers deep belief network (DBN-1 and DBN-3) and a 3 hidden layer deep network initialized by stacking basic autoencoders (SAA-3). Note that SAA-3 is equivalent to a SdA-3 with $\nu = 0\%$ destruction. As can be seen in the table, the corruption+denoising training works remarkably well as an initialization step, and in most cases yields significantly better classification performance than basic autoencoder stacking with no noise. On all but one task the SdA-3 algorithm performs on par or better than the best other algorithms, including deep belief nets. Due to space constraints, we do not report all selected hyper-parameters in the table (only showing ν). But it is worth mentioning that, for the majority of tasks, the model selection procedure chose best performing models with an *overcomplete first hidden layer representation* (typically of size 2000 for the 784-dimensional MNIST-derived tasks). This is very different from the traditional "bottleneck" autoencoders, and made possible by our denoising training procedure. All this suggests that the proposed procedure was indeed able to produce more useful feature detectors.

Next, we wanted to understand qualitatively the effect of the corruption+denoising training. To this end we display the filters obtained after initial training of the first denoising autoencoder on MNIST digits. Figure 3 shows a few of these filters as little image patches, for different noise levels. Each patch corresponds to a row of the learnt weight matrix $\mathbf{W}$, i.e. the incoming weights of one of the hidden layer neurons. The beneficial effect of the denoising training can clearly be seen. Without the denoising procedure, many filters appear to have learnt no interesting feature. They look like the filters obtained after random initialization. But when increasing the level of destructive corruption, an increasing number of filters resemble sensible feature detectors. As we move to higher noise levels, we observe a phenomenon that we expected: filters become less local, they appear sensitive to larger structures spread out across more input dimensions.

6. Conclusion and Future Work

We have introduced a very simple training principle for autoencoders, based on the objective of undoing a corruption process. This is motivated by the goal of learning representations of the input that are robust to small irrelevant changes in input. We also motivated it from a manifold learning perspective and gave an interpretation from a generative model perspective.

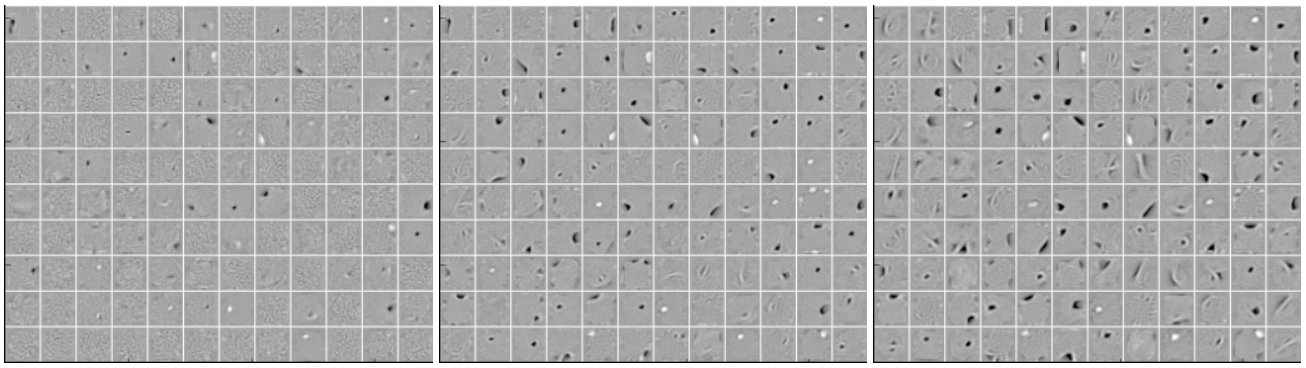
This principle can be used to train and stack autoencoders to initialize a deep neural network. A series

[3] All the datasets for these problems are available at http://www.iro.umontreal.ca/∼lisa/icml2007.

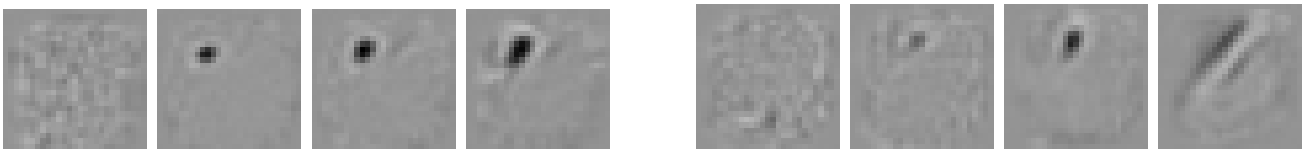
[4] Except that *rot* and *rot-bg-img*, as reported on the website from which they are available, have been regenerated since Larochelle et al. (2007), to fix a problem in the initial data generation process. We used the updated data and corresponding benchmark results given on this website.

Table 1. **Comparison of stacked denoising autoencoders (SdA-3) with other models.**
Test error rate on all considered classification problems is reported together with a 95% confidence interval. Best performer is in bold, as well as those for which confidence intervals overlap. SdA-3 appears to achieve performance superior or equivalent to the best other model on all problems except *bg-rand*. For SdA-3, we also indicate the fraction ν of destroyed input components, as chosen by proper model selection. Note that SAA-3 is equivalent to SdA-3 with $\nu = 0\%$.

Dataset	**SVM**$_{rbf}$	**SVM**$_{poly}$	**DBN-1**	**SAA-3**	**DBN-3**	**SdA-3** (ν)
basic	**3.03±0.15**	3.69±0.17	3.94±0.17	3.46±0.16	3.11±0.15	**2.80±0.14** (10%)
rot	11.11±0.28	15.42±0.32	14.69±0.31	**10.30±0.27**	**10.30±0.27**	**10.29±0.27** (10%)
bg-rand	14.58±0.31	16.62±0.33	9.80±0.26	11.28±0.28	**6.73±0.22**	10.38±0.27 (40%)
bg-img	22.61±0.37	24.01±0.37	**16.15±0.32**	23.00±0.37	**16.31±0.32**	**16.68±0.33** (25%)
rot-bg-img	55.18±0.44	56.41±0.43	52.21±0.44	51.93±0.44	47.39±0.44	**44.49±0.44** (25%)
rect	**2.15±0.13**	**2.15±0.13**	4.71±0.19	2.41±0.13	2.60±0.14	**1.99±0.12** (10%)
rect-img	24.04±0.37	24.05±0.37	23.69±0.37	24.05±0.37	22.50±0.37	**21.59±0.36** (25%)
convex	19.13±0.34	19.82±0.35	19.92±0.35	**18.41±0.34**	**18.63±0.34**	**19.06±0.34** (10%)



(a) No destroyed inputs (b) 25% destruction (c) 50% destruction



(d) Neuron A (0%, 10%, 20%, 50% destruction) (e) Neuron B (0%, 10%, 20%, 50% destruction)

Figure 3. **Filters obtained after training the first denoising autoencoder.**
(a-c) show some of the filters obtained after training a denoising autoencoder on MNIST samples, with increasing destruction levels ν. The filters at the same position in the three images are related only by the fact that the autoencoders were started from the same random initialization point.
(d) and **(e)** zoom in on the filters obtained for two of the neurons, again for increasing destruction levels.
As can be seen, with no noise, many filters remain similarly uninteresting (undistinctive almost uniform grey patches). As we increase the noise level, denoising training forces the filters to differentiate more, and capture more distinctive features. Higher noise levels tend to induce less local filters, as expected. One can distinguish different kinds of filters, from local blob detectors, to stroke detectors, and some full character detectors at the higher noise levels.

of image classification experiments were performed to evaluate this new training principle. The empirical results support the following conclusions: unsupervised initialization of layers with an explicit denoising criterion helps to capture interesting structure in the input distribution. This in turn leads to intermediate representations much better suited for subsequent learning tasks such as supervised classification. It is possible that the rather good experimental performance of Deep Belief Networks (whose layers are initialized as RBMs) is partly due to RBMs encapsulating a similar form of robustness to corruption in the representations they learn, possibly because of their stochastic nature which introduces noise in the representation during training. Future work inspired by this observation should investigate other types of corruption process, not only of the input but of the representation itself as well.

Acknowledgments

We thank the anonymous reviewers for their useful comments that helped improved the paper. We are also very grateful for financial support of this work by NSERC, MITACS and CIFAR.

References

Bengio, Y. (2007). *Learning deep architectures for AI* (Technical Report 1312). Université de Montréal, dept. IRO.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems 19* (pp. 153–160). MIT Press.

Bengio, Y., & Le Cun, Y. (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste and J. Weston (Eds.), *Large scale kernel machines*. MIT Press.

Bishop, C. M. (1995). Training with noise is equivalent to tikhonov regularization. *Neural Computation*, *7*, 108–116.

Doi, E., Balcan, D. C., & Lewicki, M. S. (2006). A theoretical analysis of robust coding over noisy overcomplete channels. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*, 307–314. Cambridge, MA: MIT Press.

Doi, E., & Lewicki, M. S. (2007). A theory of retinal population coding. *NIPS* (pp. 353–360). MIT Press.

Elad, M., & Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, *15*, 3736–3745.

Gallinari, P., LeCun, Y., Thiria, S., & Fogelman-Soulie, F. (1987). Memoires associatives distribuees. *Proceedings of COGNITIVA 87*. Paris, La Villette.

Hammond, D., & Simoncelli, E. (2007). A machine learning framework for adaptive combination of signal denoising methods. *2007 International Conference on Image Processing* (pp. VI: 29–32).

Hinton, G. (1989). Connectionist learning procedures. *Artificial Intelligence*, *40*, 185–234.

Hinton, G., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504–507.

Hinton, G. E., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, *79*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. *Proceedings of the 24th International Conference on Machine Learning (ICML'2007)* (pp. 473–480).

LeCun, Y. (1987). *Modèles connexionistes de l'apprentissage*. Doctoral dissertation, Université de Paris VI.

Lee, H., Ekanadham, C., & Ng, A. (2008). Sparse deep belief net model for visual area V2. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*. Cambridge, MA: MIT Press.

McClelland, J., Rumelhart, D., & the PDP Research Group (1986). *Parallel distributed processing: Explorations in the microstructure of cognition*, vol. 2. Cambridge: MIT Press.

Memisevic, R. (2007). *Non-linear latent factor models for revealing structure in high-dimensional data*. Doctoral dissertation, Departement of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Ranzato, M., Boureau, Y.-L., & LeCun, Y. (2008). Sparse feature learning for deep belief networks. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*. Cambridge, MA: MIT Press.

Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. *Advances in Neural Information Processing Systems (NIPS 2006)*. MIT Press.

Roth, S., & Black, M. (2005). Fields of experts: a framework for learning image priors. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 860–867).

Utgoff, P., & Stracuzzi, D. (2002). Many-layered learning. *Neural Computation*, *14*, 2497–2539.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). *Extracting and composing robust features with denoising autoencoders* (Technical Report 1316). Université de Montréal, dept. IRO.

Prediction with Expert Advice for the Brier Game

Vladimir Vovk VOVK@CS.RHUL.AC.UK
Fedor Zhdanov FEDOR@CS.RHUL.AC.UK
Computer Learning Research Centre, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK

Abstract

We show that the Brier game of prediction is mixable and find the optimal learning rate and substitution function for it. The resulting prediction algorithm is applied to predict results of football and tennis matches. The theoretical performance guarantee turns out to be rather tight on these data sets, especially in the case of the more extensive tennis data.

1. Introduction

The paradigm of prediction with expert advice was introduced in the late 1980s (see, e.g., Littlestone & Warmuth, 1994, Cesa-Bianchi et al., 1997) and has been applied to various loss functions; see Cesa-Bianchi and Lugosi (2006) for a recent book-length review. An especially important class of loss functions is that of "mixable" ones, for which the learner's loss can be made as small as the best expert's loss plus a constant (depending on the number of experts). It is known (Haussler et al., 1998; Vovk, 1998) that the optimal additive constant is attained by the "strong aggregating algorithm" proposed in Vovk (1990) (we use the adjective "strong" to distinguish it from the "weak aggregating algorithm" of Kalnishkan & Vyugin, 2005).

There are several important loss functions that have been shown to be mixable and for which the optimal additive constant has been found. The prime examples in the case of binary observations are the log loss function and the square loss function. The log loss function, whose mixability is obvious, has been explored extensively, along with its important generalizations, the Kullback–Leibler divergence and Cover's loss function.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

In this paper we concentrate on the square loss function. In the binary case, its mixability was demonstrated in Vovk (1990). There are two natural directions in which this result could be generalized:

Regression: observations are real numbers (square-loss regression is a standard problem in statistics).

Classification: observations take values in a finite set (this leads to the "Brier game", to be defined below, a standard way of measuring the quality of predictions in meteorology and other applied fields: see, e.g., Dawid, 1986).

The mixability of the square loss function in the case of observations belonging to a bounded interval of real numbers was demonstrated in Haussler et al. (1998); Haussler et al.'s algorithm was simplified in Vovk (2001). Surprisingly, the case of square-loss non-binary classification has never been analysed in the framework of prediction with expert advice. The purpose of this paper is to fill this gap. The full version (Vovk & Zhdanov, 2008) of this paper is available on arXiv.

2. Prediction Algorithm and Loss Bound

A game of prediction consists of three components: the observation space Ω, the decision space Γ, and the loss function $\lambda : \Omega \times \Gamma \to \mathbb{R}$. In this paper we are interested in the following *Brier game* (Brier, 1950): Ω is a finite and non-empty set, $\Gamma := \mathcal{P}(\Omega)$ is the set of all probability measures on Ω, and

$$\lambda(\omega, \gamma) = \sum_{o \in \Omega} \left(\gamma\{o\} - \delta_\omega\{o\}\right)^2,$$

where $\delta_\omega \in \mathcal{P}(\Omega)$ is the probability measure concentrated at ω: $\delta_\omega\{\omega\} = 1$ and $\delta_\omega\{o\} = 0$ for $o \neq \omega$. (For example, if $\Omega = \{1, 2, 3\}$, $\omega = 1$, $\gamma\{1\} = 1/2$, $\gamma\{2\} = 1/4$, and $\gamma\{3\} = 1/4$, $\lambda(\omega, \gamma) = (1/2 - 1)^2 + (1/4 - 0)^2 + (1/4 - 0)^2 = 3/8$.)

The game of prediction is being played repeatedly by a learner having access to decisions made by a pool of experts, which leads to the following prediction protocol:

Protocol 1 Prediction with expert advice

$L_0 := 0$.
$L_0^k := 0$, $k = 1, \dots, K$.
for $N = 1, 2, \dots$ **do**
  Expert k announces $\gamma_N^k \in \Gamma$, $k = 1, \dots, K$.
  Learner announces $\gamma_N \in \Gamma$.
  Reality announces $\omega_N \in \Omega$.
  $L_N := L_{N-1} + \lambda(\omega_N, \gamma_N)$.
  $L_N^k := L_{N-1}^k + \lambda(\omega_N, \gamma_N^k)$, $k = 1, \dots, K$.
end for

At each step of Protocol 1 Learner is given K experts' advice and is required to come up with his own decision; L_N is his cumulative loss over the first N steps, and L_N^k is the kth expert's cumulative loss over the first N steps. In the case of the Brier game, the decisions are probability forecasts for the next observation.

An optimal (in the sense of Theorem 1 below) strategy for Learner in prediction with expert advice for the Brier game is given by the strong aggregating algorithm. For each expert k, the algorithm maintains its weight w^k, constantly slashing the weights of less successful experts.

Algorithm 1 Strong aggregating algorithm for the Brier game

$w_0^k := 1$, $k = 1, \dots, K$.
for $N = 1, 2, \dots$ **do**
  Read the Experts' predictions γ_N^k, $k = 1, \dots, K$.
  Set $G_N(\omega) := -\ln \sum_{k=1}^K w_{N-1}^k e^{-\lambda(\omega, \gamma_N^k)}$, $\omega \in \Omega$.
  Solve $\sum_{\omega \in \Omega} (s - G_N(\omega))^+ = 2$ in $s \in \mathbb{R}$.
  Set $\gamma_N\{\omega\} := (s - G_N(\omega))^+/2$, $\omega \in \Omega$.
  Output prediction $\gamma_N \in \mathcal{P}(\Omega)$.
  Read observation ω_N.
  $w_N^k := w_{N-1}^k e^{-\lambda(\omega_N, \gamma_N^k)}$.
end for

The algorithm will be derived in Section 5. The following result (to be proved in Section 4) gives a performance guarantee for it that cannot be improved by any other prediction algorithm.

Theorem 1. *Using Algorithm 1 as Learner's strategy in Protocol 1 for the Brier game guarantees that*

$$L_N \le \min_{k=1,\dots,K} L_N^k + \ln K \tag{1}$$

for all $N = 1, 2, \dots$. If $A < \ln K$, Learner does not have a strategy guaranteeing

$$L_N \le \min_{k=1,\dots,K} L_N^k + A \tag{2}$$

for all $N = 1, 2, \dots$.

3. Experimental Results

In our first empirical study of Algorithm 1 we use historical data about 6416 matches in various English football league competitions, namely: the Premier League (the pinnacle of the English football system), the Football League Championship, Football League One, Football League Two, the Football Conference. Our data, provided by Football-Data, cover two full seasons, 2005/2006 and 2006/2007, and part of the 2007/2008 season (which ends in May shortly after the paper submission deadline). The matches are sorted first by date and then by league. In the terminology of our prediction protocol, the outcome of each match is the observation, taking one of three possible values, "home win", "draw", or "away win"; we will encode the possible values as 1, 2, and 3.

For each match we have forecasts made by a range of bookmakers. We chose eight bookmakers for which we have enough data over a long period of time, namely Bet365, Bet&Win, Gamebookers, Interwetten, Ladbrokes, Sportingbet, Stan James, and VC Bet. (And the seasons mentioned above were chosen because the forecasts of these bookmakers are available for them.)

A probability forecast for the next observation is essentially a vector (p_1, p_2, p_3) consisting of positive numbers summing to 1. The bookmakers do not announce these numbers directly; instead, they quote three betting odds, a_1, a_2, and a_3. Each number a_i is the amount which the bookmaker undertakes to pay out to a client betting on outcome i per unit stake in the event that i happens (the stake itself is never returned to the bettor, which makes all betting odds greater than 1; i.e., the odds are announced according to the "continental" rather than "traditional" system). The inverse value $1/a_i$, $i \in \{1, 2, 3\}$, can be interpreted as the bookmaker's quoted probability for the observation i. The bookmaker's quoted probabilities are usually slightly (because of the competition with other bookmakers) in his favour: the sum $1/a_1 + 1/a_2 + 1/a_3$ exceeds 1 by the amount called the *overround* (at most 0.15 in the vast majority of cases). We used

$$p_i := \frac{1/a_i}{1/a_1 + 1/a_2 + 1/a_3}, \quad i = 1, 2, 3, \tag{3}$$

as the bookmaker's forecasts; it is clear that $p_1 + p_2 + p_3 = 1$.

The results of applying Algorithm 1 to the football data, with 8 experts and 3 possible observations, are shown in Figure 1. Let L_N^k be the cumulative loss of Expert k, $k = 1, \dots, 8$, over the first N matches and L_N be the corresponding number for Algorithm 1 (i.e., we essentially continue to use the notation of Theorem 1). The dashed line corresponding to Expert k shows the excess loss $N \mapsto L_N^k - L_N$ of Expert k over Algorithm 1. The excess loss can be negative, but from Theorem 1 we know that it cannot be less than $-\ln 8$; this lower bound is also shown in Figure 1. Finally, the thick line (the positive part of the x axis) is drawn for comparison: this is the excess loss of Algorithm 1 over itself. We can see that at each moment in time the algorithm's cumulative loss is fairly close to the cumulative loss of the best expert (at that time; the best expert keeps changing over the time).

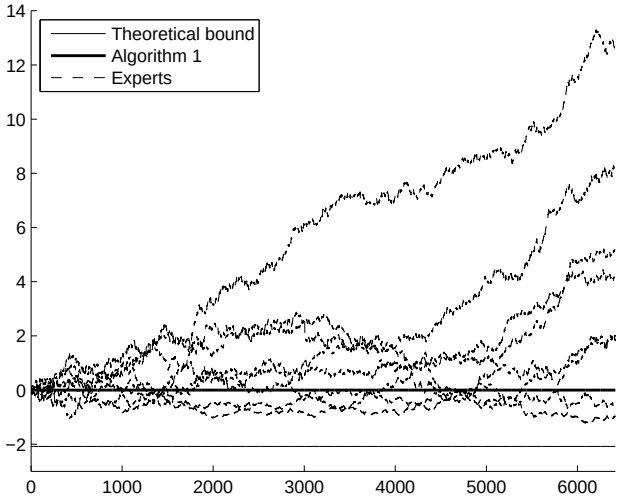


Figure 1. The difference between the cumulative loss of each of the 8 bookmakers (experts) and of Algorithm 1 on the football data. The theoretical lower bound $-\ln 8$ from Theorem 1 is also shown.

Figure 2 shows the results of another empirical study, involving data about a large number of tennis tournaments in 2004, 2005, 2006, and 2007, with the total number of matches 10,087. The tournaments include, e.g., Australian Open, French Open, Wimbledon, and US Open; the data is provided by Tennis-Data. The matches are sorted by date, then by tournament. The data contain information about the winner of each match and the betting odds of 4 bookmakers for his/her win and for the opponent's win. Therefore, now there are two possible observations (player 1's win and player 2's win). There are four bookmakers: Bet365, Centrebet, Expekt, and Pinnacle Sports. The results in Figure 2 are presented in the same way as in Figure 1. Typical values of the overround are below 0.1.

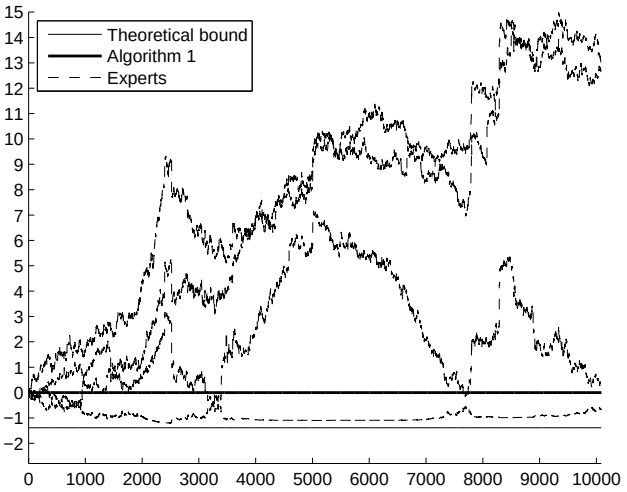


Figure 2. The difference between the cumulative loss of each of the 4 bookmakers and of Algorithm 1 on the tennis data. Now the theoretical bound is $-\ln 4$.

In both Figure 1 and Figure 2 the cumulative loss of Algorithm 1 is close to the cumulative loss of the best expert, despite the fact that some of the experts perform poorly. The theoretical bound is not hopelessly loose for the football data and is rather tight for the tennis data. The pictures look exactly the same when Algorithm 1 is applied in the more realistic manner where the weights w^k are not updated over the matches that are played simultaneously.

Our second empirical study (Figure 2) is about binary prediction, and so the algorithm of Vovk (1990) could have also been used (and would have given similar results). We included it since we are not aware of any empirical studies even for the binary case.

Other popular algorithms for prediction with expert advice that could be used instead of Algorithm 1 in our empirical studies are Kivinen and Warmuth's (1999) Weighted Average Algorithm (WAA) and Freund and Schapire's (1997) Hedge algorithm (HA). The performance guarantees for these two algorithms are much weaker than the optimal (1), especially in the case of the HA (even if the loss bound given in Freund & Schapire, 1997, is replaced by the stronger bound given in Vovk, 1998, Example 7). The weak performance guarantees show in the empirical performance of the algorithms. For the football data the maximal difference between the cumulative loss of both the WAA and the HA and the cumulative loss of the best

expert is about twice as large as that for Algorithm 1 (and so is approximately equal to the optimal bound $\ln K$ given by (1)). For the tennis data the maximal difference for the WAA is about three times as large as for Algorithm 1, and for the HA it is about twice as large; therefore, both algorithms violate the optimal bound $\ln K$. For further details, see Vovk and Zhdanov (2008).

The data used for producing Figures 1 and 2 can be downloaded from `http://vovk.net/ICML2008`.

4. Proof of Theorem 1

This proof will use some basic notions of elementary differential geometry, especially those connected with the Gauss–Kronecker curvature of surfaces. (The use of curvature in this kind of results is standard: see, e.g., Vovk, 1990, and Haussler et al., 1998.) All definitions that we will need can be found in, e.g., Thorpe, 1979.

A vector $f \in \mathbb{R}^\Omega$ (understood to be a function $f : \Omega \to \mathbb{R}$) is a *superprediction* if there is $\gamma \in \Gamma$ such that, for all $\omega \in \Omega$, $\lambda(\omega, \gamma) \le f(\omega)$; the set Σ of all superpredictions is the *superprediction set*. For each *learning rate* $\eta > 0$, let $\Phi_\eta : \mathbb{R}^\Omega \to (0,\infty)^\Omega$ be the homeomorphism defined by

$$\Phi_\eta(f) : \omega \in \Omega \mapsto e^{-\eta f(\omega)}, \quad f \in \mathbb{R}^\Omega.$$

The image $\Phi_\eta(\Sigma)$ of the superprediction set will be called the *η-exponential superprediction set*. It is known that

$$L_N \le \min_{k=1,\dots,K} L_N^k + \frac{\ln K}{\eta}$$

can be guaranteed if and only if the η-exponential superprediction set is convex (part "if" for all K and part "only if" for $K \to \infty$ are proved in Vovk, 1998; part "only if" for all K is proved by Chris Watkins, and the details can be found in, e.g., Vovk, 2007, Appendix). Comparing this with (1) and (2) we can see that we are required to prove that

- $\Phi_\eta(\Sigma)$ is convex when $\eta \le 1$;
- $\Phi_\eta(\Sigma)$ is not convex when $\eta > 1$.

Define the *η-exponential superprediction surface* to be the part of the boundary of the η-exponential superprediction set $\Phi_\eta(\Sigma)$ lying inside $(0,\infty)^\Omega$. The idea of the proof is to check that, for all $\eta < 1$, the Gauss–Kronecker curvature of this surface is nowhere vanishing. Even when this is done, however, there is still uncertainty as to in which direction the surface is bulging (towards the origin or away from it). The standard argument (as in Thorpe, 1979, Chapter 12, Theorem 6) based on the continuity of the smallest principal curvature shows that the η-exponential superprediction set is bulging away from the origin for small enough η: indeed, since it is true at some point, it is true everywhere on the surface. By the continuity in η this is also true for all $\eta < 1$. Now, since the η-exponential superprediction set is convex for all $\eta < 1$, it is also convex for $\eta = 1$.

Let us now check that the Gauss–Kronecker curvature of the η-exponential superprediction surface is always positive when $\eta < 1$ and is sometimes negative when $\eta > 1$ (the rest of the proof, an elaboration of the above argument, will be easy). Set $n := |\Omega|$; without loss of generality we assume $\Omega = \{1, \dots, n\}$.

A convenient parametric representation of the η-exponential superprediction surface is

$$\begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^{n-1} \\ x^n \end{pmatrix} = \begin{pmatrix} e^{-\eta((u^1-1)^2+(u^2)^2+\dots+(u^n)^2)} \\ e^{-\eta((u^1)^2+(u^2-1)^2+\dots+(u^n)^2)} \\ \vdots \\ e^{-\eta((u^1)^2+\dots+(u^{n-1}-1)^2+(u^n)^2)} \\ e^{-\eta((u^1)^2+\dots+(u^{n-1})^2+(u^n-1)^2)} \end{pmatrix}, \quad (4)$$

where $u^1, \dots, u^{n-1}$ are the coordinates on the surface, $u^1, \dots, u^{n-1} \in (0,1)$ subject to $u^1 + \dots u^{n-1} < 1$, and u^n is a shorthand for $1 - u^1 - \dots - u^{n-1}$. The derivative of (4) in u^1 is

$$\frac{\partial}{\partial u^1} \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^{n-1} \\ x^n \end{pmatrix} = 2\eta \times$$

$$\begin{pmatrix} (u^n - u^1 + 1)e^{-\eta((u^1-1)^2+(u^2)^2+\dots+(u^{n-1})^2+(u^n)^2)} \\ (u^n - u^1)e^{-\eta((u^1)^2+(u^2-1)^2+\dots+(u^{n-1})^2+(u^n)^2)} \\ \vdots \\ (u^n - u^1)e^{-\eta((u^1)^2+(u^2)^2+\dots+(u^{n-1}-1)^2+(u^n)^2)} \\ (u^n - u^1 - 1)e^{-\eta((u^1)^2+(u^2)^2+\dots+(u^{n-1})^2+(u^n-1)^2)} \end{pmatrix}$$

$$\propto \begin{pmatrix} (u^n - u^1 + 1)e^{2\eta u^1} \\ (u^n - u^1)e^{2\eta u^2} \\ \vdots \\ (u^n - u^1)e^{2\eta u^{n-1}} \\ (u^n - u^1 - 1)e^{2\eta u^n} \end{pmatrix},$$

the derivative in u^2 is

$$\frac{\partial}{\partial u^2}\begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^{n-1} \\ x^n \end{pmatrix} \propto \begin{pmatrix} (u^n-u^2)e^{2\eta u^1} \\ (u^n-u^2+1)e^{2\eta u^2} \\ \vdots \\ (u^n-u^2)e^{2\eta u^{n-1}} \\ (u^n-u^2-1)e^{2\eta u^n} \end{pmatrix},$$

and so on, up to

$$\frac{\partial}{\partial u^{n-1}}\begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^{n-1} \\ x^n \end{pmatrix} \propto \begin{pmatrix} (u^n-u^{n-1})e^{2\eta u^1} \\ (u^n-u^{n-1})e^{2\eta u^2} \\ \vdots \\ (u^n-u^{n-1}+1)e^{2\eta u^{n-1}} \\ (u^n-u^{n-1}-1)e^{2\eta u^n} \end{pmatrix},$$

all coefficients of proportionality being equal and positive.

Let us set $v^{i,j} := (u^n - u^i)e^{2\eta u^j}$ and $w^i := (u^n - u^i)$, for purely typographical reasons. A normal vector to the surface can be found as

$$Z := \begin{vmatrix} e_1 & \cdots & e_{n-1} & e_n \\ v^{1,1}+e^{2\eta u^1} & \cdots & v^{1,n-1} & v^{1,n}-e^{2\eta u^n} \\ \vdots & \ddots & \vdots & \vdots \\ v^{n-1,1} & \cdots & v^{n-1,n-1} & v^{n-1,n}-e^{2\eta u^n} \\ & & +e^{2\eta u^{n-1}} & \end{vmatrix}.$$

The coefficient in front of e_1 is the $(n-1)\times(n-1)$ determinant

$$\begin{vmatrix} v^{1,2} & \cdots & v^{1,n-1} & v^{1,n}-e^{2\eta u^n} \\ v^{2,2}+e^{2\eta u^2} & \cdots & v^{2,n-1} & v^{2,n}-e^{2\eta u^n} \\ \vdots & \ddots & \vdots & \vdots \\ v^{n-1,2} & \cdots & v^{n-1,n-1} & v^{n-1,n}-e^{2\eta u^n} \\ & & +e^{2\eta u^{n-1}} & \end{vmatrix}$$

$$\propto e^{-2\eta u^1}\begin{vmatrix} w^1 & \cdots & w^1 & w^1-1 \\ w^2+1 & \cdots & w^2 & w^2-1 \\ \vdots & \ddots & \vdots & \vdots \\ w^{n-1} & \cdots & w^{n-1}+1 & w^{n-1}-1 \end{vmatrix}$$

$$= e^{-2\eta u^1}\begin{vmatrix} 1 & 1 & \cdots & 1 & w^1-1 \\ 2 & 1 & \cdots & 1 & w^2-1 \\ 1 & 2 & \cdots & 1 & w^3-1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 2 & w^{n-1}-1 \end{vmatrix}$$

$$= e^{-2\eta u^1}\begin{vmatrix} 1 & 1 & \cdots & 1 & u^n-u^1-1 \\ 1 & 0 & \cdots & 0 & u^1-u^2 \\ 0 & 1 & \cdots & 0 & u^1-u^3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & u^1-u^{n-1} \end{vmatrix}$$

$$= e^{-2\eta u^1}\big((-1)^n(u^n-u^1-1)+(-1)^{n+1}(u^1-u^2) + (-1)^{n+1}(u^1-u^3)+\cdots+(-1)^{n+1}(u^1-u^{n-1})\big)$$

$$= e^{-2\eta u^1}(-1)^n\times \big((u^2+u^3+\cdots+u^n)-(n-1)u^1-1\big)$$

$$= -e^{-2\eta u^1}(-1)^n n u^1 \propto u^1 e^{-2\eta u^1} \quad (5)$$

(with a positive coefficient of proportionality, $e^{2\eta}$, in the first $\propto$; the third equality follows from the expansion of the determinant along the last column and then along the first row).

Similarly, the coefficient in front of e_i is proportional (with the same coefficient of proportionality) to $u^i e^{-2\eta u^i}$ for $i = 2,\dots,n-1$; indeed, the $(n-1)\times(n-1)$ determinant representing the coefficient in front of e_i can be reduced to the form analogous to (5) by moving the ith row to the top.

The coefficient in front of e_n is proportional to

$$e^{-2\eta u^n}\begin{vmatrix} w^1+1 & w^1 & \cdots & w^1 & w^1 \\ w^2 & w^2+1 & \cdots & w^2 & w^2 \\ \vdots & \vdots & \ddots & \vdots & \\ w^{n-2} & w^{n-2} & \cdots & w^{n-2}+1 & w^{n-2} \\ w^{n-1} & w^{n-1} & \cdots & w^{n-1} & w^{n-1}+1 \end{vmatrix}$$

$$= e^{-2\eta u^n}\begin{vmatrix} 1 & 0 & \cdots & 0 & w^1 \\ 0 & 1 & \cdots & 0 & w^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & w^{n-2} \\ -1 & -1 & \cdots & -1 & w^{n-1}+1 \end{vmatrix}$$

$$= e^{-2\eta u^n}\begin{vmatrix} 1 & 0 & \cdots & 0 & u^n-u^1 \\ 0 & 1 & \cdots & 0 & u^n-u^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & u^n-u^{n-2} \\ 0 & 0 & \cdots & 0 & nu^n \end{vmatrix} = nu^n e^{-2\eta u^n}$$

(with the coefficient of proportionality $e^{2\eta}(-1)^{n-1}$).

The Gauss–Kronecker curvature at the point with coordinates $(u^1,\dots,u^{n-1})$ is proportional (with a positive coefficient of proportionality, possibly depending on the point) to

$$\begin{vmatrix} \frac{\partial Z^{\mathrm{T}}}{\partial u^1} \\ \vdots \\ \frac{\partial Z^{\mathrm{T}}}{\partial u^{n-1}} \\ Z^{\mathrm{T}} \end{vmatrix} \quad (6)$$

(Thorpe, 1979, Chapter 12, Theorem 5, with $^{\mathrm{T}}$ standing for transposition).

Set $v^i := (1-2\eta u^i)e^{-2\eta u^i}$ and $w^i = u^i e^{-2\eta u^i}$, again for typographical reasons. A straightforward calculation allows us to rewrite determinant (6) (ignoring the

positive coefficient $((-1)^{n-1}ne^{2\eta})^n$) as

$$\begin{vmatrix} v^1 & 0 & \cdots & 0 & -v^n \\ 0 & v^2 & \cdots & 0 & -v^n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & v^{n-1} & -v^n \\ w^1 & w^2 & \cdots & w^{n-1} & w^n \end{vmatrix} \propto$$

$$\begin{vmatrix} 1-2\eta u^1 & 0 & \cdots & 0 & -1+2\eta u^n \\ 0 & 1-2\eta u^2 & \cdots & 0 & -1+2\eta u^n \\ \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & \cdots & 1-2\eta u^{n-1} & -1+2\eta u^n \\ u^1 & u^2 & \cdots & u^{n-1} & u^n \end{vmatrix}$$

$$\begin{aligned} &= u^1(1-2\eta u^2)(1-2\eta u^3)\cdots(1-2\eta u^n) \\ &+ u^2(1-2\eta u^1)(1-2\eta u^3)\cdots(1-2\eta u^n)+\cdots \\ &+ u^n(1-2\eta u^1)(1-2\eta u^2)\cdots(1-2\eta u^{n-1}) \end{aligned} \quad (7)$$

(with a positive coefficient of proportionality; to avoid calculation of the parities of various permutations, the reader might prefer to prove the last equality by induction in n, expanding the last determinant along the first column). Our goal is to show that the last expression in (7) is positive when $\eta < 1$ but can be negative when $\eta > 1$.

If $\eta > 1$, set $u^1 = u^2 := 1/2$ and $u^3 = \cdots = u^n := 0$. The last expression in (7) becomes negative. Therefore, the η-exponential superprediction set is not convex (Thorpe, 1979, Chapter 13, Theorem 1).

It remains to consider the case $\eta < 1$. Set $t_i := 1 - 2\eta u^i$, $i = 1, \dots, n$; the constraints on the t_i are

$$\begin{aligned} -1 < 1-2\eta < t_i < 1, \quad i = 1, \dots, n, \\ t_1 + \cdots + t_n = n - 2\eta > n-2. \end{aligned} \quad (8)$$

Our goal is to prove

$$(1-t_1)t_2t_3\cdots t_n + \cdots + (1-t_n)t_1t_2\cdots t_{n-1} > 0,$$

i.e.,

$$t_2t_3\cdots t_n + \cdots + t_1t_2\cdots t_{n-1} > nt_1\cdots t_n. \quad (9)$$

This reduces to

$$\frac{1}{t_1} + \cdots + \frac{1}{t_n} > n \quad (10)$$

if $t_1 \cdots t_n > 0$, and to

$$\frac{1}{t_1} + \cdots + \frac{1}{t_n} < n \quad (11)$$

if $t_1 \cdots t_n < 0$. The remaining case is where some of the t_i are zero; for concreteness, let $t_n = 0$. By (8) we have $t_1 + \cdots + t_{n-1} > n-2$, and so all of $t_1, \dots, t_{n-1}$ are positive; this shows that (9) is indeed true.

Let us prove (10). Since $t_1 \cdots t_n > 0$, all of $t_1, \dots, t_n$ are positive (if two of them were negative, the sum $t_1 + \cdots + t_n$ would be less than $n-2$; cf. (8)). Therefore,

$$\frac{1}{t_1} + \cdots + \frac{1}{t_n} > \underbrace{1 + \cdots + 1}_{n \text{ times}} = n.$$

To establish (9) it remains to prove (11). Suppose, without loss of generality, that $t_1 > 0$, $t_2 > 0, \dots$, $t_{n-1} > 0$, and $t_n < 0$. Since the function $t \in (0,1] \mapsto 1/t$ is convex, we can also assume, without loss of generality, $t_1 = \cdots = t_{n-2} = 1$. Then $t_{n-1} + t_n > 0$, and so

$$\frac{1}{t_{n-1}} + \frac{1}{t_n} < 0;$$

therefore,

$$\frac{1}{t_1} + \cdots + \frac{1}{t_{n-2}} + \frac{1}{t_{n-1}} + \frac{1}{t_n} < n-2 < n.$$

Finally, let us check that the positivity of the Gauss–Kronecker curvature implies the convexity of the η-exponential superprediction set, for $\eta \le 1$. Because of the continuity of the η-exponential superprediction surface in η we can and will assume, without loss of generality, that $\eta < 1$. The η-exponential superprediction surface will be oriented by choosing the normal vector field directed towards the origin; this can be done since

$$\begin{pmatrix} x^1 \\ \vdots \\ x^n \end{pmatrix} \propto \begin{pmatrix} e^{2\eta u^1} \\ \vdots \\ e^{2\eta u^n} \end{pmatrix}, \quad Z \propto \begin{pmatrix} -u^1 e^{-2\eta u^1} \\ \vdots \\ -u^n e^{-2\eta u^n} \end{pmatrix}, \quad (12)$$

with the first coefficient of proportionality positive (cf. (4) and the bottom row of the first determinant in (7)), and the scalar product of the two vectors in (12) is always negative.

Let us first check that the smallest principal curvature

$$k_1 = k_1(u^1, \dots, u^{n-1}, \eta)$$

of the η-exponential superprediction surface is always positive (among the arguments of k_1 we list not only the coordinates $u^1, \dots, u^{n-1}$ of a point on the surface (4) but also the learning rate $\eta \in (0,1)$). At least at some $(u^1, \dots, u^{n-1}, \eta)$ the value of $k_1(u^1, \dots, u^{n-1}, \eta)$ is positive: take a sufficiently small η and the point on the surface (4) at which the maximum of $x^1 + \cdots + x^n$ is attained (the point of the η-exponential superprediction set at which the maximum is attained will

lie on the surface since the maximum is attained at $(x^1,\dots,x^n)=(1,\dots,1)$ when $\eta=0$). Therefore, for all $(u^1,\dots,u^{n-1},\eta)$ the value of $k_1(u^1,\dots,u^{n-1},\eta)$ is positive: if k_1 had different signs at two points in the set

$$\left\{(u^1,\dots,u^{n-1},\eta)\,|\,u^1\in(0,1),\dots,u^{n-1}\in(0,1),\right.$$
$$\left.u^1+\dots+u^{n-1}<1,\eta\in(0,1)\right\},\quad(13)$$

we could connect these points by a continuous curve lying completely inside (13); at some point on the curve, k_1 would be zero, in contradiction to the positivity of the Gauss–Kronecker curvature $k_1\cdots k_{n-1}$.

Now it is easy to show that the η-exponential superprediction set is convex. Suppose there are two points A and B on the η-exponential superprediction surface such that the interval $[A,B]$ contains points outside the η-exponential superprediction set. The intersection of the plane OAB, where O is the origin, with the η-exponential superprediction surface is a planar curve; the curvature of this curve at the point between A and B closest to the origin will be negative (with the curve oriented by directing the normal vector field towards the origin), contradicting the positivity of k_1 at that point and Meusnier's theorem (cf. (12)).

5. Derivation of the Prediction Algorithm

To achieve the loss bound (1) in Theorem 1 Learner can use, as discussed earlier, the strong aggregating algorithm (see, e.g., Vovk, 2001, Section 2.1, (15)) with $\eta=1$. In this section we will find a substitution function for the strong aggregating algorithm for the Brier game with $\eta\le1$, which is the only component of the algorithm not described explicitly in Vovk (2001). Our substitution function will not require that its input, the generalized prediction, should be computed from the normalized distribution $(w^k)_{k=1}^K$ on the experts; this is a valuable feature for generalizations to an infinite number of experts (as demonstrated in, e.g., Vovk, 2001, Appendix A.1).

Suppose that we are given a generalized prediction $(l_1,\dots,l_n)^{\mathrm T}$ computed by the aggregating pseudo-algorithm from a normalized distribution on the experts. Since $(l_1,\dots,l_n)^{\mathrm T}$ is a superprediction (remember that we are assuming $\eta\le1$), we are only required to find a permitted prediction

$$\begin{pmatrix}\lambda_1\\ \lambda_2\\ \vdots\\ \lambda_n\end{pmatrix}=\begin{pmatrix}(u^1-1)^2+(u^2)^2+\dots+(u^n)^2\\ (u^1)^2+(u^2-1)^2+\dots+(u^n)^2\\ \vdots\\ (u^1)^2+(u^2)^2+\dots+(u^n-1)^2\end{pmatrix}\quad(14)$$

(cf. (4)) satisfying

$$\lambda_1\le l_1,\dots,\lambda_n\le l_n.\quad(15)$$

Now suppose we are given a generalized prediction $(L_1,\dots,L_n)^{\mathrm T}$ computed by the APA from an unnormalized distribution on the experts; in other words, we are given

$$\begin{pmatrix}L_1\\ \vdots\\ L_n\end{pmatrix}=\begin{pmatrix}l_1+c\\ \vdots\\ l_n+c\end{pmatrix}$$

for some $c\in\mathbb R$. To find (14) satisfying (15) we can first find the largest $t\in\mathbb R$ such that $(L_1-t,\dots,L_n-t)^{\mathrm T}$ is still a superprediction and then find (14) satisfying

$$\lambda_1\le L_1-t,\dots,\lambda_n\le L_n-t.\quad(16)$$

Since $t\ge c$, it is clear that $(\lambda_1,\dots,\lambda_n)^{\mathrm T}$ will also satisfy the required (15).

Proposition 1. *Define $s\in\mathbb R$ by the requirement*

$$\sum_{i=1}^n(s-L_i)^+=2.\quad(17)$$

The unique solution to the optimization problem $t\to\max$ under the constraints (16) with $\lambda_1,\dots,\lambda_n$ as in (14) will be

$$u^i=\frac{(s-L_i)^+}{2},\ \ i=1,\dots,n,\quad(18)$$
$$t=s-1-(u^1)^2-\dots-(u^n)^2.\quad(19)$$

There exists a unique s satisfying (17) since the left-hand side of (17) is a continuous, increasing (strictly increasing when positive) and unbounded above function of s. The substitution function is given by (18).

Proof of Proposition 1. Let us denote the u^i and t defined by (18) and (19) as $\overline{u}^i$ and $\overline{t}$, respectively. To see that they satisfy the constraints (16), notice that the ith constraint can be spelt out as

$$(\overline{u}^1)^2+\dots+(\overline{u}^n)^2-2\overline{u}^i+1\le L_i-\overline{t},$$

which immediately follows from (18) and (19). As a by-product, we can see that the inequality becomes an equality, i.e.,

$$\overline{t}=L_i-1+2\overline{u}^i-(\overline{u}^1)^2-\dots-(\overline{u}^n)^2,\quad(20)$$

for all i with $\overline{u}^i>0$.

We can rewrite (16) as

$$\begin{cases}t\le L_1-1+2u^1-(u^1)^2-\dots-(u^n)^2,\\ \qquad\vdots\\ t\le L_n-1+2u^n-(u^1)^2-\dots-(u^n)^2,\end{cases}\quad(21)$$

and our goal is to prove that these inequalities imply $t < \bar{t}$ (unless $u^1 = \overline{u}^1, \dots, u^n = \overline{u}^n$). Choose $\overline{u}^i$ (necessarily $\overline{u}^i > 0$ unless $u^1 = \overline{u}^1, \dots, u^n = \overline{u}^n$; in the latter case, however, we can, and will, also choose $\overline{u}^i > 0$) for which $\epsilon_i := \overline{u}^i - u^i$ is maximal. Then every value of t satisfying (21) will also satisfy

$$\begin{aligned} t &\le L_i - 1 + 2u^i - \sum_{j=1}^n (u^j)^2 \\ &= L_i - 1 + 2\overline{u}^i - 2\epsilon_i - \sum_{j=1}^n (\overline{u}^j)^2 + 2\sum_{j=1}^n \epsilon_j \overline{u}^j - \sum_{j=1}^n \epsilon_j^2 \\ &\le L_i - 1 + 2\overline{u}^i - \sum_{j=1}^n (\overline{u}^j)^2 - \sum_{j=1}^n \epsilon_j^2 \le \bar{t}, \end{aligned}$$

with the last $\le$ following from (20) and becoming $<$ when not all u^j coincide with $\overline{u}^j$. □

The detailed description of the resulting prediction algorithm was given as Algorithm 1 in Section 2. As discussed, that algorithm uses the generalized prediction $G_N(\omega)$ computed from unnormalized weights.

6. Conclusion

In this paper we only considered the simplest prediction problem for the Brier game: competing with a finite pool of experts. In the case of square-loss regression, it is possible to find efficient closed-form prediction algorithms competitive with linear functions (see, e.g., Cesa-Bianchi & Lugosi, 2006, Chapter 11). Such algorithms can often be "kernelized" to obtain prediction algorithms competitive with reproducing kernel Hilbert spaces of prediction rules. This would be an appealing research programme in the case of the Brier game as well.

Acknowledgments

We are grateful to Football-Data and Tennis-Data for providing access to the data used in this paper. This work was partly supported by EPSRC (grant EP/F002998/1). Comments by Alexey Chernov, Alex Gammerman, Yuri Kalnishkan, and anonymous referees have helped us improve the presentation.

References

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, *78*, 1–3.

Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., & Warmuth, M. K. (1997). How to use expert advice. *Journal of the Association for Computing Machinery*, *44*, 427–485.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge, England: Cambridge University Press.

Dawid, A. P. (1986). Probability forecasting. In S. Kotz, N. L. Johnson and C. B. Read (Eds.), *Encyclopedia of statistical sciences*, vol. 7, 210–218. New York: Wiley.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*, 119–139.

Haussler, D., Kivinen, J., & Warmuth, M. K. (1998). Sequential prediction of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, *44*, 1906–1925.

Kalnishkan, Y., & Vyugin, M. V. (2005). The Weak Aggregating Algorithm and weak mixability. *Proceedings of the Eighteenth Annual Conference on Learning Theory* (pp. 188–203). Berlin: Springer.

Kivinen, J., & Warmuth, M. K. (1999). Averaging expert predictions. *Proceedings of the Fourth European Conference on Computational Learning Theory* (pp. 153–167). Berlin: Springer.

Littlestone, N., & Warmuth, M. K. (1994). The Weighted Majority Algorithm. *Information and Computation*, *108*, 212–261.

Thorpe, J. A. (1979). *Elementary topics in differential geometry*. New York: Springer.

Vovk, V. (1990). Aggregating strategies. *Proceedings of the Third Annual Workshop on Computational Learning Theory* (pp. 371–383). San Mateo, CA: Morgan Kaufmann.

Vovk, V. (1998). A game of prediction with expert advice. *Journal of Computer and System Sciences*, *56*, 153–173.

Vovk, V. (2001). Competitive on-line statistics. *International Statistical Review*, *69*, 213–248.

Vovk, V. (2007). *Defensive forecasting for optimal prediction with expert advice* (Technical Report `arXiv:0708.1503` [cs.LG]). `arXiv.org` e-Print archive.

Vovk, V., & Zhdanov, F. (2008). *Prediction with expert advice for the Brier game* (Technical Report `arXiv:0708.2502v2` [cs.LG]). `arXiv.org` e-Print archive.

Sparse Multiscale Gaussian Process Regression

Christian Walder CHRISTIAN.WALDER@TUEBINGEN.MPG.DE
Kwang In Kim KIMKI@TUEBINGEN.MPG.DE
Bernhard Schölkopf BERHARD.SCHOELKOPF@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstr. 38, 72076 Tuebingen, Germany

Abstract

Most existing sparse Gaussian process (g.p.) models seek computational advantages by basing their computations on a set of m basis functions that are the covariance function of the g.p. with one of its two inputs fixed. We generalise this for the case of Gaussian covariance function, by basing our computations on m Gaussian basis functions with arbitrary diagonal covariance matrices (or length scales). For a fixed number of basis functions and any given criteria, this additional flexibility permits approximations no worse and typically better than was previously possible. We perform gradient based optimisation of the marginal likelihood, which costs $O(m^2n)$ time where n is the number of data points, and compare the method to various other sparse g.p. methods. Although we focus on g.p. regression, the central idea is applicable to all kernel based algorithms, and we also provide some results for the support vector machine (s.v.m.) and kernel ridge regression (k.r.r.). Our approach outperforms the other methods, particularly for the case of very few basis functions, *i.e.* a very high sparsity ratio.

1. Introduction

The Gaussian process (g.p.) is a popular non-parametric model for supervised learning problems. Although g.p.'s have been shown to perform well on a wide range of tasks, their usefulness is severely limited by the $O(n^3)$ time and $O(n^2)$ storage requirements where n is the number of data points. A large amount of work has been done to alleviate this problem, either by approximating the posterior distribution, or constructing degenerate covariance functions for which the exact posterior is less expensive to evaluate (Smola & Bartlett, 2000; Csató & Opper, 2002; Lawrence et al., 2002; Seeger et al., 2003; Snelson & Ghahramani, 2006) — for a unifying overview see (Quiñonero-Candela & Rasmussen, 2005). The majority of such methods achieve an $O(m^2n)$ time complexity for training where $m \ll n$ is the number of points on which the computations are based.

The g.p. can be interpreted as a linear (in the parameters) model which, due to its non-parametric nature, has potentially as many parameters to estimate as there are training points. An exception is the case where the covariance function has finite rank, such as the linear covariance function on $\mathbb{R}^d \times \mathbb{R}^d$ given by $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}'$, which has rank d. In this case the g.p. collapses to a parametric method and it is possible to derive algorithms with $O(d^2n)$ time complexity by basing the computations on d basis functions.

For non-degenerate covariance functions, most existing sparse g.p. algorithms all have in common that they base their computations on m basis functions of the form $k(\boldsymbol{v}_i, \cdot)$. Typically the set $\mathcal{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_m\}$ is taken to be a subset of the training set (Smola & Bartlett, 2000; Csató & Opper, 2002; Seeger et al., 2003). For example Seeger *et al.* (Seeger et al., 2003) employ a highly efficient approximate information gain criteria to incrementally select points from the training set in a greedy manner.

More recently Snelson and Ghahramani (2006) have shown that further improvements in the quality of the model for a given m can be made — especially for small m — by removing the restriction that $\mathcal{V}$ be a subset of the training set. For this they introduced a new sparse g.p. model which has the advantage of being closer to the full g.p., and also of being more amenable to gradient based optimisation of the marginal likelihood with respect to the set $\mathcal{V}$. A further advantage of their con-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

tinuous optimisation of $\mathcal{V}$ is that the hyper-parameters of the model can be optimised at the same time — this is more difficult when $\mathcal{V}$ is taken to be a subset of the training set, since choosing such a subset is a hard combinatoric problem.

In this paper we take a logical step forward in the development of sparse g.p. algorithms. We also base our computations on a finite set of basis functions, but remove the restriction that the basis functions be of the form $k(\boldsymbol{v}_i, \cdot)$ where k is the covariance of the g.p. This will require computing integrals involving the basis and covariance functions, and so cannot always be done in closed form. Fortunately however, closed form expressions can be obtained for arguably the most useful scenario, namely that of Gaussian covariance function (with arbitrary diagonal covariance matrix) along with Gaussian basis functions (again each with their own arbitrary diagonal covariance matrix).

The central idea is that, under some mild restrictions, we can compute the prior probability density — under the g.p. model with Gaussian covariance — of arbitrary Gaussian mixtures. Our analysis is new, but there is a precedent for it in the literature. In particular, Walder et al. (2006) employ a similar idea, but from an reproducing kernel Hilbert space (r.k.h.s.) rather than a g.p. perspective, and for a different basis and covariance function. Also related is (Gehler & Franz, 2006), which analyses from a g.p. perspective with arbitrary basis and covariance function, but with the difference that they do not take infinite limits.

Our idea has a direct r.k.h.s. analogy. Indeed the main idea is applicable to any kernel machine, but in this paper we focus on the g.p. framework. The main reason for this is that it allows us to build on the sparse g.p. model of Snelson and Ghahramani (2006), which has been shown to be amenable to gradient based optimisation of the marginal likelihood. Nonetheless we do provide some experimental results for the kernel ridge regression (k.r.r.) case, as well as an animated toy example of the support vector machine (s.v.m.), in the accompanying video.

The paper is structured as follows. Section 2 provides an introduction to g.p. regression. In Section 3 we derive the likelihood of arbitrary Gaussian mixtures under the g.p. model with Gaussian covariance, and clarify the link to r.k.h.s.'s. In Section 4 we discuss and motivate the precise probabilistic model which we use to make practical use of our theoretical results. Experimental results and conclusions are presented in Sections 5 and 6, respectively.

2. Gaussian Process Regression

We assume that we are given an independent and identically distributed (i.i.d.) sample

$$\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\} \subset \mathbb{R}^d \times \mathbb{R}$$

drawn from an unknown distribution, and the goal is to estimate $p(y|\boldsymbol{x})$. We introduce a latent variable $u \in \mathbb{R}$, and make the assumption that $p(y|u, \boldsymbol{x}) = p(y|u)$. Hence we can think of y as a noisy realisation of u, which we model by $p(y|u) = \mathcal{N}(y|u, \sigma_n^2)$ where σ_n is a hyper-parameter.[1]

The relationship $\boldsymbol{x} \to u$ is a random process $u(\cdot)$, namely a zero mean g.p. with covariance function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Typically k will be defined in terms of further hyper-parameters. We shall denote such a g.p. as $\mathcal{G}(k)$, which is defined by the fact that its joint evaluation at a finite number of input points is a zero mean Gaussian random variable with covariance

$$\mathbb{E}_{f\sim\mathcal{G}(k)}\left[f(\boldsymbol{x})f(\boldsymbol{z})\right] = k(\boldsymbol{x}, \boldsymbol{z}).$$

One can show that given the hyper-parameters, the posterior $p(\boldsymbol{u}|\mathcal{S})$, where[2] $[\boldsymbol{u}]_i = u(\boldsymbol{x}_i)$, is

$$\begin{aligned} p(\boldsymbol{u}|\mathcal{S}) &\propto p(\boldsymbol{u})\mathcal{N}(\boldsymbol{y}|\boldsymbol{u}, \sigma_n^2 I) \\ &\propto \mathcal{N}\left(\boldsymbol{u}|K_{xx}(K_{xx}+\sigma_n^2 I)^{-1}y, \sigma_n^2 K_{xx}(K_{xx}+\sigma_n^2 I)^{-1}\right), \end{aligned} \tag{1}$$

where $[K_{xx}]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Like many authors we neglect to notate the conditioning upon the hyper-parameters, both in the above expression and for the remainder of the paper. Now, it can also be shown that the latent function $u_* = u(\boldsymbol{x}_*)$ at an arbitrary test point $\boldsymbol{x}_*$ is distributed according to $p(u_*|\boldsymbol{x}_*, \mathcal{S}) = \int p(u_*|\boldsymbol{x}_*, \mathcal{S}, \boldsymbol{u})p(\boldsymbol{u}|\boldsymbol{x}_*, \mathcal{S})\,\mathrm{d}\boldsymbol{u} = \mathcal{N}(u_*|\mu_*, \sigma_*^2)$, where

$$\mu_* = \boldsymbol{y}^\top (K_{xx} + \sigma_n^2 I)^{-1}\boldsymbol{k}_*, \tag{2}$$

$$\sigma_*^2 = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^\top (K_{xx} + \sigma_n^2 I)^{-1}\boldsymbol{k}_*, \tag{3}$$

and we have defined $[\boldsymbol{k}_*]_i = k(\boldsymbol{x}_*, \boldsymbol{x}_i)$.

In a Bayesian setting, one places priors over the hyper-parameters and computes the hyper-posterior, but this usually involves costly numerical integration techniques. Alternatively one may fix the hyper-parameters to those obtained by maximising some criteria such as the marginal likelihood conditioned upon them, $p(\boldsymbol{y}|\mathcal{X}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, K_y)$, where $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$

[1] We adopt the common convention of writing $\mathcal{N}(x|\mu, \sigma)$ for the probability density at x of the Gaussian random variable with mean μ and variance σ.

[2] Square brackets with subscripts denote elements of matrices and vectors, and a colon subscript denotes an entire row or column of a matrix.

and $K_{yy} = K_{xx} + \sigma_n^2 I$ is the covariance matrix for $\boldsymbol{y}$. This can be computed using the result that

$$\log\left(p(\boldsymbol{y}|\mathcal{X})\right) \propto -\boldsymbol{y}^\top K_{yy}^{-1}\boldsymbol{y} - \log|K_{yy}| + c, \quad (4)$$

where c is a term independent of the hyper-parameters. Even when one neglects the cost of choosing the hyper-parameters however, it typically costs $O(n)$ and $O(n^2)$ time to evaluate the posterior mean and variance respectively, after an initial setup cost of $O(n^3)$.

3. Sparse Multiscale Gaussian Process Regression

In this section we – loosely speaking – derive the likelihood of a mixture of Gaussians with arbitrary diagonal covariance matrices, under a g.p. prior with a covariance function that is also a Gaussian with arbitrary diagonal covariance matrix. Let u be drawn from $\mathcal{G}(k)$. As we mentioned previously, this means that the vector of joint evaluations at an arbitrary ordered set of points $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ is a random variable, call it $\boldsymbol{u}_\mathcal{X}$, distributed according to

$$p_{\boldsymbol{u}_\mathcal{X}}(\boldsymbol{u}) = \mathcal{N}\left(\boldsymbol{u}|\boldsymbol{0}, K_{xx}\right). \quad (5)$$

Hence by definition

$$p_{\boldsymbol{u}_\mathcal{X}}(\textstyle\sum_{i=1}^m c_i\boldsymbol{u}_i) = \left|2\pi K_{xx}^{-1}\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\sum_{i,j=1}^m c_i c_j \boldsymbol{u}_i^\top K_{xx}^{-1}\boldsymbol{u}_j\right),$$

where $|\cdot|$ denotes the matrix determinant. Note that this is simply the probability density function (p.d.f.) of $\boldsymbol{u}_\mathcal{X}$ where we have set the argument to be $\sum_{i=1}^m c_i\boldsymbol{u}_i$, for some $c_i \in \mathbb{R}$. We have done this because later we will wish to determine the likelihood of a function expressed as a summation of fixed basis functions. To this end we now consider an infinite limit of the above case. Taking the limit $n \to \infty$ of uniformly distributed points[3] $\boldsymbol{x}_i$ leads to the following p.d.f. for $\mathcal{G}(k)$,

$$p_{\mathcal{G}(k)}(\textstyle\sum_{i=1}^m c_i u_i) = \left|2\pi k^{-1}\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\sum_{i,j=1}^m c_i c_j \Psi_k(u_i, u_j)\right), \quad (6)$$

where

$$\Psi_k(u_i, u_j) \triangleq \int\int k^{-1}(\boldsymbol{x}, \boldsymbol{y}) u_i(\boldsymbol{x}) u_j(\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}. \quad (7)$$

We will discuss the factor of $\left|2\pi k^{-1}\right|^{-\frac{1}{2}}$ shortly. Note that in the previous case of finite n, if we let $\boldsymbol{u} = K_{xx}\boldsymbol{\alpha}$ and assume that K_{xx} is invertible, then $\boldsymbol{\alpha} = K_{xx}^{-1}\boldsymbol{u}$. Following this finite analogy, by k^{-1} we now intend a sloppy notation for the function which, for $u = \int \alpha(\boldsymbol{x}) k(\boldsymbol{x}, \cdot)\,\mathrm{d}\boldsymbol{x}$, satisfies $\int u(\boldsymbol{x}) k^{-1}(\boldsymbol{x}, \cdot)\,\mathrm{d}\boldsymbol{x} = \alpha(\cdot)$. Hence if we define

$$M_k : \alpha \mapsto M_k\alpha = \int \alpha(\boldsymbol{x}) k(\boldsymbol{x}, \cdot)\,\mathrm{d}\boldsymbol{x},$$

then k^{-1} is by definition the Green's function (Roach, 1970) of M_k, as it satisfies

$$\int \big(M_k\alpha\big)(\boldsymbol{x}) k^{-1}(\boldsymbol{x}, \cdot)\,\mathrm{d}\boldsymbol{x} = \alpha(\cdot). \quad (8)$$

Let us now consider the covariance function given by $k(\boldsymbol{x}, \boldsymbol{y}) = cg(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma})$, where $c > 0$, $\boldsymbol{\sigma} > \boldsymbol{0} \in \mathbb{R}^d$ and g is a normalised Gaussian on $\mathbb{R}^d \times \mathbb{R}^d$ with diagonal covariance matrix, that is[4]

$$g(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma}) \triangleq |2\pi\mathrm{diag}\,(\boldsymbol{\sigma})|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\sum_{i=1}^d \frac{([\boldsymbol{x} - \boldsymbol{y}]_i)^2}{[\boldsymbol{\sigma}]_i}\right). \quad (9)$$

If we assume furthermore that our function is an arbitrary mixture of such Gaussians, so that

$$u_i(\boldsymbol{x}) = g(\boldsymbol{x}, \boldsymbol{v}_i, \boldsymbol{\sigma}_i), \quad (10)$$

then the well known integral (for the convolution of two Gaussians)

$$\int g(\boldsymbol{x}, \boldsymbol{v}_i, \boldsymbol{\sigma}_i) g(\boldsymbol{x}, \boldsymbol{v}_j, \boldsymbol{\sigma}_j)\,\mathrm{d}\boldsymbol{x} = g(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{\sigma}_i + \boldsymbol{\sigma}_j), \quad (11)$$

leads to

$$\left(\frac{1}{c} M_{cg(\cdot,\cdot,\boldsymbol{\sigma})} g(\cdot, \boldsymbol{v}_i, \boldsymbol{\sigma}_i - \boldsymbol{\sigma})\right)(\boldsymbol{x}) = g(\boldsymbol{x}, \boldsymbol{v}_i, \boldsymbol{\sigma}_i) = u_i(\boldsymbol{x}). \quad (12)$$

As the covariance function and the basis functions are all Gaussian we can obtain in closed form

$$\begin{aligned}\Psi_k(u_i, u_j) &\overset{(7,10,12)}{=} \int\int k^{-1}(\boldsymbol{x}, \boldsymbol{y}) g(\boldsymbol{x}, \boldsymbol{v}_i, \boldsymbol{\sigma}_i) \\ &\quad \cdot \left(\frac{1}{c} M_{cg(\cdot,\cdot,\boldsymbol{\sigma})} g(\cdot, \boldsymbol{v}_j, \boldsymbol{\sigma}_j - \boldsymbol{\sigma})\right)(\boldsymbol{y})\,\mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y} \\ &\overset{(8)}{=} \frac{1}{c}\int g(\boldsymbol{x}, \boldsymbol{v}_i, \boldsymbol{\sigma}_i) g(\boldsymbol{x}, \boldsymbol{v}_j, \boldsymbol{\sigma}_j - \boldsymbol{\sigma})\,\mathrm{d}\boldsymbol{x} \\ &\overset{(11)}{=} \frac{1}{c}\, g(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{\sigma}_i + \boldsymbol{\sigma}_j - \boldsymbol{\sigma}).\end{aligned}$$

[3] Although any non-vanishing distribution leads to the same result.

[4] We use diag in a sloppy fashion with two meanings — for $\boldsymbol{a} \in \mathbb{R}^n$, $\mathrm{diag}(\boldsymbol{a}) \in \mathbb{R}^{n\times n}$ is a diagonal matrix satisfying $[\mathrm{diag}(\boldsymbol{a})]_{ii} = [\boldsymbol{a}]_i$. But for $A \in \mathbb{R}^{n\times n}$, $\mathrm{diag}(A) \in \mathbb{R}^n$ is a column vector with $[\mathrm{diag}(A)]_i = [A]_{ii}$

For clarity we have noted above each equals sign the number of the equation which implies the corresponding logical step. The following expression summarises the main idea of the present section

$$p_{\mathcal{G}(cg(\cdot,\cdot,\boldsymbol{\sigma}))}\Big(\sum_{i=1}^{m} c_i g(\cdot,\boldsymbol{v}_i,\boldsymbol{\sigma}_i)\Big)$$
$$\propto \exp\Big(-\frac{1}{2}\sum_{i,j=1}^{m}\frac{1}{c}c_i c_j g(\boldsymbol{v}_i,\boldsymbol{v}_j,\boldsymbol{\sigma}_i+\boldsymbol{\sigma}_j-\boldsymbol{\sigma})\Big). \quad (13)$$

We give only an unnormalised form by neglecting the factor $\left|2\pi k^{-1}\right|^{-\frac{1}{2}}$ in (6). The neglected factor is equal to the inverse of the integral of the right hand side of the above expression with respect to all functions $\sum_{i=1}^{m} c_i g(\cdot,\boldsymbol{v}_i,\boldsymbol{\sigma}_i)$. We need not concern ourselves with choosing a measure with respect to which this integral is finite, due to the fact that, since we will be working only with ratios of the above likelihood (*i.e.* for *maximum a posteriori* (m.a.p.) estimation and marginal likelihood maximisation), we need only the unnormalised form. Note that this peculiarity is not particular to our proposed sparse approximation to the g.p., but is a property of g.p.'s in general.

Interpretation We now make two remarks regarding the expression (13). **i)** If $\boldsymbol{\sigma}_1=\boldsymbol{\sigma}_2=\cdots=\boldsymbol{\sigma}_n=\boldsymbol{\sigma}$ and we reparameterise $c_i=cc_i'$ then it simplifies to (5). **ii)** Let $c=1$ and $h(\boldsymbol{x})=\exp\left(-\frac{1}{2}\boldsymbol{x}^\top \mathrm{diag}\left(\boldsymbol{\sigma}_1\right)^{-1}\boldsymbol{x}\right)$, an unnormalised Gaussian. Using (9) and (13) we can derive the log-likelihood of h under the g.p. prior,

$$\log\Big(p_{\mathcal{G}(g(\cdot,\cdot,\boldsymbol{\sigma}))}\big(h(\cdot)\big)\Big)\propto -\sqrt{\frac{|\mathrm{diag}\left(\boldsymbol{\sigma}_1\right)|}{|\mathrm{diag}\left(2\boldsymbol{\sigma}_1-\boldsymbol{\sigma}\right)|}}. \quad (14)$$

Simple analysis of this expression shows that the most likely such function h is that with $\boldsymbol{\sigma}_1=\boldsymbol{\sigma}$. From this extremal point, as any component of $\boldsymbol{\sigma}_1$ increases, the log likelihood of h decreases without bound. Similarly decreasing any component of $\boldsymbol{\sigma}_1$ also decreases the log likelihood, and as any component of $\boldsymbol{\sigma}_1$ approaches half the value of the corresponding component of $\boldsymbol{\sigma}$, then the log likelihood decreases without bound. To be more precise, we have for all $j=1,2,\ldots,d$ that

$$\lim_{[\boldsymbol{\sigma}_1]_j\to\left(\frac{1}{2}[\boldsymbol{\sigma}]_j\right)^+}\log\Big(p_{\mathcal{G}(g(\cdot,\cdot,\boldsymbol{\sigma}))}\big(h(\cdot)\big)\Big)=-\infty.$$

An interesting consequence of the second remark is that, roughly speaking, it is not possible to recover a Gaussian function using a g.p. with Gaussian covariance, if the covariance function is more than twice as broad as the function to be recovered. Although this may at first appear to contradict proven consistency results for the Gaussian covariance function (for example (Steinwart, 2002)), this is not the case. On the contrary, such results hold only for compact domains, and our analysis is for $\mathbb{R}^d$.

An r.k.h.s. Analogy We note that (13) has a direct analogy in the theory of r.k.h.s.'s, as made clear by the following lemma. The lemma follows from (13) and the well understood relationship between every g.p. and the corresponding r.k.h.s. of functions.

Lemma 3.1. *Let $\mathcal{H}$ be the r.k.h.s. with reproducing kernel $g(\cdot,\cdot,\boldsymbol{\sigma})$. If the conditions $\boldsymbol{\sigma}_i>\frac{1}{2}\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}_j>\frac{1}{2}\boldsymbol{\sigma}$ are satisfied component-wise, then*

$$\langle g(\cdot,\boldsymbol{v}_i,\boldsymbol{\sigma}_i),g(\cdot,\boldsymbol{v}_j,\boldsymbol{\sigma}_j)\rangle_{\mathcal{H}}=g(\boldsymbol{v}_i,\boldsymbol{v}_j,\boldsymbol{\sigma}_i+\boldsymbol{\sigma}_j-\boldsymbol{\sigma}). \quad (15)$$

If either condition is not satisfied, then the corresponding function on the left hand side is not in $\mathcal{H}$.

Naturally this can also be proven directly, but doing so for the general case is more involved and we omit the details due to space limitations.[5] However, by assuming that the conditions $\boldsymbol{\sigma}_i>\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}_j>\boldsymbol{\sigma}$ are satisfied component-wise, then it is straightforward to obtain the main result. The basic idea is as follows. Using (11) we substitute $g(\cdot,\boldsymbol{v}_p,\boldsymbol{\sigma}_p)=\int g(\cdot,\boldsymbol{x}_p,\boldsymbol{\sigma})g(\boldsymbol{x}_p,\boldsymbol{v}_p,\boldsymbol{\sigma}_p-\boldsymbol{\sigma})\,\mathrm{d}\boldsymbol{x}_p$ for $p=i,j$ into the l.h.s. of (15). By linearity we can write the two integrals outside the inner product. Next we use the r.k.h.s. reproducing property — the fact that $\langle f(\cdot),g(\cdot,\boldsymbol{x},\boldsymbol{\sigma})\rangle_{\mathcal{H}}=f(\boldsymbol{x}),\forall f\in\mathcal{H},\boldsymbol{x}\in\mathbb{R}^d$ — to evaluate the inner product. Using (11) we integrate to obtain the r.h.s. of (15).

4. Inference with the Sparse Model

4.1. A Simple Approach

In the previous section we derived the g.p. likelihood over a certain restricted function space. This likelihood defines a distribution over functions of the form $\sum_{i=1}^{m} c_i g(\cdot,\boldsymbol{v}_i,\boldsymbol{\sigma}_i)$ where g as given previously is deterministic and the c_i are, by inspection of (13), normally distributed according to

$$\boldsymbol{c}\sim\mathcal{N}\left(\boldsymbol{0},U_\Psi^{-1}\right), \quad (16)$$

where $[U_\Psi]_{i,j}=\Psi_k(u_i,u_j)$. Let us write $\mathcal{U}=\{u_1,\ldots,u_m\}$ (which we refer to as the basis) and refer to the random process thus defined as $\mathcal{G}_{\mathcal{U}}(k)$. This new random process is equivalent to a full g.p. with

[5] For ICML reviewing, we can provide proof on request.

covariance function of rank at most m given by

$$\mathbb{E}_{f\sim\mathcal{G}_{\mathcal{U}}(k)}\left[f(\boldsymbol{x})f(\boldsymbol{z})\right] = \mathbb{E}_{\boldsymbol{c}\sim\mathcal{N}\left(\boldsymbol{0},U_\Psi^{-1}\right)}\left[\left(\boldsymbol{u}_{vx}^\top\boldsymbol{c}\right)\left(\boldsymbol{u}_{vz}^\top\boldsymbol{c}\right)^\top\right] = \boldsymbol{u}_{vx}^\top U_\Psi^{-1}\boldsymbol{u}_{vz}, \tag{17}$$

where $[\boldsymbol{u}_{vx}]_i = g(\boldsymbol{x},\boldsymbol{v}_i,\boldsymbol{\sigma}_i)$ and $[\boldsymbol{u}_{vz}]_i = g(\boldsymbol{z},\boldsymbol{v}_i,\boldsymbol{\sigma}_i)$.

As an aside, note that if we choose as the basis $\mathcal{U} = \{g(\cdot,\boldsymbol{x},\boldsymbol{\sigma}), g(\cdot,\boldsymbol{z},\boldsymbol{\sigma})\}$, then it is easy to verify using (17) that $\mathbb{E}_{f\sim\mathcal{G}_{\mathcal{U}}(g(\cdot,\cdot,\boldsymbol{\sigma}))}\left[f(\boldsymbol{x})f(\boldsymbol{z})\right] = \mathbb{E}_{f\sim\mathcal{G}(k)}\left[f(\boldsymbol{x})f(\boldsymbol{z})\right]$. This is analogous to a special case of the representer theorem from the theory of r.k.h.s.'s, and agrees with the interpretation that (16) is such that $\mathcal{G}_{\mathcal{U}}(k)$ approximates $\mathcal{G}(k)$ well in some sense, for the given basis $\mathcal{U}$.

Returning to the main thread, the new posterior can be derived as it was at the end of Section 2 for the exact g.p., but using the new covariance function (17). Hence after some algebra we have from (2) and (3) that, conditioned again upon the hyper-parameters, the latent function $u_* = u(\boldsymbol{x}_*)$ at an arbitrary test point is distributed according to $p_{u\sim\mathcal{G}_{\mathcal{U}}(k)}(u_*|\boldsymbol{x}_*,\mathcal{S}) = \mathcal{N}(u_*|\mu_*,\sigma_*^2)$, where

$$\mu_* = \left(U_{vx}\boldsymbol{y}\right)^\top\left(U_{vx}U_{vx}^\top + \sigma_n^2 U_\Psi\right)^{-1}\boldsymbol{u}_{v*}, \tag{18}$$

$$\sigma_*^2 = \sigma_n^2\boldsymbol{u}_{v*}^\top\left(U_{vx}U_{vx}^\top + \sigma_n^2 U_\Psi\right)^{-1}\boldsymbol{u}_{v*}, \tag{19}$$

and we have defined $[U_{vx}]_{i,j} = g(\boldsymbol{x}_j,\boldsymbol{v}_i,\boldsymbol{\sigma}_i)$, *etc.* Note that these expressions can be evaluated in $O(m)$ and $O(m^2)$ time respectively, after an initial setup or training cost of $O(m^2n)$. This is the usual improvement over the full g.p. obtained by such sparse approximation schemes. It turns out however that by employing an idea introduced by Snelson and Ghahramani (2006), we can retain these computational advantages while switching to a different model that is closer to the full g.p.

4.2. Inference with Improved Variance

A fair criticism of the previous model is that the predictive variance approaches zero far away from the basis function centres $\boldsymbol{v}_i$, as can be seen from (19). It turns out that this is particularly problematic to gradient based methods for choosing the basis (the $\boldsymbol{v}_i$ and $\boldsymbol{\sigma}_i$) by maximising the marginal likelihood (Snelson & Ghahramani, 2006). An effective but still computationally attractive way of healing the model is to switch to a different g.p. — which we denote $\tilde{\mathcal{G}}_{\mathcal{U}}(k)$ — whose covariance function satisfies

$$\mathbb{E}_{\tilde{\mathcal{G}}_{\mathcal{U}}(k)}\left[f(\boldsymbol{x})f(\boldsymbol{z})\right] = \delta_{\boldsymbol{x},\boldsymbol{z}}k(\boldsymbol{x},\boldsymbol{z}) + \overline{\delta}_{\boldsymbol{x},\boldsymbol{z}}\boldsymbol{u}_{vx}^\top U_\Psi^{-1}\boldsymbol{u}_{vz}, \tag{20}$$

where $\delta_{\boldsymbol{a},\boldsymbol{b}}$ is the Kronecker delta function and $\overline{\delta}_{\boldsymbol{a},\boldsymbol{b}} = 1 - \delta_{\boldsymbol{a},\boldsymbol{b}}$. Note that if $\boldsymbol{x} = \boldsymbol{z}$ then the covariance is that of the original g.p. $\mathcal{G}(k)$, otherwise it is that of $\mathcal{G}_{\mathcal{U}}(k)$. Unlike (17), the prior variance in this case is the same as that of the full g.p., even though in general the covariance is not. Once again the posterior can be found as before by replacing the covariance function in (2) and (3) with the right hand side of (20). In this case we obtain after some algebra the expression $p_{u\sim\tilde{\mathcal{G}}_{\mathcal{U}}(k)}(u_*|\boldsymbol{x}_*,\mathcal{S}) = \mathcal{N}(u_*|\mu_*,\sigma_*^2)$, where

$$\mu_* = \boldsymbol{u}_{v*}^\top Q^{-1}U_{vx}\left(\Lambda + \sigma_n^2 I\right)^{-1}\boldsymbol{y}, \tag{21}$$

$$\sigma_*^2 = k(\boldsymbol{x}_*,\boldsymbol{x}_*) - \boldsymbol{u}_{v*}^\top\left(U_\Psi^{-1} - Q^{-1}\right)\boldsymbol{u}_{v*}, \tag{22}$$

$\Lambda = \operatorname{diag}(\boldsymbol{\lambda})$, and

$$[\boldsymbol{\lambda}]_i = k(\boldsymbol{v}_i,\boldsymbol{v}_i) - [U_{vv}]_{:,i}^\top U_\Psi^{-1}[U_{vv}]_{:,i},$$
$$Q = U_\Psi + U_{vx}\left(\Lambda + \sigma_n^2 I\right)^{-1}U_{vx}^\top.$$

To compute the marginal likelihood we can use the expression (4). Note that it can be computed efficiently using Cholesky decompositions. In order to optimize the marginal likelihood, we also need its gradients with respect to the various parameters. Our derivation of the gradients (which closely follows (Seeger et al., 2003)) is long and tedious, and has been omitted due to space limitations. Note that by factorising appropriately, all of the required gradients can be obtained in $O(m^2n + mnd)$.

4.3. A Unifying View

We now briefly outline how the method of the previous section fits into the unifying framework of sparse g.p.'s provided by Quiñonero-Candela and Rasmussen (2005). Using Bayes rule and marginalising out the training set latent variables $\boldsymbol{u}$, we obtain the posterior

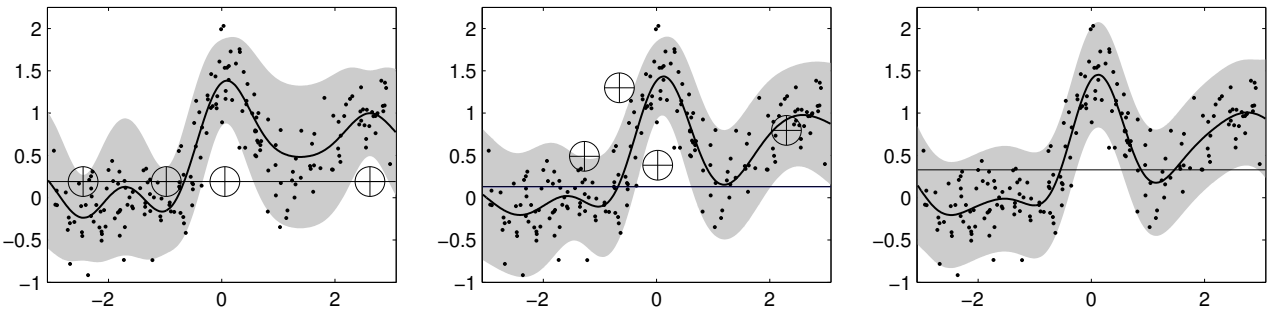
$$p(u_*|\boldsymbol{y}) = \frac{1}{p(\boldsymbol{y})}\int p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{u},u^*)\,\mathrm{d}\boldsymbol{u}.$$

Here we have neglected to notate conditioning on $\boldsymbol{x}^*$ and $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n$, and have written p instead of the more precise $p_{u\sim\mathcal{G}(k)}$. Our algorithm can be interpreted as employing two separate approximations. The first is conditional independence of $\boldsymbol{u}$ and u^* given $\boldsymbol{a}$, *i.e.*

$$p(\boldsymbol{u},u^*) = \int p(\boldsymbol{u},u^*|\boldsymbol{a})p(\boldsymbol{a})\,\mathrm{d}\boldsymbol{a} \approx \int p(\boldsymbol{u}|\boldsymbol{a})p(u^*|\boldsymbol{a})p(\boldsymbol{a})\,\mathrm{d}\boldsymbol{a},$$

where $\boldsymbol{a}$ (which is marginalised out) is taken to be

$$\left(\langle u_1,u\rangle_{\mathcal{H}}\ \ \langle u_2,u\rangle_{\mathcal{H}}\ \ \cdots\ \ \langle u_m,u\rangle_{\mathcal{H}}\right)^\top,$$



(a) *spgp-full* (b) Our *vsgp-full* (c) Exact g.p.

Figure 1. Predictive distributions (mean curve with ± two standard deviations shaded). For the *spgp-full* and *vsgp-full* algorithms, we plot the $(\boldsymbol{v}_i, \boldsymbol{\sigma}_i) \in \mathbb{R} \times \mathbb{R}$ of the basis as crossed circles. The horizontal lines denote the resulting $\boldsymbol{\sigma} \in \mathbb{R}$ of the covariance function $cg(\cdot, \cdot, \boldsymbol{\sigma})$.

the vector of inner products between the basis functions u_i and the latent function u, in the r.k.h.s. $\mathcal{H}$ associated with $k(\cdot, \cdot)$. The second approximation is

$$p(\boldsymbol{u}|\boldsymbol{a}) = \mathcal{N}\left(U_{xv}U_\Psi^{-1}\boldsymbol{v}, K_{xx} - U_{xv}U_\Psi^{-1}U_{xv}^\top\right) \approx \mathcal{N}\left(U_{xv}U_\Psi^{-1}\boldsymbol{v}, \mathrm{diag}'\left(K_{xx} - U_{xv}U_\Psi^{-1}U_{xv}^\top\right)\right).$$

where $\mathrm{diag}'(A)$ is a diagonal matrix matching A on the diagonal, and $[K_{xv}]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{v}_j)$, *etc.* Note that the first line can be shown with some algebra, whereas the second is an approximation. One can show that this leads to the result of Section 4.2, but we omit the details for brevity. Of the algorithms considered in (Quiñonero-Candela & Rasmussen, 2005), ours is closest to that of Snelson and Ghahramani (2006), however there the basis functions take the form $u_i = k(\boldsymbol{v}_i, \cdot)$, which has two implications. Firstly, $\boldsymbol{a}$ simplifies to

$$\left(u(\boldsymbol{v}_1)\ u(\boldsymbol{v}_2)\ \cdots\ u(\boldsymbol{v}_m)\right)^\top,$$

the vector of the values of u at $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$. Secondly, U_Ψ and U_{xv} simplify to K_{vv} and K_{xv}, respectively.

5. Experiments

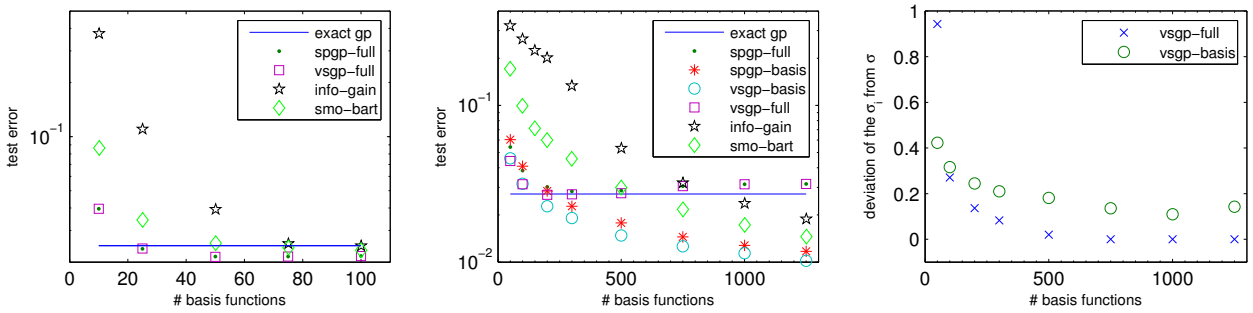
Our main goal is to demonstrate the value of being able to vary the $\boldsymbol{\sigma}_i$ individually. Note that the chief advantage of our method is in producing highly sparse solutions, and the results represent the state of the art in this respect. As such, and since the prediction cost is $O(md)$, we analyse the predictive performance of the model as a function of the number of basis functions m. Note that neither our method nor the most closely related method of Snelson and Ghahramani (2006) are particularly competitive in terms of training time. Nonetheless, there is a demand for algorithms which sacrifice training speed for testing speed, such as real-time vision and control systems, and web services in which the number of queries is large.

Let us clarify the terminology we use to refer to the various algorithms under comparison. Our new method is the variable sigma Gaussian process (v.s.g.p.). The *vsgp-full* variant consists of optimising the marginal likelihood with respect to the m basis centers $\boldsymbol{v}_i \in \mathbb{R}^d$ and length scales $\boldsymbol{\sigma}_i \in \mathbb{R}^d$ of our basis functions $u_i = g(\cdot, \boldsymbol{v}_i, \boldsymbol{\sigma}_i)$ where g is defined in (9). Also optimised are the following hyper parameters — the noise variance $\sigma_n \in \mathbb{R}$ of (1), and the parameters $c \in \mathbb{R}$ and $\boldsymbol{\sigma} \in \mathbb{R}^d$ of our original covariance function $cg(\cdot, \cdot, \boldsymbol{\sigma})$. The *vsgp-basis* variant is identical to *vsgp-full* except that σ_n, c and $\boldsymbol{\sigma}$ are determined by optimising the marginal likelihood of a full g.p. trained on a subset of the training data, and then held fixed while the $\boldsymbol{\sigma}_i$ and $\boldsymbol{v}_i$ are optimised as before. Both v.s.g.p. variants use the $\tilde{\mathcal{G}}_{\mathcal{U}}(k)$ probabilistic model of Section 4.2, where $k = cg(\cdot, \cdot, \boldsymbol{\sigma})$. For the optimisation of the sparse pseudo-input Gaussian process (s.p.g.p.) and v.s.g.p. methods we used a standard conjugate gradient type optimiser.[6]

spgp-full and *spgp-basis* correspond to the work of Snelson and Ghahramani (2006), and are identical to their v.s.g.p. counterparts except that — as with all sparse g.p. methods prior to the present work — they are forced to satisfy the constraints $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}, i = 1 \ldots m$. To initialise the marginal likelihood optimisation we take the $\boldsymbol{v}_i$ to be a *k-means* clustering of the training data. The other parameters are always initialised to the same sensible starting values, which is reasonable due to the preprocessing we employ (which is identical to that of (Seeger et al., 2003)) in order to standardise the data sets.

Figure 1 demonstrates the basic idea on a one dimensional toy problem. Using $m = 4$ basis functions is not

[6] Carl Rasmussen's `minimize.m`, which is freely available from `http://www.kyb.mpg.de/~carl`.



(a) *pumadyn-32nm* test error (b) *kin-40k* test error (c) *kin-40k* $\boldsymbol{\sigma}_i$ deviation

Figure 2. Plots (a) and (b) depict the test error as a function of basis size m. In (c) we plot against m the deviation of the $\boldsymbol{\sigma}_i$ from $\boldsymbol{\sigma}$, measured by the mean squared difference (see the text), for the *kin-40k* data set.

enough for *spgp-full* to infer a posterior similar to that of the full g.p. trained on the depicted $n = 200$ training points. The v.s.g.p. achieves a posterior closer to that of the full g.p. by employing — in comparison to the full g.p. — larger $\boldsymbol{\sigma}_i$'s and a smaller $\boldsymbol{\sigma}$. This leads to an effective covariance function — that of $\tilde{\mathcal{G}}_{\mathcal{U}}(k)$ as given by (17) — which better matches that of the full g.p. depicted in Figure 1 (c). In addition to merely observing the similarity between Figures 1 (b) and (c), we verified this last statement directly by visualising $\mathbb{E}_{\tilde{\mathcal{G}}_{\mathcal{U}}(k)}[f(\boldsymbol{x})f(\boldsymbol{z})]$ of (20) as a function of $\boldsymbol{x}$ and $\boldsymbol{z}$, but we omit the plot due to space limitations.

Figure 2 shows our experiments which, as in (Seeger et al., 2003) and (Snelson & Ghahramani, 2006), were performed on the *pumadyn-32nm* and *kin-40k* data sets.[7] Optimising the v.s.g.p. methods from a random initialisation tended to lead to inferior local optima, so we used the s.p.g.p. to find a starting point for the optimisation. This is possible because both methods optimise the same criteria, while the s.p.g.p. merely searches a subset of the space permitted by the v.s.g.p. framework. To ensure a fair comparison, we optimised the s.p.g.p. for 4000 iterations, whereas for the v.s.g.p. we optimised first the s.p.g.p. for 2000 iterations (*i.e.* fixing $\boldsymbol{\sigma}_i = \boldsymbol{\sigma}, i = 1 \dots m$), took the result as a starting point, and optimised the v.s.g.p. for a further 2000 iterations (with the $\boldsymbol{\sigma}_i$ unconstrained).

We have also reproduced with kind permission the results of Seeger *et al.* (Seeger et al., 2003), and hence have used exactly the experimental methodology described therein. The results we reproduce are from the *info-gain* and *smo-bart* methods. *info-gain* is their own method which is extremely cheap to train for a given set of hyper parameters. The method uses greedy subset selection based on a criteria which can be evaluated efficiently. *smo-bart* is similar but is based on a criteria which is more expensive to compute (Smola & Bartlett, 2000). We also show the result of training a full g.p. on a subset of the data of size 2000 and 1024 for *kin-40k* and *pumadyn-32nm*, respectively.

Neither *info-gain* nor *smo-bart* estimate the hyper-parameters, but rather fix them to the values determined by optimising the marginal likelihood of the full g.p. Hence they are most directly comparable to *spgp-basis* and *vsgp-basis*. However, *spgp-full* and *vsgp-full* correspond to the more difficult task of estimating the hyper parameters at the same time as the basis.

For *pumadyn-32nm* we do not plot *spgp-basis* and *vsgp-basis* as the results are practically identical to *spgp-full* and *vsgp-full*. This differs from (Snelson & Ghahramani, 2006), where local minima problems with *spgp-full* on the *pumadyn-32nm* data set are reported. It is unclear why our experiments did not suffer in this way — possible explanations are the choice of initial starting point, as well as the choice of optimisation algorithm. The results of the s.p.g.p. and v.s.g.p. methods on the *pumadyn-32nm* data set very similar, but both outperform the *info-gain* and *smo-bart* approaches.

The *kin-40k* results are rather different. While the $\boldsymbol{\sigma}_i$ deviated little from $\boldsymbol{\sigma}$ on the *pumadyn-32nm* data set, this was not the case for *kin-40k*, particularly for small m, as seen in Figure 2 (c) where we plot $\frac{1}{md}\sum_{i=1}^{m}\sum_{j=1}^{d}([\boldsymbol{\sigma}_i - \boldsymbol{\sigma}]_j)^2$. Our results are in agreement with those of (Snelson & Ghahramani, 2006) — our *vsgp-full* outperforms *spgp-full* for small m, which in turn outperforms both *info-gain* and *smo-bart*. However for large m both *spgp-full* and *vsgp-full*

[7] *kin-40k*: 10000 training, 30000 test, 9 attributes, see www.igi.tugraz.at/aschwaig/data.html.
pumadyn-32nm: 7168 training, 1024 test, 33 attributes, see www.cs.toronto/delve.

tend to over-fit. This is to be expected due to the use of marginal likelihood optimisation, as the choice of basis $\mathcal{U}$ is equivalent to the choice of the order of md hyper parameters for the covariance function of $\tilde{\mathcal{G}}_{\mathcal{U}}(k)$. Happily, and somewhat surprisingly, the *vsgp-full* method tends not to over-fit more than the *spgp-full*, in spite of its having roughly twice as many basis parameters. Neither *vsgp-basis* nor *spgp-basis* suffered from over-fitting however, and while they both outperform *info-gain* and *smo-bart*, our *vsgp-basis* clearly demonstrates the advantage of our new s.p.g.p. framework by consistently outperforming *spgp-basis*.

Finally, to emphasise the applicability of our idea to other kernel algorithms, we provide an accompanying video which visualises the optimisation of an s.v.m. using multiscale gaussian basis functions.

6. Conclusions

Sparse g.p. regression is an important topic which has received a lot of attention in recent years. Previous methods have based their computations on subsets of the data or pseudo input points. To relate this to our method, this is analogous to basing the computations on a set of basis functions of the form $k(\boldsymbol{v}_i, \cdot)$ where k is the covariance function and the $\boldsymbol{v}_i$ are for example the pseudo input points. We have generalised this for the case of Gaussian covariance function, by basing our computations on a set of Gaussian basis functions whose bandwidth parameters may vary independently.

This provides a new avenue for approximations, applicable to all kernel based algorithms, including g.p.'s and the s.v.m., for example. To demonstrate the utility of this new degree of freedom, we have constructed sparse g.p. and k.r.r. algorithms which outperform previous methods, particularly for very sparse solutions. As such, our approach yields state of the art performance as a function of prediction time.

References

Csató, L., & Opper, M. (2002). Sparse on-line gaussian processes. *Neural Comp.*, *14*, 641–668.

Gehler, P., & Franz, M. (2006). *Implicit wiener series, part ii: Regularised estimation* (Technical Report 148). Max Planck Institute for Biological Cybernetics.

Lawrence, N., Seeger, M., & Herbrich, R. (2002). Fast sparse gaussian process methods: The informative vector machine. *Advances in Neural Information Processing Systems 15* (pp. 609–616).

Quiñonero-Candela, J., & Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, *6*, 1935–1959.

Roach, G. F. (1970). *Green's functions*. Cambridge, UK: Cambridge University Press.

Seeger, M., Williams, C., & Lawrence, N. D. (2003). Fast forward selection to speed up sparse gaussian process regression. In C. M. Bishop and B. J. Frey (Eds.), *Workshop on ai and statistics 9*. Society for Artificial Intelligence and Statistics.

Smola, A. J., & Bartlett, P. L. (2000). Sparse greedy gaussian process regression. In T. K. Leen, T. G. Dietterich and V. Tresp (Eds.), *Advances in neural information processing systems 13*, 619–625. Cambridge, MA: MIT Press.

Snelson, E., & Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*, 1257–1264. Cambridge, MA: MIT Press.

Steinwart, I. (2002). On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, *2*, 67–93.

Walder, C., Schölkopf, B., & Chapelle, O. (2006). Implicit surface modelling with a globally regularised basis of compact support. *Proc. EUROGRAPHICS*, *25*, 635–644.

Manifold Alignment using Procrustes Analysis

Chang Wang CHWANG@CS.UMASS.EDU
Sridhar Mahadevan MAHADEVA@CS.UMASS.EDU
Computer Science Department, University of Massachusetts, Amherst, MA 01003 USA

Abstract

In this paper we introduce a novel approach to manifold alignment, based on Procrustes analysis. Our approach differs from "semi-supervised alignment" in that it results in a mapping that is defined everywhere – when used with a suitable dimensionality reduction method – rather than just on the training data points. We describe and evaluate our approach both theoretically and experimentally, providing results showing useful knowledge transfer from one domain to another. Novel applications of our method including cross-lingual information retrieval and transfer learning in Markov decision processes are presented.

1. Introduction

Manifold alignment is very useful in a variety of applications since it provides knowledge transfer between two seemingly disparate data sets. Sample applications include automatic machine translation, representation and control transfer between different Markov decision processes (MDPs), image comparison, and bioinformatics. More precisely, suppose we have two data sets $\mathcal{S}_1 = \{x_1, \cdots, x_m\}$ and $\mathcal{S}_2 = \{y_1, \cdots, y_n\}$ for which we want to find a correspondence. Working with the data in its original form can be very difficult as the data might be in high dimensional spaces and the two sets might be represented by different features. For example, $\mathcal{S}_1$ could be a collection of English documents, whereas $\mathcal{S}_2$ is a collection of Arabic documents. Thus, it may be difficult to directly compare documents from the two collections.

Even though the processing of high-dimensional data sets is challenging, for many cases, the data source may only have a limited number of degrees of freedom, implying the data set has a low intrinsic dimensionality. Similar to current work in the field, we assume kernels for computing the similarity between data points in the original space are already given. In the first step, we map the data sets to low dimensional spaces reflecting their intrinsic geometries using a standard (nonlinear or linear) dimensionality reduction approach. For example, using a graph-based nonlinear dimensionality reduction method provides a discretized approximation to the manifolds, so the new representations characterize the relationships between points but not the original features. By doing this, we can compare the embeddings of the two sets instead of their original representations. Generally speaking, if two data sets $\mathcal{S}_1$ and $\mathcal{S}_2$ have similar intrinsic geometry structures, they have similar embeddings. In our second step, we apply Procrustes analysis to align the two low dimensional embeddings of the data sets based on a number of landmark points. Procrustes analysis, which has been used for statistical shape analysis and image registration of 2D/3D data (Luo et al., 1999), removes the translational, rotational and scaling components from one set so that the optimal alignment between the two sets can be achieved.

There is a growing body of work on manifold alignment. Ham et al. (Ham et al., 2005) align the manifolds leveraging a set of correspondences. In their approach, they map the points of the two data sets to the same space by solving a constrained embedding problem, where the embeddings of the corresponding points from different sets are constrained to be identical. The work of Lafon et al. (Lafon et al., 2006) is based on a similar framework as ours. They use Diffusion Maps to embed the nodes of the graphs corresponding to the aligned sets, and then apply affine matching to align the resulting clouds of points.

Our approach differs from semi-supervised alignment (Ham et al., 2005) in that it results in a mapping that is defined everywhere rather than just on the known data points (provided a suitable dimensionality

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

reduction method like LPP (He et al., 2003) or PCA is used). Recall that semi-supervised alignment is defined only on the known data points and it is hard to handle the new test points (Bengio et al., 2004). Our method is also faster, since it requires computing eigendecompositions of much smaller matrices. Compared to affine matching, which changes the shape of one given manifold to achieve alignment, our approach keeps the manifold shape untouched. This property preserves the relationship between any two data points in each individual manifold in the process of alignment. The computation times for affine matching and Procrustes analysis are similar, both run in $O(N^3)$ (where N is the number of instances).

Given the fact that dimensionality reduction approaches play a key role in our approach, we provide a theoretical bound for the difference between subspaces spanned by low dimensional embeddings of the two data sets. This bound analytically characterizes when the two data sets can be aligned well. In addition to the theoretical analysis of our algorithm, we also report on several novel applications of our alignment approach.

The rest of this paper is as follows. In Section 2 we describe the main algorithm. In Section 3 we explain the rationale underlying our approach, and prove a bound on the difference between the subspaces spanned by low dimensional embeddings of the two data sets being aligned. We describe some novel applications and summarize our experimental results in Section 4. Section 5 provides some concluding remarks.

2. Manifold Alignment

2.1. The Problem

Given two data sets along with additional pairwise correspondences between a subset of the training instances, we want to determine a correspondence between the remaining instances in the two data sets. Formally speaking, we have two sets: $\mathcal{S}_1 = \mathcal{S}_1^l \bigcup \mathcal{S}_1^u = \{x_1, \cdots, x_m\}$, $\mathcal{S}_2 = \mathcal{S}_2^l \bigcup \mathcal{S}_2^u = \{y_1, \cdots, y_n\}$, and the subsets $\mathcal{S}_1^l$ and $\mathcal{S}_2^l$ are in pairwise alignment. We want to find a mapping f, which is more precisely defined in Section 3.1, to optimally match the points between $\mathcal{S}_1^u$ and $\mathcal{S}_2^u$.

2.2. The Algorithm

Assume the kernel K_i for computing the similarity between data points in each of the two data sets is already given. The algorithmic procedure is stated below. For the sake of concreteness, in the procedure, Laplacian eigenmap (Belkin et al., 2003) is used for dimensionality reduction.

1. **Constructing the relationship matrices:**
   - Construct the weight matrices W_1 for $\mathcal{S}_1$ and W_2 for $\mathcal{S}_2$ using K_i, where $W_1(i,j) = K_1(x_i, x_j)$ and $W_2(i,j) = K_2(y_i, y_j)$.
   - Compute Laplacian matrices $\mathcal{L}_1 = I - D_1^{-0.5} W_1 D_1^{-0.5}$ and $\mathcal{L}_2 = I - D_2^{-0.5} W_2 D_2^{-0.5}$, where D_k is a diagonal matrix ($D_k(i,i) = \sum_j W_k(i,j)$) and I is the identity matrix.
2. **Learning low dimensional embeddings of the data sets:**
   - Compute selected eigenvectors of $\mathcal{L}_1$ and $\mathcal{L}_2$ as the low dimensional embeddings of the data sets $\mathcal{S}_1$ and $\mathcal{S}_2$. Let X, X_U be the d dimensional embeddings of $\mathcal{S}_1^l$ and $\mathcal{S}_1^u$, Y, Y_U be the d dimensional embeddings of $\mathcal{S}_2^l$ and $\mathcal{S}_2^u$, where $\mathcal{S}_1^l$, $\mathcal{S}_2^l$ are in pairwise alignment and $|\mathcal{S}_1^l| = |\mathcal{S}_2^l|$.
3. **Finding the optimal alignment of X and Y:**
   - Translate the configurations in X, X_U, Y and Y_U, so that X, Y have their centroids $(\sum_{i=1}^{|\mathcal{S}_1^l|} X_i/|\mathcal{S}_1^l|, \sum_{i=1}^{|\mathcal{S}_2^l|} Y_i/|\mathcal{S}_2^l|)$ at the origin.
   - Compute the singular value decomposition (SVD) of $Y^T X$, that is $U\Sigma V^T = SVD(Y^T X)$.
   - $Y^* = kYQ$ is the optimal mapping result that minimizes $\|X - Y^*\|_F$, where $\|.\|_F$ is Frobenius norm, $Q = UV^T$ and $k = trace(\Sigma)/trace(Y^T Y)$.
4. **Apply Q and k to find correspondences between $\mathcal{S}_1^u$ and $\mathcal{S}_2^u$.**
   - $Y_U^* = kY_U Q$.
   - For each element x in X_U, its correspondence in $Y_U^* = \arg\min_{y^* \in Y_U^*} \|y^* - x\|$.

Depending on the approach that we want to use, there are several variations of Step 1. For example, if we are using PCA, then we use the covariance matrices instead of Laplacian matrices; similarly, if we are using LPP (He et al., 2003), then we construct the weight matrices W_1^l for $\mathcal{D}_1^l$, W_2^l for $\mathcal{D}_2^l$ using K_i and then learn the projections. Note that when PCA or LPP is used, then the low dimensional embedding will be defined everywhere rather than just on the training points.

3. Justification

In this section, we prove two theorems. Theorem 1 shows why the algorithm is valid. Given the fact that dimensionality reduction approaches play a key role in our approach, Theorem 2 provides a theoretical bound for the difference between subspaces spanned by low dimensional embeddings of the two data sets. This bound analytically characterizes when the two data sets can be aligned well.

3.1. Optimal Manifold Alignment

Procrustes analysis seeks the isotropic dilation and the rigid translation, reflection and rotation needed to best match one data configuration to another (Cox et al., 2001). Given low dimensional embeddings X and Y (defined in Section 2), the most convenient way to do translation is to translate the configurations in X and Y so that their centroids are at the origin. Then the problem is simplified as: finding Q and k so that $\|X - kYQ\|_F$ is minimized, where $\|\cdot\|_F$ is Frobenius norm. The matrix Q is orthonormal, giving a rotation and possibly a reflection, k is a re-scale factor to either stretch or shrink Y. Below, we show that the optimal solution is given by the *SVD* of Y^TX. A detailed review of Procrustes analysis can be found in (Cox et al., 2001).

Theorem 1: Let X and Y be low dimensional embeddings of the points with known correspondences in data set $\mathcal{S}_1$, $\mathcal{S}_2$, and X_i matches Y_i for each i. If Singular Value Decomposition (*SVD*) of Y^TX is $U\Sigma V^T$, then $Q = UV^T$ and $k = trace(\Sigma)/trace(Y^TY)$ minimize $\|X - kYQ\|_F$.
Proof:
The problem is formalized as:
$\{k_{opt}, Q_{opt}\} = \arg\min_{k,Q} \|X - kYQ\|_F$. (1.1)

It is easy to verify that
$\|X - kYQ\|_F^2 = trace(X^TX) + k^2 \cdot trace(Y^TY) - 2k \cdot trace(Q^TY^TX)$. (1.2)

Since $trace(X^TX)$ is a constant, the minimization problem is equivalent to $\{k_{opt}, Q_{opt}\} = \arg\min_{k,Q}(k^2 \cdot trace(Y^TY) - 2k \cdot trace(Q^TY^TX))$. (1.3)

Differentiating with respect to k, we have
$2k \cdot trace(Y^TY) = 2 \cdot trace(Q^TY^TX)$,
i.e. $k = trace(Q^TY^TX)/trace(Y^TY)$. (1.4)

(1.3) and (1.4) show that the minimization problem reduces to $Q_{opt} = \arg\max_Q (trace(Q^TY^TX))^2$. (1.5)

Case 1:
If $trace(Q^TY^TX) \geq 0$, then the problem becomes $Q_{opt} = \arg\max_Q \ trace(Q^TY^TX)$. (1.6)

Using Singular Value Decomposition, we have $Y^TX = U\Sigma V^T$, where U and V are orthonormal, and Σ is a diagonal matrix having as its main diagonal all the positive singular values of Y^TX. So $\max_Q trace(Q^TY^TX) = \max_Q trace(Q^TU\Sigma V^T)$. (1.7)

It is well known that for two matrices A and B, $trace(AB) = trace(BA)$, so $\max_Q trace(Q^TU\Sigma V^T) = \max_Q trace(V^TQ^TU\Sigma)$. (1.8)

For simplicity, we use Z to represent V^TQ^TU. We know Q, U and V are all orthonormal matrices, so Z is also orthonormal. It is well known that any element in an orthonormal matrix, say B, is in [-1,1] (otherwise B^TB is not an identity matrix). So we know $trace(Z\Sigma) = Z_{1,1}\Sigma_{1,1} + \cdots + Z_{c,c}\Sigma_{c,c} \leq \Sigma_{1,1} + \cdots + \Sigma_{c,c}$ (1.9) , which implies $Z = I$ maximizes $trace(Z\Sigma)$, where I is an identity matrix. (1.10)

Obviously, the solution to $Z = I$ is $Q = UV^T$. (1.11)

Case 2:
If $trace(Q^TY^TX) < 0$, then the problem becomes $Q_{opt} = \arg\min_Q \ trace(Q^TY^TX)$. (1.12)

Following the similar procedure shown above, we have $trace(Z\Sigma) = Z_{1,1}\Sigma_{1,1} + \cdots + Z_{c,c}\Sigma_{c,c} \geq -\Sigma_{1,1} - \cdots - \Sigma_{c,c}$ (1.13) , which implies that $Z = -I$ minimizes $trace(Z\Sigma)$. (1.14)

Obviously, the solution to $Z = -I$ is $Q = -UV^T$. (1.15)

Considering (1.5), it is easy to verify that $Q = UV^T$ and $Q = -UV^T$ return the same results, so $Q = UV^T$ is always the optimal solution to (1.5), no matter whether $trace(Q^TY^TX)$ is positive or not. Further, we can simplify (1.4), and have $k = trace(\Sigma)/trace(Y^TY)$. (1.16) □

3.2. Theoretical Analysis

Many dimensionality reduction approaches first compute a relationship matrix, and then project the data onto a subspace spanned by the "top" eigenvectors of the matrix. The "top" eigenvectors mean some subset of eigenvectors that are of interest. They might be eigenvectors corresponding to largest, smallest, or

A is a $N \times N$ relationship matrix computed from $\mathcal{S}_1$.
B is a $N \times N$ relationship matrix computed from $\mathcal{S}_2$.
$E = B - A$.

$\mathcal{X}$ denotes a subspace of the column space of A spanned by top M eigenvectors of A.
$\mathcal{Y}$ denotes a subspace of the column space of B spanned by top M eigenvectors of B.
X is a matrix whose columns are an orthonormal basis of $\mathcal{X}$.
Y is a matrix whose columns are an orthonormal basis of $\mathcal{Y}$.

δ_A^1 is the set of top M eigenvalues of A, δ_A^2 includes all eigenvalues of A except those in δ_A^1.
δ_B^1 is the set of top M eigenvalues of B, δ_B^2 includes all eigenvalues of B except those in δ_B^1.

d_1 is the eigengap between δ_A^1 and δ_A^2, i.e. $d_1 = \min_{\lambda_i \in \delta_A^1, \lambda_j \in \delta_A^2} |\lambda_i - \lambda_j|$.
$d = \delta_A^1 - \delta_B^2$.

P denotes the orthogonal projection onto subspace $\mathcal{X}$.
Q denotes the orthogonal projection onto subspace $\mathcal{Y}$.

$\| \cdot \|$ denotes *Operator Norm*, i.e. $\|L\|_{\mu,\nu} = \max_{\nu(x)=1} \mu(Lx)$, where μ, ν are simply $\| \cdot \|_2$.

Figure 1. Notation used in Theorem 2.

even arbitrary eigenvalues. One example is Laplacian eigenmap, where we project the data onto the subspace spanned by the "smoothest" eigenvectors of the graph Laplacian. Another example is PCA, where we project the data onto the subspace spanned by the "largest" eigenvectors of the covariance matrix. In this section, we study the general approach, which provides a general framework for each individual algorithm such as Laplacian eigenmap. We assume the two given data sets $\mathcal{S}_1$ and $\mathcal{S}_2$ do not differ significantly, so the related relationship matrices A and B are "very similar". We study the difference between the embedding subspaces corresponding to the two relationship matrices. Notation used in the proof is in Figure 1. The difference between orthogonal projections $\|Q-P\|$ characterizes the distance between the two subspaces. The proof of the theorem below is based on the perturbation theory of spectral subspaces, where $E = B-A$ can be thought as the perturbation to A. The only assumption we need to make is for any i and j, $|E_{i,j}| = |B_{i,j} - A_{i,j}| \leq \tau$.

Theorem 2: If the absolute value of each element in E is bounded by τ, and $\tau \leq 2\varepsilon d_1/(N(\pi + 2\varepsilon))$, then the difference between the two embedding subspaces $\|Q - P\|$ is at most ε.

Proof:
From the definition of operator norm, we know $\|E\| = \max_{k_1,k_2,\cdots,k_N} \sqrt{\sum_i (\sum_j k_j E_{i,j})^2}$, given $\sum_i k_i^2 = 1$. (2.1)

We can verify the following inequality always holds: $\sum_i (\sum_j k_j E_{i,j})^2 \leq N \sum_j k_j^2 \sum_i E_{i,j}^2$. (2.2)

From (2.1) and (2.2), we have $\sum_i (\sum_j k_j E_{i,j})^2 \leq N^2\tau^2 \sum_j k_j^2 = N^2\tau^2$. (2.3)

Combining (2.1) and (2.3), we have: $\|E\| \leq N\tau$. (2.4)

It can be shown that if A and E are bounded self-adjoint operators on a separable Hilbert space, then the spectrum of $A+E$ is in the closed $\|E\|$-neighborhood of the spectrum of A (Kostrykin et al., 2003). From (Kostrykin et al., 2003), we also have the following inequality: $\|Q^\perp P\| \leq \pi\|E\|/2d$. (2.5)

We know A has an isolated part δ_A^1 of the spectrum separated from its remainder δ_A^2 by gap d_1. To guarantee $A + E$ also has separated components, we need to assume $\|E\| < d_1/2$. Thus (2.5) becomes $\|Q^\perp P\| \leq \pi\|E\|/2(d_1 - \|E\|)$. (2.6)

Interchanging the roles of δ_A^1 and δ_A^2, we have the analogous inequality: $\|QP^\perp\| \leq \pi\|E\|/2(d_1 - \|E\|)$. (2.7)

Since $\|Q - P\| = max\{\|Q^\perp P\|, \|QP^\perp\|\}$ (2.8), we have $\|Q - P\| \leq \pi\|E\|/2(d_1 - \|E\|)$. (2.9)

We define $R = Q - P$, and from (2.9), we get $\|R\| \leq \pi\|E\|/2(d_1 - \|E\|)$. (2.10)

(2.10) implies that if $\|E\| \leq 2d_1\varepsilon/(2\varepsilon + \pi)$, then $\|R\| \leq \varepsilon$. (2.11)

So we have the following conclusion: if the absolute value of each element in E is bounded by τ, and $\tau \leq 2\varepsilon d_1/(N(\pi + 2\varepsilon))$, then the difference of the subspaces spanned by top M eigenvectors of A and B is at most ε. □

Theorem 2 tells us that if the eigengap (between δ_A^1 and δ_A^2) is large, then the subspace corresponding to the top M eigenvectors of A is insensitive to perturbations. In other words, the algorithm can tolerate larger differences between A and B. So when we are selecting eigenvectors to form a subspace, the eigengap is an important factor to be considered. The reasoning behind this is that if the magnitudes of the relevant eigenval-

ues do not change too much, the top M eigenvectors will not be overtaken by other eigenvectors, thus the related space is more stable. Our result in essence connects the difference between the two relationship matrices to the difference between the subspaces spanned by their low dimensional embeddings.

4. Applications and Results

In this section, we first use a toy example to illustrate how our algorithm works, then we apply our approach to transfer knowledge from one domain to another. We present results applying our approach to two real world problems: cross-lingual information retrieval and transfer learning in Markov decision processes (MDPs).

4.1. A Toy Example

In this example, we directly align two manifolds and use some pictures to illustrate how our algorithm works. The two manifolds come from real protein tertiary structure data.

Protein 3D structure reconstruction is an important step in Nuclear Magnetic Resonance (NMR) protein structure determination. Basically, it finds a map from distances to coordinates. A protein 3D structure is a chain of amino acids. Let n be the number of amino acids in a given protein and $C_1, \cdots, C_n$ be the coordinate vectors for the amino acids, where $C_i = (C_{i,1}, C_{i,2}, C_{i,3})^T$ and $C_{i,1}, C_{i,2}$, and $C_{i,3}$ are the x, y, z coordinates of amino acid i (in biology, one usually uses atom but not amino acid as the basic element in determining protein structure. Since the number of atoms is huge, for simplicity, we use amino acid as the basic element). Then the distance $d_{i,j}$ between amino acids i and j can be defined as $d_{i,j} = \|C_i - C_j\|$. Define $A = \{d_{i,j}, i, j = 1, \cdots, n\}$, and $C = \{C_i, i = 1, \cdots, n\}$. It is easy to see that if C is given, then we can immediately compute A. However, if A is given, it is non-trivial to compute C. The latter problem is called Protein 3D structure reconstruction. In fact, the problem is even more tricky, since only the distances between neighbors are reliable, and this makes A an incomplete distance matrix. The problem has been proved to be NP-complete for general sparse distance matrices (Hogben, 2006). In real life, people use other techniques, such as angle constraints and human experience, together with the partial distance matrix to determine protein structures.

With the information available to us, NMR techniques might find multiple estimations (models), since more than one configuration can be consistent with the distance matrix and the constraints. Thus, the result is an ensemble of models, rather than a single structure. Most usually, the ensemble of structures, with perhaps 10 - 50 members, all of which fit the NMR data and retain good stereochemistry is deposited with the Protein Data Bank (PDB) (Berman et al., 2000). Models related to the same protein should be similar and comparisons between the models in this ensemble provides some information on how well the protein conformation was determined by NMR.

In this test, we study a Glutaredoxin protein PDB-1G7O (this protein has 215 amino acids in total), whose 3D structure has 21 models. Since such models are already low dimensional (3D) embeddings of the distance matrices, we skip Step 1 and 2 in our algorithm. We pick up Model 1 and Model 21 for test. These two models are related to the same protein, so it makes sense to treat them as manifolds to test our techniques. We denote Model 1 by Manifold A, which is represented by matrix S_1. We denote Model 21 by Manifold B, which is represented by matrix S_2. Obviously, both S_1 and S_2 are 215×3 matrices. To evaluate our re-scale factor, we manually stretch manifold A by letting $S_1 = 4 \cdot S_1$. Manifold A and B (row vectors of S_1 and S_2 represent points in the 3D space) are shown in Figure 2(A) and Figure 2(B). In biology, such chains are called protein backbones. For the purpose of comparison, we also plot both manifolds on the same graph (Figure 2(C)). It is clear that manifold A is much larger than B, and the orientations of A and B are quite different.

To align the two manifolds, we uniformly selected 1/4 amino acids as correspondence resulting in matrix X and Y, where row i of X (from S_1) matches row i of Y (from S_2) and both X and Y are 54×3 matrices. We run our algorithm from Step 3. Our algorithm identifies the re-scale factor k as 4.2971, and the rotation matrix Q as

$$\mathbf{Q} = \begin{pmatrix} 0.56151 & -0.53218 & 0.63363 \\ 0.65793 & 0.75154 & 0.048172 \\ -0.50183 & 0.38983 & 0.77214 \end{pmatrix}.$$

S_2^*, the new representation of S_2, is computed as $S_2^* = kS_2Q$. We plot S_2^* and S_1 in the same graph (Figure 2(D)). The result shows that Manifold B is rotated and enlarged to the similar size as A, and now the two manifolds are aligned very well.

4.2. Cross-lingual Information Retrieval

In information retrieval, manifold alignment can be used to find correspondences between documents. One example is finding the exact correspondences between documents in different languages. Such systems are quite useful, since they allow users to query a docu-

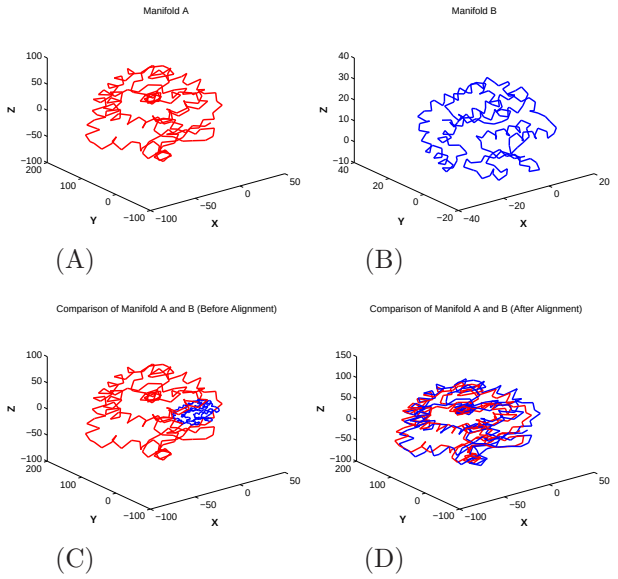


Figure 2. (A): Manifold A; (B): Manifold B; (C): Comparison of Manifold A(red) and B(blue) before alignment; (D): Comparison of Manifold A(red) and B(blue) after alignment.

ment in their native language and retrieve documents in a foreign language. Assume that we are given two document collections. For example, one in English and one in Arabic. We are also given some training correspondences between documents that are exact translations of each other. The task is: for each English or Arabian document in the untranslated set, to find the most similar document in the other corpus.

We apply our manifold alignment approach to this problem. The topical structure of each collection can be thought as a manifold over documents. Each document is a sample from the manifold. We are interested in the case where the underlying topical manifolds of two languages are similar. Our procedure for aligning collections consists of two steps: learning low dimensional embeddings of the two manifolds and aligning the low dimensional embeddings. To compute similarity of two documents in the same collection, we assume that document vectors are language models (multinomial term distributions) estimated using the document text. By treating documents as probability distributions, we can use distributional affinity to detect topical relatedness between documents. More precisely, a multinomial diffusion kernel is used for this particular application. The kernel used here is the same as the one used in (Diaz et al., 2007), where more detailed description is provided. Dimensionality reduction approaches are then used to learn the low dimensional embeddings. After shifting the centroids of the documents in each collection to the origin point, we apply our approach to learn the re-scale factor k and rotation Q from the training correspondences and then apply them to the untranslated set.

In our experiments, we used two document collections (one in English, one in Arabic, manually translated), each of which has 2119 documents. Correspondences between 25% of them were given and used to learn the mapping between them. The remaining 75% were used for testing. We used Laplacian eigenmap and LPP (the projection was learned from the data points in the correspondence) to learn the low dimensional embeddings, where top 100 eigenvectors were used to construct the embeddings. Our testing scheme is as follows: for each given Arabic document, we retrieve its top j most similar English documents. The probability that the true match is among this top j documents is used to show the goodness of the method. We also used the same data set to test the semi-supervised manifold alignment method proposed in (Ham et al., 2005), where top 100 eigenvectors were used for low dimensional embeddings. A fourth method (called baseline method) was also tested. The baseline method is as follows: assume that we have m correspondences in the training set, then document x is represented by a vector V with length m, where $V(i)$ is the similarity of x and the i^{th} document in the training correspondences. The baseline method maps the documents from different collections to the same embedding space - $\mathcal{R}^m$. Experiment results are shown in Figure 3.

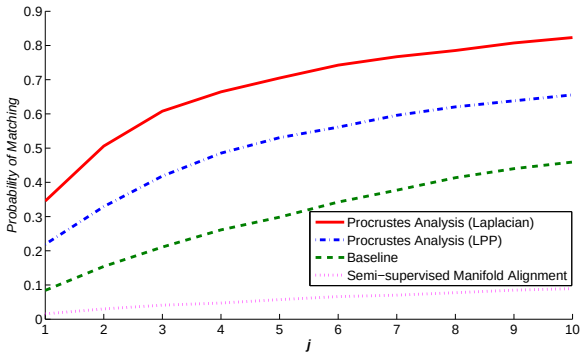


Figure 3. Cross-lingual information retrieval test.

Compared to semi-supervised manifold alignment method, the performance of Prucrustes (with Laplacian eigenmap) is significantly better. For each given Arabic document, if we retrieve 3 most relevant English documents, then the true match has a 60% probability of being among the 3. If we retrieve 10 most relevant English documents, then we have about 80% probability of getting the true match. Further, our method is much faster. Semi-supervised manifold

alignment method requires solving an eigenvalue problem over a $(n_1 + n_2 - m) \times (n_1 + n_2 - m)$ matrix, where n_i is the total number of the documents in collection i, and m is the number of training correspondences. Using our approach, the most time consuming step is finding the low dimensional embeddings with Laplacian eigenmap, which requires solving eigenvalue problems over a $n_1 \times n_1$ matrix and a $n_2 \times n_2$ matrix. We also compute the *SVD* over a $d \times d$ matrix, where d is the dimension of the low dimensional embeddings and is usually much smaller than n. In the experiments, Procrustes (with Laplacian eigenmap) is roughly 2 times faster than semi-supervised manifold alignment. Procrustes (with LPP) also returns reasonably good results: if we retrieve 10 most relevant English documents, then we have a 60% probability of getting the true match. Procrustes (with LPP) results in a mapping that is defined everywhere rather than just on the training data points and it also requires less time. Another interesting result is that the baseline algorithm also performs quite well, and better than semi-supervised alignment method. One reason that semi-supervised manifold alignment method is not working well is that mappings of the corresponding points are constrained to be identical. This might lead to "over fitting" problems for some applications.

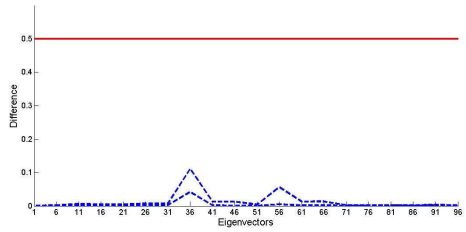
4.3. Transfer Learning in Markov Decision Process

Transfer learning studies how to re-use knowledge learned from one domain or task to a related domain or task. In this section, we investigate transfer learning in Markov decision processes (MDPs) following the approach of "proto-value functions" (PVFs), where the Laplacian eigenmap method is used to construct basis functions (Mahadevan, 2005). In a MDP, a value function is a mapping from states to real numbers, where the value of a state represents the long-term reward achieved starting from that state, and executing a particular policy. PVFs are an orthonormal basis spanning all value functions of an MDP on a state space manifold. They are computed as follows: First, create a weight matrix that reflects the topology of the state space using a series of random walks; Second, compute the graph Laplacian of the weight matrix; Third, select the smoothest k eigenvectors of this graph Laplacian as PVFs. If the state space is the same and only the reward function is changed, then the PVFs can be directly transferred to the new domain. One interesting question related to PVFs is how to transfer the old PVFs to a new domain when the new state space is only slightly different from the old one. In this section, we answer this question with our techniques.

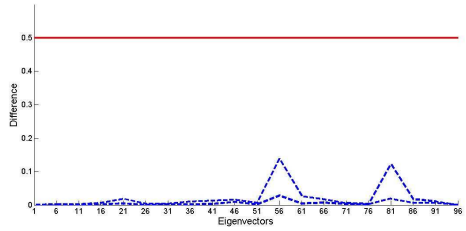
Let columns of Y denote PVFs of the current MDP. Given the procedure on how to generate PVFs, we know the rows of Y are also the low dimensional representations of the data points on the current state space manifold. Let rows of X represent the low dimensional embedding of the new manifold. Assume centroids of both X and Y are at the origin. By using isotropic dilation, reflection and rotation to align the two state space manifolds, we may find the optimal k and Q such that the two manifolds are aligned well. Our argument is that the new PVFs are YQ. The reason is as follows: suppose we have already found the optimal k and Q that minimize $\|X - kYQ\|_F$, then Y will be changed to kYQ in the process of alignment. k can be skipped, since it is well known that kYQ and YQ span the same space. The only thing that we need to show is the columns of YQ are orthonormal to each other (a requirement of PVFs). The proof is quite simple: $(YQ)^T YQ = Q^T Y^T YQ = Q^T IQ = I$, where I is an identity matrix. This means different columns of YQ are orthogonal to each other and norm of each column is 1, so YQ is orthonormal.

The conclusion shown above works when two state space manifolds are similar. Here, we still need to answer one more question: "under what conditions are the two manifolds similar?". Theorem 2 provides an answer to this question. Theorem 2 numerically bounds the difference between two spaces given the difference between the relevant relationship matrices. For this case, the relationship matrices are the Laplacian matrices used to model the state spaces. In this test, we run experiments to verify the bound. We investigate two reinforcement learning tasks. The inverted pendulum task requires balancing a pendulum of unknown mass and length by applying force to a cart attached to the pendulum. The state space is defined by two variables: the vertical angle of the pendulum, and the angular velocity of the pendulum. The mountain car task is to get a simulated car to the top of a hill as quickly as possible. The car does not have enough power to get there immediately, and so must oscillate on the hill to build up the necessary momentum. The state space is the position and velocity of the car.

We first generate two different sets of sampled states for the pendulum task and compute their related normalized graph Laplacian matrices A and B. We compute the top i non-trivial eigenvectors of A and B, and directly compute the difference between the spaces spanned by them. Theorem 2 says if the absolute value of each element in $A - B$ is bounded by τ, and $\tau \leq 2\varepsilon d_1/(N(\pi+2\varepsilon))$, then the difference of the spaces spanned by top i eigenvectors of A and B is at most



(A) Pendulum Task



(B) Mountain Car Task

Figure 4. (A): Bound for Pendulum task. (B): Bound for Mountain car task. For both tasks, ε is 0.5, true values (Max and Min in 5 tests) of the difference between two spaces are in dotted lines.

ε. We set ε be 0.5, and let τ be $\varepsilon d_1/(N(\pi+2\varepsilon))$. Here d_1 is the eigengap between top i eigenvectors and the other eigenvetors, N is 500. Based on our theorem, the difference between spaces should not be larger than ε. In our experiments, we tried 20 different values for i=1, 6, 11, $\cdots$, 96. For each i, we ran 5 tests. We carried out the same experiment on the Mountain Car task. Figure 4(A) and 4(B) respectively show the results from Pendulum task and Mountain car task. For each figure, we plot ε and the maximum and minimum difference values of the 5 tests for various values of i. For this application, the bound is loose, but the bound given in Theorem 2 is a general theoretical bound and for other applications, it might be tight. We also empirically evaluate the PVFs transfer performance. The results (not included) show that we can learn a good policy by using PVFs from a similar domain.

5. Conclusions

In this paper we introduce a novel approach to manifold alignment based on Procrustes Analysis. When used with a suitable dimensionality reduction method, our approach results in a mapping defined everywhere rather than just on the training data points. We also study the conditions under which low dimensional embeddings of two data sets can be aligned well. We presented novel applications of our approach, including cross-lingual information retrieval and transfer learning in Markov decision processes.

Acknowledgments

We thank the reviewers for their helpful comments. This project was supported in part by the National Science Foundation under grant IIS-0534999.

References

Belkin, M., Niyogi, P. (2003) Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15.

Bengio, Y. et al. (2004) Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. *NIPS* 16.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N. Weissig, H., Shindyalov, I. N., Bourne, P. E. (2000) The protein data bank. *Nucleic Acids Research*, 28:235–242.

Cox, M. F., Cox, M. A. A. (2001) Multidimensional scaling. Chapman and Hall.

Diaz, F., Metzler, D. (2007) Pseudo-aligned multilingual corpora. *The International Joint Conference on Artificial Intelligence(IJCAI) 2007.* 2727-2732.

Ham, J., Lee, D., Saul, L. (2005) Semisupervised alignment of manifolds. 10^{th} *International Workshop on Artificial Intelligence and Statistics.* 120-127.

He, X., Niyogi, P. (2003) Locality preserving projections. *The Annual Conference on Neural Information Processing Systems (NIPS) 16.*

Hogben, L. (2006) Handbook of linear algebra. Chapman/Hall CRC Press.

Kostrykin, V., Makarov, K. A., Motovilov, A. K. (2003) On a subspace perturbation problem. *Proc. of the American Mathematical Society.* 131:3469-3476.

Lafon, S., Keller, Y., Coifman, R. R. (2006) Data fusion and multi-cue data matching by diffusion maps. *IEEE transactions on Pattern Analysis and Machine Intelligence.* 28(11):1784-1797.

Luo, B., Hancock, W.R. (1999) Feature matching with Procrustes alignment and graph editing. *7th International Conference on Image Processing and its Applications.*

Mahadevan, S. (2005) Proto-value functions: developmental reinforcement learning. *The 22nd International Conference on Machine Learning (ICML).*

Dirichlet Component Analysis: Feature Extraction for Compositional Data

Hua-Yan Wang WANGHY@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception (Ministry of Education), Peking University

Qiang Yang QYANG@CSE.UST.HK
Department of Computer Science and Engineering, Hong Kong University of Science and Technology

Hong Qin QIN@CS.SUNYSB.EDU
Department of Computer Science, State University of New York at Stony Brook

Hongbin Zha ZHA@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception (Ministry of Education), Peking University

Abstract

We consider feature extraction (dimensionality reduction) for compositional data, where the data vectors are constrained to be positive and constant-sum. In real-world problems, the data components (variables) usually have complicated "correlations" while their total number is huge. Such scenario demands feature extraction. That is, we shall de-correlate the components and reduce their dimensionality. Traditional techniques such as the Principle Component Analysis (PCA) are not suitable for these problems due to unique statistical properties and the need to satisfy the constraints in compositional data. This paper presents a novel approach to feature extraction for compositional data. Our method first identifies a family of dimensionality reduction projections that preserve all relevant constraints, and then finds the optimal projection that maximizes the estimated Dirichlet precision on projected data. It reduces the compositional data to a given lower dimensionality while the components in the lower-dimensional space are de-correlated as much as possible. We develop theoretical foundation of our approach, and validate its effectiveness on some synthetic and real-world datasets.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

1. Introduction

Compositional data (positive constant-sum real vectors) are frequently encountered in various scientific disciplines and industrial applications. They quantitatively describe the parts that comprise the entire entity. In geology, scientists investigate relative proportion of different minerals in rocks. In microeconomics, household expenditure in different commodity/service groups is recorded as relative proportion. In information retrieval, documents are usually represented as relative frequencies of words in a prescribed vocabulary. Generally, compositional data are natural representations when the variables (features) are essentially probabilities of complementary and mutually exclusive events. The variables (features) in compositional data are referred to as *components* in this paper.

Feature extraction is often applied in machine learning when the datasets are large and complex. The same is needed for compositional data. The need for feature extraction arises from four aspects. First, prediction performance in classification and regression can benefit from a lower dimensional representation with de-correlated *components* to avoid the curse of dimensionality. Second, feature extraction may improve overall domain understanding, e.g., we could expect the learned *components* to represent latent independent sources from which the data are generated. Third, the computational expense of subsequent data processing can be reduced with a lower dimensionality. Finally, reducing data to two or three dimensions facilitates visualization and further analysis by domain experts.

However, traditional feature extraction techniques are

not suitable for compositional data due to several reasons. First, the traditional measurement of "correlation"[1] implicated by multivariate Gaussian and PCA only captures a linear relationship between two random variables. In contrast, the "curved" nature (Aitchison, 1983) of compositional data and the "spurious correlation" (Pearson, 1896) induced by the constant-sum constraint make it problematic to interpret correlation as merely a linear relationship. We thus need a new concept of "correlation" for compositional data. Second, the positive and constant-sum constraints for compositional data are not considered in most dimensionality reduction techniques, and simply modifying them to accommodate these constraints may induce biases.

PCA is one of the most widely used techniques for feature extraction. Given a target dimension k, PCA identifies an orthogonal projection to a k dimensional subspace that maximizes the estimated Gaussian variance of the projected data. Moreover, the covariance matrix is diagonalized such that the variables are decorrelated. Our approach adapts this framework for compositional data. In particular, we first identify a family of projections that preserve a simplex constraint as substitutes for the orthogonal projections in PCA. Then, we find an optimal projection that minimizes the "Dirichlet correlation" among the projected *components*, as a substitute for maximizing the estimated Gaussian variance in PCA. The Dirichlet correlation among the *components* is defined as the estimated Dirichlet precision on projected data. The *components* are better de-correlated and separated with a smaller Dirichlet correlation. The notion of Dirichlet correlation extends the traditional "linear" interpretation of correlation connoted in the covariance structure of multivariate Gaussian and PCA. Because of our approach's affinity to the Dirichlet distribution, we call it *Dirichlet component analysis (DCA)*.

Although the Dirichlet distribution is a natural parametric family on the simplex, its role in modeling compositional data is not well studied. As pointed out in (Aitchison, 1982), the "ultimate independence" property of the Dirichlet family prevents us from directly applying it to model compositional data. Consequently, the use of Dirichlet family in compositional data analysis has been superseded by the *log-ratio* framework (eliminating the constraints by a transformation to $\mathbb{R}^N$) originated from (Aitchison, 1982). For example, the *centered log-ratio* is defined as dividing all *components* by their geometric mean and then applying the *log* function. Although this framework has been very successful, it has certain problems. The *log-ratio* well captures variability in the central area of the simplex, but encounters singularity in peripheral areas. For example, in sparse compositional data (e.g., term frequencies in documents with thousands of terms) the log-ratio is not well defined as most denominators would be zero.

In this paper, we make three main contributions. 1) We identify a rich family of dimensionality reduction transformations for compositional data, as an alternative to existing compositional operators such as *subcomposition*, *amalgamation*, and *partition* (Aitchison, 1982). 2) We exploit the Dirichlet family for compositional data analysis to capture data variability beyond traditional concepts of statistical correlation. 3) We show that the entire framework of DCA is effective and conceptually succinct, and validate its effectiveness on two synthetic datasets and two real-world datasets.

2. Dirichlet Component Analysis

2.1. The Projection Family

Compositional data are positive constant-sum vectors. Without loss of generality, we assume all *components* to sum to one:

$$\mathbf{x} = (x_1, x_2, \cdots, x_N)^T, \quad x_i \geq 0 \text{ for all } i, \quad \sum_{i=1}^{N} x_i = 1 \tag{1}$$

All points satisfying these constraints constitute the $(N-1)$-simplex, denoted as $\mathbb{S}^N$. As low dimensional examples, $\mathbb{S}^3$ is a triangle and $\mathbb{S}^4$ is a tetrahedron.

Given a target dimension K ($K \leq N$), our first aim in dimensionality reduction is to identify a family of projections from $\mathbb{S}^N$ to $\mathbb{S}^K$.

Proposition 1 *For linear projections*

$$\mathbf{y} = \mathbf{R}\,\mathbf{x} \quad \text{where} \qquad \mathbf{R} = (r_{ij})_{K\times N} \tag{2}$$

$\mathbf{y}$ *is in* $\mathbb{S}^K$ *for all* $\mathbf{x}$ *in* $\mathbb{S}^N$ ***if and only if***
1) $r_{ij} \geq 0$ *for all* i, j.
2) $\sum_{i=1}^{K} r_{ij} = 1$, *for* $j = 1, \cdots, N$.

The proof is quite straightforward and we omit it here for brevity.

Such projections could be viewed as *rearranging* mass from the N original *components* to the K new *components*, while the law of conservation of mass is satisfied. Hence we refer to such linear transforms from $\mathbb{S}^N$ to $\mathbb{S}^K$ as *rearrangements*.

Unfortunately, we could have degenerate *rearrangements* when some rows of $\mathbf{R}$ are close to zero and

[1] It is measured by the Pearson's correlation coefficient.

as a result, the corresponding new *component* is almost ignored in the *rearranging* process. Without *a priori* knowledge we should treat the K new *components* equally, which gives rise to the family of *balanced rearrangements*:

Definition 1 (Balanced Rearrangement) *A linear projection $\mathbf{R}\,\mathbf{x} = \mathbf{y}$ is a **balanced rearrangement**, if $\mathbf{R} = (r_{ij})_{K\times N}$ satisfies:*
1) $r_{ij} \geq 0$ for all i, j.
2) $\sum_{i=1}^{K} r_{ij} = 1$, for $j = 1, \cdots, N$.
3) $\sum_{j=1}^{N} r_{ij} = N/K$, for $i = 1, \cdots, K$.

The *balanced* is described by the following proposition, which gives rise to a univariate (symmetric) Dirichlet family, as we will discuss in Section 2.2.

Proposition 2 *If $\mathbf{R}_{K\times N}$ is a balanced rearrangement matrix, $\mathbf{x}$ is a random vector in $\mathbb{S}^N$ satisfying $\mathbf{E}(x_i) = \frac{1}{N}$ for all i, then $\mathbf{y} = \mathbf{R}\,\mathbf{x}$ is a random vector in $\mathbb{S}^K$ and $\mathbf{E}(y_i) = \frac{1}{K}$ for all i.*

The proof is straightforward given the linearity of the expectation operator.

The space of balanced rearrangement projections from $\mathbb{S}^N$ to $\mathbb{S}^K$ is a $NK - N - K + 1$ dimensional vector space, which is closed with respect to the operator of weighted average. This property is useful in developing the optimization algorithm in Section 3:

Proposition 3 *If $\mathbf{R_1}$ and $\mathbf{R_2}$ are balanced rearrangement matrices, α and β are positive real numbers, then $(\alpha\mathbf{R_1} + \beta\mathbf{R_2})/(\alpha+\beta)$ is a balanced rearrangement matrix.*

This is easy to validate from the definition of balanced rearrangement.

A noticeable property of balanced rearrangements is the "shrinking effects" stated as follows:

Proposition 4 *Let $\min(\mathbf{x})$ be the minimum component of $\mathbf{x}$. $\mathbf{R}_{K\times N}$ is a balanced rearrangement matrix with $K \leq N$, then $\min(\mathbf{Rx}) \geq \min(\mathbf{x})$ for all $\mathbf{x}$ in $\mathbb{S}^N$.*

The proof is obvious as long as we notice that each component of $\mathbf{Rx}$ is N/K times a weighted average of the components of $\mathbf{x}$, where equality holds only in some trivial cases. For example, $\mathbf{R}$ is the identify matrix or $\mathbf{x} = (1/N, 1/N, \cdots, 1/N)$.

Intuitively, Proposition 4 states that the balanced rearrangements always make data points "shrink" toward the central area of the simplex, which is undesirable because it diminishes variabilities of data[2]. To solve this problem, we induce the *regularization* operator for compositional data. As shown in Figure 1, we impose on the data points a parallel move along the direction $x_1 = x_2 = \cdots = x_N$, and then project the data points back to the simplex by radial projection:

Definition 2 (Regularization) *Given a compositional dataset $\mathcal{X} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^M\}$, a **regularization** on the dataset is denoted as: $\widetilde{\mathcal{X}} = \{\widetilde{\mathbf{x}^1}, \widetilde{\mathbf{x}^2}, \cdots, \widetilde{\mathbf{x}^M}\}$, where $\widetilde{\mathbf{x}^i} = \frac{1}{\sum_{j=1}^{N}(x_j^i - \delta)}(x_1^i - \delta, x_2^i - \delta, \cdots, x_N^i - \delta,)$ for $i = 1, 2, \cdots, M$, and the **regularization factor** $\delta = \min(\min(\mathbf{x^1}), \min(\mathbf{x^2}), \cdots, \min(\mathbf{x^M}))$.*

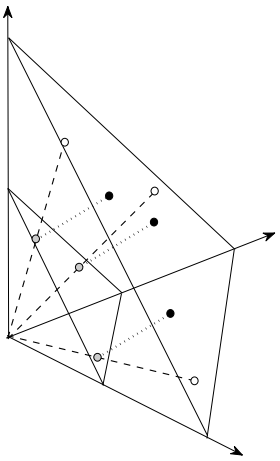


Figure 1. Regularization of compositional data points (black) is performed by parallel projection to the gray points, then radial projection to the white points.

The *regularization* operator can be viewed as a "scaling", which preserves Euclidean geometrical properties such as distance (allowing a constant scaling factor) and angle. Intuitively it "expands" the data points and compensates for the "shrinking effect" of balanced rearrangements. Its usefulness will be illustrated in a toy example in Section 2.3.1.

2.2. Dirichlet Correlation

The Dirichlet distribution (3) is conjugate prior of the multinomial, which is quite natural for compositional data arisen from independent *components*.

$$\mathrm{Dir}(\mathbf{x} \mid \alpha) = \frac{\Gamma(\sum_{i=1}^{N} \alpha_i)}{\prod_{i=1}^{N} \Gamma(\alpha_i)} \prod_{i=1}^{N} x_i^{\alpha_i - 1} \tag{3}$$

Parameters $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_N)$ could be summarized by the Dirichlet precision $\sum_{i=1}^{N} \alpha_i$ and the Dirichlet

[2] Actually, as we will show, it also increases the Dirichlet correlation among components, which is undesirable.

mean $(\alpha_1, \alpha_2, \cdots, \alpha_N)/\sum_{i=1}^{N} \alpha_i$. The Dirichlet mean actually encodes the expectation of each *component*:

$$\mathbf{E}_{\mathrm{Dir}}(x_i) = \alpha_i / \sum_{j=1}^{N} \alpha_j \tag{4}$$

Without domain knowledge, we assume the *components* in original data to be equally important. According to Proposition 2, the feature extraction process should not "prefer" any new *component*. We therefore adopt a uniform Dirichlet mean:

$$\mathrm{Dir}(\mathbf{x} \mid \alpha_0) = \frac{\Gamma(N\alpha_0)}{\Gamma(\alpha_0)^N} \prod_{i=1}^{N} x_i^{\alpha_0 - 1} \tag{5}$$

The traditional concept of "correlation" (Pearson product-moment correlation coefficient) encodes linear relationships between *components* (variables). With strong linear relationships, some *components* are redundant and the total amount of information declines. In information theory, the amount of information is measured by "uncertainty" of a distribution. The Gaussian distribution with larger variances is more "uncertain", thus is preferred in PCA. For the Dirichlet distribution (5), a smaller α_0 indicates higher "uncertainty" (amount of information) and less "correlation" among the *components* (see Figure 2), which coincides with the traditional statistical interpretation of "correlation". Hence we define correlation for compositional data in terms of α_0:

Definition 3 (Dirichlet Correlation) *Given i.i.d. compositional data set $\mathcal{X} = \{\mathbf{x^1}, \mathbf{x^2}, \cdots, \mathbf{x^M}\}$ arisen from (5), the **Dirichlet correlation** among the components with respect to $\mathcal{X}$ is defined as the maximum likelihood estimation of α_0.*

Note that α_0 is the overall (not pairwise) "correlation" among *all* components.

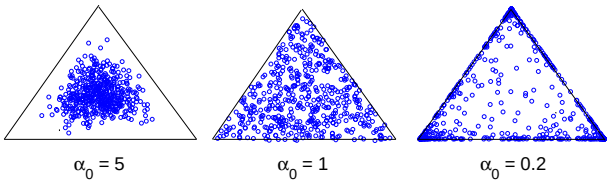


Figure 2. Points sampled from the univariate Dirichlet distribution (5) on $\mathbb{S}^3$ with different α_0.

The intuitive interpretation of the Dirichlet correlation is shown in Figure 2: 1) when $\alpha_0 > 1$, the distribution is bump-shaped, where the *components* are highly correlated and are likely to mix together in samples; 2) when $\alpha_0 = 1$, the distribution is uniform, and any proportion of mixture is equally preferred; 3) when $\alpha_0 < 1$, the distribution is valley-shaped with peaks at simplex vertices, and the *components* are better de-correlated such that the components present themselves as more purified elements in the data samples.

With the specially designed transform family and correlation measure for compositional data, we define Dirichlet component analysis (DCA) as follows:

Definition 4 (Dirichlet Component Analysis) *Given i.i.d. compositional data set $\mathcal{X} = \{\mathbf{x^1}, \mathbf{x^2}, \cdots, \mathbf{x^M}\}$ with N components, and the target dimension K, **Dirichlet component analysis** (DCA) applies a balanced rearrangement $\bar{\mathbf{R}}_{K\times N}$ and a regularization on $\mathcal{X}$ to minimize the Dirichlet correlation among the resulted K components:*

$$\bar{\mathbf{R}} = \operatorname*{argmin}_{\mathbf{R}} \operatorname*{argmax}_{\alpha_0} \mathrm{Dir}(\widetilde{\mathbf{R}(\mathcal{X})} \mid \alpha_0) \tag{6}$$

where $\widetilde{\mathbf{R}(\mathcal{X})}$ denotes that we first apply balanced rearrangement $\mathbf{R}$ to $\mathcal{X}$, and then apply a *regularization* according to Definition 2. The *i.i.d.* assumption is for factorization of the joint likelihood. The optimization problem will be discussed in Section 3.

2.3. Illustrative Examples

2.3.1. Example 1: Composition of Rocks in Geology

In this example, suppose that some rock samples are collected in a geological study in an attempt to analyze their composition. Original representation of each rock sample is a point in $\mathbb{S}^3$ (see Figure 3 left) indicating relative proportion of 3 minerals. The data points demonstrate three peaks that correspond to three substances that have fixed compositions in terms of the minerals. These peaks are formed because the formation of different substances depends on certain geological factors that vary from site to site. Hence a particular substance tends to dominate rock samples collected from some particular site. The substances had been decomposed by the chemical tests on the rocks, so that we only observe proportions of minerals.

Given the target dimension of three, DCA obtains a new representation of the rock samples (see Figure 3 right). The learned new components correspond to three underlying substances in the rock samples. Three peaks are found near the vertices of the simplex, which indicates that the new *components* are "decorrelated" in the sense that the samples tend to be

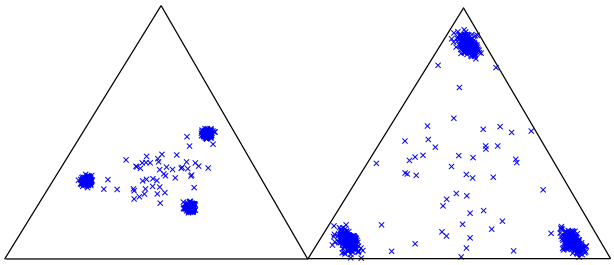


Figure 3. Left: synthetic data, composition of rock samples (small 'x') in terms of the old *components*. Right: representation in terms of new *components* (right) obtained by optimization algorithm discussed in Section 3.

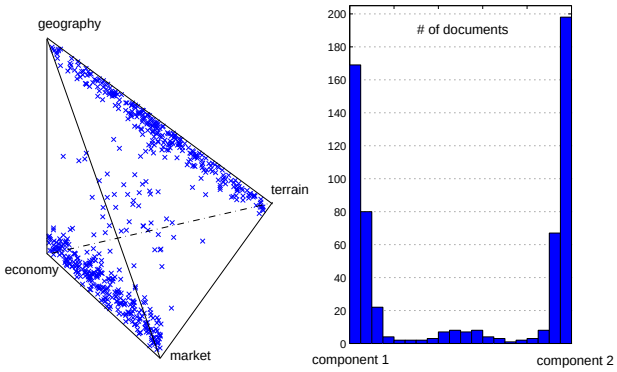


Figure 4. Left: Term frequencies of four words on $\mathbb{S}^4$, each small 'x' denotes a document. The data are synthetic. Right: new representation obtained by the optimization algorithm discussed in Section 3. The histogram illustrates the distribution of documents on $\mathbb{S}^2$.

explained by individual *components* instead of linear combinations of multiple *components*. This effect of de-correlation could be interpreted analogous to PCA. In PCA, we diagonalize the covariance matrix in order that variance in data is separately "explained" by individual variables rather than linear combinations of multiple variables. In our representation, we reveal more information about the rocks' substances because the individual components are easier to explain in further statistical analysis. In contrast, we note that PCA cannot be used to solve this rock analysis problem because by its nature this problem cannot be resolved through an orthogonal transformation.

2.3.2. Example 2: Term Frequencies in Document Retrieval

We consider a simplified bag-of-words model for document retrieval, where relative frequency values of four terms are counted in a set of documents. Each document is represented as a point in $\mathbb{S}^4$ (see Figure 4 left).

Predictably, many documents would mention both "economy" and "market" a lot, and many documents would mention both "terrain" and "geography" a lot, which gives rise to two ridge-shaped modes, corresponding to two underlying classes in these documents (one concerns economical issues, and the other discusses geological issues). Reducing the dimensionality is very likely to boost the prediction performance in classification tasks because it helps avoid overfitting (the curse of dimensionality), especially in more sophisticated high-dimensional document datasets.

Given the target dimension of two, DCA identifies two latent *components* (see Figure 4 right). The projection actually merges two pairs of semantically close *components*, and the resulting representation best preserves the information that distinguishes the two classes. Note that as an unsupervised approach, DCA cannot see any class label—all it does is minimizing the Dirichlet correlation. Although applying PCA to this toy case may have similar effects, our approach greatly outperforms PCA in higher dimensional cases, because it is specially designed for compositional data (as shown on a real-world dataset in Section 4.2).

3. Optimization

The optimization problem of DCA as defined in (6) lacks an explicit analytical loss function. Moreover, the *regularization* operator adds to the difficulty in identifying gradients or judging convexity in the parameter space.

Maximum likelihood estimation of Dirichlet precision can be carried out efficiently (Minka, 2003). The solution space is closed with respect to weighted average (Proposition 3), which motivates us to use the *genetic algorithm* (Goldberg, 1988), in which the weighted average serves as the *crossover* operator[3]. Although *genetic algorithm* is generally inefficient, it is still tractable with additional acceleration tricks. Nevertheless, genetic algorithm is just one of many choices in the optimization of DCA.

The algorithm is formalized in Algorithm 1, where "BR" is abbreviation for balanced rearrangement matrix; "DC" is abbreviation for Dirichlet correlation; "MAX" is the maximum number of iterations allowed; "SIZE" is the size of population. The *fitness* score is

[3]We do not use the *mutation* operator in our algorithm.

Algorithm 1 Genetic Algorithm for DCA

Input: dataset $\mathcal{X} \subset \mathbb{S}^N$, target dimension K
Initialize population of BR, denoted as P_0.
for $iter = 0$ **to** MAX $- 1$ **do**
  for $j = 0$ **to** SIZE $- 1$ **do**
    Apply BR_j in P_{iter} to $\mathcal{X}$
    Apply the regularization operator
    Estimate DC for transformed data
  end for
  Find minimum DC in P_{iter}
  if converged **then**
    break
  end if
  Put the BR with minimum DC into P_{iter+1}
  Compute *fitness* score for all BR
  Reduce SIZE
  for $j = 1$ **to** SIZE $- 1$ **do**
    Sample two BR from P_{iter}, probability proportional to their *fitness* scores
    Put their average, weighted by *fitness* scores, into P_{iter+1},
  end for
end for

computed as:

$$fitness = -\log(\min(\frac{DC}{\text{median } DC}, 1)), \qquad (7)$$

where "median DC" is the median Dirichlet correlation in current population. This is a key trick to accelerate the algorithm, because it prunes half of the population by assigning zero *fitness* scores. The pruning is based on the intuitive observation that: 1) the *regularization* factor is a continuous function of the BR matrix given the dataset $\mathcal{X}$; 2) the Dirichlet precision is a continuous function of the *regularization* factor and BR matrix. Hence the target function is approximately continuous and smooth in the solution space. Retaining 50% good candidate solutions in each generation is sufficient. Since the total diversity of the population diminishes, the population size could be reduced accordingly in each iteration.

4. Experimental Results

4.1. The Llobregat River Basin Hydrogeochemistry

We investigate the hydrogeochemistry dataset from the Llobregat River Basin (northeast Spain)[4] with DCA. This dataset had been studied in (Tolosana-Delgado, 2005) using factor analysis under the log-ratio framework, with which they obtained interpretable latent factors. Applying our approach on this dataset yields even more interesting results.

[4]This dataset "Hydrochem.txt" is available online at: http://rss.acs.unt.edu/Rdoc/library/compositions/data/

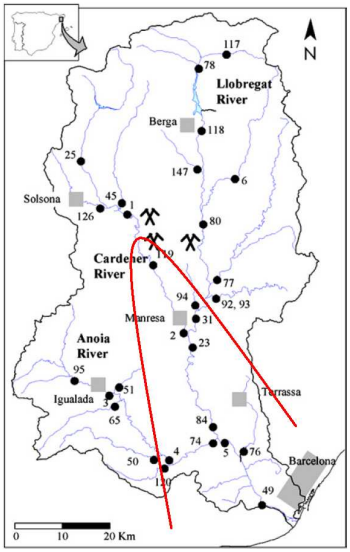


Figure 5. Sampling sites in the Llobregat River Basin, classified as "upstream" and "downstream" by the red line.

The dataset consists of 485 samples, each being a 14 dimensional compositional vector representing the concentrations of major ions (e.g. H^+, Na^+, NH_4^+, Cl^-, HCO_3^-, etc.) in the water samples. These samples are collected monthly over a certain period of time from 31 sites in the Llobregat River Basin. We classify the sites into two categories: *upstream* and *downstream*, separated by the red line (see figure 5). The 485 water samples are also classified into two categories according to the site from which they had been collected.

DCA is applied on this dataset with a target dimension of three to facilitate visualization. Visualization of high-dimensional data is crucial in disciplines such as geology, chemistry, etc., because it facilitates further analysis by domain experts.

Interestingly, although there is no location information in this dataset (locations are known from labels unseen for DCA), the two categories are well separated in the latent representation (see Figure 6). This underlying pattern is attributable to various geological and anthropogenic factors thoroughly described in (Tolosana-Delgado, 2005), which we omit here for brevity. These new patterns that are discovered by DCA was not reported in (Tolosana-Delgado, 2005), a fact highlighting the power of DCA in knowledge discovery.

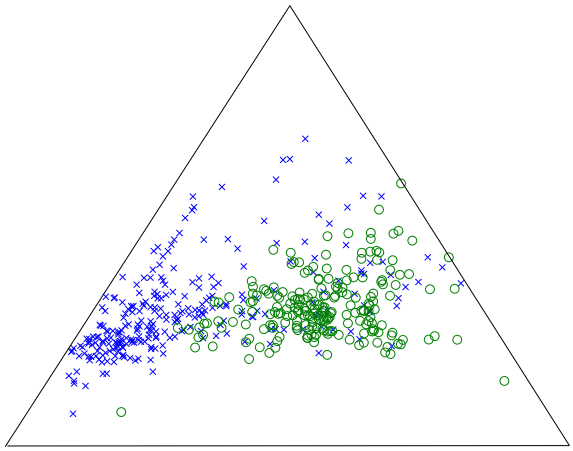


Figure 6. Latent representation of the hydrogeochemistry dataset on $\mathbb{S}^3$, learned by DCA. The two classes are well separated, see text for explanation.

4.2. Twenty newsgroups dataset

Using the 20 newsgroup data set [5], we consider a classification task of the "alt" class (798 documents) versus the "misc" class (965 documents). We show that our approach avoids overfitting and improves the prediction accuracy when we train the classifiers with a very small number of training examples, in which case the problem of overfitting could be the severest. In the preprocessing step, the "stop words" and scarce words with less than 10 total occurrences are removed. Thus the dataset we used consists of 1763 documents, where each document is represented by a 2711 dimensional sparse vector of relative word frequency values, which satisfy the constant-sum constraint.

The dimensionality is reduced to K with DCA, PCA, and LDA (latent Dirichlet allocation) (Blei, 2003), respectively. We then used a linear SVM to classify these low dimensional representations as well as the original high dimensional data for comparison. Performance results on the test test dataset are plotted with a varying number of training samples (see Figure 7) for target dimensions of $K = 10$, 20, and 50. Different choices of training data may affect the prediction performance, especially in our case where the size of training set is very small. So the performance results in our experiments are averaged over 500 different random choices of the training set. The advantage of DCA in improving prediction is clear when comparing to other techniques with the same target dimensionality, especially with very small training sets.

[5]The dataset is available at http://people.csail.mit.edu/ jrennie/20Newsgroups/

Although the highly specialized technique for bag-of-words data (LDA) beats our approach in some cases, these cases are extreme (the target dimension is 10 and the number of training samples exceeds 30). The results also justify the applicability of DCA on sparse compositional data, for which the traditional *log-ratio* framework (Aitchison, 1982) is not applicable.

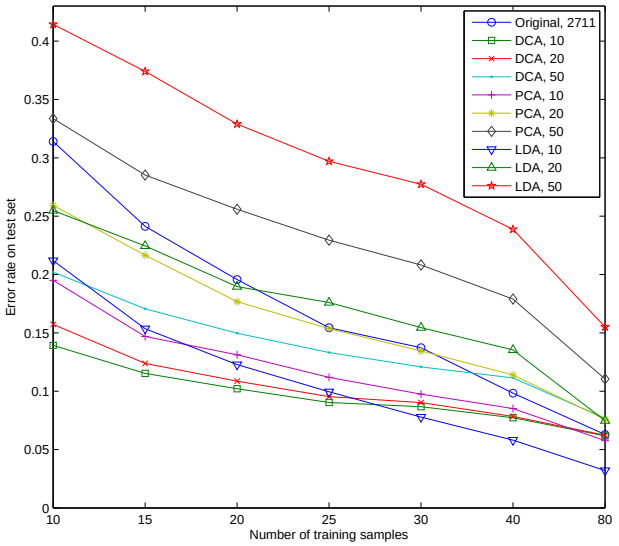


Figure 7. The "alt" versus "misc" classification performances using linear SVM. Representations with different dimensionality obtained by different methods are compared. E.g., "LDA, 10" indicates 10 dimensional representation obtained by latent Dirichlet allocation.

To make the optimization step tractable for high-dimensional data, we employed several variations in implementation. In order to reduce the solution space, we used a restricted family of transformations rather than general rearrangements. The restricted family is "amalgamations" (binary rearrangement matrices) introduced in (Aitchison, 1982), and the "balanced" requirement is not imposed. This is inspired from the toy example in Section 2.3, for which the optimal rearrangement matrix is actually an amalgamation.

5. Related Work

Feature extraction for de-correlating and reducing variables date back to K. Pearson's original idea (Pearson, 1901) on PCA. There have been a large body of research papers in the statistics and machine learning literature that address this issue, including ICA (Hyvärinen, 2001), kernel PCA (Schölkopf, 1998), etc. Directed and undirected graphical models (Blei, 2003; Welling, 2004) have also been exploited to handle this

problem, where they treat the target variables as latent nodes in the graph. Besides, the manifold assumption motivates a family of non-linear methods (Tenenbaum, 2000; Roweis, 2000), in which they use coordinates on the manifold to encode original high dimensional data.

Statistical analysis of compositional data has received a lot of concern since J. Aitchison's seminal work (Aitchison, 1982). The author proposed to transform from $\mathbb{S}^N$ to $\mathbb{R}^{N+1}$ by *log-ratio* functions, and transplanted PCA to $\mathbb{S}^N$ under the log-ratio framework (Aitchison, 1983). Our approach is an alternative PCA-like technique on $\mathbb{S}^N$, which focuses on different statistical properties (Dirichlet correlation) of data. Moreover, the *log-ratio* is not well-defined for sparse compositional data. In contrast, our approach do not have this problem. Algebraic-geometric structures (Pawlowsky-Glahn, 2001) on the simplex had been investigated, which facilitate analysis of relationship among compositional data points. Unsupervised metric learning for compositional data had been addressed in the machine learning literature (Lebanon, 2003; Wang, H.-Y., 2007).

6. Discussion

A major unresolved issue in the DCA framework is the theoretical implication of the *regularization* operator (see Figure 1), which is not compatible with the popular *log-ratio* framework, because it does not preserve the ratio between different *components*. Nevertheless, the *regularization* operator preserves Euclidean geometrical properties such as distance (allowing a constant scaling factor) and angle. Although these properties are not emphasized in the log-ratio framework, they are nonetheless meaningful as long as classification or regression tasks are concerned.

Acknowledgement

This work was supported in part by NKBRPC No. 2004CB318000, NHTRDP 863 Grant No. 2006AA01Z302, and No. 2007AA01Z336. Qiang Yang thanks Hong Kong CERG grants 621307 and and CAG grant HKBU1/05C.

References

Aitchison, J. (1982). "The statistical analysis of compositional data", *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol.44, No.2, pp.139-177

Aitchison, J. (1983). "Principal component analysis for compositional data", *Biometrika*, Vol.70, No.1, pp.57-65

Blei, D.M., Ng, A.Y., Jordan, M.I. (2003). "Latent Dirichlet allocation", *Journal of Machine Learning Research*, 3, pp.993-1022

Goldberg, D.E. and Holland, J.H. (1988) "Genetic Algorithms and Machine Learning", *Machine Learning* Volume 3, Numbers 2-3.

Hyvärinen, A., Karhunen, J., Oja, E. (2001). "*Independent Component Analysis*", John Wiley & Sons.

Lebanon, G. (2003). "Learning Riemannian metrics", In *Proceedings of the 19th UAI*

Minka, T.P. (2003) "Estimating a Dirichlet distribution", Technical Report, Microsoft Research

Pawlowsky-Glahn, V. and Egozcue, J.J. (2001). "Geometric approach to statistical analysis on the simplex", *Stochastic Environ. Res. Risk Assess. (SERRA)*, v.15, no.5, pp.384-398.

Pearson, K. (1896). "On a form of spurious correlation which may arise when indices are used in the measurements of organs", *Proceedings of the Royal Society of London*, 60, pp.489-502

Pearson, K. (1901). "On lines and planes of closest fit to systems of points in space", *Philosophical Magazine*, 2, (6), pp.559-572

Roweis, S. and Saul, L. (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding", *Science*, 290, pp.2323-2326

Schölkopf, B., Smola, A., Müller, K.R. (1998). "Nonlinear component analysis as a kernel eigenvalue problem", *Neural Computation*, Vol 10, p.1299-1319,

Tenenbaum, J., Silva, V., Langford, J. (2000). "A global geometric framework for nonlinear dimensionality reduction", *Science*, 290, pp.2319-2323

Tolosana-Delgado, R., Otero, N., Pawlowsky-Glahn, V. and Soler, A. (2005). "Latent Compositional Factors in the Llobregat River Basin (Spain) Hydrogeochemistry", *Mathematical Geology*, Vol.37, No.7, pp.681-702

Wang, H.-Y., Zha, H., Qin, H. (2007). "Dirichlet aggregation: unsupervised learning towards an optimal metric for proportional data", In *Proceedings of the 24th ICML*

Welling, M., Rosen-Zvi, M., Hinton, G. (2004). "Exponential family harmoniums with an application to information retrieval", In *NIPS*, volume 16

Adaptive p-Posterior Mixture-Model Kernels for Multiple Instance Learning

Hua-Yan Wang WANGHY@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception (Ministry of Education), Peking University

Qiang Yang QYANG@CSE.UST.HK
Department of Computer Science and Engineering, Hong Kong University of Science and Technology

Hongbin Zha ZHA@CIS.PKU.EDU.CN
Key Laboratory of Machine Perception (Ministry of Education), Peking University

Abstract

In multiple instance learning (MIL), how the instances determine the bag-labels is an essential issue, both algorithmically and intrinsically. In this paper, we show that the *mechanism* of how the instances determine the bag-labels is different for different application domains, and does not necessarily obey the traditional assumptions of MIL. We therefore propose an *adaptive* framework for MIL that adapts to different application domains by learning the domain-specific *mechanisms* merely from labeled bags. Our approach is especially attractive when we are encountered with novel application domains, for which the *mechanisms* may be different and unknown. Specifically, we exploit mixture models to represent the composition of each bag and an adaptable kernel function to represent the relationship between the bags. We validate on synthetic MIL datasets that the kernel function automatically adapts to different *mechanisms* of how the instances determine the bag-labels. We also compare our approach with state-of-the-art MIL techniques on real-world benchmark datasets.

1. Introduction

Multiple instance learning (MIL) has become an active area of investigation in machine learning since it was first put forward for drug activity prediction(Dietterich, 1997). In MIL, we consider "instance-bags", which are unordered sets of instances. Each instance is represented as a feature vector. According to the original definition, a bag of instances is labeled as positive if at least one of its instances is positive, and it is labeled as negative if all of its instances are negative. In real-world applications of MIL, the focus is on assigning labels to bags rather than instances. Many methods have been proposed to solve the MIL problem, including Axis-Parallel Rectangles (Dietterich, 1997), Diverse Density (Maron, 1998), EM-DD (Zhang, 2001), Citation k-NN (Wang, J., 2000), and variations of SVM (Andrews, 2003; Gartner, 2002; Kwok, 2007; Bunescu, 2007).

A major difficulty of MIL arises from the ambiguity caused by not knowing which instances determined the bag labels. According to the original definition of MIL (Dietterich, 1997), a bag can be labeled as positive based on just one positive instance in it. However, since the instance-labels are unknown in the outset, we need to leverage the available information conveyed by all instances to determine the label of a bag. This motivates us to carefully examine the underlying *mechanism* of how the bag labels are determined by the instances within the bag.

Firstly, examining the algorithmic aspect of the *mechanism* we could conclude that, even if there is an unambiguous intrinsic *mechanism* (*e.g.*, a bag is positive *iff* at least one instance is positive), it can hardly benefit a MIL algorithm deterministically due to the unknown instance labels. Instead, possible instance labels are usually leveraged in a probabilistic manner. For example, (Zhang, 2001) computes posteriors of instance labels in an EM-like algorithm; (Kwok, 2007) marginalizes a kernel function over possible instance

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

labels. In other words, virtually *all* instances in a bag can contribute to a bag label.

Our second observation is that the intrinsic *mechanisms* of how the instances determine the bag-labels can vary in different application domains of MIL; these *mechanisms* do not necessarily obey the original definition of MIL. Instead, they must be relaxed to allow more flexibility. Recall that in the original definition of MIL, a bag is positive *iff* at least one instance is positive. This clearly defines MIL problems in some applications such as drug activity prediction, because the "positive" instances in these applications could serve as *strong* or even *definite* evidence for labeling a bag as positive. For example, if a molecule binds well to some target protein (positive instance), the molecule undoubtedly binds well and the associated bag is labeled positive. However, in other applications, this restriction is too limiting. In many real world applications, the bag-label determining *mechanism* can allow a bag label to be negative even when there exists a positive instance in it; such relationship between instances and bags should be probabilistic in nature. For example, in content-based image retrieval (Zhang, 2002), images are represented as "bags" of localized features (regions). The low-level instance representation, describing color, texture, and shape, may have no direct correspondence to high-level image-labels (*e.g.*, human faces, buildings, the sky *etc.*). Instead, they only serve as *weak* evidences that should be integrated together to determine the image-label. Intuitively, the localized image feature descriptors can be "a region appears like a human eye or a region appears like a human nose". The decision to label an image as a "human face" should leverage many such pieces of evidence, because a non-face images (negative bags) can also contain some other object that appears like a human nose (positive instance). In this example, a positive instance can be found in a negative bag, which violates the traditional definition of MIL. Therefore, our solution to the MIL problem should be flexible enough to allow for different bag-label determining *mechanisms*.

The *mechanisms* of how the instances determine the bag labels is an essential issue in MIL. However, to our best knowledge, none of existing MIL techniques has explicitly addressed the issue of the cross-domain differences of this *mechanism*. In this paper, we propose a new framework for MIL that includes the original definition of MIL as a special case, and yet allows for more flexible cases. Our solution is to automatically *adapt* the instance-to-bag-label *mechanism* to accommodate the differences in various formulations of the MIL problems. Our main contribution is to capture the *mechanism* by a simple model, embodied in a parameter p of a kernel function (Schölkopf, 2001) defined over the bags. This parameter is learned from labeled bags in the training data without *a priori* knowledge of that *mechanism*.

Our *adaptive* framework for MIL is supported by a number of motivations. First, explicitly describing the *mechanism* (as (Dietterich, 1997) did for drug activity prediction) for an application domain calls for strong domain knowledge. Second, a hand-crafted *mechanism* could be subjective and unreliable. Third, designing different MIL methods for different application domains is inefficient, given the large number of applications that has a potential to be formalized as MIL. Thus it is better to design an *adaptive* formalism for this task.

In our framework, a two-phase learning procedure is adopted to characterize a kernel function on the bags, which can be used as a distance function in classification via algorithms such as SVM, or as a similarity measure for information retrieval.

The first learning phase exploits the unlabeled instances with a mixture model to characterize the intrinsic structures of the feature space of instances. Each bag is represented by some *aggregate posteriors* on a mixture of components, which summarizes the bag as compositions of different "patterns".

While the first learning phase adapts to different characteristics of the instance space, the adaptive nature of our approach is shown mostly in the second learning phase. We define a kernel mapping by computing a power p of the aggregate posteriors. As we will show in the rest of the paper, the parameter p explicitly captures the domain-specific *mechanism* of how the instances determine the bag-labels, where the parameter p is learned by optimizing an objective function defined over the labeled bags. In this way, our framework can adapt MIL algorithms to different instance-to-bag-label *mechanisms* in many application domains, even if we have no *a priori* knowledge about them.

2. The p-Posterior Mixture Model Kernel

2.1. Aggregate posteriors

We use lowercase $\mathbf{x}$ to denote instances, and uppercase $\mathbf{X}$ to denote bags. In MIL, we are provided with a training set $\{(\mathbf{X}_i, y_i)\}_{i=1}^N$ consisting of labeled bags, where $y_i \in \{+1, -1\}$ are labels[1]. Let $\{\mathbf{x}_i\}_{i=1}^n$

[1] Although we address two-class classification in this paper, it is straightforward to generalize our approach to multi-class classification and continuous-output regression.

be all instances available for the learning algorithm, where each instance $\mathbf{x}_i$ resides in a feature space $\mathbb{R}^D$. The training set includes both the instances in labeled bags and possibly a large number of other unlabeled instances, because the unlabeled instances are often much easier to obtain than the labeled bags in many applications[2]. The distribution of instances in the sampling domain could demonstrate sophisticated patterns due to the underlying unknown *generative model* of instances. Previous approaches usually impose over simplified assumptions on the generative model; for example, APR (Dietterich, 1997) assumes that "positive" instances reside in an axis-parallel rectangle, and Diverse Density (Maron, 1998) assumes that the "positive" instances demonstrate a Gaussian-like pattern around some concept point. In our approach, a mixture model approximates the underlying generative model of instances, which is much more flexible and informative. We make no additional constraint on the instances used; the instances are chosen for training as long as they are from the same underlying generative model. Note that this is different from semi-supervised learning (Zhu, 2005), for which we usually require the unlabeled samples to come from the specific classes of labeled samples.

We approximate the underlying generative model of instances by several mixture models in $\mathbb{R}^D$. Fitting the mixture models to all unlabeled instances with a given number of mixture components K results in the optimal parameters and weights $\{(\mathbf{\Lambda}_i, w_i)\}_{i=1}^K$. For Gaussian mixture models (GMM) adopted in our experiments, we have $\{(\mu_i, \mathbf{\Sigma}_i, w_i)\}_{i=1}^K$.

Given the above, the likelihood of an instance $\mathbf{x}$ under the i-th mixture component is denoted as:

$$p_i(\mathbf{x}) := \Pr(x|\mathbf{\Lambda}_i) \tag{1}$$

For a bag of instances $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^M$ and a mixture model $\{(\mathbf{\Lambda}_i, w_i)\}_{i=1}^K$, we define the aggregation posteriors of a bag on the mixture components:

Definition 1 (Aggregate Posteriors) *The **aggregate posteriors** of a bag of instances $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^M$ with respect to the mixture model $\{(\mathbf{\Lambda}_i, w_i)\}_{i=1}^K$ is denoted as:*

$$\psi(\mathbf{X}) := \mathcal{C} \sum_{i=1}^{M} \left(\frac{w_1 p_1(\mathbf{x}_i)}{\sum_{j=1}^K w_j p_j(\mathbf{x}_i)}, \cdots, \frac{w_K p_K(\mathbf{x}_i)}{\sum_{j=1}^K w_j p_j(\mathbf{x}_i)} \right)$$

where $\mathcal{C}$ is a normalizing operator indicating dividing a vector by the sum of all its elements.

[2]For example, we can extract image regions (instances) in thousands of arbitrary unlabelled images collected from the Internet. This is much easier than manually labeling even a small number of these images which are bags.

It is straightforward to validate that $\frac{w_j p_j(\mathbf{x}_i)}{\sum_{j=1}^K w_j p_j(\mathbf{x}_i)}$ is the posterior probability that $\mathbf{x}_i$ is generated from the j-th mixture component. The normalizing operator $\mathcal{C}$ is induced such that the kernel function (defined later) would be unbiased towards sizes of bags; "large" bags and "small" bags are treated equally. The aggregate posteriors summarize frequencies of different "pattern" within the bag, which could be viewed as a "Bayesian" histogram because a frequentist would replace the K-component mixture model with K-means clustering, and replace the posteriors with a deterministic vote. Thus, the aggregate posteriors degenerate to a normalized histogram.

The first learning phase of our framework is itself endowed with much flexibility and can be customized for specific situations. For example, in some applications the number of available unlabeled instances may be small. We therefore have to reduce the degree of freedom in the mixture model accordingly. For example, we could add the restriction that the components of the Gaussian mixture model have diagonal covariance, or even isotropic covariance[3]. When the dimensionality of the instance space is too high to fit a Gaussian mixture model, we can also adopt the frequentist's point of view by representing the training bags as histograms obtained by K-means clustering of the instances.

2.2. The order-p kernel mapping

We have defined a mapping from bags of instances $\mathbf{X}$ to aggregate posteriors $\psi(\mathbf{X}) \in \mathbb{S}^K$, where $\mathbb{S}^K$ is the $(K-1)$-simplex that consists of all positive constant-sum real vectors. The aggregate posteriors summarize frequencies of different patterns[4] within the bag. For example, consider a toy case where $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ are three bags in some MIL problem, and we have:

$$\begin{aligned}
\psi(\mathbf{X}_1) &= (0, \quad 0.3, \quad 0.5, \quad 0.2), \\
\psi(\mathbf{X}_2) &= (0, \quad 0.2, \quad 0.6, \quad 0.2), \\
\psi(\mathbf{X}_3) &= (0.2, \quad 0.1, \quad 0.6, \quad 0.1).
\end{aligned}$$

We will carefully examine this toy case for an intuitive understanding of our approach. Aggregate posteriors of these bags all demonstrate relatively high values on the third mixture component, and low values on others. According to the definition of aggregate posteriors, the bags all have "major patterns" represented by the third mixture component, with a lot

[3]An isotropic covariance matrix is in the form $\lambda\mathbf{I}$.

[4]The "patterns" are represented by the components of the mixture model.

of instances contributing to that pattern, and "minor patterns" represented by other components, with fewer instances contributing to them.

The kernel function for the bags serves as a similarity measure that affects the decisions in label prediction. Therefore how to define the kernel function depends on the intrinsical *mechanism* that the bag-labels are determined. Since the *mechanism* varies in different application domains, the kernel function should vary accordingly. On one hand, in some applications such as drug activity prediction, positive bags are determined by a few (at least one, actually) positive instances serving as *strong* evidences, and there can be many negative instances in positive bags. Hence the "minor patterns" in the aggregate posteriors are endowed with considerable significance, given that the "major patterns" could be dominated by overwhelming negative instances. On the other hand, in other applications such as image classification, the positive bags are determined by integrating a lot of low-level *weak* evidences from instances. Hence we should focus on the "major patterns" in accordance with the voting-like *mechanism*. The "minor patterns", however, should be underrated because they are attributable to random noise and outliers. For example, in an image classification task, the "minor patterns" could be generated by the image background clutter.

For the toy case, the similarity in minor patterns between $\mathbf{X}_1$ and $\mathbf{X}_2$ is greater than that between $\mathbf{X}_2$ and $\mathbf{X}_3$, but the similarity in major patterns between $\mathbf{X}_2$ and $\mathbf{X}_3$ is greater than that between $\mathbf{X}_1$ and $\mathbf{X}_2$. According to our previous analysis, whether $\mathbf{X}_2$ should be considered more similar to $\mathbf{X}_1$ or $\mathbf{X}_3$ depends on whether we should place more emphasis on major or minor patterns; the latter in turn depends on the domain-specific instance-to-bag-label *mechanism*. To endow the kernel function with such flexibility, we define the p-posterior-mixture-model (*ppmm*) kernel:

Definition 2 (p-Posterior-Mixture-Model Kernel) *The p-posterior-mixture-model (ppmm) kernel function on a pair of bags $\mathbf{X}_1$ and $\mathbf{X}_2$ is defined as*

$$\kappa_p(\mathbf{X}_1, \mathbf{X}_2) := < \psi(\mathbf{X}_1)^p, \psi(\mathbf{X}_2)^p >$$

where $p \in (0, \infty)$, and $< \bullet, \bullet >$ denotes the standard inner-product in $\mathbb{R}^K$.

For the toy case, it is easy to validate that:

$$\begin{aligned} \kappa_p(\mathbf{X}_1, \mathbf{X}_2) &> \kappa_p(\mathbf{X}_2, \mathbf{X}_3), \quad \text{if} \quad p < 1; \\ \kappa_p(\mathbf{X}_1, \mathbf{X}_2) &= \kappa_p(\mathbf{X}_2, \mathbf{X}_3), \quad \text{if} \quad p = 1; \\ \kappa_p(\mathbf{X}_1, \mathbf{X}_2) &< \kappa_p(\mathbf{X}_2, \mathbf{X}_3), \quad \text{if} \quad p > 1. \end{aligned}$$

The parameter p tunes the kernel in a way that a larger p makes it put more emphasis on major patterns, and a smaller p draws more attention to the minor patterns. According to our previous analysis, we can predict that a larger p is preferred in applications such as image classification, and a smaller p is preferred in applications such as drug activity prediction. However these judgements are based on the fact that we already have sufficient *a priori* knowledge about these two application domains. If we encounter a novel application domain of MIL, for which we have no *a priori* knowledge, the p-posterior-mixture-model kernel can be adapted to that novel domain by learning the domain-specific instance-to-bag-label *mechanism*. Learning the *mechanism* is implemented by optimizing an objective function of p defined on labeled bags.

Given labeled bags $\{(\mathbf{X}_i, y_i)\}_{i=1}^N$, $y_i \in \{+1, -1\}$, we learn the parameter p via maximizing the *alignment* (Cristianini, 2002) between the p-posterior-mixture-model kernel and the ideal kernel, which measures the kernel's degree of agreement with the bag-labels:

$$\arg\max_p \frac{< \mathbf{K}_p \, , \, \mathbf{y}\mathbf{y}^T >_F}{\sqrt{< \mathbf{K}_p \, , \, \mathbf{K}_p >_F < \mathbf{y}\mathbf{y}^T \, , \, \mathbf{y}\mathbf{y}^T >_F}} \qquad (2)$$

where $< \bullet, \bullet >_F$ denotes the Frobenius inner-product between matrices. $\mathbf{K}_p$ is the p-posterior-mixture-model kernel matrix.

The optimization problem in (2) is easily resolved by exhaustive search within a certain interval of p (*e.g.* $p \in (0, 3]$ in our later experiments). Because the target function is extremely easy to evaluate and empirically quite smooth, and the search space is only one dimensional, even the exhaustive search is fast and scales linearly with respect to the interval considered.

3. Experiments

3.1. Synthetic data

To empirically validate our analysis in previous sections, we simulate three different multiple instance learning datasets endowed with different instance-to-bag-label *mechanisms*.

MIL dataset 1 is synthesized as follows: 1) randomly generate isotropic-covariance Gaussian mixture models in $\mathbb{R}^D$ with K equally weighted components, from which $N \times S$ instances are sampled; 2) one mixture component is randomly chosen, and the instances generated by that component are labeled as positive; 3) all $N \times S$ instances are randomly put into N bags, with S instances in each; 4) each bag is labeled as positive

iff there is at least one positive instance in it.

MIL dataset 2 and 3 are synthesized similarly. But the instances generated by $\frac{K}{5}$ mixture components are labeled as positive in MIL dataset 2, and each bag is labeled as positive *iff* positive instances in the bag exceed 20%. The instances generated by $\frac{K}{2}$ mixture components are labeled as positive in MIL dataset 3, and each bag is labeled as positive *iff* positive instances in the bag exceed 50%.

Although the synthetic datasets are endowed with different instance-to-bag-label *mechanisms*, all other aspects of these datasets are the same, which can *not* be exploited by the algorithm to distinguish these datasets. They all have approximately the same ratio of positive and negative bags if K and S are properly chosen. Although these tasks have different ratios of positive and negative instances, the instance labels are kept from the learning algorithm, which only observe 50%-50% bag-labels. Our approach is expected to discover the *mechanism* difference among these datasets in such a challenging setting. Moreover, in the first learning phase, the number of mixture components is deliberately set to be different from K, in order to simulate the fact that the characteristics of the underlying true generative model are usually unknown.

We repeated this experiment for many different choices of the bag size S, mixture model size K, instance space dimensionality D, and we observed that the optimal p value is almost always the smallest for MIL dataset 1, intermediate for MIL dataset 2, and the largest for MIL dataset 3. In Figure 1 we plotted the kernel alignment as a function of p in a typical run of the experiment with $K = 20$, $D = 5$, $S = 13$, and total number of bags $N = 200$. Note that this specific setting results in approximately the same number of positive bags and negative bags in all datasets.

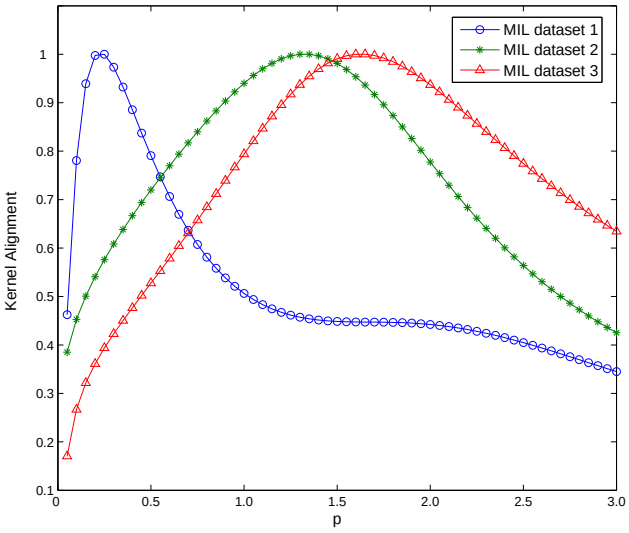


Figure 1. p versus kernel alignment in synthetic MIL dataset 1, 2, and 3. Kernel alignment values are normalized by its maximum value in either dataset. The optimal p values in these datasets justified our previous analysis.

3.2. MIL benchmark datasets

We tested our method on standard MIL benchmark datasets[5] (Andrews, 2003), which consist of MIL tasks in various application domains including drug activity prediction, image classification, and text classification.

3.2.1. Drug activity prediction

The concept of multiple instance learning had been originated from the application of drug activity prediction. In this application, the molecules are regarded as bags, and various shapes a molecule can adopt constitute instances within the bag. A molecule is considered a "positive" bag if it binds well to some target protein, which is true if at least one of its shapes (instances) binds well. The instances are represented as vectors describing that shape. In bio-chemical experiments, we can only observe whether a molecule binds well or not, but if the molecule binds well, we cannot further identify which shape(s) binds well and contributes to the positive bag-label.

Datasets of drug activity prediction for MIL are MUSK1 and MUSK2. The MUSK1 dataset consists of 47 positive bags, 45 negative bags, and totally 476 instances, each represented as a 166 dimensional vector. The MUSK2 dataset consists of 39 positive bags, 63 negative bags, and totally 6598 instances.

3.2.2. Image classification

Content-based image classification/retrieval is another application domain of multi-instance learning. Its major difficulty arises from the fact that an image consists of not only the object-of-interest, which determines its category label, but also background clutter, which may take up even a larger portion of the image. To segment the object-of-interest from background clutter is yet a challenging open problem in computer vision. A common strategy to perform classification without identifying the object-of-interest beforehand is to represent an image by many localized feature vectors instead of a single global feature description. Each localized feature is computed based on a small region of the image,

[5]The datasets used in this section are available online at http://www.cs.columbia.edu/ andrews/mil/datasets.html

so we could expect that the object-of-interest is captured by a number of local features, even if there are also other irrelevant local features arisen from background clutter. It is therefore quite natural to formalized content-based image classification/retrieval as a multi-instance learning problem, where images are the bags and local features are the instances.

The MIL benchmark dataset includes three image classification tasks—to discriminate images that contain *elephant*, *tiger*, and *fox* from irrelevant images, respectively. Each image (bag) is segmented into a set of regions (instances), and each region is represented as a 230 dimensional vector describing its color, texture, and shape characteristics. Each classification tasks has 100 positive bags and 100 negative bags.

3.2.3. Text classification

Another application domain of multiple instance learning is text classification/retrieval. A document can be divided into a number of segments, which could have different topic focuses. And the category label of the whole document (bag) should be decided by taking into account all these segments (instance), which constitutes a multiple instance learning problem.

The MIL benchmark dataset contains text data chosen from the OHSUMED (Hersh, 1994) dataset on medical issues. We perform two text classification tasks: TREC1 and TREC2. Each consists of 200 positive documents (bags) and 200 negative documents. Each document is segmented into overlapping 50-word-passages, which results in over 3000 passages (instances) in either of the tasks. Each passage is indexed by a sparse high-dimensional (over 60,000 index terms) feature vector.

3.2.4. Results

In order to make our results comparable to previous published results on these datasets, our experiments are conducted in the same way as in most previous works. For each classification task, we use 10-fold cross-validation. Classification accuracies are measured on the 10% hold-out data. Our method is compared with a number of existing multiple instance learning techniques. We replicated the results reported in their original papers for comparison if their results are measured similarly (using 10-fold cross-validation). Some results not available in their papers are marked as N/A (see Table 1).

For our approach (The PPMM Kernel), the only parameter that has to be manually set is the dimensionality K of aggregate posteriors (*i.e.* number of mixture components). We set $K = 30$ for drug activity prediction data and image data, and $K = 40$ for text data, since the instances in the text data have higher dimensionality and there are more labeled bags for training. Note that K is chosen subjectively but not carefully tuned for each task—tuning the parameter for each task could results in higher classification performance but may be impractical in real-world scenarios. Other implementation details of our approach in these experiments are the same as in Section 3.1, except that we adopt K-means clustering and histogram representations for these tasks, because they generally have high dimensional instance representation, and relatively small number of instances. Due to the local-optimal nature of K-means clustering, we tried multiple randomly seeded runs of algorithm, and chose the best one based on their performances on the training set.

One major advantage of our approach is the capacity to utilize a large number of unlabeled instances, but no extra unlabeled instance is available for the benchmark datasets, which implicates that the potential performance of our approach could possibly be underestimated in these experiments. Nevertheless, our approach performs generally better than or comparable with other MIL techniques (see Table 1). Since the benchmark datasets for MIL are rather small (number of bags ranges from tens to hundreds), slight differences in classification accuracy should not be overly emphasized. Instead, the most encouraging result we obtained is the optimal p values learned in these datasets. Note that the p values for drug activity prediction tasks (MUSK1 and MUSK2) are generally smaller than that for image classification tasks (ELEPHANT, TIGER, and FOX). Although the p value learned in the FOX dataset is smaller than other image datasets, we can further observe that all methods perform unsatisfactorily on the FOX dataset, which may indicate that this classification task itself could be impractical, hence the learned p value may be unreliable. Interestingly, comparing Table 1 and Figure 1 we could observe that the p values learned in real-world drug activity prediction tasks are close to that learned in synthetic task 1, and the p values learned in real-world image classification tasks are close to that learned in task 2 and task 3. We can also observe that the p values for text classification tasks (TREC1 and TREC2) are also small; possibly this is because the instance representation in the text domain are also high-level, and serving as *strong* evidences for bag-labels. In contrast, instance representation in the image domain are generally low-level (*e.g.* color, texture, shape), and they can only be considered as *weak* evidences for the

Table 1. Empirical results of multiple instance learning methods, the last row shows the optimal p value learned in each task. MUSK1 and MUSK2 are drug activity prediction datasets. ELEPHANT, TIGER, FOX are image classification datasets. TREC1 and TREC2 are text classification datasets. Best performance in each task is in bold. The average performance over all tasks is shown in the last column.

DATESETS:	MUSK1	MUSK2	ELEPHANT	TIGER	FOX	TREC1	TREC2	**Average**
APR (DIETTERICH, 1997)	92.4%	89.2%	N/A	N/A	N/A	N/A	N/A	N/A
DD (MARON, 1998)	88.0%	84.0%	N/A	N/A	N/A	N/A	N/A	N/A
EM-DD (ZHANG, 2001)	84.8%	84.9%	78.3%	72.1%	56.1%	85.8%	84.0%	78.0%
CITATION k-NN (WANG, J., 2000)	91.3%	86.0%	80.5%	78.0%	60.0%	87.0%	81.0%	80.5%
mi-SVM (ANDREWS, 2003)	87.4%	83.6%	82.0%	78.9%	58.2%	93.6%	78.2%	80.3%
MI-SVM (ANDREWS, 2003)	77.9%	84.3%	81.4%	**84.0%**	59.4%	**93.9%**	**84.5%**	80.8%
MISS-SVM (ZHOU, 2007)	87.6%	80.0%	N/A	N/A	N/A	N/A	N/A	N/A
MG-ACC KERNEL (KWOK, 2007)	90.1%	**90.4%**	N/A	N/A	N/A	N/A	N/A	N/A
PPMM KERNEL (this paper)	**95.6%**	81.2%	**82.4%**	80.2%	**60.3%**	93.3%	79.5%	**81.8%**
OPTIMAL VALUE OF p	0.7	0.15	2.1	1.3	0.8	0.75	0.4	

high-level bag-labels (*i.e.* elephant, tiger, fox).

4. Related Work

The concept of multiple instance learning was originally proposed in (Dietterich, 1997) for the application of drug activity prediction. The author assumes that positive instances all reside in an axis-parallel rectangle (APR), which implicates specific constraints that the shape should satisfy in order to bind well to some target protein. Although this assumption can be appropriate for this specific application, it is not clear how to adapt it to other applications, which may have more complex intrinsic structures in the instance space, and different instance-to-bag-label *mechanisms*.

Diverse Density (DD) (Maron, 1998) is another general framework for MIL. The author assumes that positive instances form a Gaussian-like pattern around some "concept point" in the instance space, which is expected to be close to at least one point in each positive bag and far away from all instances in negative bags. This assumption on the structure of instance space could also be over-simplified for some applications. And the algorithm, by definition, relies on the instance-to-bag-label *mechanism* in the original definition of multiple instance learning.

Citation k-NN adapts the memory based classification method k-NN to MIL, which considers not only the *references*, but also the *citers* as neighbors of a bag in determine its label, in order to be less affected by the negative instances in positive bags. It had been empirically proved to be more robust than standard k-NN. Nevertheless, the role of instance-to-bag-label *mechanism* is not clear in this framework.

Support vector machines (SVM) and the kernel trick (Schölkopf, 2001) have been very successful in traditional supervised learning. There also have been many attempts to apply them to MIL. These works falls into two major categories as summarized in (Kwok, 2007). The first family of methods try to modify the optimization problem of SVM, such as MI-SVM and mi-SVM (Andrews, 2003), which may result in non-convex optimization problems and suffer from local minima. The second family of methods design kernel functions on the bags, including (Gartner, 2002) and (Kwok, 2007). Our approach also falls into the second category, but it possesses a unique characteristic as *adapts* to various application domains with different instance-to-bag-label *mechanisms*.

The aggregate posteriors are essentially positive constant-sum real vectors, which reside in a simplex. Data in a simplex had been addressed from the metric learning perspective (Lebanon, 2003; Wang, H.-Y., 2007), which are related to our approach because the kernel defined for aggregate posteriors also gives rise to a distance metric on the simplex.

The idea of defining a kernel function based on posterior probabilities on mixture models had also been exploited in (Hertz, 2006). The author proposed the *KernelBoost* algorithm for learning with a large number of unlabeled data and few labeled data, in which the weak kernel mappings are defined as posterior probabilities on mixture models.

The two-phase learning scheme in our approach makes use of both unlabeled instances and labeled bags. It

is therefore conceptually related to semi-supervised learning (Zhu, 2005) and self-taught learning (Raina, 2007).

5. Conclusion

In this paper, we proposed a novel framework for *adapting* multiple instance learning to different *mechanisms* of how the instances determine the bag-labels. We showed that this *mechanism* is different in different application domains of multiple instance learning, and our approach well captures this domain-specific *mechanism* through learning with unlabeled instances and labeled bags.

To the best of our knowledge, this paper is the first work that addresses the problem of *adapting* multiple instance learning to different application domains with different instance-to-bag-label *mechanisms*. The major advantage of such a self-adaptive framework lies in that, if we are encountered with some novel application domain, which could be well formalized as multiple instance learning, but we have no *a priori* knowledge about the instance-to-bag-label *mechanisms* in that domain, we can learn the *mechanisms* from labeled bags, and design a kernel function *adapted* to this *mechanism*.

Acknowledgement

This work was supported in part by NKBRPC No. 2004CB318000, NHTRDP 863 Grant No. 2006AA01Z302, and No. 2007AA01Z336. Qiang Yang thanks Hong Kong CERG grants 621307 and and CAG grant HKBU1/05C.

Hua-Yan Wang would like to thank Haiyan Sun for her encouragements and insightful comments.

References

Andrews, S., Tsochantaridis, I., and Hofmann, T. (2003). Support vector machines for multiple-instance learning. In *NIPS* 15

Bunescu, R.C., Mooney, R.J. (2007) Multiple Instance Learning for Sparse Positive Bags. In *Proceedings of the 24th ICML*

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. (2002). On kernel-target alignment. In *NIPS* 14

Dietterich, T.G., Lathrop, R.H. and Lozano-Perez, T. (1997). Solving the multiple instance problem with axisparallel rectangles. *Artificial Intelligence*, 89, 31-71.

Gartner, T., Flach, P., Kowalczyk, A., and Smola, A. (2002). Multi-instance kernels. IIn *Proceedings of the 19th ICML*

Hersh, W., Buckley, C., Leone, T.J., Hickam, D. (1994). OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*

Hertz, T., Hillel, A.B., Weinshall, D. (2006). Learning a Kernel Function for Classification with Small Training Samples. In *Proceedings of the 23rd ICML*

Kwok, J.T. and Cheung, Pak-Ming. (2007). Marginalized Multi-Instance Kernels. In *IJCAI'07*

Lebanon, G. (2003). "Learning Riemannian metrics", In *Proceedings of the 19th UAI*

Maron, O., Lozano-Pérez, T. (1998). A Framework for Multiple-Instance Learning. In *NIPS* 10

Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y. (2007). Self-taught learning: Transfer Learning from Unlabeled Data. In *Proceedings of the 24st ICML*

Schölkopf, B., Smola, A.J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press

Wang, H.-Y., Zha, H., Qin, H. (2007). "Dirichlet aggregation: unsupervised learning towards an optimal metric for proportional data", In *Proceedings of the 24th ICML*

Wang, J. and Zucker, J.-D. (2000). Solving the multiple-instance problem: a lazy learning approach. *Proceedings of the 17th ICML*

Zhang, Q., Goldman, S.A. (2001). EM-DD: An improved multiple instance learning technique. In *NIPS* 13

Zhang, Q., Goldman, S.A., Yu, W. and Fritts, J. (2002). Content-based image retrieval using multiple-instance learning. In *Proceedings of the 19th ICML*

Zhou, Z.-H., and Xu, J.-M. (2007). On the Relation Between Multi-Instance Learning and Semi-Supervised Learning. In *Proceedings of the 24th ICML*

Zhu, X. (2005). Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin, Madison

Graph Transduction via Alternating Minimization

Jun Wang JWANG@EE.COLUMBIA.EDU
Department of Electrical Engineering, Columbia University

Tony Jebara JEBARA@CS.COLUMBIA.EDU
Department of Computer Science, Columbia University

Shih-Fu Chang SFCHANG@EE.COLUMBIA.EDU
Department of Electrical Engineering, Columbia University

Abstract

Graph transduction methods label input data by learning a classification function that is regularized to exhibit smoothness along a graph over labeled and unlabeled samples. In practice, these algorithms are sensitive to the initial set of labels provided by the user. For instance, classification accuracy drops if the training set contains weak labels, if imbalances exist across label classes or if the labeled portion of the data is not chosen at random. This paper introduces a propagation algorithm that more reliably minimizes a cost function over both a function on the graph and a binary label matrix. The cost function generalizes prior work in graph transduction and also introduces node normalization terms for resilience to label imbalances. We demonstrate that global minimization of the function is intractable but instead provide an alternating minimization scheme that incrementally adjusts the function and the labels towards a reliable local minimum. Unlike prior methods, the resulting propagation of labels does not prematurely commit to an erroneous labeling and obtains more consistent labels. Experiments are shown for synthetic and real classification tasks including digit and text recognition. A substantial improvement in accuracy compared to state of the art semi-supervised methods is achieved. The advantage are even more dramatic when labeled instances are limited.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

1. Introduction

Graph transduction refers to a family of algorithms that achieve state of the art performance in semi-supervised learning and classification. These methods incur a tradeoff between a classification function's accuracy on labeled examples and a regularizer term that encourages the function to remain smooth over a weighted graph connecting the data samples. The weighted graph and the minimized function ultimately propagate label information from labeled data to unlabeled data to provide the desired transductive predictions. Popular algorithms for graph transduction include the Gaussian fields and harmonic functions based method (*GFHF*) (Zhu et al., 2003) as well as the local and global consistency method (*LGC*) (Zhou et al., 2004). Other closely related methods include the manifold regularization framework proposed in (Sindhwani et al., 2005; Belkin et al., 2006) where graph Laplacian regularization terms are combined with regularized least squares (*RLS*) or support vector machine (*SVM*) function estimation criteria. These methods lead to graph-regularized variants denoted as Laplacian *RLS* (*LapRLS*) and Laplacian *SVM* (*LapSVM*) respectively. For certain synthetic and real data problems, graph transduction approaches do achieve promising performance. However, this article identifies several realistic settings and labeling situations where this performance can be compromised. An alternative algorithm which generalizes the previous techniques is proposed by defining a joint iterative optimization over the classification function *and* a balanced label matrix.

Even if one assumes the graph structures used in the above methods faithfully describe the data manifold, graph transduction algorithms may still be misled by problems in the label information. Figure 1 depicts several cases where the label information leads to invalid graph transduction solutions for all the aforemen-

tioned algorithms. The top row of Figure 1 shows a separable pair of manifolds where unbalanced label information affects the propagation results. Although a clear separation region is visible between the two manifolds, the imbalance in the labels misleads the previous algorithms which prefer assigning points to the class with the majority of labels. In the bottom row of Figure 1, a non-separable problem is shown where two otherwise separable manifolds are peppered with noisy outlier samples. Here, the outliers do not belong to either class but once again interfere with the propagation of label information. In both situations, conventional transductive learning approaches such as *GFHF*, *LGC*, *LapRLS*, and *LapSVM* fail to give acceptable labeling results.

In order to handle such situations, we extend the graph transduction optimization problem by casting it as a *joint* optimization over the classification function *and* the labels. The optimization is solved iteratively and remedies the instability previous methods seem to have vis-a-vis the initial labeling. In our novel framework, initial labels simply act as the starting value of the label matrix variable which is incrementally refined until convergence. The overall minimization over the continuous classification function and the binary label matrix proceeds by an alternating minimization over each term separately and converges to a local minimum. Moreover, to handle the imbalanced labels issue, a node regularizer term is introduced to balance the label matrix among different classes. These two fundamental changes to the graph transduction problem produce significantly better performance on both artificial and real datasets.

The remainder of this paper is organized as the follows. In Section 2, we revisit the graph regularization framework of (Zhou et al., 2004) and extend it into a bivariate graph optimization problem. A corresponding algorithm is provided that solves the new optimization problem by iterative alternating minimization. Section 3 provides experimental validation for the algorithm on both toy and real classification datasets, including text classification and digital recognition. Comparisons with leading semi-supervised methods are made. Concluding remarks and a discussion are then provided in Section 4.

2. Graph Transduction

Consider the dataset $\mathcal{X} = (\mathcal{X}_l, \mathcal{X}_u)$ of labeled inputs $\mathcal{X}_l = \{\mathbf{x}_1, \cdots, \mathbf{x}_l\}$ and unlabeled inputs $\mathcal{X}_u = \{\mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$ along with a small portion of corresponding labels $\{y_1, \cdots, y_l\}$, where $y_i \in \mathcal{L} = \{1, \cdots, c\}$. For transductive learning, the objective is to infer the labels $\{y_{l+1}, \cdots, y_n\}$ of the unlabeled data $\{\mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$, where typically $l << n$. The graph transduction methods define an undirected graph represented by $\mathcal{G} = \{\mathcal{X}, \mathcal{E}\}$, where the set of node or vertices is $\mathcal{X} = \{\mathbf{x}_i\}$ and the set of edges is $\mathcal{E} = \{e_{ij}\}$. Each sample $\mathbf{x}_i$ is treated as the node on the graph and the weight of edge e_{ij} is w_{ij}. Typically, one uses a kernel function $k(\cdot)$ over pairs of points to recover weights, in other words $w_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ with the RBF kernel being a popular choice. The weights for edges are used to build a weight matrix which is denoted by $\mathbf{W} = \{w_{ij}\}$. Similarly, the node degree matrix $\mathbf{D} = diag([d_1, \cdots, d_n])$ is defined as $d_i = \sum_{j=1}^{n} w_{ij}$. The binary label matrix $\mathbf{Y}$ is described as $Y \in \mathcal{B}^{n \times c}$ with $\mathbf{Y}_{ij} = 1$ if $\mathbf{x}_i$ has label $y_i = j$ and $\mathbf{Y}_{ij} = 0$ otherwise. This article will often refer to row and column vectors of such matrices, for instance, the ith row and jth column vectors of $\mathbf{Y}$ are denoted as $Y_{i\cdot}$ and $Y_{\cdot j}$, respectively. The graph Laplacian is defined as $\mathbf{\Delta} = \mathbf{D} - \mathbf{W}$ and the normalized graph Laplacian is $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{\Delta}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$.

2.1. Consistent Label Propagation

Graph based semi-supervised learning methods propagate label information from labeled nodes to unlabeled nodes by treating all samples as nodes in a graph and using edge-based affinity functions between all pairs of nodes to estimate the weight of each edge. Most methods then define a continuous classification function $F \in \mathcal{R}^{n \times c}$ that is estimated on the graph to minimize a cost function. The cost function typically enforces a tradeoff between the smoothness of the function on the graph of both labeled and unlabeled data and the accuracy of the function at fitting the label information for the labeled nodes. Such is the case for a large variety of graph based semi-supervised learning techniques ranging from the the mincuts method (Blum & Chawla, 2001), the Gaussian fields and harmonic functions (*GFHF*) method, and the local and global consistency (*LGC*) method. A detailed survey of these methods is available in (Zhu, 2005).

In trading off smoothness for accuracy, both *GFHF* and *LGC* approaches attempt to preserve consistency on the data manifold during the optimization of the classification function. The loss function for both methods involves the additive contribution of two penalty terms the global smoothness Q_{smooth} and local fitness Q_{fit} as shown below:

$$\mathbf{F}^* = \arg\min_{\mathbf{F}} \mathcal{Q}(\mathbf{F}) = \arg\min_{\mathbf{F}} \{Q_{smooth}(\mathbf{F}) + Q_{fit}(\mathbf{F})\} \quad (1)$$

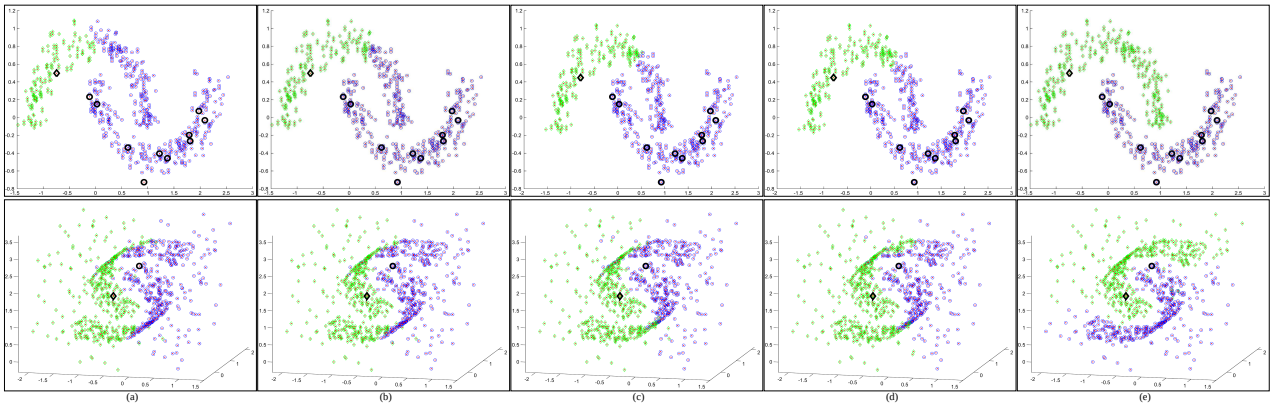


Figure 1. A demonstration with artificial data of the sensitivity graph transduction exhibits for certain initial label settings. The top row shows how imbalanced labels adversely affect even a well-separated 2D two-moon dataset. The bottom row shows a 3D two-moon data where graph transduction is again easily misled by the introduction of a cloud of outliers. Large markers indicate known labels and the two-color small markers represent the predicted classification results. Columns depict the results from (a) the *GFHF* method (Zhu et al., 2003); (b) the *LGC* method (Zhou et al., 2004); (c) the *LapRLS* method (Belkin et al., 2006); (d) the *LapSVM* method (Belkin et al., 2006); and (e) Our method (*GTAM*).

In particular, recall that *LGC* uses an elastic regularizer framework with the following cost function (Zhou et al., 2004).

$$\mathcal{Q}(\mathbf{F}) = \frac{1}{2}\left(\sum_{i,j=1}^{n} w_{ij}\left\|\frac{\mathbf{F}_{i\cdot}}{\sqrt{\mathbf{D}_{ii}}} - \frac{\mathbf{F}_{j\cdot}}{\sqrt{\mathbf{D}_{jj}}}\right\|^2 + \mu\sum_{i=1}^{n}\|\mathbf{F}_{i\cdot} - \mathbf{Y}_{i\cdot}\|^2\right) \tag{2}$$

where the coefficient μ balances global smoothness and local fitting penalty terms. If we set $\mu = \infty$ and use a standard graph Laplacian for the smoothness term, the above framework reduces to the harmonic function formulation as shown in (Zhu et al., 2003).

While *LGC* and *GFHF* formulations remain popular and have been empirically validated in the past, it is possible to discern some key limitations. First, the optimization can be broken up into a separate parallel problems since the cost function decomposes into terms that only depend on individual columns of the matrix $\mathbf{F}$. Because each column of $\mathbf{F}$ indexes the labeling of a single class, such a decomposition reveals that biases may arise if the input labels are disproportionately imbalanced. In practice, both propagation algorithms tend to prefer predicting the class with the majority of labels. Second, both learning algorithms are extremely dependent on the initial labels provided in $\mathbf{Y}$. This is seen in practice but can also be explained mathematically by fact that $\mathbf{Y}$ is starts off extremely sparse and has many unknown terms. Third, when the graph contains background noise and makes class manifolds nonseparable, these graph transduction approaches fail to output reasonable classification results. These difficulties were illustrated in Figure 1 and seem to plague many graph transduction approaches. However, the proposed method, graph transduction via alternating minimization (*GTAM*) appears resilient.

To address these problems, we will make modifications to the cost function in Eq. 1. The first one is to explicitly show the optimization over both the classification function $\mathbf{F}$ and the binary label matrix $\mathbf{Y}$:

$$(\mathbf{F}^*, \mathbf{Y}^*) = \arg\min_{\mathbf{F}\in\mathcal{R}^{n\times c}, \mathbf{Y}\in\mathcal{B}^{n\times c}} \mathcal{Q}(\mathbf{F}, \mathbf{Y}). \tag{3}$$

Where $\mathcal{B}^{n\times c}$ is the set of all binary matrices $\mathbf{Y}$ of size $n \times c$ that satisfy $\sum_j \mathbf{Y}_{ij} = 1$ and, for the labeled data $\mathbf{x}_i \in \mathcal{X}_l$, $\mathbf{Y}_{ij} = 1$ if $y_i = j$. More specifically, our loss function is:

$$\mathcal{Q}(\mathbf{F}, \mathbf{Y}) = \frac{1}{2}\mathrm{tr}\left\{\mathbf{F}^T\mathbf{L}\mathbf{F} + \mu(\mathbf{F} - \mathbf{V}\mathbf{Y})^T(\mathbf{F} - \mathbf{V}\mathbf{Y})\right\} \tag{4}$$

where we have introduced the matrix $\mathbf{V}$ which is a node regularizer to balance the influence of labels from different classes. The matrix $\mathbf{V} = diag(\mathbf{v})$ is a function of the current label matrix $\mathbf{Y}$:

$$\mathbf{v} = \sum_{j=1}^{c} \frac{\mathbf{Y}_{\cdot j} \odot \mathbf{D}\vec{\mathbf{1}}}{\mathbf{Y}_{\cdot j}^T\mathbf{D}\vec{\mathbf{1}}} \tag{5}$$

where the symbol $\odot$ denotes the Hadamard product and column vector $\vec{\mathbf{1}}$ represents $\vec{\mathbf{1}} = [1\cdots 1]^T$. This node regularizer permits us to work with a normalized version of the label matrix $\mathbf{Z}$ defined as: $\mathbf{Z} = \mathbf{V}\mathbf{Y}$.

By definition, we see that the normalized label matrix satisfies $\sum_i \mathbf{Z}_{ij} = 1$. Using the normalized label matrix $\mathbf{Z}$ in a graph regularization allows labeled nodes with high degree to contribute more during the graph diffusion and label propagation process. However, the

total diffusion of each class is kept equal and normalized to be one. Therefore, the influence of different classes is balanced even if the given class labels are imbalanced. If class proportion information is known a priori, it can be integrated by scaling the diffusion with the prior class proportion. However, because of the nature of graph transduction and unknown class prior knowledge, equal class balancing leads to generally more reliable solutions than label proportional weighting. This intuition is in line with prior work that uses class proportion information in transductive inference such as (Chapelle et al., 2007) where class proportion is enforced as a hard constraint on the labels or in (Mann & McCallum, 2007) where such information is used as a regularizer. We next discuss the alternating minimization procedure which is the key modification to the overall framework.

2.2. Alternating Minimization Procedure

In our proposed graph regularization framework, the cost function involves two variables to be optimized. While simultaneously recovering both solutions is intractable due to the mixed integer programming problem over binary $\mathbf{Y}$ and continuous $\mathbf{F}$, we will propose a greedy alternating minimization approach. The first update of the continuous classification function $\mathbf{F}$ is straightforward since the resulting cost function is convex and unconstrained allowing us to recover the optimal $\mathbf{F}$ by setting the partial derivative $\frac{\partial \mathcal{Q}}{\partial \mathbf{F}}$ to be zero. However, since $\mathbf{Y} \in \mathcal{B}$ is a binary matrix and subject to linear constraints of the form $\sum_j \mathbf{Y}_{ij} = 1$, the other step in our alternating minimization requires solving a linearly constrained max cut problem which is NP (Karp, 1972). Due to the alternating minimization outer loop, investigating guaranteed approximation schemes (Goemans & Williamson, 1995) to solve a constrained max cut problem for $\mathbf{Y}$ is unjustified due to the solution's dependence on the dynamically varying classification function $\mathbf{F}$ during the alternating minimization procedure. Instead, we use a greedy gradient based approach to incrementally update $\mathbf{Y}$, while keeping the classification function $\mathbf{F}$ at the corresponding optimal setting. Moreover, because the node regularizer term $\mathbf{V}$ normalizes the labeled data, we also interleave updates of $\mathbf{V}$ based on the revised $\mathbf{Y}$.

Minimization for F :
The classification function $\mathbf{F} \in \mathcal{R}^{n \times c}$ is continuous and its loss terms are convex allowing the minimum to be recovered by zeroing the partial derivative:

$$\frac{\partial \mathcal{Q}}{\partial \mathbf{F}^*} = 0 \implies \mathbf{L}\mathbf{F}^* + \mu(\mathbf{F}^* - \mathbf{V}\mathbf{Y}) = 0$$
$$\implies \mathbf{F}^* = (\mathbf{L}/\mu + \mathbf{I})^{-1}\mathbf{V}\mathbf{Y} = \mathbf{P}\mathbf{V}\mathbf{Y} \quad (6)$$

where we denote $\mathbf{P} = (\mathbf{L}/\mu + \mathbf{I})^{-1}$ as the propagation matrix and assume the graph is symmetrically built (i.e. $\mathbf{L} = \mathbf{L}^T$).

Greedy minimization of Y:
To update $\mathbf{Y}$, first replace $\mathbf{F}$ in Eq. 4 by its optimal vlue $\mathbf{F}^*$ from the solution of Eq. 6.

$$\begin{aligned}\mathcal{Q}(\mathbf{Y}) =& \frac{1}{2}\mathrm{tr}(\mathbf{Y}^T\mathbf{V}^T\mathbf{P}^T\mathbf{L}\mathbf{P}\mathbf{V}\mathbf{Y} \\ &+\mu(\mathbf{P}\mathbf{V}\mathbf{Y} - \mathbf{V}\mathbf{Y})^T(\mathbf{P}\mathbf{V}\mathbf{Y} - \mathbf{V}\mathbf{Y})) \\ =& \frac{1}{2}\mathrm{tr}\left(\mathbf{Y}^T\mathbf{V}^T\left[\mathbf{P}^T\mathbf{L}\mathbf{P} + \mu(\mathbf{P}^T - \mathbf{I})(\mathbf{P} - \mathbf{I})\right]\mathbf{V}\mathbf{Y}\right)\end{aligned} \quad (7)$$

The optimization still involves the node regularizer $\mathbf{V}$ in Eq. 5, which depends on $\mathbf{Y}$ and normalizes the label matrix over columns. Due to the dependence on the current estimate of $\mathbf{F}$ and $\mathbf{V}$, only an incremental step will be taken greedily to reduce $\mathcal{Q}(\mathbf{Y})$. In each iteration, we find position (i^*, j^*) in the matrix $\mathbf{Y}$ and change the binary value of $\mathbf{Y}_{i^*j*}$ from 0 to 1. The direction with largest negative gradient guides our choice of binary step on $\mathbf{Y}$. Therefore, we need to evaluate $\| \bigtriangledown \mathcal{Q}_{\mathbf{Y}} \|$ and find the largest negative value to determine (i^*, j^*).

Note that setting $\mathbf{Y}_{i^*,j^*} = 1$ is equivalent to a similar operation on the normalized label matrix $\mathbf{Z}$ by setting $\mathbf{Z}_{i^*,j^*} = \epsilon, 0 < \epsilon < 1$, and $\mathbf{Y}, \mathbf{Z}$ have one to one correspondence. Thus, the greedy optimization of $\mathcal{Q}$ with respect to $\mathbf{Y}$ is equivalent to greedy minimization of $\mathcal{Q}$ with respect to $\mathbf{Z}$. More formally: $\frac{\partial \mathcal{Q}}{\partial \mathbf{Y}} = \frac{\partial \mathcal{Q}}{\partial \mathbf{Z}}\frac{\partial \mathbf{Z}}{\partial \mathbf{Y}}$ and with straightforward algebra we see that:

$$(i^*, j^*) = \arg\min_{i,j} \frac{\partial \mathcal{Q}}{\partial \mathbf{Y}} = \arg\min_{i,j} \frac{\partial \mathcal{Q}}{\partial \mathbf{Z}} \quad (8)$$

Then we can rewrite the loss function using the variable $\mathbf{Z}$ as:

$$\begin{aligned}\mathcal{Q}(\mathbf{Z}) &= \frac{1}{2}\mathrm{tr}\left(\mathbf{Z}^T\left[\mathbf{P}^T\mathbf{L}\mathbf{P} + (\mathbf{P}^T - \mathbf{I})(\mathbf{P} - \mathbf{I})\right]\mathbf{Z}\right) \\ &= \frac{1}{2}\mathrm{tr}\left(\mathbf{Z}^T\mathbf{A}\mathbf{Z}\right)\end{aligned} \quad (9)$$

where $\mathbf{A}$ represents $\mathbf{A} = \mathbf{P}^T\mathbf{L}\mathbf{P} + (\mathbf{P}^T - \mathbf{I})(\mathbf{P} - \mathbf{I})$. Notice that $\mathbf{A}$ is symmetric if the graph is symmetrically built. We derive the gradient of the above loss function and recover it with respect to $\mathbf{Y}$ as:

$$\frac{\partial \mathcal{Q}}{\partial \mathbf{Z}} = \mathbf{A}\mathbf{Z} = \mathbf{A}\mathbf{V}\mathbf{Y} \quad (10)$$

As described earlier, we search the gradient matrix $\nabla_{\mathbf{Z}}\mathcal{Q}$ to find the minimal element for updating

$$(i^*, j^*) = \arg\min_{\mathbf{x}_i \in \mathcal{X}_u, 1 \le j \le c} \nabla_{\mathbf{Z}_{ij}}\mathcal{Q} \quad (11)$$

Then update the label matrix by setting $\mathbf{Y}_{i^*j^*} = 1$. Because of the binary nature of $\mathbf{Y}$, we simply set $\mathbf{Y}_{i^*j^*} = 1$ instead of using a continuous gradient approach. In the $t+1$th iteration, the node regularizer $\mathbf{v}^{t+1}$ can be recalculated with the updated $\mathbf{Y}^{t+1}$.

The update $\mathbf{Y}$ is indeed greedy. Therefore, it could oscillate and backtrack from predicted labelings in previous iterations without convergence guarantees. We propose a straightforward way to guarantee convergence and avoid backtracking, inconsistency or unstable oscillation in the greedy propagation of labels. Once an unlabeled point has been labeled, its labeling can no longer be changed. Thus, we remove the most recently labeled point (i^*, j^*) from future consideration and only permit the algorithm to search for the minimal gradient entries corresponding to the remaining unlabeled examples. Thus, to avoid changing the labeling of previous predictions, the new labeled node $\mathbf{x}_{i^*}$ will be removed from $\mathcal{X}_u$ and added to $\mathcal{X}_l$.

In the following, we summarize the updating rules from step t to $t+1$ in the alternative minimization scheme. Although the optimal $\mathbf{F}^*$ is being computed in each iteration as shown in Eq. 6, we do not explicitly need to update it. Instead, it is implicitly used in Eq. 8 to directly update $\mathbf{Y}$.

$$\begin{aligned}
\nabla_{\mathbf{Z}}\mathcal{Q}^t &= \mathbf{A}diag(\mathbf{v}^t)\mathbf{Y}^t \qquad (12)\\
(i^*, j^*) &= \arg\min_{\mathbf{x}_i \in \mathcal{X}_u, 1 \le j \le c} \nabla_{\mathbf{Z}_{ij}}\mathcal{Q}^t \\
\mathbf{Y}^{t+1}_{i^*j^*} &= 1 \\
\mathbf{v}^{t+1} &= \sum_{j=1}^{c} \frac{\mathbf{Y}^{t+1}_{\cdot j} \odot \mathbf{D}\vec{\mathbf{1}}}{{\mathbf{Y}^{t+1}_{\cdot j}}^T \mathbf{D}\vec{\mathbf{1}}} \\
\mathcal{X}_l^{t+1} &\longleftarrow \mathcal{X}_l^t + \mathbf{x}_{i^*} \;\; ; \;\; \mathcal{X}_u^{t+1} \longleftarrow \mathcal{X}_u^t - \mathbf{x}_{i^*}
\end{aligned}$$

The procedure above repeats until all points have been labeled.

2.3. Algorithm Summary and Convergence

From the above discussion, our method is unique in that it optimizes the loss function over both continuous-valued $\mathbf{F}$ space and binary-valued $\mathbf{Y}$ space. Starting from a few given labels, the method iteratively and greedily updates the label matrix $\mathbf{Y}$, node regularizer $\mathbf{v}$, and gradient matrix $\nabla_{\mathbf{Z}}\mathcal{Q}$. In each individual iteration, new labeled samples are obtained to drive a better graph propagation in the next iteration. In our approach, we directly acquire new labels instead of calculating $\mathbf{F}^*$ and then conducting a mapping to $\mathbf{Y}$, which is the regular procedure in other graph transduction methods like *LGC* and *GFHF*. This unique feature makes the proposed algorithm very efficient since we only update the gradient matrix $\nabla_{\mathbf{Z}}\mathcal{Q}$ in each iteration. Furthermore, similar to the graph superposition approach introduced in (Wang et al., 2008), the calculation of the node regularizer $\mathbf{v}$ and gradient matrix $\nabla_{\mathbf{Z}}\mathcal{Q}$ can be more efficient by incremental updating as a result of the newly gained labels.

Due to greedy assignment, the algorithm can only loop the alternative minimization (or the gradient computation equivalently) at most $n-l$ times. The update of the graph gradient, finding the largest element in the gradient and the matrix algebra involved can be done efficiently by modifying only a single entry in $\mathbf{Y}$ per loop. Each minimization step over $\mathbf{F}$ and $\mathbf{Y}$ thus requires $\mathcal{O}(n^2)$ time and the total runtime of the greedy *GTAM* algorithm is $\mathcal{O}(n^3)$. Empirically, the value of the loss function $\mathcal{Q}$ decreases rapidly in the the first dozen iterations and achieves steady convergence afterward. This phenomenon indicates that the label propagation loop could be early stopped by solving for the labels from the optimized $\mathbf{F}^*$ (Eq. 6) after only a few iterations. The above algorithm chart summarizes the proposed *GTAM* method.

3. Experiments

In this section, we demonstrate the superiority of the proposed *GTAM* method in comparison to state of the art semi-supervised learning methods over both synthetic and real data. For instance, on the WebKB data, previous work shows that *LapSVM* and *LapRLS* are better than other semi-supervised approaches, such as Transductive SVMs *TSVM* (Joachims, 1999) and ∇*TSVM*. Therefore, we only compare our method with *LapRLS*, *LapSVM* and two related methods, $LapRLS_{joint}$ and $LapSVM_{joint}$ (Sindhwani et al., 2005). In all experiments, we used the same parameter settings reported in the literature. The *GTAM* approach only requires a single μ parameter which controls the tradeoff between the global smoothness and local fitting terms in the cost function. Although our experiments show that *GTAM* is fairly robust to the setting of μ, we set $\mu = 99$ throughout all experiments.

For all real implementations of graph-based methods, one needs a construction method that builds a graph from the training data $\mathcal{X}$, which involves a procedure for computing the weight of links via a kernel or similarity function. Typically, practitioners use RBF kernels for image recognition and cosine distances for text classification (Zhou et al., 2004; Ng et al., 2001; Chapelle et al., 2003; Hein & Maier, 2006). However, finding adequate parameters for the kernel or similarity function, such as the RBF kernel size δ, is not always straightforward particularly if labeled data is scarce. Empirical evidence has shown that the prop-

Algorithm 1 Graph Transduction via Alternating Minimization (*GTAM*)

Input: data set $\mathcal{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_l, \mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$, labeled subset $\mathcal{X}_l = \{\mathbf{x}_1, \cdots, \mathbf{x}_l\}$, unlabeled subset $\mathcal{X}_u = \{\mathbf{x}_{l+1}, \cdots, \mathbf{x}_n\}$, labels $\{y_1, \cdots, y_j, \cdots, y_l\}$, where $y_j \in \mathcal{L} = \{1, \cdots, l\}$. Affinity matrix $\mathbf{W} = \{w_{ij}\}$, node degree matrix $\mathbf{D}$, initial label matrix $\mathbf{Y}^0$;

Initialization:

iteration counter $t = 0$;

normalized graph Laplacian $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{\Delta}\mathbf{D}^{-1/2}$;

propagation matrix $\mathbf{P} = (\mathbf{L}/\mu + \mathbf{I})^{-1}$;

matrix $\mathbf{A} = \mathbf{P}^T\mathbf{L}\mathbf{P} + (\mathbf{P}^T - \mathbf{I})(\mathbf{P} - \mathbf{I})$;

node regularizer $\mathbf{v}^0 = \sum_{j=1}^{c} \frac{\mathbf{Y}^0_{\cdot j} \odot \mathbf{D}\vec{\mathbf{1}}}{\mathbf{Y}^{0\,T}_{\cdot j}\mathbf{D}\vec{\mathbf{1}}}$.

repeat

Compute graph gradient:

$\mathbf{Z}^t = diag(\mathbf{v}^t)\mathbf{Y}^t$, $\nabla_{\mathbf{Z}^t}\mathcal{Q}^t = \mathbf{A}\mathbf{Z}^t$;

Find the optimal element in $\nabla_{\mathbf{Z}^t}\mathcal{Q}^t$:

$(i^*, j^*) = \arg\min_{\mathbf{x}_i \in \mathcal{X}_u, 1 \le j \le c} \nabla_{\mathbf{Z}_{ij}}\mathcal{Q}^t$;

Update label matrix to obtain $\mathbf{Y}^{t+1}$ by setting:

$\mathbf{Y}^{t+1}_{i^* j^*} = 1$; also $\mathbf{y}_{i^*} = j^*$;

Update node regularizer by:

$\mathbf{v}^{t+1} = \sum_{j=1}^{c} \frac{\mathbf{Y}^{t+1}_{\cdot j} \odot \mathbf{D}\vec{\mathbf{1}}}{\mathbf{Y}^{t+1\,T}_{\cdot j}\mathbf{D}\vec{\mathbf{1}}}$;

Remove $\mathbf{x}_{i^*}$ from $\mathcal{X}_u$: $\mathcal{X}_u^{t+1} \longleftarrow \mathcal{X}_u^t - \mathbf{x}_{i^*}$;

Add $\mathbf{x}_{i^*}$ to $\mathcal{X}_l$: $\mathcal{X}_l^{t+1} \longleftarrow \mathcal{X}_l^t + \mathbf{x}_{i^*}$;

Update iteration counter: $t = t + 1$;

until $\mathcal{X}_u^t = \emptyset$

Output:

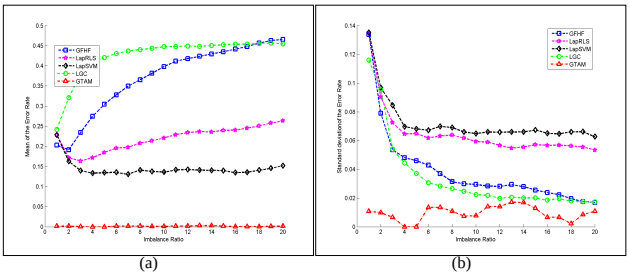
The labels of unlabeled samples $\{y_{l+1}, \cdots, y_n\}$.

agation results highly depend on the kernel parameter selection. Motivated by the approach reported in (Hein & Maier, 2006), we use an adaptive kernel size based on the mean distance of k-nearest neighborhoods ($k = 6$) for the experiments on real USPS handwritten digit data. On the WebKB data, we use the same graph construction suggested by (Sindhwani et al., 2005). For each dataset, the same graph is used for all the compared transductive learning approaches.

3.1. Two Moon Synthetic Data

Figure 1 illustrated synthetic experiments on $2D$ and 3D two-moon data. Despite the near-perfect classification results reported on such datasets in the literature (Zhou et al., 2004), we showed how small perturbations to the problem can have adverse effects on prior algorithms. The prior methods are overly sensitive to locations of the initial labels, ratios of the two-class labels, and the level of ambient noise or outliers.

A more thorough experimental study is also possible



(a) (b)

Figure 2. Performance comparison of *LGC*, *GFHF*, *LapRLS*, *LapSVM*, and *GTAM* on noisy 3D two moon data. Only one label is given for one class, while the other class has a varying number of labels, shown as imbalance ratio on the horizontal axis: (a) The mean of the test error; (b) The standard deviation of the test error.

for the two-moon data by exploring the effect of class imbalance. We start by fixing one class to have one observed label and select r labels from the other class. Here, r is also the imbalance ratio and the range we explore is $1 \le r \le 20$. These experiments use the 3D noisy two-moon data which contain 300 positive and 300 negative sample points as well as 200 additional background noise samples. Multiple round tests (100 trails) are evaluated for each imbalance condition by calculating the average prediction accuracy on the relevant 600 samples. For a fair comparison, we use the same graph Laplacian, which is constructed using k-NN ($k = 6$) neighbors with RBF weights. Moreover, the parameter for *LGC* is set as $\alpha = 0.99$. The parameters for *LapRLS* and *LapSVM* are $\gamma_A = 1, \gamma_I = 1$.

Figure 2 demonstrates the performance advantage of the proposed *GTAM* approach versus the *LGC*, *GFHF*, *LapRLS*, and *LapSVM* methods. From the figure, we can conclude that all the four strawman approaches are extremely sensitive to the initial labels and label class imbalance since none of them can produce perfect accuracy and the error rates of *LGC* and *GFHF* are dramatically increased when the label class becomes more imbalanced *even though more information is being provided to the algorithm.* However, *GTAM* is clearly superior, achieving the best accuracy regardless of the imbalance ratio and despite contamination with noisy samples. In fact only 1 or 2 of the 100 trails for each individual setting of r were imperfect using the *GTAM* method.

3.2. WebKB Dataset

For validation on real data, we first evaluated our method using the WebKB dataset, which has been widely used in semi-supervised learning experiments (Joachims, 2003; Sindhwani et al., 2005). The WebKB dataset contains two document categories, *course* and

non-course. Each document has two types of information, the webpage text content called *page* representation and *link* or pointer representation. For fair comparison, we applied the same feature extraction procedure as discussed in (Sindhwani et al., 2005), obtained 1051 samples with 1840-dimensional *page* attributes and 3000 *link* attributes. The graph was built based on cosine-distance neighbors with Gaussian weights (number of nearest neighbors is 200 as in (Sindhwani et al., 2005)). We compared our method with four of the best known approaches, *LapRLS*, *LapSVM*, and the two problem specific methods, $LapRLS_{joint}$, $LapSVM_{joint}$ reported in (Sindhwani et al., 2005). All the compared approaches used the same graph construction procedure and all parameter settings were set according to (Sindhwani et al., 2005), in other words $\gamma_A = 10^{-6}, \gamma_I = 0.01$. We varied the number of labeled data to measure the performance gain with increasing supervision. For each fixed number of labeled samples, 100 random trails were tested. The means of the test errors are shown in Figure 3.

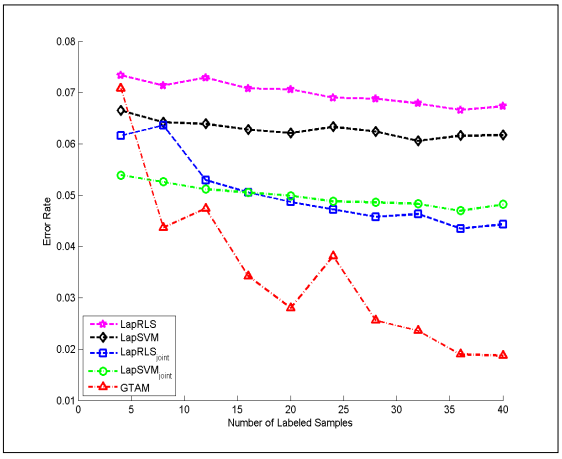


Figure 3. Performance comparison on text classification (WebKB dataset). The horizontal axis represents the number of randomly observed labels (guaranteeing there is at least one label for each class). The vertical axis shows the average error rate over 100 random trials.

As the Figure reveals, the proposed *GTAM* method achieved significantly better accuracy than all the other methods, except for the extreme case when only four labeled samples were available. The performance gain grows rapidly when the number of labeled samples increases, although in some cases the error rate does not drop monotonically.

3.3. USPS digit data

We also evaluated the proposed method in an image recognition task. Specifically, we used the data in (Zhou et al., 2004) for handwritten digit classification experiments. To evaluate the algorithms, we reveal a subset of the labels (randomly chosen and guaranteeing at least one labeled example is available for each digit). We compared our method with *LGC* and *GFHF*, *LapRLS*, and *LapSVM*. The error rates are calculated based on the average over 20 trials.

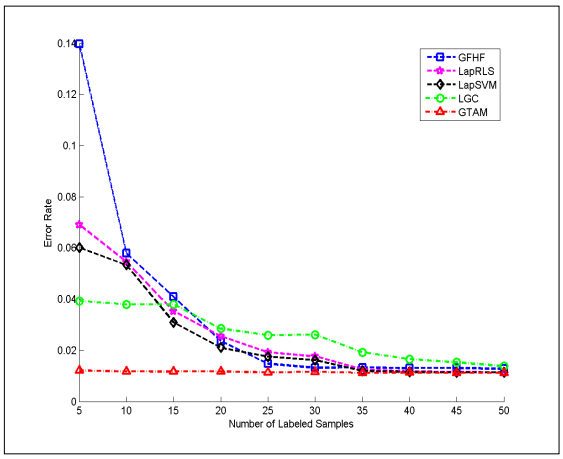


Figure 4. Performance comparison on handwritten digit classification (USPS database). The horizontal axis shows the total number of randomly observed labels (guaranteeing there is at least one label for each class). The vertical axis shows the average error rate over 20 random trials.

From Figure 4, we can conclude that *GTAM* significantly improved the classification accuracy, compared to the other approaches, especially when very few labeled samples are available. The mean accuracies of *GTAM* are consistently low for different numbers of labels and the standard deviation values are also very small (10^{-4} level). This demonstrates that the *GTAM* method is insensitive to the numbers and specified locations of the initially given labels. Only 1% of the test digit images were mislabeled. These failure cases are presented in Figure 5 and are often ambiguous or extremely poorly drawn digits. Compared to the performance on WebKB dataset shown in Figure 3, the USPS digit database experiments exhibit even more promising results. One possible reason is that the USPS digit dataset has relatively more samples (3874) and a lower feature dimensionality (256), compared to the WebKB dataset (which has 1840 samples in 4800 dimensions). Therefore the graph construction procedure is more reliable and the estimation of graph gradients in our algorithm is more robust.

Figure 5. USPS handwritten digit samples which are incorrectly classified.

4. Conclusion and Discussion

Existing graph-based transductive learning methods hinge on good labeling information and can easily be misled if the labels are not distributed evenly across classes, if the choice of initial label locations is varied or if excessive noise or outliers corrupt the underlying manifold structure. These degenerate settings seem to plague real world problems as well, compromising the performance of state-of-the-art graph transduction methods. Our experiments over synthetic data sets (two moon data sets) and real data sets (USPS digits and WebKB) confirm the shortcomings of existing tools.

This article addresses these shortcomings and proposes a novel graph based semi-supervised learning method, graph transduction via alternating minimization (*GTAM*). Therein, both the classification function and the label matrix are treated as variables in a cost function that is iteratively minimized. While the optimal classification function can be estimated exactly, greedy optimization is applied to update the label matrix. The algorithm iterates an alternating minimization between both variables and is guaranteed to converge via a greedy scheme. In each individual iteration, through the graph gradient, the unlabeled node with the largest cost reduction is labeled. We gradually update the label matrix by adding more labeled samples while keeping the classification function at its optimal setting. Furthermore, we enforce normalization of the label matrix to avoid degeneracies. This results in an algorithm that can cope with all the aforementioned degeneracies and in practice achieves significant gains in accuracy while remaining efficient and cubic in the number of samples. Future work will include out of sample extensions of this method such that new data points can be added to the training and test set without requiring a full retraining procedure.

5. Acknowledgments

The authors would like to thank Mr. Yongtao Su and Shouzhen Liu for their valuable comments. We also thank Mr. We Liu and Deli Zhao for the collection of the artificial data. TJ was supported by ONR Award N000140710507 (ModNo: 07PR04918-00) and NSF Award IIS-0347499.

References

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *JMLR*, *7*, 2399–2434.

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. *Proc. 18th ICML* (pp. 19–26).

Chapelle, O., Sindhwani, V., & Keerthi, S. (2007). Branch and Bound for Semi-Supervised Support Vector Machines. *Proc. of NIPS*.

Chapelle, O., Weston, J., & Scholkopf, B. (2003). Cluster kernels for semi-supervised learning. *Proc. NIPS*, *15*, 1.

Goemans, M., & Williamson, D. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, *42*, 1115–1145.

Hein, M., & Maier, M. (2006). Manifold denoising. *Proc. NIPS*, *19*.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *Proc. of the ICML*, 200–209.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. *Proc. of ICML*, 290–297.

Karp, R. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations*, *43*, 85–103.

Mann, G., & McCallum, A. (2007). Simple, robust, scalable semi-supervised learning via expectation regularization. *Proc. of the ICML*, 593–600.

Ng, A., Jordan, M., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Proc. NIPS*, *14*, 849–856.

Sindhwani, V., Niyogi, P., & Belkin, M. (2005). Beyond the point cloud: from transductive to semi-supervised learning. *Proc. of ICML*.

Wang, J., Chang, S.-F., Zhou, X., & Wong, T. C. S. (2008). Active microscopic cellular image annotation by superposable graph transduction with imbalanced labels. *IEEE CVPR*. Alaska, USA.

Zhou, D., Bousquet, O., Lal, T., Weston, J., & Scholkopf, B. (2004). Learning with local and global consistency. *Proc. NIPS* (pp. 321–328).

Zhu, X. (2005). *Semi-supervised learning literature survey* (Technical Report 1530). Computer Sciences, University of Wisconsin-Madison.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *Proc. 20th ICML*.

On Multi-View Active Learning and the Combination with Semi-Supervised Learning

Wei Wang WANGW@LAMDA.NJU.EDU.CN
Zhi-Hua Zhou ZHOUZH@LAMDA.NJU.EDU.CN
National Key Laboratory for Novel Software Technology, Nanjing University, China

Abstract

Multi-view learning has become a hot topic during the past few years. In this paper, we first characterize the sample complexity of multi-view *active learning*. Under the α-*expansion* assumption, we get an exponential improvement in the sample complexity from usual $\widetilde{O}(\frac{1}{\epsilon})$ to $\widetilde{O}(\log \frac{1}{\epsilon})$, requiring neither strong assumption on data distribution such as the data is distributed uniformly over the unit sphere in $\mathbb{R}^d$ nor strong assumption on hypothesis class such as linear separators through the origin. We also give an upper bound of the error rate when the α-*expansion* assumption does not hold. Then, we analyze the combination of multi-view active learning and semi-supervised learning and get a further improvement in the sample complexity. Finally, we study the empirical behavior of the two paradigms, which verifies that the combination of multi-view active learning and semi-supervised learning is efficient.

1. Introduction

Learning from labeled data is well-established in machine learning, but labeling the training data is time consuming, sometimes may be very expensive since it may need human efforts. In many machine learning applications, unlabeled data can often be obtained abundantly and cheaply, so there has recently been substantive interest in using large amount of unlabeled data together with labeled data to achieve better learning performance.

There are two popular paradigms for using unlabeled data to complement labeled data. One is *semi-supervised learning*. Some approaches use a generative model for the classifier and employ EM to model the label estimation or parameter estimation process (Dempster et al., 1977; Miller & Uyar, 1997; Nigam et al., 2000); some approaches use the unlabeled data to regularize the learning process in various ways, e.g., defining a graph on the data set and then enforcing the label smoothness over the graph as a regularization term (Belkin et al., 2001; Zhu et al., 2003; Zhou et al., 2005); some approaches use the *multi-view* setting to train learners and then let the learners to label unlabeled examples (Blum & Mitchell, 1998; Goldman & Zhou, 2000; Zhou & Li, 2005). The multi-view setting is first formalized by Blum and Mitchell (1998), where there are several disjoint subsets of features (each subset is called as a *view*), each of which is sufficient for learning the target concept. For example, the web page classification task has two views, i.e., the text appearing on the page itself and the anchor text attached to hyper-links pointing to this page (Blum & Mitchell, 1998); the speech recognition task also has two views, i.e., sound and lip motion (de Sa & Ballard, 1998).

Another important paradigm for using unlabeled data to complement labeled data, which is the focus of this paper, is *active learning* (Cohn et al., 1994; Freund et al., 1997; Tong & Koller, 2001; Melville & Mooney, 2004). In active learning, the learners actively ask the user to label the *most informative* examples and hope to learn a good classifier with as few labeled examples as possible.

There have been many theoretical analyses on the sample complexity of *single-view* active learning. For some simple learning tasks the sample complexity of active learning can be $O(\log \frac{1}{\epsilon})$ which is exponentially improved in contrast to $O(\frac{1}{\epsilon})$ of *passive learning* taking into account the desired accuracy bound ϵ. Unfortunately, such an exponential improvement is not always achievable in active learning. Dasgupta (2006) illustrated that if the hypothesis class $\mathcal{H}$ is linear separators in $\mathbb{R}^2$ and if the data distribution is some density

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

supported on the perimeter of the unit circle, there are some target hypotheses in $\mathcal{H}$ for which $\Omega(\frac{1}{\epsilon})$ labels are needed to find a classifier with error rate less than ϵ, no matter what active learning approach is used. Under the strong assumptions that the hypothesis class is linear separators through the origin, that the data is distributed uniformly over the unit sphere in $\mathbb{R}^d$, and that the learning task is a *realizable* case (i.e., there exists a hypothesis perfectly separating the data), the sample complexity of active learning is $\widetilde{O}(d \log \frac{1}{\epsilon})$ taking into account the desired accuracy bound ϵ (Freund et al., 1997; Dasgupta et al., 2005) [1]. For some known data distribution $\mathbb{D}$ and specific hypothesis class, Dasgupta (2006) gave the coarse sample complexity bounds for realizable active learning. The study of sample complexity of active learning for realizable case without strong assumptions on the data distribution and the hypothesis class remains an open problem.

All the above results were obtained under the *single-view* setting. The first algorithm for active learning in multi-view setting is *co-testing* (Muslea et al., 2000; Muslea et al., 2006). It focuses on the set of *contention points* (i.e., unlabeled examples on which different views predict different labels) and asks the user to label some of them. This is somewhat related to Query-by-Committee (Freund et al., 1997) since *co-testing* also uses more than one learners to identify the most informative unlabeled examples to query, but the typical Query-by-Committee works under a single-view setting while *co-testing* exploits the multi-views explicitly. It was reported that *co-testing* outperforms existing active learners on a variety of real-world domains such as wrapper induction, Web page classification, advertisement removal and discourse tree parsing. To the best of our knowledge, however, there is no theoretical result on the sample complexity of multi-view active learning.

In this paper, we first theoretically analyze the sample complexity of multi-view active learning under the α-*expansion* assumption which is first mentioned by Balcan et al. (2005) and prove that the sample complexity of multi-view active learning can be exponentially improved to $\widetilde{O}(\log \frac{1}{\epsilon})$. A clear advantage is that we do not use strong assumptions which were employed in most previous studies, such as the hypothesis class is linear separators through the origin and the data is distributed uniformly over the unit sphere in $\mathbb{R}^d$. In case the α-*expansion* assumption does not hold, we give an upper bound of the error rate. Second, we analyze the combination of multi-view active learning and semi-supervised learning and get an further improvement in the sample complexity. Finally, we study the empirical behavior of the two paradigms, which verifies that the combination of multi-view active learning and semi-supervised learning is more efficient than pure multi-view active learning.

The rest of this paper is organized as follows. After introducing some preliminaries in Section 2, we analyze the sample complexity of multi-view active learning in Section 3. Then we analyze the sample complexity of the combination of multi-view active learning and semi-supervised learning in Section 4 and study the empirical behavior in Section 5. Finally we conclude the paper in Section 6.

2. Preliminaries

In the multi-view setting, an example x is described with several different disjoint sets of features. Without loss of generality, in this paper we only consider the *two-view* setting for the sake of simplicity. Suppose that the example space $X = X_1 \times X_2$ is with some unknown distribution $\mathbb{D}$, X_1 and X_2 are the two views, and $Y = \{-1, 1\}$ is the label space. Let $c = (c_1, c_2)$ be the underlying target concept, where c_1 and c_2 are the underlying target concepts in the two views, respectively. Suppose that the example space is consistent, that is, there is no such example $x = (x_1, x_2)$ that $c_1(x_1) \neq c_2(x_2)$ in X. Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be the hypothesis class in each view, respectively. For any $h_j \in \mathcal{H}_j$ and $x = (x_1, x_2)$ we say $x_j \in h_j$ if and only if $h_j(x_j) = c_j(x_j)$ $(j = 1, 2)$. In this way any hypothesis in $\mathcal{H}_j$ can be thought of as a subset of X_j.

In each round of iterative multi-view active learning, the learners ask the user to label some unlabeled examples and add them into the labeled training data. These newly labeled examples provide more information about the data distribution. In this paper, we consider the *co-testing*-like Paradigm 1 described in Table 1. In Paradigm 1, the learners ask the user to label some contention points to refine the classifiers. If the confident set of each view is expanding by considering the other view together, Paradigm 1 may succeed. Intuitively, we can use the α-*expansion* assumption to analyze the process.

Suppose $S_1 \subseteq X_1$ and $S_2 \subseteq X_2$ denote the examples that are correctly classified in each view, respectively. Let $Pr(\mathbf{S}_1 \wedge \mathbf{S}_2)$ denote the probability mass on examples that are correctly classified in both views, while $Pr(\mathbf{S}_1 \oplus \mathbf{S}_2)$ denotes the probability mass on examples that are correctly classified only in one view (i.e., examples disagreed by the two classifiers). Now we give

[1] The $\widetilde{O}$ notation is used to hide factors $\log\log(\frac{1}{\epsilon})$, $\log(d)$ and $\log(\frac{1}{\delta})$

Input:
Unlabeled data set $\mathcal{U} = \{x^1, x^2, \cdots, \}$, where each example x^t is given as a pair (x_1^t, x_2^t)
Process:
Ask the user to label m_0 unlabeled examples drawn randomly from $\mathbb{D}$ to compose the labeled data set $\mathcal{L}$
Iterate $i = 0, 1, \cdots, s$
Train two classifiers h_1^i and h_2^i consistent with $\mathcal{L}$ in each view, respectively;
Apply h_1^i and h_2^i to the unlabeled data set $\mathcal{U}$ and find out the contention points set $\mathcal{Q}_i$;
Ask the user to label m_{i+1} unlabeled examples drawn randomly from $\mathcal{Q}_i$, then add them into $\mathcal{L}$ and delete them from $\mathcal{U}$.
Output:
$h_{final} = \text{combine}(h_1^s, h_2^s)$

Table 1. Paradigm 1: Multi-view active learning

our definition on α-*expansion*.

Definition 1 *$\mathbb{D}$ is α-expansion if for any $S_1 \subseteq X_1$, $S_2 \subseteq X_2$, we have*

$$Pr(\boldsymbol{S}_1 \oplus \boldsymbol{S}_2) \geq \alpha \min[Pr(\boldsymbol{S}_1 \wedge \boldsymbol{S}_2), Pr(\overline{\boldsymbol{S}_1} \wedge \overline{\boldsymbol{S}_2})].$$

We say that $\mathbb{D}$ is α-expanding with respect to hypothesis class $\mathcal{H}_1 \times \mathcal{H}_2$ if the above holds for all $S_1 \in \mathcal{H}_1 \cap X_1$, $S_2 \in \mathcal{H}_2 \cap X_2$ (here we denote by $\mathcal{H}_j \cap X_j$ the set $\{h \cap X_j : h \in \mathcal{H}_j\}$ for $j = 1, 2$).

Note that Definition 1 on α-*expansion* is almost the same as that in Balcan et al. (2005). To guarantee the success of iterative *co-training*, they made several assumptions such as that the learning algorithm used in each view is confident about being positive and is able to learn from positive examples only, and that the distribution $\mathbb{D}^+$ over positive examples is expanding. There are many concept classes, however, are not learnable from positive examples only. Apparently, all problems which satisfy the definition of Balcan et al. (2005) also satisfy our definition.

We will make use of the following lemma when deriving our sample complexity bound (Anthony & Bartlett, 1999).

Lemma 1 *Let $\mathcal{H}$ be a set of functions from X to $\{-1, 1\}$ with finite VC-dimension $V \geq 1$. Let $\mathbb{P}$ be an arbitrary, but fixed probability distribution over $X \times \{-1, 1\}$. For any ϵ, $\delta > 0$, if we draw a sample from $\mathbb{P}$ of size $N(\epsilon, \delta) = \frac{1}{\epsilon}(4V \log(\frac{1}{\epsilon}) + 2\log(\frac{2}{\delta}))$, then with probability $1 - \delta$, all hypotheses with error $\geq \epsilon$ are inconsistent with the data.*

3. Sample Complexity of Multi-View Active Learning

There are many strategies to combine the classifiers in Paradigm 1, for example, *weighted voting*, *majority voting* or *winner-take-all* (Muslea et al., 2006). In this paper, we use the following simple combination scheme for binary classification:

$$h_{com}^i(x) = \begin{cases} h_1^i(x_1) & \text{if } h_1^i(x_1) = h_2^i(x_2) \\ \textit{random guess} & \text{if } h_1^i(x_1) \neq h_2^i(x_2) \end{cases} \quad (1)$$

Assuming that the data distribution $\mathbb{D}$ is α-expanding with respect to hypothesis class $\mathcal{H}_1 \times \mathcal{H}_2$, we will analyze how many labels the user should label to achieve classifiers with error rate no larger than ϵ. We consider the iterative process and let $S_1^i \subseteq X_1$ and $S_2^i \subseteq X_2$ where S_1^i and S_2^i corresponds to the classifiers $h_1^i \in \mathcal{H}_1$ and $h_2^i \in \mathcal{H}_2$ in the i-th round, respectively. The initial m_0 unlabeled examples are randomly picked from $\mathbb{D}$ and labeled by the user according to the target concept c. Suppose m_0 is sufficient for learning two classifiers h_1^0 and h_2^0 whose error rates are at most $1/4$ (i.e., $Pr(\mathbf{S}_1^0) \geq 1 - 1/4$ and $Pr(\mathbf{S}_2^0) \geq 1 - 1/4$), and thus $Pr(\mathbf{S}_1^0 \wedge \mathbf{S}_2^0) \geq 1/2$. The α-*expansion* condition suggests

$$Pr(\mathbf{S}_1^0 \oplus \mathbf{S}_2^0) \geq \alpha Pr(\overline{\mathbf{S}_1^0} \wedge \overline{\mathbf{S}_2^0}).$$

In each round of Paradigm 1, the learners ask the user to label some unlabeled examples according to the target concept c and add them into the labeled data set. Then the two classifiers are refined. Some example x in X might be predicted with different labels between the i-th and $(i+1)$-th round. Intuitively, in order to get the classifiers improved in Paradigm 1, the reduced size of confident set should be no more than the size of contention set. Moreover, considering that there is no noise in the labeled data since all the labels are given by the user according to the target concept, and that the amount of labeled training examples are monotonically increasing, the asymptotic performance of PAC learners increase, we can assume that

$$Pr(\overline{\mathbf{S}_j^{i+1}} \mid \mathbf{S}_1^i \wedge \mathbf{S}_2^i) \leq \frac{\alpha Pr(\mathbf{S}_1^i \oplus \mathbf{S}_2^i)}{16 Pr(\mathbf{S}_1^i \wedge \mathbf{S}_2^i)} \quad (j \in \{1, 2\}) \quad (2)$$

Intuitively, by multiplying the denominator at the right-hand to the left-hand (16 is used for a faster convergence; it can be 2 for an easier understanding), Eq. 2 implies that the total reduced size of confident sets on both views after using the newly labeled contention points is no more than the size of contention set. Apparently, all problems that satisfy the assumption of Balcan et al. (2005) also satisfy Eq. 2. Now we give our main theorem.

Theorem 1 *For data distribution $\mathbb{D}$ α-expanding with respect to hypothesis class $\mathcal{H}_1 \times \mathcal{H}_2$, let ϵ and δ denote the final desired accuracy and confidence parameters. If $s = \lceil \frac{\log \frac{\alpha}{8\epsilon}}{\log \frac{1}{C}} \rceil$ and $m_i = \frac{16}{\alpha}(4V \log(\frac{16}{\alpha}) + 2\log(\frac{8(s+1)}{\delta}))$ $(i = 0, 1, \cdots, s)$, Paradigm 1 will generate a classifier with error rate no more than ϵ with probability $1 - \delta$.*

Here, $V = \max[VC(\mathcal{H}_1), VC(\mathcal{H}_2)]$ where $VC(\mathcal{H})$ denotes the VC-dimension of the hypothesis class $\mathcal{H}$ and constant $C = \frac{\alpha/4+1/\alpha}{1+1/\alpha}$.

Proof. In Paradigm 1, we use Eq. 1 to combine the two classifiers, thus the error rate of the combined classifier h^i_{com} is

$$\begin{aligned} error_{h^i_{com}} &= Pr(\overline{\mathbf{S}_1^i} \wedge \overline{\mathbf{S}_2^i}) + \frac{1}{2} Pr(\mathbf{S}_1^i \oplus \mathbf{S}_2^i) \\ &\leq Pr(\overline{\mathbf{S}_1^i} \wedge \overline{\mathbf{S}_2^i}) + Pr(\mathbf{S}_1^i \oplus \mathbf{S}_2^i) \\ &= Pr(\overline{\mathbf{S}_1^i \wedge \mathbf{S}_2^i}) \end{aligned}$$

With $m_0 = \frac{16}{\alpha}(4V \log(\frac{16}{\alpha}) + 2\log(\frac{8(s+1)}{\delta}))$, using Lemma 1 we have $Pr(\overline{\mathbf{S}_1^0}) \leq \frac{\alpha}{16}$ and $Pr(\overline{\mathbf{S}_2^0}) \leq \frac{\alpha}{16}$ with probability $1 - \frac{\delta}{4(s+1)}$. Generally, we have that an arbitrary S_j^i $(j = 1, 2)$ being consistent with the examples in $\mathcal{L}$ has an error rate at most $\frac{\alpha}{16}$ with probability $1 - \frac{\delta}{4(s+1)}$. So we have $Pr(\mathbf{S}_1^i \wedge \mathbf{S}_2^i) \geq 1 - \frac{\alpha}{8}$ with probability $1 - \frac{\delta}{2(s+1)}$. Without loss of generality, consider $0 < \alpha \leq 1$ and therefore $1 - \frac{\alpha}{8} > \frac{1}{2}$. Thus the α-*expansion* condition suggests

$$Pr(\mathbf{S}_1^i \oplus \mathbf{S}_2^i) \geq \alpha Pr(\overline{\mathbf{S}_1^i} \wedge \overline{\mathbf{S}_2^i}). \tag{3}$$

For $i \geq 1$, the learners ask the user to label m_i unlabeled examples drawn randomly from $S_1^{i-1} \oplus S_2^{i-1}$ according to the target concept c and obtain two new classifiers S_1^i and S_2^i. Similarly, if $m_i = \frac{16}{\alpha}(4V \log(\frac{16}{\alpha}) + 2\log(\frac{8(s+1)}{\delta}))$, using Lemma 1 we have

$$Pr(\overline{\mathbf{S}_j^i} \mid \mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) \leq \frac{\alpha}{16} \quad (j \in \{1, 2\})$$

with probability $1 - \frac{\delta}{4(s+1)}$. So we get that

$$Pr(\overline{\mathbf{S}_1^i \wedge \mathbf{S}_2^i} \mid \mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) \leq \frac{\alpha}{8}$$

with probability $1 - \frac{\delta}{2(s+1)}$. Considering Eq. 2 we have

$$Pr(\overline{\mathbf{S}_1^i \wedge \mathbf{S}_2^i} \mid \mathbf{S}_1^{i-1} \wedge \mathbf{S}_2^{i-1}) \leq \frac{\alpha Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1})}{8 Pr(\mathbf{S}_1^{i-1} \wedge \mathbf{S}_2^{i-1})} .$$

Since

$$\begin{aligned} Pr(\overline{\mathbf{S}_1^i \wedge \mathbf{S}_2^i}) = {} & Pr(\overline{\mathbf{S}_1^i \wedge \mathbf{S}_2^i} \mid \overline{\mathbf{S}_1^{i-1}} \wedge \overline{\mathbf{S}_2^{i-1}}) \\ & \cdot Pr(\overline{\mathbf{S}_1^{i-1}} \wedge \overline{\mathbf{S}_2^{i-1}}) \\ & + Pr(\overline{\mathbf{S}_1^i \wedge \mathbf{S}_2^i} \mid \mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) \\ & \cdot Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) \\ & + Pr(\overline{\mathbf{S}_1^i \wedge \mathbf{S}_2^i} \mid \mathbf{S}_1^{i-1} \wedge \mathbf{S}_2^{i-1}) \\ & \cdot Pr(\mathbf{S}_1^{i-1} \wedge \mathbf{S}_2^{i-1}), \end{aligned}$$

we have

$$Pr(\overline{\mathbf{S}_1^i \wedge \mathbf{S}_2^i}) \leq \frac{\alpha}{4} Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) + Pr(\overline{\mathbf{S}_1^{i-1}} \wedge \overline{\mathbf{S}_2^{i-1}}) .$$

From Eq. 3 we can get that

$$Pr(\overline{\mathbf{S}_1^{i-1}} \wedge \overline{\mathbf{S}_2^{i-1}}) \leq Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1})/\alpha .$$

Thus, considering

$$Pr(\overline{\mathbf{S}_1^{i-1} \wedge \mathbf{S}_2^{i-1}}) = Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) + Pr(\overline{\mathbf{S}_1^{i-1}} \wedge \overline{\mathbf{S}_2^{i-1}}),$$

we have

$$\begin{aligned} & \frac{Pr(\overline{\mathbf{S}_1^i \wedge \mathbf{S}_2^i})}{Pr(\overline{\mathbf{S}_1^{i-1} \wedge \mathbf{S}_2^{i-1}})} \\ \leq {} & \frac{\frac{\alpha}{4} Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) + Pr(\overline{\mathbf{S}_1^{i-1}} \wedge \overline{\mathbf{S}_2^{i-1}})}{Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) + Pr(\overline{\mathbf{S}_1^{i-1}} \wedge \overline{\mathbf{S}_2^{i-1}})} \\ \leq {} & \frac{\frac{\alpha}{4} Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) + Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1})/\alpha}{Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1}) + Pr(\mathbf{S}_1^{i-1} \oplus \mathbf{S}_2^{i-1})/\alpha} \\ = {} & \frac{\alpha/4 + 1/\alpha}{1 + 1/\alpha} . \end{aligned}$$

Now we get

$$\begin{aligned} Pr(\overline{\mathbf{S}_1^s \wedge \mathbf{S}_2^s}) &\leq (\frac{\alpha/4 + 1/\alpha}{1 + 1/\alpha})^s Pr(\overline{\mathbf{S}_1^0 \wedge \mathbf{S}_2^0}) \\ &\leq \frac{\alpha}{8} (\frac{\alpha/4 + 1/\alpha}{1 + 1/\alpha})^s . \end{aligned}$$

So when $s = \lceil \frac{\log \frac{\alpha}{8\epsilon}}{\log \frac{1}{C}} \rceil$ where C is a constant and $\frac{\alpha/4+1/\alpha}{1+1/\alpha} < 1$, we have $Pr(\overline{\mathbf{S}_1^s \wedge \mathbf{S}_2^s}) \leq \epsilon$. In other words, we get a classifier h^s_{com} whose error rate is no more than ϵ with probability $1 - \delta$. □

From Theorem 1 we know that we only need to label $\sum_{i=0}^{s} m_i = O(\log \frac{1}{\epsilon} \log(\log \frac{1}{\epsilon}))$ examples to get a classifier with error rate no more than ϵ with probability

$1-\delta$. Thus, we achieve an exponential improvement in sample complexity from $\widetilde{O}(\frac{1}{\epsilon})$ to $\widetilde{O}(\log\frac{1}{\epsilon})$ as in Dasgupta et al. (2005) and Balcan et al. (2007). Note that we have not assumed a specific data distribution and a specific hypothesis class which were assumed in the studies of Dasgupta et al. (2005) and Balcan et al. (2007). From the proof of Theorem 1 we can also know that the proportion $\frac{\alpha}{16}$ in Eq. 2 can be relaxed to close to $\frac{\alpha}{2}$. Such relaxation will not affect the exponential improvement, but will reduce the convergence speed.

Further, considering that not every data distribution $\mathbb{D}$ is α-expanding with respect to hypothesis class $\mathcal{H}_1 \times \mathcal{H}_2$, we will give a coarse upper bound of the generalization error for Paradigm 1 for cases when the *α-expansion* assumption does not hold.

Let $Pr(\mathbf{S}_1^i \oplus \mathbf{S}_2^i) = \alpha_i Pr(\overline{\mathbf{S}_1^i} \wedge \overline{\mathbf{S}_2^i})$ $(i = 0, 1, \cdots)$. If the *α-expansion* assumption does not hold in Paradigm 1, for any $\epsilon > 0$ and any integer $N > 0$, the size of the set $\{\alpha_i\colon\ i > N \wedge \alpha_i < \epsilon\}$ is infinite. We set a parameter $\epsilon_c > 0$ as the stop condition. When $Pr(\mathbf{S}_1^i \oplus \mathbf{S}_2^i)$ is less than ϵ_c, we terminate the iteration in Paradigm 1. Now we make the definition on *expanded region with respect to ϵ_c*.

Definition 2 *Let γ_{ϵ_c} denote the expanded region with respect to ϵ_c in Paradigm 1,*

$$\gamma_{\epsilon_c} = Pr(\overline{\boldsymbol{S}_1^0} \wedge \overline{\boldsymbol{S}_2^0}) - Pr(\overline{\boldsymbol{S}_1^i} \wedge \overline{\boldsymbol{S}_2^i}),$$

where $i = \min\{i : Pr(\boldsymbol{S}_1^i \oplus \boldsymbol{S}_2^i) < \epsilon_c \wedge i \geq 1\}$.

After i rounds the region in which both classifiers wrongly predict becomes smaller and smaller, from $Pr(\overline{\mathbf{S}_1^0} \wedge \overline{\mathbf{S}_2^0})$ to $Pr(\overline{\mathbf{S}_1^i} \wedge \overline{\mathbf{S}_2^i})$. This *expanded region* can be thought of as an approximation of $\Sigma_{k=1}^{i} Pr(\mathbf{S}_1^k \oplus \mathbf{S}_2^k)$.

Theorem 2 *When the α-expansion assumption does not hold, set $\epsilon_c > 0$ to terminate Paradigm 1. The error rate of h_{com}^i can be smaller than h_{com}^0 for $\gamma_{\epsilon_c} + \frac{1}{2}(Pr(\boldsymbol{S}_1^0 \oplus \boldsymbol{S}_2^0) - \epsilon_c)$.*

Proof. Considering $error_{h_{com}^i} = Pr(\overline{\mathbf{S}_1^i} \wedge \overline{\mathbf{S}_2^i}) + \frac{1}{2}Pr(\mathbf{S}_1^i \oplus \mathbf{S}_2^i)$ and $Pr(\mathbf{S}_1^i \oplus \mathbf{S}_2^i) < \epsilon_c$, we have that $error_{h_{com}^0} - error_{h_{com}^i}$ is larger than $\gamma_{\epsilon_c} + \frac{1}{2}(Pr(\mathbf{S}_1^0 \oplus \mathbf{S}_2^0) - \epsilon_c)$. □

Theorem 2 implies that Paradigm 1 could not boost the performance to arbitrarily high and gives a coarse upper bound of the error rate, when the *α-expansion* assumption does not hold. The improvement depends on the *expanded region* γ and the disagreement between the initial two classifiers. The larger the *expanded region* γ, the better the improvement of Paradigm 1. Theorem 2 can also be applied to one-shot *co-training* (Balcan et al., 2005).

4. Sample Complexity of Combination of Multi-View Active Learning and Semi-Supervised Learning

We can try to reduce the sample complexity further by combining multi-view active learning with semi-supervised learning. Previously this has been tried in some applications and led to good results (Zhou et al., 2006), yet to the best of our knowledge, there is no theoretical analysis which supports such argument. For computational simplicity, we consider the following case in this section. Suppose that the hypothesis class $\mathcal{H}_j$ is the subset of mappings from X_j to $[-1, 1]$ and $y = sign(c(x))$, $c = (c_1, c_2)$ is the underlying target concept, where c_1 and c_2 is the underlying target concept in each view, respectively. Let $d(f, g)$ denote the probability that the two classifiers $f \in \mathcal{H}_j$ and $g \in \mathcal{H}_j$ predict different labels on an example x_j drawn randomly from X_j, then

$$d(f, g) = Pr_{x_j \in X_j}\Big(sign\big(f(x_j)\big) \neq sign\big(g(x_j)\big)\Big).$$

Suppose that for any $f, g \in \mathcal{H}_j$, there exists some constant $L_1 > 0$ to hold that $|f(x_j) - g(x_j)| \leq L_1 \cdot d(f, g) \cdot \|x_j\|_2$, where $\|x_j\|_2$ denotes the 2-norm of x_j. Without loss of generality, suppose that there exists some constant $L_2 > 0$ to hold that $\|x_j\|_2 \leq L_2$ for $x_j \in X_j$ $(j = 1, 2)$. Now we have the following theorem for Paradigm 2 which combines multi-view active learning with semi-supervised learning.

Theorem 3 *For data distribution $\mathbb{D}$ α-expanding with respect to hypothesis class $\mathcal{H}_1 \times \mathcal{H}_2$, let ϵ and δ denote the final desired accuracy and confidence parameters. If $s = \lceil \frac{\log\frac{\alpha}{8\epsilon}}{\log\frac{1}{C}} \rceil$, $m_0 = \frac{1}{L}(4V\log(\frac{1}{L}) + 2\log(\frac{8(s+1)}{\delta}))$ and $m_i = \frac{16}{\alpha}(4V\log(\frac{16}{\alpha}) + 2\log(\frac{8(s+1)}{\delta}))$ $(i = 1, 2, \cdots,)$, Paradigm 2 will generate a classifier with error rate no more than ϵ with probability $1 - \delta$.*

Here, $V = \max[VC(\mathcal{H}_1), VC(\mathcal{H}_2)]$ where $VC(\mathcal{H})$ denotes the VC-dimension of the hypothesis class $\mathcal{H}$, constant $C = \frac{\alpha/4+1/\alpha}{1+1/\alpha}$ and constant $L = \min[\frac{\alpha}{16}, \frac{1}{16L_1L_2}]$.

Proof. In Paradigm 2, we also use Eq. 1 to combine the two classifiers. With $m_0 = \frac{1}{L}(4V\log(\frac{1}{L}) + 2\log(\frac{8(s+1)}{\delta}))$ where constant $L = \min[\frac{\alpha}{16}, \frac{1}{16L_1L_2}]$, using Lemma 1 we have $Pr(\overline{\mathbf{S}_1^0}) \leq \frac{1}{L}$ and $Pr(\overline{\mathbf{S}_2^0}) \leq \frac{1}{L}$ with probability $1 - \frac{\delta}{4(s+1)}$. Generally, we have that an

Input:

Unlabeled data set $\mathcal{U} = \{x^1, x^2, \cdots, \}$, where each example x^t is given as a pair (x_1^t, x_2^t)

Threshold thr

Process:

Ask the user to label m_0 unlabeled examples drawn randomly from $\mathbb{D}$ to compose the labeled data set $\mathcal{L}$

Iterate $i = 0, 1, \cdots, s$

Set counter n_1^{i+1} to 0. If $\mathbb{D}$ is expanding, set counter n_2^{i+1} to $+\infty$; Otherwise, set counter n_2^{i+1} to 0;

Train two classifiers h_1^i and h_2^i consistent with $\mathcal{L}$ in each view, respectively;

Apply h_1^i and h_2^i to the unlabeled data set $\mathcal{U}$ and find out the contention points set $\mathcal{Q}_i$;

for $k = 1, \cdots, m_{i+1}$

Draw an example $x^k = (x_1^k, x_2^k)$ randomly from $\mathcal{Q}_i$;

if $|h_1^i(x_1^k)| > thr$ **then** $y^k = sign(h_1^i(x_1^k))$;

else if $|h_2^i(x_2^k)| > thr$ **then** $y^k = sign(h_2^i(x_2^k))$;

else ask the user to label x^k and $n_1^{i+1} = n_1^{i+1} + 1$;

Add (x^k, y^k) into $\mathcal{L}$ and delete it from $\mathcal{U}$ and $\mathcal{Q}_i$.

end for

for $w = 1, 2, \cdots$

if $n_2^{i+1} \geq m_{i+1} - n_1^{i+1}$ **break**;

Draw an example $x^w = (x_1^w, x_2^w)$ randomly from $\mathcal{U} - \mathcal{Q}_i$;

if $|h_1^i(x_1^w)| > thr$ **then** $y^w = sign(h_1^i(x_1^w))$;

else if $|h_2^i(x_2^w)| > thr$ **then** $y^w = sign(h_2^i(x_2^w))$;

else ask the user to label x^w and $n_2^{i+1} = n_2^{i+1} + 1$;

Add (x^w, y^w) into $\mathcal{L}$ and delete it from $\mathcal{U}$.

end for

Output:

$h_{final} = \text{combine}(h_1^s, h_2^s)$

Table 2. Paradigm 2: Combination of multi-view active learning and semi-supervised learning

arbitrary S_j^i $(j = 1, 2)$ being consistent with the examples in $\mathcal{L}$ has an error rate at most $\frac{1}{L}$ with probability $1 - \frac{\delta}{4(s+1)}$. So, for any example $x = (x_1, x_2)$,

$$|h_j^i(x_j) - c_j(x_j)| \leq L_1 \cdot L_2 \cdot d(h_j^i, c_j) \leq \frac{1}{16} .$$

We can set the threshold thr in Paradigm 2 to $\frac{1}{16}$. If $|h_j^i(x_j)| > \frac{1}{16}$, h_j^i and c_j make the same prediction on x_j. When $s = \lceil \frac{\log \frac{\alpha}{8\epsilon}}{\log \frac{1}{C}} \rceil$, from the proof of Theorem 1 we have $Pr(\overline{\mathbf{S}_1^s \wedge \mathbf{S}_2^s}) \leq \epsilon$. Thus we get a classifier h_{com}^s whose error rate is no more than ϵ with probability $1 - \delta$ using Paradigm 2. □

The sample complexity of Paradigm 2 is $m_0 + \sum_{i=1}^{s} n_1^i$, which is much smaller than that of Paradigm 1. From Theorem 3 we know that the sample complexity can be further reduced by combining multi-view active learning with semi-supervised learning, however, it needs a stronger assumption on the hypothesis class $\mathcal{H}_1 \times \mathcal{H}_2$. If this assumption holds, in contrast to Paradigm 1, when α-*expansion* does not hold, we can query $\sum_{i=1}^{s}(m_i - n_1^i)$ more examples on which both classifiers have small margin, which can help to reduce the size of the region $\overline{S_1} \wedge \overline{S_2}$.

5. Empirical Study

In this section we empirically study the performance of the Paradigms 1 and 2 on a real-world data set, i.e., the *course* data set (Blum & Mitchell, 1998). This data set has two views (*pages* view and *links* view) and contains 1,051 examples each corresponds to a web page, and the task is to predict whether an unseen web page is a course page or not. There are 230 positive examples (roughly 22%). We randomly use 25% data as the test set and use the remaining 75% data as the unlabeled set $\mathcal{U}$ in Tables 1 and 2. Then, we randomly draw 10 positive and 30 negative examples from $\mathcal{U}$ to generate the initial m_0 labeled examples.

In practice, the thr in Paradigm 2 can be determined by cross validation on labeled examples. Here in our experiments, for the ease of comparison, we do not set thr and instead, we fix the number of examples to be queried in both Paradigms. Thus, we can study their performance under the same number of queries. In detail, in the i-th round, Paradigm 1 picks out two contention points randomly to query; while Paradigm

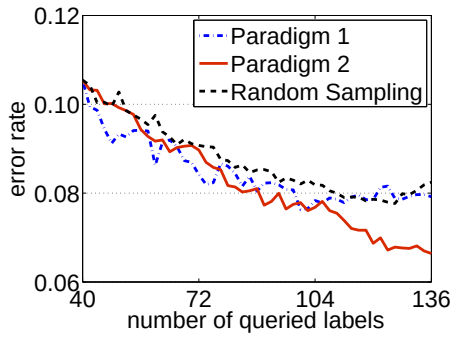


Figure 1. Comparison of the performances

2 picks out the example with the smallest absolute sum of the two classifiers' outputs from $\mathcal{Q}_i$ and $\mathcal{U}-\mathcal{Q}_i$ respectively to query, and picks out the example with the largest absolute sum of the two classifiers' outputs from $\mathcal{Q}_i$ and $\mathcal{U}-\mathcal{Q}_i$ respectively to label as $sign\big(h_1^i(x_1)+h_2^i(x_2)\big)$. That is, the two examples to be queried in Paradigm 2 are $\arg\min_{x\in\mathcal{Q}_i}\big(|h_1^i(x_1)+h_2^i(x_2)|\big)$ and $\arg\min_{x\in\mathcal{U}-\mathcal{Q}_i}\big(|h_1^i(x_1)+h_2^i(x_2)|\big)$, while the two examples Paradigm 2 labels for itself by semi-supervised learning are $\arg\max_{x\in\mathcal{Q}_i}\big(|h_1^i(x_1)+h_2^i(x_2)|\big)$ and $\arg\max_{x\in\mathcal{U}-\mathcal{Q}_i}\big(|h_1^i(x_1)+h_2^i(x_2)|\big)$. We use Random Sampling as the baseline and implement the classifiers with SMO in WEKA (Witten & Frank, 2005). The experiments are repeated for 20 runs and Figure 1 plots the average error rates of the three methods against the number of examples that have been queried.

It can be found from Figure 1 that with the same number of queried examples, although there are some fluctuation, the performance of Paradigm 1 is generally better than that of Random Sampling, while the performance of Paradigm 2 is better than that of the others. In particular, the advantage of Paradigm 2 becomes more prominent as the number of queries increases. This is not difficult to understand since with more labeled data the learners become stronger and thus the labels obtained from the semi-supervised learning process become more helpful.

Overall, the empirical study verifies that comparing with pure active learning, the combination of multi-view active learning and semi-supervised learning can reduce the sample complexity.

6. Conclusion

In this paper, we first characterize the sample complexity of multi-view active learning and get an exponential improvement in the sample complexity from $\widetilde{O}(\frac{1}{\epsilon})$ to $\widetilde{O}(\log\frac{1}{\epsilon})$. The α-*expansion* assumption we employed is weaker than assumptions taken by previous theoretical studies on active learning, such as that the data is distributed uniformly over the unit sphere in $\mathbb{R}^d$ and that the hypothesis class is linear separators through the origin. We also give an upper bound of the error rate for cases where the α-*expansion* assumption does not hold. Then, we analyze the combination of multi-view active learning with semi-supervised learning and get that such a combination can reduce the sample complexity further, which is verified by an empirical study. This provides an explanation to that why the method described in (Zhou et al., 2006) can lead to good results.

Our work is the first theoretical analysis on the sample complexity of realizable multi-view active learning. Recently, *non-realizable* active learning, where there does not exist a hypothesis perfectly separating the data, starts to attract attention (Balcan et al., 2006; Balcan et al., 2007; Dasgupta et al., 2008). Extending our work to non-realizable multi-view active learning is a future work.

Acknowledgments

This research was supported by the National Science Foundation of China (60635030, 60721002), the Foundation for the Author of National Excellent Doctoral Dissertation of China (200343) and the National High Technology Research and Development Program of China (2007AA01Z169).

References

Anthony, M., & Bartlett, P. L. (Eds.). (1999). *Neural network learning: Theoretical foundations*. Cambridge, UK: Cambridge University Press.

Balcan, M.-F., Beygelzimer, A., & Langford, J. (2006). Agnostic active learning. *Proceedings of the 23rd International Conference on Machine Learning* (pp. 65–72). Pittsburgh, PA.

Balcan, M.-F., Blum, A., & Yang, K. (2005). Co-training and expansion: Towards bridging theory and practice. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*, 89–96. Cambridge, MA: MIT Press.

Balcan, M.-F., Broder, A. Z., & Zhang, T. (2007). Margin based active learning. *Proceedings of the 20th Annual Conference on Learning Theory* (pp. 35–50). San Diego, CA.

Belkin, M., Matveeva, I., & Niyogi, P. (2001). Reg-

ularization and semi-supervised learning on large graphs. *Proceedings of the 17th Annual Conference on Learning Theory* (pp. 624–638). Banff, Canada.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory* (pp. 92–100). Madison, WI.

Cohn, D. A., Atlas, L. E., & Ladner, R. E. (1994). Improving generalization with active learning. *Machine Learning*, *15*, 201–221.

Dasgupta, S. (2006). Coarse sample complexity bounds for active learning. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*, 235–242. Cambridge, MA: MIT Press.

Dasgupta, S., Hsu, D., & Monteleoni, C. (2008). A general agnostic active learning algorithm. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in neural information processing systems 20*, 353–360. Cambridge, MA: MIT Press.

Dasgupta, S., Kalai, A. T., & Monteleoni, C. (2005). Analysis of perceptron-based active learning. *Proceedings of the 18th Annual Conference on Learning Theory* (pp. 249–263). Bertinoro, Italy.

de Sa, V. R., & Ballard, D. H. (1998). Category learning through multi-modality sensing. *Neural Computation*, *10*, 1097–1117.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, *39*, 1–38.

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*, 133–168.

Goldman, S., & Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. *Proceedings of the 17th International Conference on Machine Learning* (pp. 327–334). San Francisco, CA.

Melville, P., & Mooney, R. J. (2004). Diverse ensembles for active learning. *Proceedings of the 21st International Conference on Machine Learning* (pp. 584–591). Banff, Canada.

Miller, D. J., & Uyar, H. S. (1997). A mixture of experts classifier with learning based on both labelled and unlabelled data. In M. Mozer, M. I. Jordan and T. Petsche (Eds.), *Advances in neural information processing systems 9*, 571–577. Cambridge, MA: MIT Press.

Muslea, I., Minton, S., & Knoblock, C. A. (2000). Selective sampling with redundant views. *Proceedings of the 17th National Conference on Artificial Intelligence* (pp. 621–626). Austin, TX.

Muslea, I., Minton, S., & Knoblock, C. A. (2006). Active learning with multiple views. *Journal of Artificial Intelligence Research*, *27*, 203–233.

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

Tong, S., & Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, *2*, 45–66.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, CA: Morgan Kaufmann. 2nd edition.

Zhou, D., Schölkopf, B., & Hofmann, T. (2005). Semi-supervised learning on directed graphs. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*, 1633–1640. Cambridge, MA: MIT Press.

Zhou, Z.-H., Chen, K.-J., & Dai, H.-B. (2006). Enhancing relevance feedback in image retrieval using unlabeled data. *ACM Transactions on Information Systems*, *24*, 219–244.

Zhou, Z.-H., & Li, M. (2005). Semi-supervised learning with co-training. *Proceedings of the 19th International Joint Conference on Artificial Intelligence* (pp. 908–913). Edinburgh, Scotland.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. *Proceedings of the 20th International Conference on Machine Learning* (pp. 912–919). Washington, DC.

Fast Solvers and Efficient Implementations for Distance Metric Learning

Kilian Q. Weinberger KILIAN@YAHOO-INC.COM
Yahoo! Research, 2821 Mission College Blvd, Santa Clara, CA 9505

Lawrence K. Saul SAUL@CS.UCSD.EDU
CSE Department, University of California, San Diego 9500 Gilman Drive, La Jolla, CA 92093-0404

Abstract

In this paper we study how to improve nearest neighbor classification by learning a Mahalanobis distance metric. We build on a recently proposed framework for distance metric learning known as large margin nearest neighbor (LMNN) classification. Our paper makes three contributions. First, we describe a highly efficient solver for the particular instance of semidefinite programming that arises in LMNN classification; our solver can handle problems with billions of large margin constraints in a few hours. Second, we show how to reduce both training and testing times using metric ball trees; the speedups from ball trees are further magnified by learning low dimensional representations of the input space. Third, we show how to learn different Mahalanobis distance metrics in different parts of the input space. For large data sets, the use of locally adaptive distance metrics leads to even lower error rates.

1. Introduction

Many algorithms for pattern classification and machine learning depend on computing distances in a multidimensional input space. Often, these distances are computed using a Euclidean distance metric—a choice which has both the advantages of simplicity and generality. Notwithstanding these advantages, though, the Euclidean distance metric is not very well adapted to most problems in pattern classification.

Viewing the Euclidean distance metric as overly simplistic, many researchers have begun to ask how to learn or adapt the distance metric itself in order to achieve better results (Xing et al., 2002; Chopra et al., 2005; Frome et al., 2007). Distance metric learning is an emerging area of statistical learning in which the goal is to induce a more powerful distance metric from labeled examples. The simplest instance of this problem arises in the context of k-nearest neighbor (kNN) classification using Mahalanobis distances. Mahalanobis distances are computed by linearly transforming the input space, then computing Euclidean distances in the transformed space. A well-chosen linear transformation can improve kNN classification by decorrelating and reweighting elements of the feature vector. In fact, significant improvements have been observed within several different frameworks for this problem, including neighborhood components analysis (Goldberger et al., 2005), large margin kNN classification (Weinberger et al., 2006), and information-theoretic metric learning (Davis et al., 2007).

These studies have established the general utility of distance metric learning for kNN classification. However, further work is required to explore its promise in more difficult regimes. In particular, larger data sets raise new and important challenges in scalability. They also present the opportunity to learn more adaptive and sophisticated distance metrics.

In this paper, we study these issues as they arise in the recently proposed framework of large margin nearest neighbor (LMNN) classification (Weinberger et al., 2006). In this framework, a Mahalanobis distance metric is trained with the goal that the k-nearest neighbors of each example belong to the same class while examples from different classes are separated by a large margin. Simple in concept, useful in practice, the ideas behind LMNN classification have also inspired other related work in machine learning and computer vision (Torresani & Lee, 2007; Frome et al., 2007).

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

The role of the margin in LMNN classification is inspired by its role in support vector machines (SVMs). Not surprisingly, given these roots, LMNN classification also inherits various strengths and weaknesses of SVMs (Schölkopf & Smola, 2002). For example, as in SVMs, the training procedure in LMNN classification reduces to a convex optimization based on the hinge loss. However, as described in section 2, naïve implementations of this optimization do not scale well to larger data sets.

Addressing the challenges and opportunities raised by larger data sets, this paper makes three contributions. First, we describe how to optimize the training procedure for LMNN classification so that it can readily handle data sets with tens of thousands of training examples. In order to scale to this regime, we have implemented a special-purpose solver for the particular instance of semidefinite programming that arises in LMNN classification. In section 3, we describe the details of this solver, which we have used to tackle problems involving billions of large margin constraints. To our knowledge, problems of this size have yet to be tackled by other recently proposed methods (Goldberger et al., 2005; Davis et al., 2007) for learning Mahalanobis distance metrics.

As the second contribution of this paper, we explore the use of metric ball trees (Liu et al., 2005) for LMNN classification. These data structures have been widely used to accelerate nearest neighbor search. In section 4, we show how similar data structures can be used for faster training and testing in LMNN classification. Ball trees are known to work best in input spaces of low to moderate dimensionality. Mindful of this regime, we also show how to modify the optimization in LMNN so that it learns a low-rank Mahalanobis distance metric. With this modification, the metric can be viewed as projecting the original inputs into a lower dimensional space, yielding further speedups.

As the third contribution of this paper, we describe an important extension to the original framework for LMNN classification. Specifically, in section 5, we show how to learn different Mahalanobis distance metrics for different parts of the input space. The novelty of our approach lies in learning a collection of different local metrics to maximize the margin of correct kNN classification. The promise of this approach is suggested by recent, related work in computer vision that has achieved state-of-the-art results on image classification (Frome et al., 2007). Our particular approach begins by partitioning the training data into disjoint clusters using class labels or unsupervised methods. We then learn a Mahalanobis distance metric for each cluster. While the training procedure couples the distance metrics in different clusters, the optimization remains a convex problem in semidefinite programming. The globally coupled training of these metrics also distinguishes our approach from earlier work in adaptive distance metrics for kNN classification (Hastie & Tibshirani, 1996). To our knowledge, our approach yields the best kNN test error rate on the extensively benchmarked MNIST data set of handwritten digits that does not incorporate domain-specific prior knowledge (LeCun et al., 1998; Simard et al., 1993). Thus, our results show that we can exploit large data sets to learn more powerful and adaptive distance metrics for kNN classification.

2. Background

Of the many settings for distance metric learning, the simplest instance of the problem arises in the context of kNN classification using Mahalanobis distances. A Mahalanobis distance metric computes the squared distances between two points $\vec{x}_i$ and $\vec{x}_j$ as:

$$d^2_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^\top \mathbf{M} (\vec{x}_i - \vec{x}_j), \qquad (1)$$

where $\mathbf{M} \succeq 0$ is a positive semidefinite matrix. When $\mathbf{M}$ is equal to the identity matrix, eq. (1) reduces to the Euclidean distance metric. In distance metric learning, the goal is to discover a matrix $\mathbf{M}$ that leads to lower kNN error rates than the Euclidean distance metric.

Here we briefly review how Mahalanobis distance metrics are learned for LMNN classification (Weinberger et al., 2006). Let the training data consist of n labeled examples $\{(\vec{x}_i, y_i)\}_{i=1}^n$ where $\vec{x}_i \in \mathcal{R}^d$ and $y_i \in \{1, \ldots, c\}$, where c is the number of classes. For LMNN classification, the training procedure has two steps. The first step identifies a set of k similarly labeled *target neighbors* for each input $\vec{x}_i$. Target neighbors are selected by using prior knowledge (if available) or by simply computing the k nearest (similarly labeled) neighbors using Euclidean distance. We use the notation $j \rightsquigarrow i$ to indicate that $\vec{x}_j$ is a target neighbor of $\vec{x}_i$. The second step adapts the Mahalanobis distance metric so that these target neighbors are closer to $\vec{x}_i$ than all other differently labeled inputs. The Mahalanobis distance metric is estimated by solving a problem in semidefinite programming. Distance metrics obtained in this way were observed to yield consistent and significant improvements in kNN error rates.

The semidefinite program in LMNN classification arises from an objective function which balances two terms. The first term penalizes large distances between inputs and their target neighbors. The second term penalizes small distances between differently la-

beled inputs; specifically, a penalty is incurred if these distances do not exceed (by a finite margin) the distances to the target neighbors of these inputs. The terms in the objective function can be made precise with further notation. Let $y_{ij} \in \{0, 1\}$ indicate whether the inputs $\vec{x}_i$ and $\vec{x}_j$ have the same class label. Also, let $\xi_{ijl} \geq 0$ denote the amount by which a differently labeled input $\vec{x}_l$ invades the "perimeter" around input $\vec{x}_i$ defined by its target neighbor $\vec{x}_j$. The Mahalanobis distance metric $\mathbf{M}$ is obtained by solving the following semidefinite program:

Minimize $\sum_{j \rightsquigarrow i} \left[d^2_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) + \mu \sum_l (1 - y_{il}) \xi_{ijl} \right]$
subject to:
(a) $d^2_{\mathbf{M}}(\vec{x}_i, \vec{x}_l) - d^2_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) \;\geq\; 1 - \xi_{ijl}$
(b) $\xi_{ijl} \geq 0$
(c) $\mathbf{M} \succeq 0$.

The constant μ defines the trade-off between the two terms in the objective function; in our experiments, we set $\mu = 1$. The constraints of type **(a)** encourage inputs ($\vec{x}_i$) to be at least one unit closer to their k target neighbors ($\vec{x}_j$) than to any other differently labeled input ($\vec{x}_l$). When differently labeled inputs $\vec{x}_l$ invade the local neighborhood of $\vec{x}_i$, we refer to them as *impostors*. Impostors generate positive slack variables ξ_{ijl} which are penalized in the second term of the objective function. The constraints of type **(b)** enforce nonnegativity of the slack variables, and the constraint **(c)** enforces that $\mathbf{M}$ is positive semidefinite, thus defining a valid (pseudo)metric. Noting that the squared Mahalanobis distances $d^2_{\mathbf{M}}(\vec{x}_i, \vec{x}_j)$ are linear in the matrix $\mathbf{M}$, the above optimization is easily recognized as an semidefinite program.

3. Solver

The semidefinite program in the previous section grows in complexity with the number of training examples (n), the number of target neighbors (k), and the dimensionality of the input space (d). In particular, the objective function is optimized with respect to $O(kn^2)$ large margin constraints of type **(a)** and **(b)**, while the Mahalanobis distance metric $\mathbf{M}$ itself is a $d \times d$ matrix. Thus, for even moderately large and/or high dimensional data sets, the required optimization (though convex) cannot be solved by standard off-the-shelf packages (Borchers, 1999).

In order to tackle larger problems in LMNN classification, we implemented our own special-purpose solver. Our solver was designed to exploit the particular structure of the semidefinite program in the previous section. The solver iteratively re-estimates the Mahalanobis distance metric $\mathbf{M}$ to minimize the objective function for LMNN classification. The amount of computation is minimized by careful book-keeping from one iteration to the next. The speed-ups from these optimizations enabled us to work comfortably on data sets with up to $n = 60,000$ training examples.

Our solver works by eliminating the slack variables ξ_{ijl} from the semidefinite program for LMNN classification, then minimizing the resulting objective function by sub-gradient methods. The slack variables are eliminated by folding the constraints **(a)** and **(b)** into the objective function as a sum of "hinge" losses. The hinge function is defined as $[z]_+ = z$ if $z > 0$ and $[z]_+ = 0$ if $z < 0$. In terms of this hinge function, we can express ξ_{ijl} as a function of the matrix $\mathbf{M}$:

$$\xi_{ijl}(\mathbf{M}) \;=\; \left[1 + d^2_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) - d^2_{\mathbf{M}}(\vec{x}_i, \vec{x}_l)\right]_+ \qquad (2)$$

Finally, writing the objective function only in terms of the matrix $\mathbf{M}$, we obtain:

$$\varepsilon(\mathbf{M}) = \sum_{j \rightsquigarrow i} \left[d^2_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) + \mu \sum_l (1 - y_{il}) \xi_{ijl}(\mathbf{M}) \right]. \qquad (3)$$

This objective function is not differentiable due to the hinge losses that appear in eq. (2). Nevertheless, because it is convex, we can compute its sub-gradient and use standard descent algorithms to find its minimum. At each iteration of our solver, the optimization takes a step along the sub-gradient to reduce the objective function, then projects the matrix $\mathbf{M}$ back onto the cone of positive semidefinite matrices. Iterative methods of this form are known to converge to the correct solution, provided that the gradient step-size is sufficiently small (Boyd & Vandenberghe, 2004).

The gradient computation can be done most efficiently by careful book-keeping from one iteration to the next. As simplifying notation, let $\mathbf{C}_{ij} = (\vec{x}_i - \vec{x}_j)(\vec{x}_i - \vec{x}_j)^\top$. In terms of this notation, we can express the squared Mahalanobis distances in eq. (8) as:

$$d_{\mathbf{M}}(\vec{x}_i, \vec{x}_j) = \mathrm{tr}(\mathbf{C}_{ij}\mathbf{M}). \qquad (4)$$

To evaluate the gradient, we denote the matrix $\mathbf{M}$ at the t^{th} iteration as $\mathbf{M}^t$. At each iteration, we also define a set $\mathcal{N}^t$ of triplet indices such that $(i, j, l) \in \mathcal{N}^t$ if and only if the triplet's corresponding slack variable exceeds zero: $\xi_{ijl}(\mathbf{M}^t) > 0$. With this notation, we can write the gradient $\mathbf{G}^t = \frac{\partial \varepsilon}{\partial \mathbf{M}}\big|_{\mathbf{M}^t}$ at the t^{th} iteration as:

$$\mathbf{G}^t \;=\; \sum_{j \rightsquigarrow i} \mathbf{C}_{ij} + \mu \sum_{(i,j,l) \in \mathcal{N}^t} (\mathbf{C}_{ij} - \mathbf{C}_{il}). \qquad (5)$$

Computing the gradient requires computing the outer products in $\mathbf{C}_{ij}$; it thus scales quadratically in the

input dimensionality. As the set $\mathcal{N}^t$ is potentially large, a naïve computation of the gradient would be extremely expensive. However, we can exploit the fact that the gradient contribution from each active triplet (i, j, l) does not depend on the degree of its margin violation. Thus, the changes in the gradient from one iteration to the next are determined entirely by the differences between the sets $\mathcal{N}^t$ and $\mathcal{N}^{t+1}$. Using this fact, we can derive an extremely efficient update that relates the gradient at one iteration to the gradient at the previous one. The update subtracts the contributions from triples that are no longer active and adds the contributions from those that just became active:

$$\mathbf{G}^{t+1} = \mathbf{G}^t - \mu \sum_{(i,j,l)\in\mathcal{N}^t-\mathcal{N}^{t+1}} (\mathbf{C}_{ij} - \mathbf{C}_{il}) + \mu \sum_{(i,j,l)\in\mathcal{N}^{t+1}-\mathcal{N}^t} (\mathbf{C}_{ij} - \mathbf{C}_{il}) \tag{6}$$

For small gradient step sizes, the set $\mathcal{N}^t$ changes very little from one iteration to the next. In this case, the right hand side of eq. (6) can be computed very fast.

To further accelerate the solver, we adopt an active set method. This method is used to monitor the large margin constraints that are actually violated. Note that computing the set $\mathcal{N}^t$ at each iteration requires checking every triplet (i, j, l) with $j \rightsquigarrow i$ for a potential margin violation. This computation scales as $O(nd^2 + kn^2d)$, making it impractical for large data sets. To avoid this computational burden, we observe that the great majority of triples do not incur margin violations: in particular, for each training example, only a very small fraction of differently labeled examples typically lie nearby in the input space. Consequently, a useful approximation is to check only a subset of likely triples for margin violations per gradient computation and only occasionally perform the full computation. We set this active subset to the list of all triples that have ever violated the margin, ie $\bigcup_{i=1}^{t-1} \mathcal{N}^i$. When the optimization converges, we verify that the working set $\mathcal{N}^t$ does contain all active triples that incur margin violations. This final check is needed to ensure convergence to the correct minimum. If the check is not satisfied, the optimization continues with the newly expanded active set.

Table 1 shows how quickly the solver works on problems of different sizes. The results in this table were generated by learning a Mahalanobis distance metric on the MNIST data set of 28×28 grayscale handwritten digits (LeCun et al., 1998). The digits were preprocessed by principal component analysis (PCA) to reduce their dimensionality from $d = 784$ to $d = 169$. We experimented by learning a distance metric from different subsets of the training examples. The experiments were performed on a standard desktop machine with a 2.0 GHz dual core 2 processor. For each experiment, the table shows the number of training examples, the CPU time to converge, the number of active constraints, the total number of constraints, and the kNN test error (with $k = 3$). Note that for the full MNIST training set, the semidefinite program has over three billion large margin constraints. Nevertheless, the active set method converges in less than four hours—from a Euclidean distance metric with 2.33% test error to a Mahalanobis distance metric with 1.72% test error.

N	time	|active set|	|total set|	train error	test error
60	**9s**	**844**	**3.2K**	**0%**	**29.37%**
600	**37s**	**6169**	**323K**	**0%**	**10.79%**
6000	**4m**	**50345**	**32M**	**0.48%**	**3.13%**
60000	**3h25m**	**540037**	**3.2B**	**0%**	**1.72%**

Table 1. Statistics of the solver on subsets of the data set of MNIST handwritten digits. See text for details.

4. Tree-Based Search

Nearest neighbor search can be accelerated by storing training examples in hierarchical data structures (Liu et al., 2005). These data structures can also be used to reduce the training and test times for LMNN classification. In this section, we describe how these speedups are obtained using metric ball trees.

4.1. Ball trees

We begin by reviewing the use of ball trees (Liu et al., 2005) for fast kNN search. Ball trees recursively partition the training inputs by projecting them onto directions of large variance, then splitting the data on the mean or median of these projected values. Each subset of data obtained in this way defines a hypersphere (or "ball") in the multidimensional input space that encloses its training inputs. The distance to such a hypersphere can be easily computed for any test input; moreover, this distance provides a lower bound on the test input's distance to any of the enclosed training inputs. This bound is illustrated in Fig. 1. Let S be the set of training inputs inside a specific ball with radius r. The distance from a test input $\vec{x}_t$ to any training input $\vec{x}_i \in S$ is bounded from below by:

$$\forall \vec{x}_i \in S \quad \|\vec{x}_t - \vec{x}_i\| \geq \max(\|\vec{x}_t - \vec{c}\|_2 - r, 0). \tag{7}$$

These bounds can be exploited in a tree-based search for nearest neighbors. In particular, if the distance to the currently k^{th} closest input $\vec{x}_j$ is smaller than the bound from eq. (7), then all inputs inside the ball S can be pruned away. This pruning of unexplored subtrees can significantly accelerate kNN search. The same basic strategy can also be applied to kNN search

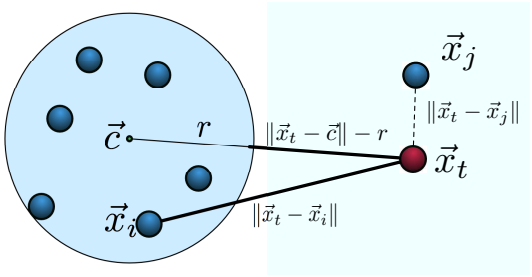


Figure 1. How ball trees work: for any input $\vec{x}_i \in S$, the distance $\|\vec{x}_t - \vec{x}_i\|$ is bounded from below by eq. (7) . If a training example $\vec{x}_j$ is known to be closer to $\vec{x}_t$, then the inputs inside the ball can be ruled out as nearest neighbors.

under a Mahalanobis distance metric.

4.2. Speedups

We first experimented with ball trees to reduce the *test* times for LMNN classification. In our experiments, we observed a factor of $3x$ speed-up for 40-dimensional data and a factor of $15x$ speedup for 15-dimensional data. Note that these speedups were measured relative to a highly optimized baseline implementation of kNN search. In particular, our baseline implementation rotated the input space to align its coordinate axes with the principal components of the data; the coordinate axes were also sorted in decreasing order of variance. In this rotated space, distance computations were terminated as soon as any partially accumulated results (from leading principal components) exceeded the currently smallest k distances from the kNN search in progress.

We also experimented with ball trees to reduce the *training* times for LMNN classification. To reduce training times, we integrated ball trees into our special-purpose solver. Specifically, ball trees were used to accelerate the search for so-called "impostors". Recall that for each training example $\vec{x}_i$ and for each of its similarly labeled target neighbors $\vec{x}_j$, the impostors consist of all differently labeled examples $\vec{x}_l$ with $d_M(\vec{x}_i, \vec{x}_l)^2 \leq d_M(\vec{x}_i, \vec{x}_j)^2 + 1$. The search for impostors dominates the computation time in the training procedure for LMNN classification. To reduce the amount of computation, the solver described in section 3 maintains an active list of previous margin violations. Nevertheless, the overall computation still scales as $O(n^2 d)$, which can be quite expensive. Note that we only need to search for impostors among training examples with different class labels. To speed up training, we built one ball tree for the training examples in each class and used them to search for impostors (as the ball-tree creation time is negligible in comparison with the impostor search, we re-built it in every iteration). We observed the ball trees to yield speedups ranging from a factor of $1.9x$ with 10-dimensional data to a factor of $1.2x$ with 100 dimensional data.

4.3. Dimensionality reduction

Across all our experiments, we observed that the gains from ball trees diminished rapidly with the dimensionality of the input space. This observation is consistent with previous studies of ball trees and NN search. When the data is high dimensional, NN search is plagued by the so-called "curse of dimensionality". In particular, distances in high dimensions tend to be more uniform, thereby reducing the opportunities for pruning large subtrees.

The curse of dimensionality is often addressed in ball trees by projecting the stored training inputs into a lower dimensional space. The most commonly used methods for dimensionality reduction are random projections and PCA. Despite their widespread use, however, neither of these methods is especially geared to preserve (or improve) the accuracy of kNN classification.

We experimented with two methods for dimensionality reduction in the particular context of LMNN classification. Both methods were based on learning a low-rank Mahalanobis distance metric. Such a metric can be viewed as projecting the original inputs into a lower dimensional space. In our first approach, we performed a singular value decomposition (SVD) on the full rank solution to the semidefinite program in section 2. The full rank solution for the distance metric was then replaced by a low rank approximation based on its leading eigenvectors. We call this approach LMNN-SVD. In our second approach, we followed a suggestion from previous work on LMNN classification (Torresani & Lee, 2007). In this approach, we explicitly parameterized the Mahalanobis distance metric as a low-rank matrix, writing $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$, where $\mathbf{L}$ is a rectangular matrix. To obtain the distance metric, we optimized the same objective function as before, but now in terms of the explicitly low-rank linear transformation $\mathbf{L}$. The optimization over $\mathbf{L}$ is not convex unlike the original optimization over $\mathbf{M}$, but a (possibly local) minimum can be computed by standard gradient-based methods. We call this approach LMNN-RECT.

Fig. 2 shows the results of kNN classification from both these methods on the MNIST data set of handwritten digits. For these experiments, the raw MNIST images (of size 28× 28) were first projected onto their 350 leading principal components. The training procedure for LMNN-SVD optimized a full-rank distance metric in this 350 dimensional space, then extracted a low-rank distance metric from its leading eigenvectors.

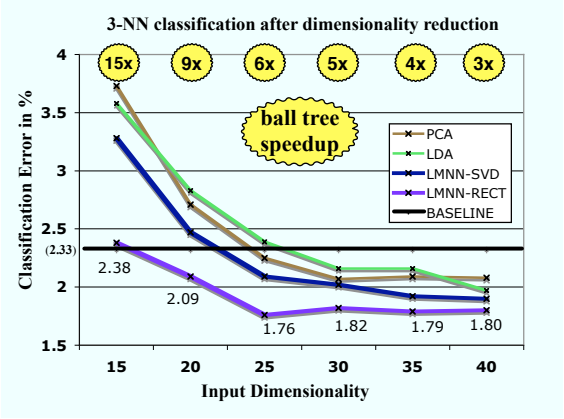


Figure 2. Graph of kNN error rate (with $k = 3$) on different low dimensional representations of the MNIST data set.

The training procedures for LMNN-RECT optimized a low-rank rectangular matrix of size $r \times 350$, where r varied from 15 to 40. Also shown in the figure are the results from further dimensionality reduction using PCA, as well as the baseline kNN error rate in the original (high dimensional) space of raw images. The speedup from ball trees is shown at the top of the graph. The amount of speedup depends significantly on the amount of dimensionality reduction, but very little on the particular method of dimensionality reduction. Of the three methods compared in the figure, LMNN-RECT is the most effective, improving significantly over baseline kNN classification while operating in a much lower dimensional space. Overall, these results show that aggressive dimensionality reduction can be combined with distance metric learning.

5. Multiple Metrics

The originally proposed framework for LMNN classification has one clear limitation: the same Mahalanobis distance metric is used to compute distances everywhere in the input space. Writing the metric as $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$, we see that Mahalanobis distances are equivalent to Euclidean distances after a global linear transformation $\vec{x} \rightarrow \mathbf{L}\vec{x}$ of the input space. Such a transformation cannot adapt to nonlinear variabilities in the training data.

In this section, we describe how to learn different Mahalanobis distance metrics in different parts of the input space. We begin by simply describing how such a collection of local distance metrics is used at test time. Assume that the data set is divided into p disjoint partitions $\{P_\alpha\}_{\alpha=1}^p$, such that $P_\alpha \cap P_\beta = \{\}$ for any $\alpha \neq \beta$ and $\bigcup_\alpha P_\alpha = \{\vec{x}_i\}_{i=1}^n$. Also assume that each partition P_α has its own Mahalanobis distance metric $\mathbf{M}_\alpha$ for use in kNN classification. Given a test vector $\vec{x}_t$, we compute its squared distance to a training input $\vec{x}_i$ in partition α_i as:

$$d^2_{\mathbf{M}_{\alpha_i}}(\vec{x}_t, \vec{x}_i) = (\vec{x}_t - \vec{x}_i)^\top \mathbf{M}_{\alpha_i}(\vec{x}_t - \vec{x}_i). \qquad (8)$$

These distances are then sorted as usual to determine nearest neighbors and label the test input. Note, however, how different distance metrics are used for training inputs in different partitions.

We can also use these metrics to compute distances between training inputs, with one important caveat. Note that for inputs belonging to different partitions, the distance between them will depend on the particular metric used to compute it. This asymmetry does not present any inherent difficulty since, in fact, the dissimilarity measure in kNN classification is not required to be symmetric. Thus, even on the training set, we can use multiple metrics to measure distances and compute meaningful leave-one-out kNN error rates.

5.1. Learning algorithm

In this section we describe how to learn multiple Mahalanobis distance metrics for LMNN classification. Each of these metrics is associated with a particular cluster of training examples. To derive these clusters, we experimented with both unsupervised methods, such as the k-means algorithm, and fully supervised methods, in which each cluster contains the training examples belonging to a particular class.

Before providing details of the learning algorithm, we make the following important observation. Multiple Mahalanobis distance metrics for LMNN classification cannot be learned in a decoupled fashion—that is, by solving a collection of simpler, independent problems of the type already considered (e.g., one within each partition of training examples). Rather, the metrics must be learned in a coordinated fashion so that the distances from different metrics can be meaningfully compared for kNN classification. In our framework, such comparisons arise whenever an unlabeled test example has potential nearest neighbors in two or more different clusters of training examples.

Our learning algorithm for multiple local distance metrics $\{\mathbf{M}_\alpha\}_{\alpha=1}^p$ generalizes the semidefinite program for ordinary LMNN classification in section 2. First, we modify the objective function so that the distances to target neighbors $\vec{x}_j$ are measured under the metric $\mathbf{M}_{\alpha_j}$. Next, we modify the large margin constraints in **(a)** so that the distances to potential impostors $\vec{x}_l$ are measured under the metric $\mathbf{M}_{\alpha_l}$. Finally, we replace the single positive semidefinite constraint in **(c)**

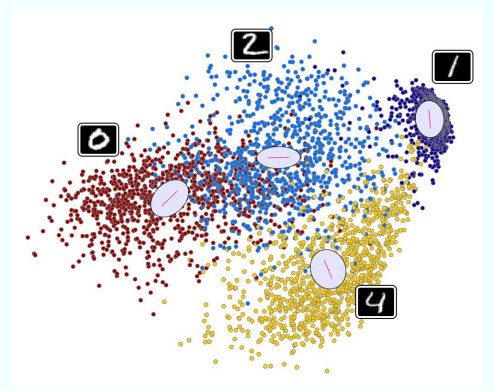


Figure 3. Visualization of multiple local distance metrics for MNIST handwritten digits. See text for details.

by multiple such constraints, one for each local metric $\mathbf{M}_\alpha$. Taken together, these steps lead to the new semidefinite program:

Minimize $\sum_{j \rightsquigarrow i} \left[d^2_{\mathbf{M}_{\alpha_j}}(\vec{x}_i, \vec{x}_j) + \mu \sum_l (1 - y_{il}) \xi_{ijl} \right]$
subject to:
(a) $d^2_{\mathbf{M}_{\alpha_l}}(\vec{x}_i, \vec{x}_l) - d^2_{\mathbf{M}_{\alpha_j}}(\vec{x}_i, \vec{x}_j) \;\geq\; 1 - \xi_{ijl}$
(b) $\xi_{ijl} \geq 0$
(c) $\mathbf{M}_\alpha \succeq 0$.

Note how the new constraints in **(a)** couple the different Mahalanobis distance metrics. By jointly optimizing these metrics to minimize a single objective function, we ensure that the distances they compute can be meaningfully compared for kNN classification.

5.2. Results

We evaluated the performance of this approach on five publicly available data sets: the MNIST data set[1] of handwritten digits ($n = 60000$, $c = 10$), the 20-Newsgroups data set[2] of text messages ($n = 18827$, $c = 20$), the Letters data set[3] of distorted computer fonts ($n = 14000$, $c = 26$), the Isolet data set[4] of spoken letters ($n = 6238$, $c = 26$), and the YaleFaces[5] data set of face images ($n = 1690$, $c = 38$). The data sets were preprocessed by PCA to reduce their dimensionality. The amount of dimensionality reduction varied with each experiment, as discussed below.

To start, we sought to visualize the multiple metrics learned in a simple experiment on MNIST handwritten digits of zeros, ones, twos, and fours. For ease of visualization, we worked with only the leading two principal components of the MNIST data set. Fig. 3 shows these two dimensional inputs, color-coded by class label. With these easily visualized inputs, we minimized the objective function in section 5.1 to learn a specialized distance metric for each type of handwritten digit. The ellipsoids in the plot reveal the directions amplified by the local distance metric of each digit class. Notably, each distance metric learns to amplify the direction perpendicular to the decision boundary for the nearest, competing class of digits.

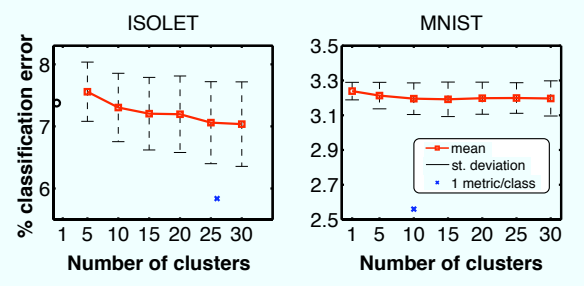


Figure 4. Test kNN error rates on the Isolet and MNIST data sets as a function of the number of distance metrics.

Our next experiments examined the performance of LMNN classification as a function of the number of distance metrics. In these experiments, we used PCA to reduce the input dimensionality to $d = 50$; we also only worked with a subset of $n = 10000$ training examples of MNIST handwritten digits. To avoid overfitting, we used an "early stopping" approach while monitoring the kNN error rates on a held-out validation set consisting of 30% of the training data.

Fig. 4 shows the test kNN error rates on the Isolet and MNIST data sets as a function of the number of distance metrics. In these experiments, we explored both unsupervised and supervised methods for partitioning the training inputs as a precursor to learning local distance metrics. In the unsupervised setting, the training examples were partitioned by k-means clustering, with the number of clusters ranging from 1 to 30 (just 1 cluster is identical to single-matrix LMNN). As k-means clustering is prone to local minima, we averaged these results over 100 runs. The figure shows the average test error rates in red, as well as their standard deviations (via error bars). In the supervised setting, the training examples were partitioned by their class labels, resulting in the same number of clusters as classes. The test error rates in these experiments are shown as blue crosses. In both the unsupervised and supervised settings, the test error rates decreased with the use of multiple metrics. However, the improvements were far greater in the supervised setting.

[1] http://yann.lecun.com/exdb/mnist/
[2] http://people.csail.mit.edu/jrennie/20Newsgroups
[3] http://www.ics.uci.edu/~mlearn/databases/letter-recognition/letter-recognition.names
[4] http://archive.ics.uci.edu/ml/
[5] http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html

	Error in %	mnist	20news	letters	isolet	yalefaces
train	LMNN	1.72	14.91	3.62	3.59	**6.48**
	Multiple Metrics	**1.18**	**13.66**	**3.2**	**3.08**	6.4
test	LMNN	1.19	9.73	3.54	0.7	**3.54**
	Multiple Metrics	**0.04**	**7.08**	**1.55**	**0**	3.57

Figure 5. The classification train- and testerror rates with one metric (LMNN) and multiple metrics. The value of k was set by cross validation.

Finally, our last experiments explored the improvement in kNN error rates when one distance metric was learned for the training examples in each class. In these experiments, we used the full number of training examples for each data set. In addition, we used PCA to project the training inputs into a lower dimensional subspace accounting for at least 95% of the data's total variance. Fig. 5 shows generally consistent improvement in training and test kNN error rates, though overfitting is an issue, especially on the 20-NewsGroups and YaleFaces data sets. This overfitting is to be expected from the relatively large number of classes and high input dimensionality of these data sets: the number of model parameters in these experiments grows linearly in the former and quadratically in the latter. On these data sets, only the use of a validation set prevents the training error from vanishing completely while the test error skyrockets. On the other hand, a significant improvement in the test error rate is observed on the largest data set, that of MNIST handwritten digits. On this data set, multiple distance metrics yield a 1.18% test error rate—a highly competitive result for a method that does not take into account prior domain knowledge (LeCun et al., 1998).

6. Discussion

In this paper, we have extended the original framework for LMNN classification in several important ways: by describing a solver that scales well to larger data sets, by integrating metric ball trees into the training and testing procedures, by exploring the use of dimensionality reduction for further speedups, and by showing how to train different Mahalanobis distance metrics in different parts of the input space. These extensions should prove useful in many applications of kNN classification. More generally, we also hope they spur further work on problems in distance metric learning and large-scale semidefinite programming, both areas of great interest in the larger field of machine learning.

Acknowledgments

This research is based upon work supported by the National Science Foundation under Grant No. 0238323.

References

Borchers, B. (1999). CSDP, a C library for semidefinite programming. *Optimization Methods and Software 11(1):613-623*.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a similiarty metric discriminatively, with application to face verification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-05)*. San Diego, CA.

Davis, J., Kulis, B., Jain, P., Sra, S., & Dhillon, I. (2007). Information-theoretic metric learning. *Proceedings of the 24th International Conference on Machine Learning (ICML-07)*. Corvallis, OR.

Frome, A., Singer, Y., Sha, F., & Malik, J. (2007). Learning globally-consistent local distance functions for shape-based image retrieval and classification. *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07)*. Rio de Janeiro, Brazil.

Goldberger, J., Roweis, S., Hinton, G., & Salakhutdinov, R. (2005). Neighbourhood components analysis. *Advances in Neural Information Processing Systems 17* (pp. 513–520). Cambridge, MA: MIT Press.

Hastie, T., & Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *18*, 607–616.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86(11)*, 2278–2324.

Liu, T., Moore, A. W., Gray, A., & Yang, K. (2005). An investigation of practical approximate nearest neighbor algorithms. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*, 825–832. Cambridge, MA: MIT Press.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge, MA: MIT Press.

Simard, P. Y., LeCun, Y., & Decker, J. (1993). Efficient pattern recognition using a new transformation distance. *Advances in Neural Information Processing Systems* (pp. 50–58). San Mateo, CA: Morgan Kaufman.

Torresani, L., & Lee, K. (2007). Large margin component analysis. In B. Schölkopf, J. Platt and T. Hofmann (Eds.), *Advances in neural information processing systems 19*. Cambridge, MA: MIT Press.

Weinberger, K., Blitzer, J., & Saul, L. (2006). Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf and J. Platt (Eds.), *Advances in neural information processing systems 18*. Cambridge, MA: MIT Press.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2002). Distance metric learning, with application to clustering with side-information. *Advances in Neural Information Processing Systems 14*. Cambridge, MA: MIT Press.

Deep Learning via Semi-Supervised Embedding

Jason Weston[*] JASONW@NEC-LABS.COM
Frédéric Ratle[†] FREDERIC.RATLE@GMAIL.COM
Ronan Collobert[*] COLLOBER@NEC-LABS.COM
(∗) NEC Labs America, 4 Independence Way, Princeton, NJ 08540 USA
(†) IGAR, University of Lausanne, Amphipôle, 1015 Lausanne, Switzerland

Abstract

We show how nonlinear embedding algorithms popular for use with *shallow* semi-supervised learning techniques such as kernel methods can be applied to deep multilayer architectures, either as a regularizer at the output layer, or on each layer of the architecture. This provides a simple alternative to existing approaches to *deep* learning whilst yielding competitive error rates compared to those methods, and existing *shallow* semi-supervised techniques.

1. Introduction

Embedding data into a lower dimensional space or the related task of clustering are unsupervised dimensionality reduction techniques that have been intensively studied. Most algorithms are developed with the motivation of producing a useful analysis and visualization tool.

Recently, the field of semi-supervised learning (Chapelle et al., 2006), which has the goal of improving generalization on supervised tasks using unlabeled data, has made use of many of these techniques. For example, researchers have used nonlinear embedding or cluster representations as features for a supervised classifier, with improved results.

Most of these architectures are *disjoint* and *shallow*, by which we mean the unsupervised dimensionality reduction algorithm is trained on unlabeled data separately as a first step, and then its results are fed to a supervised classifier which has a shallow architecture such as a (kernelized) linear model. For example, several methods learn a clustering or a distance measure based on a nonlinear manifold embedding as a first step (Chapelle et al., 2003; Chapelle & Zien, 2005). Transductive Support Vector Machines (TSVMs) (Vapnik, 1998) (which employs a kind of clustering) and LapSVM (Belkin et al., 2006) (which employs a kind of embedding) are examples of methods that are *joint* in their use of unlabeled data and labeled data, but their architecture is still *shallow*.

Deep architectures seem a natural choice in hard AI tasks which involve several *sub-tasks* which can be coded into the layers of the architecture. As argued by several researchers (Hinton et al., 2006; Bengio et al., 2007) semi-supervised learning is also natural in such a setting as otherwise one is not likely to ever have enough labeled data to perform well.

Several authors have recently proposed methods for using unlabeled data in deep neural network-based architectures. These methods either perform a greedy layer-wise pre-training of weights using unlabeled data alone followed by supervised fine-tuning (which can be compared to the *disjoint* shallow techniques for semi-supervised learning described before), or learn unsupervised encodings at multiple levels of the architecture jointly with a supervised signal. Only considering the latter, the basic setup we advocate is simple:

1. Choose an unsupervised learning algorithm.
2. Choose a model with a deep architecture.
3. The unsupervised learning is plugged into any (or all) layers of the architecture as an *auxiliary task*.
4. Train supervised and unsupervised tasks using the same architecture *simultaneously*.

The aim is that the unsupervised method will improve accuracy on the task at hand. However, the unsupervised methods so far proposed for deep architectures are in our opinion somewhat complicated and

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

restricted. They include a particular kind of generative model (a restricted Boltzmann machine) (Hinton et al., 2006), autoassociators (Bengio et al., 2007), and a method of sparse encoding (Ranzato et al., 2007). Moreover, in all cases these methods are not compared with, and appear on the surface to be completely different to, algorithms developed by researchers in the field of semi-supervised learning.

In this article we advocate simpler ways of performing deep learning by leveraging *existing* ideas from semi-supervised algorithms so far developed in *shallow* architectures. In particular, we focus on the idea of combining an *embedding*-based regularizer with a supervised learner to perform semi-supervised learning, such as is used in Laplacian SVMs (Belkin et al., 2006). We show that this method can be: (i) generalized to multi-layer networks and trained by stochastic gradient descent; and (ii) is valid in the *deep* learning framework given above.

Our experimental evaluation is then split into three parts: (i) stochastic training of semi-supervised multi-layered architectures is compared with existing semi-supervised approaches on several benchmarks, with positive results; (ii) a demonstration of how to use semi-supervised regularizers in *deep* architectures by plugging them into any layer of the architecture is shown on the well-known MNIST dataset; and (iii) a case-study is presented using these techniques for deep-learning of semantic role labeling of English sentences.

The rest of the article is as follows. In Section 2 we describe existing techniques for semi-supervised embedding. In Section 3 we describe how to generalize these techniques to the task of *deep learning*. Section 4 reviews existing techniques for deep learning, Section 5 gives an experimental comparison between all these approaches, and Section 6 concludes.

2. Semi-Supervised Embedding

A key assumption in many semi-supervised algorithms is the structure assumption[1]: points within the same structure (such as a cluster or a manifold) are likely to have the same label. Given this assumption, the aim is to use unlabeled data to uncover this structure. In order to do this many algorithms such as cluster kernels (Chapelle et al., 2003), LDS (Chapelle & Zien, 2005), label propagation (Zhu & Ghahramani, 2002) and LapSVM (Belkin et al., 2006), to name a few, make use of regularizers that are directly related to unsupervised embedding algorithms. To understand these methods we will first review some relevant approaches to linear and nonlinear embedding.

[1]This is often referred to as the cluster assumption or the manifold assumption (Chapelle et al., 2006).

2.1. Embedding Algorithms

We will focus on a rather general class of embedding algorithms that can be described by the following type of optimization problem: given the data $x_1, \ldots, x_U$ find an embedding $f(x_i)$ of each point x_i by minimizing

$$\sum_{i,j=1}^{U} L(f(x_i, \alpha), f(x_j, \alpha), W_{ij})$$

w.r.t. α, subject to

Balancing constraint.

This type of optimization problem has the following main ingredients:

- $f(x) \in \mathbb{R}^n$ is the embedding one is trying to learn for a given example $x \in \mathbb{R}^d$. It is parametrized by α. In many techniques $f(x_i) = f_i$ is a lookup table where each example i is assigned an independent vector f_i.
- L is a loss function between pairs of examples.
- The matrix W of weights W_{ij} specifying the similarity or dissimilarity between examples x_i and x_j. This is supplied in advance and serves as a kind of label for the loss function.
- A balancing constraint is often required for certain objective functions so that a trivial solution is not reached.

Many well known algorithms fit into this framework.

Multidimensional scaling (MDS) is a classical algorithm that attempts to preserve the distance between points, whilst embedding them in a lower dimensional space, e.g. by using the loss function

$$L(f_i, f_j, W_{ij}) = (||f_i - f_j|| - W_{ij})^2$$

MDS is equivalent to PCA if the metric is Euclidean (Williams, 2001).

ISOMAP (Tenenbaum et al., 2000) is a nonlinear embedding technique that attempts to capture manifold structure in the original data. It works by defining a similarity metric that measures distances along the manifold, e.g. W_{ij} is defined by the shortest path on the neighborhood graph. One then uses those distances to embed using conventional MDS.

Laplacian Eigenmaps (Belkin & Niyogi, 2003) learn manifold structure by emphasizing the preservation of *local distances*. One defines the distance metric between the examples by encoding them in the Laplacian $L = W - D$, where $D_{ii} = \sum_j W_{ij}$ is diagonal. Then, the following optimization is used:

$$\sum_{ij} L(f_i, f_j, W_{ij}) = \sum_{ij} W_{ij} ||f_i - f_j||^2 = f^\top L f \quad (1)$$

subject to the balancing constraint:

$$f^\top D f = I \text{ and } f^\top D 1 = 0. \quad (2)$$

Siamese Networks (Bromley et al., 1993) are also a classical method for nonlinear embedding. Neural networks researchers think of such models as a network with two identical copies of the same function, with the same weights, fed into a "distance measuring" layer to compute whether the two examples are similar or not, given labeled data. In fact, this is exactly the same as the formulation given at the beginning of this Section.

Several loss functions have been proposed for *siamese networks*, here we describe a margin-based loss proposed by the authors of (Hadsell et al., 2006):

$$L(f_i, f_j, W_{ij}) = \begin{cases} ||f_i - f_j||^2 & \text{if } W_{ij} = 1, \\ \max(0, m - ||f_i - f_j||^2) & \text{if } W_{ij} = 0 \end{cases} \quad (3)$$

which encourages similar examples to be close, and dissimilar ones to have a distance of at least m from each other. Note that no balancing constraint is needed with such a choice of loss as the margin constraint inhibits a trivial solution. Compared to using constraints like (2) this is much easier to optimize by gradient descent.

2.2. Semi-Supervised Algorithms

Several *semi-supervised* classification algorithms have been proposed which take advantage of the algorithms described in the last section. Here we assume the setting where one is given $L + U$ examples x_i, but only the first L have a known label y_i.

Label Propagation (Zhu & Ghahramani, 2002) adds a Laplacian Eigenmap type regularization to a nearest-neighbor type classifier:

$$\min_f \sum_{i=1}^{L} ||f_i - y_i||^2 + \lambda \sum_{i,j=1}^{L+U} W_{ij} ||f_i - f_j||^2 \quad (4)$$

The algorithm tries to give two examples with large weighted edge W_{ij} the same label. The neighbors of neighbors tend to also get the same label as each other by transitivity, hence the name *label propagation*.

LapSVM (Belkin et al., 2006) uses the Laplacian Eigenmaps type regularizer with an SVM: minimize

$$||w||^2 + \gamma \sum_{i=1}^{L} H(y_i f(x_i)) + \lambda \sum_{i,j=1}^{L+U} W_{ij} ||f(x_i) - f(x_j)||^2 \quad (5)$$

where $H(x) = \max(0, 1 - x)$ is the hinge loss.

Other Methods In (Chapelle & Zien, 2005) a method called *graph* is suggested which combines a modified version of ISOMAP with an SVM. The authors also suggest to combine modified ISOMAP with TSVMs rather than SVMs, and call it *Low Density Separation* (LDS).

3. Semi-supervised Embedding for Deep Learning

We would like to use the ideas developed in semi-supervised learning for *deep learning*. Deep learning consists of learning a model with several layers of non-linear mapping. In this article we will consider multi-layer networks with M layers of hidden units that give a C-dimensional output vector:

$$f_i(x) = \sum_{j=1}^{d} w_j^{O,i} \; h_j^M(x) + b^{O,i}, \quad i = 1, \ldots, C \quad (6)$$

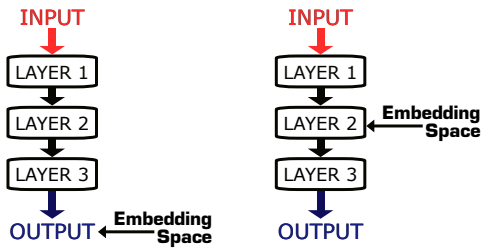
where w^O are the weights for the output layer, and typically the k^{th} layer is defined as

$$h_i^k(x) = S\left(\sum_j w_j^{k,i} \; h_j^{k-1}(x) + b^{k,i} \right), \; k > 1 \quad (7)$$

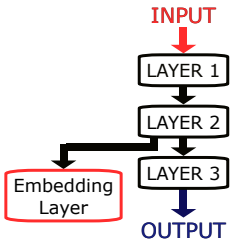
$$h_i^1(x) = S\left(\sum_j w_j^{1,i} \; x_j + b^{1,i} \right) \quad (8)$$

and S is a non-linear squashing function such as tanh. Here, we describe a standard *fully connected* multi-layer network but prior knowledge about a particular problem could lead one to other network designs. For example in sequence and image recognition time delay and convolutional networks (TDNNs and CNNs) (LeCun et al., 1998) have been very successful. In those approaches one introduces layers that apply convolutions on their input which take into account locality information in the data, i.e. they learn features from image patches or windows within a sequence.

The general method we propose for *semi-supervised deep learning* is to add a semi-supervised regularizer in deep architectures in one of three different modes, as shown in Figure 1:



(a) Output (b) Internal



(c) Auxiliary

Figure 1. Three modes of embedding in deep architectures.

(a) Add a semi-supervised loss (regularizer) to the supervised loss on the entire network's output (6):

$$\sum_{i=1}^{L} \ell(f(x_i), y_i) + \lambda \sum_{i,j=1}^{L+U} L(f(x_i), f(x_j), W_{ij}) \quad (9)$$

This is most similar to the *shallow* techniques described before, e.g. equation (5).

(b) Regularize the k^{th} hidden layer (7) directly:

$$\sum_{i=1}^{L} \ell(f(x_i), y_i) + \lambda \sum_{i,j=1}^{L+U} L(f^k(x_i), f^k(x_j), W_{ij}) \quad (10)$$

where $f^k(x) = (h_1^k(x), \ldots, h_{N_k}^k(x))$ is the output of the network up to the k^{th} hidden layer.

(c) Create an auxiliary network which shares the first k layers of the original network but has a new final set of weights:

$$g_i(x) = \sum_j w_j^{AUX,i} \; h_j^k(x) + b^{AUX,i} \quad (11)$$

We train this network to *embed* unlabeled data simultaneously as we train the original network on *labeled* data.

In our experiments we use the loss function (3) for embedding, and the hinge loss

$$\ell(f(x), y) = \sum_{c=1}^{C} H(y(c) f_c(x)),$$

Algorithm 1 *Embed*NN

Input: labeled data (x_i, y_i), $i = 1, \ldots, L$, unlabeled data x_i, $i = L+1, \ldots, U$, set of functions $f(\cdot)$, and embedding functions $g^k(\cdot)$, see Figure 1 and equations (9), (10) and (11).
repeat
  Pick a random *labeled* example (x_i, y_i)
  Make a gradient step to optimize $\ell(f(x_i), y_i)$
  for each embedding function $g^k(\cdot)$ **do**
    Pick a random pair of neighbors x_i, x_j.
    Make a gradient step for $\lambda L(g^k(x_i), g^k(x_j), 1)$
    Pick a random unlabeled example x_n.
    Make a gradient step for $\lambda L(g^k(x_i), g^k(x_n), 0)$
  end for
until stopping criteria is met.

for labeled examples, where $y(c) = 1$ if $y = c$ and -1 otherwise. For neighboring points, this is the same regularizer as used in LapSVM and Laplacian Eigenmaps. For non-neighbors, where $W_{ij} = 0$, this loss "pulls" points apart, thus inhibiting trivial solutions without requiring difficult constraints such as (2). To achieve an embedding *without* labeled data the latter is necessary or all examples would collapse to a single point in the embedding space. We therefore prefer this regularizer to using (1) alone. Pseudocode of our approach is given in Algorithm 1.

Labeling unlabeled data as neighbors Training neural networks online using stochastic gradient descent is fast and can scale to millions of examples. A possible bottleneck with our approach is computation of the matrix W, that is, computing which unlabeled examples are neighbors and have value $W_{ij} = 1$. Embedding algorithms often use k-nearest neighbor for this task, and although many methods for its fast computation do exist, this could still be slower than we would like. One possibility is to approximate it with sampling techniques.

However, there are also many other ways of collecting neighboring unlabeled data, notably if one is given *sequence* data such as in audio, video or text problems. For example, one can take images from two consecutive frames of video as a neighboring pair with $W_{ij} = 1$. Such pairs are likely to have the same label, and are collected cheaply. In Section 5 we apply this kind of idea to text and train a semi-supervised semantic role labeler using an unlabeled set of 631 million words.

When do we expect this approach to work? One can see our approach as an instance of multi-task learning (Caruana, 1997) using unsupervised auxiliary

tasks. In common with other semi-supervised learning approaches, and indeed other deep learning approaches, we only expect this to work if $p(x)$ is useful for the supervised task $p(y|x)$, i.e. if the structure assumption is true. We believe many natural tasks have this property.

We note that an alternative multi-task learning scheme is presented in (Ando & Zhang, 2005) and applied to neural networks in (Ahmed et al., 2008) which instead constructs auxiliary *supervised* tasks from unlabeled data by constructing tasks with labels y^*. This is useful when $p(y^*|x)$ is correlated to $p(y|x)$, however an expert must engineer a useful target y^*.

4. Existing Approaches to Deep Learning

Hinton and coworkers (2006) proposed the Deep Belief Net (DBN) which is a multi-layered network first trained as a generative model with unlabeled data before being subsequently trained in supervised mode. It is based around iteratively training Restricted Boltzmann machines (RBMs) for each layer. An RBM is a two-layer network in which visible, binary stochastic pixels v are connected to hidden binary stochastic feature detectors h. The probability assigned to an example x is:

$$P(x) = \sum_{h \in \mathcal{H}} P(x,h) = \sum_{h \in \mathcal{H}} \frac{e^{-E(x,h)}}{Z}$$

$$E(x,h) = -\sum_{i \in \text{pixels}} w_i^P v_i - \sum_{j \in \text{features}} w_j^F h_j - \sum_{i,j} v_j h_j w_{ij}$$

The idea is to obtain large values for the training examples, and small values elsewhere just as in any maximum likelihood density estimator. This is trained with a procedure called contrastive divergence whereby one pushes up the energy on training data and pushes down the energy on samples generated by the model. The authors used this method to pretrain a deep neighborhood component analysis model (DBN-NCA) and a regularized version that simultaneously trains an autoencoder (DBN-rNCA) (Salakhutdinov & Hinton, 2007).

The authors of (Bengio et al., 2007) suggested a simpler scheme: define an autoencoder that given an input x tries to encode it in a low dimensional space $z = f_{enc}(x)$, and then decode it again to reproduce it as well as possible, e.g. so that

$$||x - f_{dec}(f_{enc}(x))||^2$$

is small. (Actually you can also view RBMs in this way, see (Ranzato et al., 2007).) The idea is to use an autoencoder as a regularizer which is trained on unlabeled data. If the autoencoder is linear it corresponds to PCA (Japkowicz et al., 2000) and hence also MDS, making a clear link to the embedding algorithms we discussed in Section 2.1. The authors claim that autoassociators have the advantage "that almost any parametrizations of the layers are possible, as long as the training criterion is continuous in the parameters [...] the class of probabilistic models for which [DBNs] can be applied is currently more limited."

Finally, recently the authors of (Ranzato et al., 2007) introduced another method of deep learning which also amounts to a kind of encoder/decoder architecture, called SESM. In this case they choose to learn large, sparse codes as they believe these are good for classification. They choose an encoder $f_{enc}(x) = w^\top x + b_{enc}$ and a decoder with shared weights $f_{dec}(z) = wS(z) + b_{dec}$. They then optimize the following loss:

$$\alpha_e ||z - f_{enc}(x)||_2^2 + ||x - f_{dec}(z)||_2^2 + \alpha_s h(z) + \alpha_r ||w||_1$$

where the first term makes the output of the encoder close to the code z (which is also learnt), the second term makes the decoder try to reproduce the input, and the third and fourth terms sparsify the codes z and the weights of the encoder and decoder w. α_e, α_s and α_r are all hyperparameters. The training requires an online coordinate descent scheme because both z and w are being optimized.

We believe all of the methods just described are significantly more complicated than our approach. Our embedding approach can also be seen as an encoder $f_{enc}(x)$ that embeds data into a low dimensional space. However we do not need to decode during training (or indeed at all). Further, if the data is high dimensional and sparse there is a significant speedup from not having to decode.

Finally, existing approaches advocate greedy layer-wise training, followed by a "fine-tuning" step using the supervised signal. The intention is that the unsupervised learning provides a better initialization for supervised learning, and hence a better final local minimum. Our approach does not use a pre-training step, but instead *directly* optimizes our new objective function. We advocate that it is the new choice of objective that can provide improved results.

5. Experimental Evaluation

We test our approach on several datasets summarized in Table 1.

Small-scale experiments g50c, Text and Uspst are small-scale datasets often used for semi-supervised

Table 1. Datasets used in our experiments. The first three are small scale datasets used in the same experimental setup as found in (Chapelle & Zien, 2005; Sindhwani et al., 2005; Collobert et al., 2006). The following six datasets are large scale. The Mnist 1h,6h,1k,3k and 60k variants are MNIST with a labeled subset of data, following the experimental setup in (Collobert et al., 2006). SRL is a Semantic Role Labeling task (Pradhan et al., 2004) with one million labeled training examples and 631 million unlabeled examples.

data set	classes	dims	points	labeled
g50c	2	50	500	50
Text	2	7511	1946	50
Uspst	10	256	2007	50
Mnist1h	10	784	70k	100
Mnist6h	10	784	70k	600
Mnist1k	10	784	70k	1000
Mnist3k	10	784	70k	3000
Mnist60k	10	784	70k	60000
SRL	16	-	631M	1M

learning experiments (Chapelle & Zien, 2005; Sindhwani et al., 2005; Collobert et al., 2006). We followed the same experimental setup, averaging results of ten splits of 50 labeled examples where the rest of the data is unlabeled. In these experiments we test the embedding regularizer on the output of a neural network (see equation (9) and Figure 1(a)). We define a two-layer neural network (NN) with hu hidden units. We define W so that the 10 nearest neighbors of i have $W_{ij} = 1$, and $W_{ij} = 0$ otherwise. We train for 50 epochs of stochastic gradient descent and fixed $\lambda = 1$, but for the first 5 we optimized the supervised target alone (without the embedding regularizer). This gives two free hyperparameters: the number of hidden units $hu = \{0, 5, 10, 20, 30, 40, 50\}$ and the learning rate $lr = \{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$.

We report the optimum choices of these values optimized by 5-fold cross validation and by optimizing on the test set in Table 2. Note the datasets are very small, so cross validation is unreliable. Several methods from the literature optimized their hyperparameters using the test set (those that are not marked with (cv)). Our *Embed*NN is competitive with state-of-the-art semi-supervised methods based on SVMs, even outperforming them in some cases.

MNIST experiments We compare our method in all three different modes (Figure 1) with conventional semi-supervised learning (TSVM) using the same data split and validation set as in (Collobert et al., 2006). We also compare to several deep learning methods: RBMs, SESM and DBN-NCA and DBN-rNCA (however, they are trained on a different data split). In

Table 2. Results on Small-Scale Datasets. We report the best test error over the hyperparameters of our method, *Embed*NN, as in the methodology of (Chapelle & Zien, 2005) as well as the error when optimizing the parameters by cross-validation, *Embed*NN$^{(cv)}$. LDS$^{(cv)}$ and LapSVM$^{(cv)}$ also use cross-validation.

	g50c	Text	Uspst
SVM	8.32	18.86	23.18
TSVM	5.80	5.71	17.61
LapSVM$^{(cv)}$	5.4	10.4	12.7
LDS$^{(cv)}$	5.4	5.1	15.8
Label propagation	17.30	11.71	21.30
Graph SVM	8.32	10.48	16.92
NN	10.62	15.74	25.13
*Embed*NN	5.66	5.82	15.49
*Embed*NN$^{(cv)}$	6.78	6.19	15.84

Table 3. Results on MNIST with 100, 600, 1000 and 3000 labels. A two-layer Neural Network (NN) is compared to an NN with Embedding regularizer (*Embed*NN) on the output (O), i^{th} layer (Ii) or auxiliary embedding from the i^{th} layer (Ai) (see Figure 1). A convolutional network (CNN) is also tested in the same way. We compare to SVMs and TSVMs. RBM, SESM, DBN-NCA and DBN-rNCA (marked with $(*)$) taken from (Ranzato et al., 2007; Salakhutdinov & Hinton, 2007) are trained on a different data split.

	Mnst1h	Mnst6h	Mnst1k	Mnst3k
SVM	23.44	8.85	7.77	4.21
TSVM	16.81	6.16	5.38	3.45
RBM$^{(*)}$	21.5	-	8.8	-
SESM$^{(*)}$	20.6	-	9.6	-
DBN-NCA$^{(*)}$	-	10.0	-	3.8
DBN-rNCA$^{(*)}$	-	8.7	-	3.3
NN	25.81	11.44	10.70	6.04
Embed^{O}NN	17.05	5.97	5.73	3.59
Embed^{I1}NN	16.86	9.44	8.52	6.02
Embed^{A1}NN	17.17	7.56	7.89	4.93
CNN	22.98	7.68	6.45	3.35
Embed^{O}CNN	11.73	3.42	3.34	2.28
Embed^{I5}CNN	7.75	3.82	2.73	1.83
Embed^{A5}CNN	7.87	3.82	2.76	2.07

Table 4. Mnist1h dataset with deep networks of 2, 6, 8, 10 and 15 layers; each hidden layer has 50 hidden units. We compare classical NN training with *Embed*NN where we either learn an embedding at the output layer (^{O}) or an auxiliary embedding on all layers at the same time (^{ALL}). .

	2	4	6	8	10	15
NN	26.0	26.1	27.2	28.3	34.2	47.7
Embed^{O}NN	19.7	15.1	15.1	15.0	13.7	11.8
Embed^{ALL}NN	18.2	12.6	7.9	8.5	6.3	9.3

Table 5. Full Mnist60k dataset with deep networks of 2, 6, 8, 10 and 15 layers, using either 50 or 100 hidden units. We compare classical NN training with $Embed^{ALL}$NN where we learn an auxiliary embedding on all layers at the same time.

	2	4	6	8	10	15
NN (HUs=50)	2.9	2.6	2.8	3.1	3.1	4.2
$Embed^{ALL}$NN	2.8	1.9	2.0	2.2	2.4	2.6
NN (HUs=100)	2.0	1.9	2.0	2.2	2.3	3.0
$Embed^{ALL}$NN	1.9	1.5	1.6	1.7	1.8	2.4

these experiments we consider 2-layer networks (NN) and 6-layer convolutional neural nets (CNN) for embedding. We optimize the parameters of NN ($hu = \{50, 100, 150, 200, 400\}$ hidden units and learning rates as before) on the validation set. The CNN architecture is fixed: 5 layers of image patch-type convolutions, followed by a linear layer of 50 hidden units, similar to (LeCun et al., 1998). The results given in Table 3 show the effectiveness of embedding in all three modes, with both NNs and CNNs.

Deeper MNIST experiments We then conducted a similar set of experiments but with very deep architectures – up to 15 layers, where each hidden layer has 50 hidden units. Using Mnist1h, we first compare conventional NNs to $Embed^{ALL}$NN where we learn an auxiliary nonlinear embedding (50 hidden units and a 10 dimensional embedding space) on each layer, as well as $Embed^{O}$NN where we only embed the outputs. Results are given in Table 4. When we increase the number of layers, NNs trained with conventional backpropagation overfit and yield steadily *worse* test error (although they are easily capable of achieving zero training error). In contrast, $Embed^{ALL}$NN *improves* with increasing depth due to the semi-supervised "regularization". Embedding on *all* layers of the network has made *deep learning* possible. $Embed^{O}$NN (embedding on the outputs) also helps, but not as much.

We also conducted some experiments using the full MNIST dataset, Mnist60k. Again using deep networks of up to 15 layers using either 50 or 100 hidden units $Embed^{ALL}$NN outperforms standard NN. Results are given in Table 5. Increasing the number of hidden units is likely to improve these results further, e.g. using 4 layers and 500 hidden units on each layer, one obtains 1.27% using $Embed^{ALL}$NN.

Semantic Role Labeling The goal of semantic role labeling (SRL) is, given a sentence and a relation of interest, to label each word with one of 16 tags that indicate that word's semantic role with respect to the action of the relation. For example the sentence *"The cat eats the fish in the pond"* is labeled in the following way: *"The*$_{ARG0}$ *cat*$_{ARG0}$ *eats*$_{REL}$ *the*$_{ARG1}$ *fish*$_{ARG1}$ *in*$_{ARGM-LOC}$ *the*$_{ARGM-LOC}$ *pond*$_{ARGM-LOC}$*"* where ARG0 and ARG1 effectively indicate the subject and object of the relation "eats" and ARGM-LOC indicates a locational modifier. The PropBank dataset includes around 1 million labeled words from the Wall Street Journal. We follow the experimental setup of (Collobert & Weston, 2007) and train a 5-layer convolutional neural network for this task, where the first layer represents the input sentence words as 50-dimensional vectors. Unlike (Collobert & Weston, 2007), we do not give any prior knowledge to our classifier. In that work words were stemmed and clustered using their parts-of-speech. Our classifier is trained using only the original input words.

Table 6. A deep architecture for Semantic Role Labeling with no prior knowledge outperforms state-of-the-art systems ASSERT and SENNA that incorporate knowledge about parts-of-speech and parse trees. A convolutional network (CNN) is improved by learning an auxiliary embedding ($Embed^{A1}$CNN) for words represented as 100-dimensional vectors using the entire Wikipedia website as unlabeled data.

Method	Test Error
ASSERT (Pradhan et al., 2004)	16.54%
SENNA (Collobert & Weston, 2007)	16.36%
CNN [no prior knowledge]	18.40%
$Embed^{A1}$CNN [no prior knowledge]	14.55%

We attempt to improve this system by, as before, learning an *auxiliary embedding* task. Our embedding is learnt using unlabeled sentences from the Wikipedia web site, consisting of 631 million words in total using the scheme described in Section 3. The same lookup table of word vectors as in the supervised task is used as input to an 11 word window around a given word, yielding 550 features. Then a linear layer projects these features into a 100 dimensional embedding space. All windows of text from Wikipedia are considered neighbors, and non-neighbors are constructed by replacing the middle word in a sentence window with a random word. Our lookup table indexes the most frequently used 30,000 words, and all other words are assigned index 30,001.

The results in Table 6 indicate a clear improvement when learning an auxiliary embedding. ASSERT (Pradhan et al., 2004) is an SVM parser-based system with many hand-coded features, and SENNA is a NN which uses part-of-speech information to build its word vectors. In contrast, our system is the only state-

of-the-art method that does not use prior knowledge in the form of features derived from parts-of-speech or parse tree data. This application will be described in more detail in a forthcoming paper.

6. Conclusion

In this work, we showed how one can improve supervised learning for deep architectures if one jointly learns an embedding task using unlabeled data. Our results both confirm previous findings and generalize them. Researchers using *shallow* architectures already showed two ways of using embedding to improve generalization: (i) embedding unlabeled data as a *separate* pre-processing step (i.e., first layer training) and; (ii) using embedding as a regularizer (i.e., at the output layer). More importantly, we generalized these approaches to the case where we train a semi-supervised embedding *jointly* with a supervised *deep* multi-layer architecture on any (or all) layers of the network, and showed this can bring real benefits in complex tasks.

References

Ahmed, A., Yu, K., Xu, W., & Gong, Y. (2008). Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. *ECCV*. Submitted.

Ando, R., & Zhang, T. (2005). A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *The Journal of Machine Learning Research*, *6*, 1817–1853.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*, 1373–1396.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: a geometric framework for learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, *7*, 2399–2434.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems, NIPS 19*.

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Sackinger, E., & Shah, R. (1993). Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, *7*.

Caruana, R. (1997). Multitask Learning. *Machine Learning*, *28*, 41–75.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Adaptive computation and machine learning. Cambridge, Mass., USA: MIT Press.

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. *NIPS 15* (pp. 585–592). Cambridge, MA, USA: MIT Press.

Chapelle, O., & Zien, A. (2005). Semi-supervised classification by low density separation. *AISTATS* (pp. 57–64).

Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Large scale transductive svms. *Journal of Machine Learning Research*, *7*, 1687–1712.

Collobert, R., & Weston, J. (2007). Fast semantic extraction using a novel neural network architecture. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 25–32.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Comp.*, *18*, 1527–1554.

Japkowicz, N., Hanson, S., & Gluck, M. (2000). Nonlinear autoassociation is not equivalent to PCA. *Neural Computation*, *12*, 531–545.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J., & Jurafsky, D. (2004). Shallow semantic parsing using support vector machines. *Proceedings of HLT/NAACL-2004*.

Ranzato, M., Huang, F., Boureau, Y., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07)*. IEEE Press.

Salakhutdinov, R., & Hinton, G. (2007). Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. *AISTATS*.

Sindhwani, V., Niyogi, P., & Belkin, M. (2005). Beyond the point cloud: from transductive to semi-supervised learning. *International Conference on Machine Learning, ICML*.

Tenenbaum, J., de Silva, V., & Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Vapnik, V. N. (1998). *Statistical learning theory*. John Wiley and Sons, New York.

Williams, C. (2001). On a connection between kernel PCA and metric multidimensional scaling. *Advances in Neural Information Processing Systems, NIPS 13*.

Zhu, X., & Ghahramani, Z. (2002). *Learning from labeled and unlabeled data with label propagation* (Technical Report CMU-CALD-02-107). Carnegie Mellon University.

Acknowledgements

Thanks to Sandra Cordero for producing the figures. Frédéric Ratle is funded by the SNF, Grant no. 105211-107862.

Efficiently Learning Linear-Linear Exponential Family Predictive Representations of State

David Wingate WINGATED@UMICH.EDU
Satinder Singh BAVEJA@UMICH.EDU
Computer Science and Engineering, University of Michigan, 2260 Hayward, Ann Arbor, MI 48109

Abstract

Exponential Family PSR (EFPSR) models capture stochastic dynamical systems by representing state as the parameters of an exponential family distribution over a short-term window of future observations. They are appealing from a learning perspective because they are fully observed (meaning expressions for maximum likelihood do not involve hidden quantities), but are still expressive enough to both capture existing models and predict new models. While maximum-likelihood learning algorithms for EFPSRs exist, they are not computationally feasible. We present a new, computationally efficient, learning algorithm based on an approximate likelihood function. The algorithm can be interpreted as attempting to induce stationary distributions of observations, features and states which match their empirically observed counterparts. The approximate likelihood, and the idea of matching stationary distributions, may apply to other models.

1. Introduction

One of the basic problems in modeling controlled, partially observable, stochastic dynamical systems is representing and tracking state. In a reinforcement learning context, the state of the system is important because it can be used to make predictions about the future, or to control the system optimally. Often, state is viewed as an unobservable, latent variable, but models with *predictive representations of state* (Littman et al., 2002) propose an alternative: PSRs represent state as *statistics about the future*.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

The original PSR models used the probability of specific, detailed futures called *tests* as the statistics of interest. Recent work has introduced the more general notion of using parameters that model the distribution of length n futures as the statistics of interest (Rudary et al., 2005; Wingate, 2008). To clarify this, consider an agent interacting with the system. It observes a series of observations $o_1...o_t$, which we call a *history* h_t (where subscripts denote time). Given any history, there is some distribution over the next n observations: $p(O_{t+1}...O_{t+n}|h_t) \equiv p(F^n|h_t)$ (where O_{t+i} is the random variable representing an observation i steps in the future, and F^n is a mnemonic for *future*). We emphasize that this distribution directly models observable quantities in the system.

The *Exponential Family PSR* is a new family of models of partially observable, stochastic dynamical systems. EFPSR models assume that the distribution $p(F^n|h_t)$ has an exponential family form, and that the parameters of that distribution are the state of the system (Wingate, 2008). This idea has been shown to unify a number of existing models of dynamical systems: for example, if $p(F^n|h_t)$ is assumed to be Gaussian (and certain other choices are made), the model can capture any domain modeled by a Kalman filter.

Existing algorithms for learning EFPSR models from data are based on maximizing exact likelihood, but the algorithms are slow. This paper presents an efficient algorithm for one particular EFPSR, named the Linear-Linear EFPSR. We begin by presenting an approximate likelihood function, and then show that the terms needed to maximize it can be efficiently computed by virtue of the linearity of the Linear-Linear EFPSR's state update. The resulting algorithm is computationally efficient, and can be interpreted in terms of stationary distributions of observations, features and states. It allows us to begin to learn models of domains which are too large (in terms of the amount of data required, and in terms of the complexity of the observation space) to tackle with any other EFPSR.

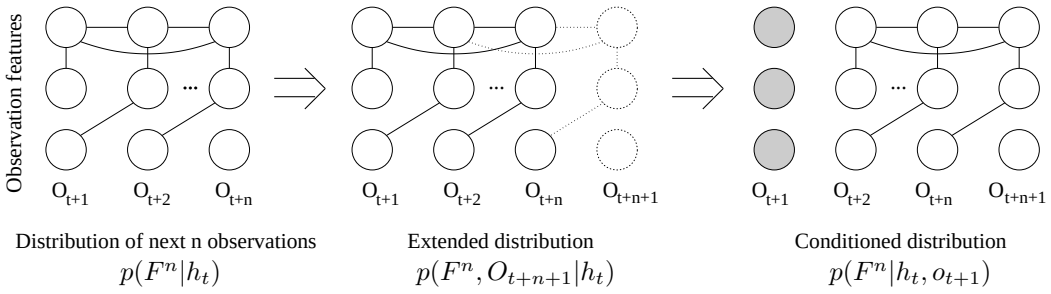


Figure 1. An illustration of extending and conditioning.

2. The Exponential Family PSR

We first review the EFPSR family of models, including how state is represented and how it is maintained.

State. The EFPSR defines state as the parameters of an exponential family distribution modeling $p(F^n|h_t)$, which is a window of n future observations. To emphasize that these parameters represent state, we will refer to them as s_t. The form of the distribution is:

$$p(F^n = f^n|h_t; s_t) = \exp\left\{s_t^\top \phi(f^n) - \log Z(s_t)\right\}, \quad (1)$$

with both $\{\ \phi(f^n),\ s_t\ \} \in \mathbb{R}^{l\times 1}$. The vector $\phi(f^n)$ is a feature vector which controls the particular form of the distribution. For example, $\phi(X) = [X, X^2]$, yields a Gaussian, but $\phi(X) = [X, \log(X)]$ yields a gamma. Since the distribution is over the future, ϕ can be thought of as *features of the future*.

As the agent interacts with the system, $p(F^n|h_t)$ changes because h_t changes; therefore the parameters s_t and hence state change. The feature vector $\phi(f^n)$ does not change over time.

Maintaining State. In addition to selecting the form of $p(F^n|h_t)$, there is a dynamical component: given the parameters of $p(F^n|h_t)$, how can we incorporate a new observation to find the parameters of $p(F^n|h_t, o_{t+1})$? That is, how can we update state? Our strategy is to *extend and condition*.

Extend. We assume that we have the parameters of $p(F^n|h_t)$, denoted s_t. We *extend* the distribution of $F^n|h_t$ to include O_{t+n+1}, which forms a new variable $F^{n+1}|h_t$, and we assume it has the distribution $p(F^n, O_{t+n+1}|h_t) = p(F^{n+1}|h_t)$. This is a temporary distribution over $(n+1)$ observations.

To perform the extension, we define an *extension function* which maps the current state vector to the parameters of the extended distribution:

$$s_t^+ = \text{extend}(s_t; \theta),$$

where θ is a vector of parameters controlling the extension function (and hence, the overall dynamics).

The extension function helps govern the kinds of dynamics that the model can capture. For example, in the PLG family of work, a linear extension allows the model to capture linear dynamics (Rudary et al., 2005), while a non-linear extension allows the model to capture non-linear dynamics (Wingate, 2008).

Condition. Once we have extended the distribution to model the $n+1$'st observation in the future, we then condition on the *actual* observation o_{t+1}, which results in the parameters of $p(F^n|h_{t+1})$:

$$s_{t+1} = \text{condition}(s_t^+, o_{t+1}),$$

which is our state at time $t+1$.

The entire process of extending and conditioning is illustrated in Fig. 1. We have drawn graphs to suggest that there can be structure in the distributions, and to informally hint at the fact that the form of the distribution does not change over time. This, and other constraints on the features and extension function, are discussed in detail elsewhere (Wingate, 2008).

2.1. The Linear-Linear EFPSR

The EFPSR is a family of models. Specific members of the family are chosen by selecting two things: the features ϕ, and an extension function. For example, if $p(F^n|h_t)$ is Gaussian, and a special extension function is chosen, the predictively defined version of a linear dynamical system (Kalman filter) is recovered.

The Linear-Linear EFPSR chooses features and an extension function designed to make it both analytically tractable and efficiently approximable. The extension function is linear, and features are chosen such that conditioning is always a linear operation (hence the name, "Linear-Linear"). In this paper, we also assume the base observations are vectors of binary random variables. If they are not, we assume that binary features are extracted from the observations, and discard the original observations (we call these *atomic* features, to distinguish them from the higher-order features defined by ϕ).

Features. Let each base observation O_t be a vector $\in \{0,1\}^d$; therefore, each $F^n|h_t \in \{0,1\}^{nd}$. We restrict all features comprising the feature vector ϕ to be conjunctions of the atomic binary variables in the base observations. For example, if each $O_t \in \{0,1\}^3$, there could be a feature $\phi(o_t)_k$ which is a conjunction of the second and third components of the observation: $\phi(o_t)_k = (o_t)_2(o_t)_3$. By selecting features this way, the resulting distribution can be conditioned with an operator that is nonlinear in the observation o_{t+1}, but linear in the state s_t. We therefore define the linear conditioning operator $G(o_{t+1})$ to be a matrix which transforms s_t^+ into s_{t+1}: $s_{t+1} = G(o_{t+1})s_t^+$. See (Wingate, 2008) for details.

Extension function. We choose a linear extension:

$$s_t^+ = As_t + B.$$

$A \in \mathbb{R}^{k\times l}$ and $B \in \mathbb{R}^{k\times 1}$ are our model parameters.

The combination of a linear extension and a linear conditioning operator means that the entire extend-and-condition operation (ie, state update) is a linear operation:

$$s_{t+1} = G(o_{t+1})\left(As_t + B\right).$$

This will be critical in the sequel.

3. Learning with Exact Likelihood

We now briefly sketch how to learn a Linear-Linear EFPSR model from data by maximizing exact likelihood. We do this to point out the two primary computational bottlenecks that motivate this paper.

We assume we are given a sequence of T observations, $[o_1 \cdots o_T]$, which we stack to create a sequence of samples from the $F^n|h_t$'s: $f_t|h_t = [o_{t+1} \cdots o_{t+n}|h_t]$. The likelihood of the training data is $p(o_1, o_2 ... o_T) = \prod_{t=1}^T p(o_t|h_t)$, but we will find it more convenient to measure the likelihood of the corresponding f_t's: $p(o_1, o_2 ... o_T) \approx n\prod_{t=1}^T p(f_t|h_t)$ (maximizing this is equivalent to maximizing the standard likelihood).

The expected log-likelihood of the training f_t's under the model defined in Eq. 1 is

$$\mathcal{LL} = \frac{1}{T}\left(\sum_{t=1}^{T} -s_t^\top \phi(f_t) - \log Z(s_t)\right) \qquad (2)$$

Our goal is to maximize this quantity. Any optimization method can be used to maximize the log-likelihood. Two popular choices are gradient ascent and quasi-Newton methods, such as (L-)BFGS, which require the gradient of the likelihood with respect to the parameters, which we will now compute.

We can differentiate with respect to our parameters:

$$\frac{\partial \mathcal{LL}}{\partial \{A, B\}} = \sum_{t=1}^{T} \frac{\partial \mathcal{LL}}{\partial s_t}^\top \frac{\partial s_t}{\partial \theta} \qquad (3)$$

and with respect to each state:

$$\begin{aligned}\frac{\partial \mathcal{LL}}{\partial s_t} &= \frac{\partial}{\partial s_t}\left[-s_t^\top \phi(f_t) - \log Z(s_t)\right] \\ &= \mathrm{E}_{s_t}[\phi(F^n|h_t)] - \phi(f_t) \qquad (4)\end{aligned}$$

where $\mathrm{E}_{s_t}[\phi(F^n|h_t)] \in \mathbb{R}^{l\times 1}$ is the vector of expected sufficient statistics at time t.

The gradient of s_t with respect to A is given by

$$\frac{\partial s_t}{\partial A} = G(o_{t+1})\left(A\frac{\partial s_{t-1}}{\partial A} + s_{t-1}^\top \otimes I\right),$$

where $\otimes$ is the Kronecker product, and I is an identity matrix the same size as A. The gradient of the state with respect to B is

$$\frac{\partial s_t}{\partial B} = G(o_{t+1})\left(A\frac{\partial s_{t-1}}{\partial B} + I\right).$$

Note that the gradients at time t are temporally recursive – they depend all previous gradients.

There are two bottlenecks which motivate this paper:

1. Computing $\mathrm{E}_{s_t}[\phi(F^n|h_t)]$ is a standard inference problem in exponential family models, and is computationally expensive because it scales exponentially with the number of atomic observation variables included in the domain of $p(F^n|h_t)$. Even approximate inference is NP-hard (Dagum & Luby, 1993), and it must be done T times.

2. The gradients are temporally recursive, but can be computed in a single pass through the data. However, the process is expensive. For the discussion, assume that we have l features in $\phi(f_t)$, and that we have k features in the extended distribution. This means that the matrix $A \in \mathbb{R}^{k\times l}$, that the vector $s_t \in \mathbb{R}^l$, and that there are kl total parameters describing A. The term $\partial s_t/\partial A$ is a matrix, with l rows and kl columns. Given $\partial s_{t-1}/\partial A$, part of computing $\partial s_t/\partial A$ involves multiplying $\partial s_t/\partial A$ by A. This is an expensive matrix-matrix multiplication, which scales poorly as the number of features in the model grows, and it must be performed T times to get the true gradient of the likelihood, which scales poorly as the size of the training set grows.

To summarize, the exact learning algorithm does not scale well with either the number of training samples T, the dimension of the observations, the window size n, or the number of features $|\phi|$.

4. Approximate Likelihood

We now turn to the main contribution of this paper. In order to achieve an efficient learning algorithm, we will present an approximate expression for likelihood, named $\widehat{\mathcal{LL}}$, and show that its gradient can be efficiently computed. We will also examine what happens in the limit as $T \to \infty$. The quantity $\widehat{\mathcal{LL}}$ could be used with any model, not just the Linear-Linear EFPSR, but we will show that the Linear-Linear EFPSR allows us to compute the needed terms easily.

We now present our approximate log-likelihood $\widehat{\mathcal{LL}}$, which is an approximate lower bound on the exact likelihood. To begin, we will make one central assumption:

Assumption 4.1. We assume that $\text{Cov}[s_t, \phi(f_t)] = 0$ and that $\text{Cov}[s_t, o_t] = 0$, $\forall t$.

This assumes that the state does not covary with observable quantities. It implies that $\text{E}[s_t^\top \phi(f_t)] = \text{E}[s_t]^\top \text{E}[\phi(f_t)]$, which will be repeatedly used in the following derivation. This is not as severe of an assumption as it may appear to be – in particular, that this does not imply that s_t and $\phi(f_t)$ are independent.

We derive $\widehat{\mathcal{LL}}$ using Assumption 4.1 and a lower bound based on Jensen's inequality:

$$\begin{aligned}
\mathcal{LL} &= \frac{1}{T}\left(\sum_{t=1}^{T} -s_t^\top \phi(f_t) - \log Z(s_t)\right) \\
&= \text{E}_T\left[-s_t^\top \phi(f_t) - \log Z(s_t)\right] \\
&= \text{E}_T\left[-s_t^\top \phi(f_t)\right] - \text{E}_T\left[\log Z(s_t)\right] \\
&\approx \text{E}_T\left[-s_t\right]^\top \text{E}_T\left[\phi(f_t)\right] - \text{E}_T\left[\log Z(s_t)\right] \\
&\geq \text{E}_T\left[-s_t\right]^\top \text{E}_T\left[\phi(f_t)\right] - \log Z(\text{E}_T\left[s_t\right]) \\
&\equiv \widehat{\mathcal{LL}}
\end{aligned}$$

where we have defined the operator

$$\text{E}_T[X] \equiv \frac{1}{T}\sum_{t=1}^{T} X.$$

The fourth line in the derivation follows because of Assumption 4.1. The fifth line is obtained by a double application of Jensen's inequality:

$$\begin{aligned}
\text{E}[-\log Z(s_t)] &= \text{E}\left[-\log(\int \exp(-s_t^\top \phi(F))dF)\right] \\
&\geq -\log(\text{E}\left[\int \exp(-s_t^\top \phi(F))dF\right]) \\
&\geq -\log(\int \exp(\text{E}\left[-s_t^\top \phi(F)\right])dF) \\
&\approx -\log(\int \exp(\text{E}\left[-s_t\right]^\top \text{E}\left[\phi(F)\right])dF) \\
&= -\log Z(\text{E}[s_t]).
\end{aligned}$$

Algorithm 1 LEARN-EFPSRS-W-APPROX-LL

Input: $\text{E}_T[o_t]$, $\text{E}_T[\phi(f_t)]$
Initialize $A = 0$, $B = 0$.
repeat
  $(\widehat{\mathcal{LL}}, \nabla_A \widehat{\mathcal{LL}}, \nabla_B \widehat{\mathcal{LL}})$=GRADS-OF-APPROX-LL($\text{E}_T[o_t], \text{E}_T[\phi(f_t)], A, B$)
  // Use the gradients in an optimizer. Steepest
  // descent would look like this:
  $A = A + \alpha \nabla_A \widehat{\mathcal{LL}}$
  $B = B + \alpha \nabla_B \widehat{\mathcal{LL}}$
until $\widehat{\mathcal{LL}}$ is maximized
Return A, B

The second and third lines follow because of the convexity of the functions $-\log$ and exp, and the fourth line follows by Assumption 4.1.

The approximate log-likelihood involves several new terms, which we now explain. Consider $\text{E}_T[s_t]$. Because this is an unconditional expectation, as $T \to \infty$, this can be interpreted as the stationary distribution of states induced by a particular setting of the parameters of the model.

At first glance, this term would appear to defeat the point of our approximations: it appears to depend on T and on the model parameters, which means that we would have to recompute it, at cost T, every time the parameters change (as they would inside any sort of optimization loop). Fortunately, because it is the stationary distribution of states, it can be efficiently computed in the case of the Linear-Linear EFPSR as the solution to a linear system of equations in a way that does not depend on T.

The other terms have similar interpretations. $\text{E}_T[\phi(f_t)]$ is empirically observed stationary distribution of features of n-step windows of observations. Since it does not depend on the model parameters, it can be computed once at the beginning of learning in a single pass through the data. The quantity $\log Z(\text{E}_T[s_t])$ is the log partition function Z computed using the vector $\text{E}_T[s_t]$, and can be computed in the same way as the partition function associated with any ordinary state s_t.

4.1. Computing the Approximate Likelihood

Can the approximate log-likelihood $\widehat{\mathcal{LL}}$ and its derivatives be computed efficiently? The answer is yes: Appendix A shows that in the case of the Linear-Linear EFPSR, both $\widehat{\mathcal{LL}}$ and the derivative of $\widehat{\mathcal{LL}}$ with respect to the model parameters can be computed efficiently. The computation does not depend on T (the amount

of training data), and only involves the solution to two sparse linear systems of equations. Inference must be performed on the graphical model only once. In addition, the expensive matrix-matrix multiplications are completely eliminated.

4.2. Algorithm Summary

Let us pause for a brief summary. The exact log-likelihood $\mathcal{LL}$ in Eq. 2 is intractable to maximize. However, we have introduced $\widehat{\mathcal{LL}}$, and shown that it and its derivatives can be computed efficiently. Putting everything together, we see that this learning algorithm is attempting:

- to find a setting of the parameters A and B
- which generate a stationary distribution of states $\mathrm{E}_T[s_t]$,
- based on a transition operator defined using the stationary distribution of observations $\mathrm{E}_T[o_t]$,
- which imply a stationary distribution of features of length n trajectories $\mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F^n|h_t)]$ as close as possible to the empirically observed stationary distribution of features of length n trajectories $\mathrm{E}_T[\phi(f_t)]$.

With gradients in hand, any optimization method may be used to find the optimal settings for A and B. The final gradient algorithm is shown in Algorithm 2 (in Appendix A), and a simple companion steepest descent optimizer is shown in Algorithm 1.

5. A Low-Rank Parameterization

We briefly turn our attention to the parameter matrices A and B. So far, we have implicitly assumed that the matrix A is reasonably sized, but this assumption is false in the case of a large number of features.

To clarify this, recall that our state s_t is a vector $\in \mathbb{R}^{l\times 1}$, where l is the number of features of the future. When we extend and condition, we implicitly compute s_t^+, which is a vector of parameters describing $n+1$ observations: $s_t^+ = As_t + B$. If we assume that there are k extended features, the A matrix is $\in \mathbb{R}^{k\times l}$.

One of the goals of EFPSRs is to be able to use many features in order to capture state. If the number of features l is very large (say, tens of thousands, or even millions), the number of extended features k will be even larger, and the matrix A will be too large to work with. For example, if there are 10,000 features, and if the extended distribution has 15,000 features, the matrix $A \in \mathbb{R}^{15{,}000\times 10{,}000}$, which is simply too large.

$\widehat{\mathcal{LL}}$ has another property which suggests a solution

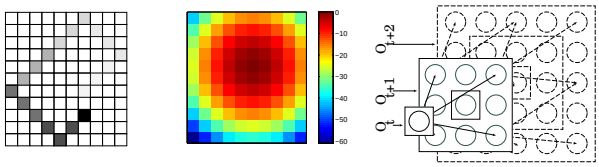


Figure 2. The setup of the bouncing ball problem.

to this problem: the gradients $\nabla_A \widehat{\mathcal{LL}}$ have a natural rank-one form, and therefore mesh well with singular value decomposition (SVD) update algorithms (Brand, 2006). Instead of maintaining the full matrix A, we can maintain a low-rank SVD of A. Given the SVD of A and a rank-one gradient update, the parameters of the updated SVD can be efficiently computed. The entire process can be meshed with a rank-aware line search. The advantage is that the full matrix A is never computed, but exact line searches can be conducted. See (Wingate, 2008) for more details.

6. Experiments and Results

We now evaluate the quality of our approximations. For large problems, we cannot compute the exact likelihoods to compare with, and since we are using approximate likelihoods, it is not clear what a comparison would mean. Instead, we use reinforcement learning to help measure the quality of the model: we use the states generated by the EFPSR as the input to an reinforcement learning planner. We conclude that our model is good if the RL agent is able to use it generate performance comparable to that of the true model. For comparison, we also tested RL using the raw observations as state (called the "reactive," or first-order Markov policy), and a random policy.

6.1. Planning in the EFPSR

We used the Natural Actor Critic (or NAC) algorithm (Peters et al., 2005) to test our model. NAC requires two things: a stochastic, parameterized policy and the gradients of the log probability of that policy. We used a softmax function of a linear projection of the state: the probability of taking action a_i from state s_t given the policy parameters θ is $p(a_i; s_t, \theta) = \exp\left\{s_t^\top \theta_i\right\} / \sum_{j=1}^{|\mathcal{A}|} \exp\left\{s_t^\top \theta_j\right\}$, where θ is to be learned. See (Wingate, 2008) for details.

6.2. Bouncing Ball

The first test domain is called the Bouncing Ball domain. In this domain, the observations are factored in a way that is closely related to the dynamics of the system. This domain was hand-crafted to be compatible with the EFPSR: the domain has significant structure

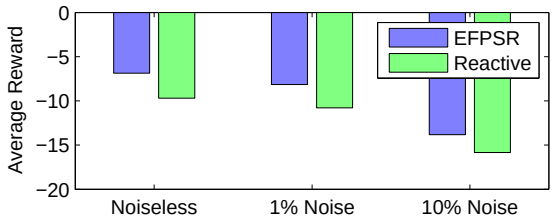


Figure 3. Results on the bouncing ball domain.

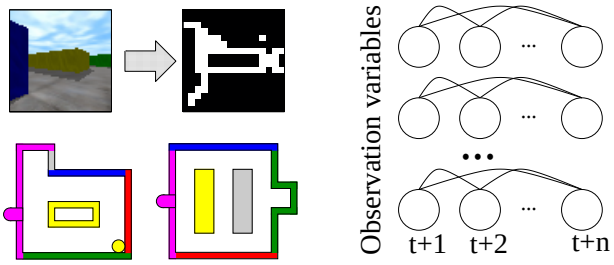


Figure 4. Setup of the vision domain.

in the observations, and basically requires the use of a model which is able to capture that structure.

Figure 2 describes the domain pictorially. The left figure shows the the ball bouncing. At each timestep, the agent observes an 11x10 array of pixels which may be black or white. One of these pixels represents the "ball," which bounces diagonally around the box (shown as a gray trail in the figure). The agent has two actions: 0 means "do nothing," and 1 means "reverse the direction of the ball." The reward signal is shown in the middle. This domain is episodic: every 50 timesteps, the ball is reset to a random position. We define three different versions of the domain. In the noiseless version, the agent sees the exact position of the ball. This domain is second-order Markov with $11 \times 10 = 110$ observations. The second version adds a p=1% chance of flipping white pixels to black. This domain is no longer second-order Markov, and has 2^{110} possible observations. The third version uses p=10%.

Figure 2 shows the features we used, which are hand-coded to correspond with the known dynamics. We set $n = 2$ and added singleton features for each observation. Pairwise features were added for each variable to its diagonal neighbors in the next timestep (to capture the diagonal motion of the ball). The extended distribution $p(F^3|h_t)$ used quartets consisting of an action and observation at time t, and diagonal observations at time $t+1$ and $t+2$. There were 584 features describing $p(F^2|h_t)$ and 1,292 features describing $p(F^3|h_t)$.

We used the timeless gradients, the low rank approximation of A, and 100,000 training samples. Figure 3 collects the results. The EFPSR is able to consistently improve over the best reactive policy, generating a policy with 30% higher reward in the noiseless version, a policy with 25% higher reward when p=1%, and a policy with 13% higher reward when p=10%. It is an open question as to whether different feature sets would improve these results further.

6.3. Robot Vision Domain

Together, the combination of the Linear-Linear EFPSR, the approximate maximum likelihood objective function, and the low-rank decomposition of the parameter matrix allow experimentation on domains with hundreds of observation variables and tens of thousands of features, which is larger than any other model with a predictive representation of state. Here, we apply the entire suite of techniques to the task of visual navigation, where a robot must navigate a maze using only features of camera images as observations.

Figure 4 explains the setup. The latent state space consists of a position x, y and orientation θ. The experiments used two different maps (bottom left). The agent has four actions: move forward, move backward, turn left and turn right. We tested two kinds of dynamics: in the "coarse" dynamics, the agent took large steps and turns, and in the "fine" dynamics, the agent took small steps and turns. The initial observations are 64x64 color images, from which binary features are extracted (upper left). We tried two different sets of binary features. The first set consisted of 884 features like edges, corners and colors, and the second feature set was a post-processed version of the first. The idea of the second set was to create higher-order features which represented things like walls and hallways. To do this, images from Maze #1 were clustered according to the latent states, and then the binary features were averaged together to create a sort of filter. New images were tested against each filter, triggering if the response exceeded a threshold. There were 373 of these features. Note that while the images were all taken from Maze #1, they were also used in Maze #2, where the colors, hall geometry, etc. were all different.

We set $n = 3$. For the feature vector $\phi()$, we used "streamer features." These connect each observation variable only to its temporal successors (Fig. 4, right). There were between 12,000 and 50,000 total features in the final feature set. We trained on 200,000 samples generated with a random policy. For the NAC parameters, we used a TD rate of $\lambda = 0.85$, a step-size $\alpha = 10.0$, gradient termination test $\epsilon = 0.001$ and remembering factor $\beta = 0.0$.

Figure 5 shows the results. The random policy performed the same in both domains, regardless of map or dynamics. Higher rewards were obtained in general

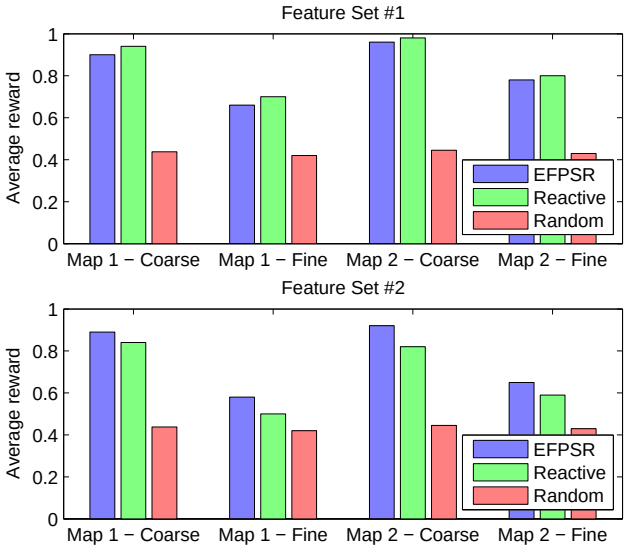


Figure 5. Results on the vision domain.

with coarse dynamics, regardless of map, feature set, or learning algorithm (presumably because the agent can reach high-reward regions more quickly).

The difference between the two feature sets that is most interesting. Using feature set #1, the EFPSR performs just under the performance of the reactive policy, regardless of map or dynamics. Perhaps this means that the EFPSR was unable to capture any meaningful dynamics, and instead learned to predict the identity function, with some noise. This would result in a policy equivalent to the reactive policy.

The results are reversed for feature set #2. Here, the EFPSR consistently outperforms the reactive policy. Together, these observations imply a coherent story. For both feature sets, we used the same set of streamer features. One plausible explanation for the results is that low-order conjunctions of more abstract features gives more modeling benefit than low-order conjunctions of granular, low-level features. It is easy to imagine that low-order conjunctions of granular features is insufficient to capture useful abstract structure in the domain. For example, to represent the corner of a wall, the agent might need a conjunction of 10 features, but we only had fourth order conjunctions. This was part of the motivation for feature set #2: because the camera images were clustered according to latent states, they were typically images of the same thing, from slightly different positions and angles. Using this feature set, the highest-order conjunction was still four or five, but these conjunctions may represent more abstract knowledge: if one feature represents "pink wall" and another represents "pink corner," perhaps a low-order conjunction could express "I'm looking at a pink wall, but if I turn left, I'll see a pink corner." The idea that low-order conjunctions of more abstract features gives more modeling benefit than low-order conjunctions of granular, low-level features suggests several directions for future improvement of these results.

Not reflected in the performance graphs is the computation required. Learning the model was relatively easy, taking only about 30 seconds. Because of the intensive rendering and relatively large size of the domains, the NAC algorithm required about a day to generate the policies to be reported. Informal calculations indicated that it would take about a week to get a single gradient with exact likelihood.

7. Conclusions and Future Work

We have presented a computationally efficient learning algorithm for the Linear-Linear EFPSR model and illustrated it on two domains. Our main contribution is an approximate likelihood, and the insight that maximizing it is equivalent to attempting to match stationary distributions. This idea may find traction in other learning problems. While evaluation of the model and learning algorithm is challenging, it is only by virtue of these approximations that we were able to attempt at all domains like the Bouncing Ball or the Robot Vision domain, which have continuous state spaces, rich observations, and tens of thousands of features. For both domains, we obtained better-than-reactive control policies, suggesting that information from history has successfully been incorporated into the state representation. This is a positive result considering the size of the data set and the number of features involved. Future work needs to address the problem of learning good atomic features and the graphical structure, since these appear to be key factors affecting performance.

A. Computing $\widehat{\mathcal{LL}}$ and Its Derivatives

To compute $\widehat{\mathcal{LL}}$ we must compute three terms: $\mathrm{E}_T[s_t]$ (the stationary distribution of states), $\mathrm{E}_T[\phi(f_t)]$ (which is computed once from data), and the log partition function $\log Z(\mathrm{E}_T[s_t])$. We begin with $\mathrm{E}_T[s_t]$. Recall that our goal is to compute this term in a way that is independent of T. This will be possible using Assumption 4.1, the linearity of the state update, and an insight related to stationary distributions:

$$\begin{aligned}
\mathrm{E}_T[s_t] &= \mathrm{E}_T\left[G(o_t)\left(As_{t-1}+B\right)\right] \\
&\approx \mathrm{E}_T\left[G(o_t)A\right]\mathrm{E}_T\left[s_{t-1}\right] + \mathrm{E}_T\left[G(o_t)\right]B \\
&= \mathrm{E}_T\left[G(o_t)A\right]\mathrm{E}_T\left[s_t\right] + B_G \\
&= G(\mathrm{E}_T[o_t])A\mathrm{E}_T\left[s_t\right] + B_G \\
&= \left(I - G(\mathrm{E}_T[o_t])A\right)^{-1} B_G \qquad (5)
\end{aligned}$$

where I is an appropriately sized identity matrix, and where $B_G = G(\mathrm{E}_T[o_t])B$. The second line follows by Assumption 4.1.

The third and fourth lines are both interesting for different reasons. The fourth line follows by the linearity of the operator $G(\cdot)$. The matrix $G(\mathrm{E}[o_t])$ can be interpreted as the expected transition operator, and is a simple function of the stationary distribution of observations $\mathrm{E}_T[o_t]$. The third line follows by the limiting properties of our expectations: we assume that $\mathrm{E}_T[s_t] = \mathrm{E}_T[s_{t-1}]$ because as $T \to \infty$, both represent the stationary distribution of states.

The result is that $\mathrm{E}_T[s_t]$ can be computed as the solution to a linear system of equations. Note that $G(E_T[o_t])$ will typically be very sparse, and a designer may force the A part to be sparse or low-rank. If so, a matrix-vector product can be computed efficiently, and an iterative solver should be used to solve Eq. 5.

Computing Derivatives. We now compute the derivatives of $\widehat{\mathcal{LL}}$ with respect to A and B:

$$\frac{\partial \widehat{\mathcal{LL}}}{\partial \{A, B\}} = \frac{\partial \widehat{\mathcal{LL}}}{\partial \mathrm{E}_T[s_t]}^\top \frac{\partial \mathrm{E}_T[s_t]}{\partial \{A, B\}}$$

We begin with the left-hand term:

$$\frac{\partial \widehat{\mathcal{LL}}}{\partial \mathrm{E}_T[s_t]} = \mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)] - \mathrm{E}_T[\phi(f_t)] \equiv \Delta$$

This result has an appealing intuitive interpretation. $\mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)]$ can be interpreted as the expected features that would be obtained if inference were performed using $\mathrm{E}_T[s_t]$ as the state – in other words, it represents the stationary distribution of features under the model. Since $\mathrm{E}_T[\phi(f_t)]$ represents the empirically observed stationary distribution, we see that the gradient wishes to match the two. If we use a variational method to compute the log partition function $\log Z(\mathrm{E}_T[s_t])$, which is needed to determine the value of the log-likelihood, then the expected features $\mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)]$ are available as a byproduct of the optimization. This is a pleasing efficiency.

However, we are not done. We still must find the transition parameters which allow us to move the expected sufficient statistics closer:

$$\frac{\partial \mathrm{E}_T[s_t]}{\partial A} = (I - G(\mathrm{E}[o_t])A)^{-1} \left(\frac{\partial}{\partial A} G(\mathrm{E}[o_t])A\right) \mathrm{E}_T[s_t]$$

We now find it convenient to remember that the full derivative also includes the term $\partial \widehat{\mathcal{LL}} / \partial \mathrm{E}_T[s_t] \equiv \Delta$, which is a column vector. Let $\Gamma \equiv \Delta^\top (I - G(\mathrm{E}[o_t])A)^{\top -1} G(\mathrm{E}[o_t])$. Then:

$$\Delta^\top \frac{\partial \mathrm{E}_T[s_t]}{\partial A} = \frac{\partial}{\partial A} \Gamma A \mathrm{E}_T[s_t] = \Gamma^\top \mathrm{E}_T[s_t]^\top$$

Algorithm 2 GRADS-OF-APPROX-LL

Input: $\mathrm{E}_T[o_t]$, $\mathrm{E}_T[\phi(f_t)]$, A, B

// Compute stationary distribution of states
$\mathrm{E}_T[s_t] = (I - G(\mathrm{E}_T[o_t])A)^{-1} B$

// Use $\mathrm{E}_T[s_t]$ to perform inference
Compute $\mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)]$ and $\log Z(\mathrm{E}_T[s_t])$

// Compute the approximate log-likelihood:
$\widehat{\mathcal{LL}} = -\mathrm{E}_T[s_t]^\top \mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)] - \log Z(\mathrm{E}_T[s_t])$.

// Compute the gradient:
$\Delta = \mathrm{E}[\phi(f_t)] - \mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)]$.
$\Gamma = \Delta^\top (I - G(\mathrm{E}[o_t])A)^{\top -1} G(\mathrm{E}[o_t])$
$\nabla_A \widehat{\mathcal{LL}} = \Gamma^\top \mathrm{E}_T[s_t]^\top$ ← note: a rank-one matrix
$\nabla_B \widehat{\mathcal{LL}} = G(\mathrm{E}_T[o_t])^\top \Delta$

Return $\widehat{\mathcal{LL}}, \nabla_A \widehat{\mathcal{LL}}, \nabla_B \widehat{\mathcal{LL}}$

The derivative with respect to B is similar:

$$\begin{aligned} \Delta^\top \frac{\partial \mathrm{E}_T[s_t]}{\partial B} &= \Delta^\top \frac{\partial}{\partial B} [G(\mathrm{E}[o_t])(A\mathrm{E}_T[s_t] + B)] \\ &= \frac{\partial}{\partial B} \Delta^\top G(\mathrm{E}[o_t])B = \Delta^\top G(\mathrm{E}[o_t]) \end{aligned}$$

The completed algorithm is shown in Algorithm 2.

Acknowledgments

Both authors were supported by NSF grant IIS-0413004. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the NSF.

References

Brand, M. (2006). Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, *415*, 20–30.

Dagum, P., & Luby, M. (1993). Approximate probabilistic reasoning in bayesian belief networks is NP-hard. *Artificial Intelligence*, *60*, 141–153.

Littman, M. L., Sutton, R. S., & Singh, S. (2002). Predictive representations of state. *Neural Information Processing Systems (NIPS)* (pp. 1555–1561).

Peters, J., Vijayakumar, S., & Schaal, S. (2005). Natural actor-critic. *European Conference on Machine Learning (ECML)* (pp. 280–291).

Rudary, M. R., Singh, S., & Wingate, D. (2005). Predictive linear-Gaussian models of stochastic dynamical systems. *Uncertainty in Artificial Intelligence* (pp. 501–508).

Wingate, D. (2008). *Exponential family predictive representations of state*. PhD thesis, University of Michigan.

Fully Distributed EM for Very Large Datasets

Jason Wolfe JAWOLFE@CS.BERKELEY.EDU
Aria Haghighi ARIA42@CS.BERKELEY.EDU
Dan Klein KLEIN@CS.BERKELEY.EDU
Computer Science Division, University of California, Berkeley, CA 94720

Abstract

In EM and related algorithms, E-step computations distribute easily, because data items are independent given parameters. For very large data sets, however, even storing all of the parameters in a single node for the M-step can be impractical. We present a framework that fully distributes the entire EM procedure. Each node interacts only with parameters relevant to its data, sending messages to other nodes along a junction-tree topology. We demonstrate improvements over a MapReduce topology, on two tasks: word alignment and topic modeling.

1. Introduction

With dramatic recent increases in both data scale and multi-core environments, it has become increasingly important to understand how machine learning algorithms can be efficiently parallelized. Many computations, such as the calculation of expectations in the E-step of the EM algorithm, decompose in obvious ways, allowing subsets of data to be processed independently. In some such cases, the MapReduce framework (Dean & Ghemawat, 2004) is appropriate and sufficient (Chu et al., 2006). Specifically, MapReduce is suitable when its centralized reduce operation can be carried out efficiently. However, this is not always the case. For example, in modern machine translation systems, many millions of words of example translations are aligned using unsupervised models trained with EM (Brown et al., 1994). In this case, one quickly gets to the point where no single compute node can store the model parameters (expectations over word pairs in this case) for all of the data at once, and communication required for a centralized reduce operation dominates computation time. The common solutions in practice are either to limit the total training data or to process manageable chunks independently. Either way, the complete training set is not fully exploited.

In this paper, we propose a general framework for distributing EM and related algorithms in which not only is the computation distributed, as in the map and reduce phases of MapReduce, but the storage of parameters and expected sufficient statistics is also fully distributed and maximally localized. No single node needs to store or manipulate all of the data or all of the parameters. We describe a range of network topologies and discuss the tradeoffs between communication bandwidth, communication latency, and per-node memory requirements. In addition to a general presentation of the framework, a primary focus of this paper is the presentation of experiments in two application cases: word alignment for machine translation (using standard EM) and topic modeling with LDA (using variational EM). We show empirical results on the scale-up of our method for both applications, across several topologies.

Previous related work in the sensor network literature has discussed distributing estimation of Gaussian mixtures using a tree-structured topology (Nowak, 2003); this can be seen as a special case of the present approach. Paskin et al. (2004) present an approximate message passing scheme that uses a junction tree topology in a related way, but for a different purpose. In addition, Newman et al. (2008) present an asynchronous sampling algorithm for LDA; we discuss this work further, below. None of these papers have discussed the general case of distributing and decoupling parameters in M-step calculations, the main contribution of the current work.

2. Expectation Maximization

Although our framework is more broadly applicable, we focus on the EM algorithm (Dempster et al., 1977), a technique for finding maximum likelihood param-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

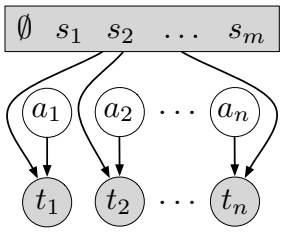


Figure 1: IBM Model 1 word alignment model. The top sentence is the source, and the bottom sentence is the target. Each target word is generated by a source word determined by the corresponding alignment variable.

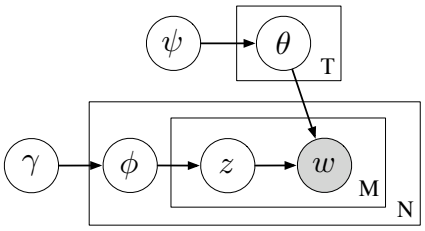


Figure 2: Latent Dirichlet Allocation model. Each word is generated from a topic vocabulary distribution and each topic is generated from a document topic distribution.

eters of a probabilistic model with latent or hidden variables. In this setting, each datum d_i consists of a pair (x_i, h_i) where x_i is the set of observed variables and h_i are unobserved. We assume a joint model over $P(x_i, h_i|\theta)$ with parameters θ. Our goal is to find a θ that maximizes the marginal observed log-likelihood $\sum_{i=1}^{m} \log P(x_i|\theta)$. Each iteration consists of two steps:

$$q_i(h_i) \leftarrow P(h_i|x_i, \theta) \qquad \text{[E-Step]}$$

$$\theta \leftarrow \arg\max_{\theta} \sum_{i=1}^{m} \mathbf{E}_{q_i} P(x_i|h_i, \theta) \qquad \text{[M-Step]}$$

where the expectation in the M-Step is taken with respect to the distribution $q(\cdot)$ over the latent variables found in the E-Step. When $P(\cdot|\theta)$ is a member of the exponential family, the M-Step reduces to solving a set of equations involving expected sufficient statistics under the distribution. Thus, the E-Step consists of collecting expected sufficient statistics $\eta = \mathbf{E}_\theta P(\boldsymbol{\eta}|X)$ with respect to q_i for each datum x_i. We briefly present two EM applications we use for experiments.

2.1. Word Alignment

Word alignment is the task of linking words in a corpora of parallel sentences. Each parallel sentence pair consists of a source sentence S and its translation T into a target language.[1] The model we present here is known as IBM Model 1 (Brown et al., 1994).[2] In this model, each word of T is generated from some word of S or from a null word $\emptyset$ prepended to each source sentence. The null word allows words to appear in the target sentence without any evidence in the source. Model 1 is a mixture model, in which each mixture component indicates which source word is responsible for generating the target word (see figure 1).

[1] Sometimes in the word alignment literature the roles of S and T are reversed to reflect the decoding process.

[2] Although there are more sophisticated models for this task, our concern is with efficiency in the presence of many parameters. More complicated models do not contain substantially more parameters.

The formal generative model is as follows: (1) Select a length n for the translation T based upon $|S| = m$ (typically uniform over a large range). (2) For each $j = 1, \ldots, n$, uniformly choose some source alignment position $a_j \in \{0, 1, \ldots, m\}$. (3) For each $j = 1, \ldots, n$, choose target word t_j based on source word s_{a_j} with probability $\theta_{s_{a_j} t_j}$

In the data, the alignment variables a are unobserved, and the parameters are the multinomial distributions $\theta_{s\cdot}$ for each source word s. The expected sufficient statistics are expected alignment counts between each source and target word that appear in a parallel sentence pair. These expectations can be obtained from the posterior probability of each alignment,

$$P(a_j = i|S, T, \theta) = \frac{\theta_{s_i t_j}}{\sum_{i'} \theta_{s_{i'} t_j}}$$

The E-Step computes the above posterior for each alignment variable; these values are added to the current expected counts of (s, t) pairings, denoted by η_{st}. The M-Step consists of the following update: $\theta_{st} \leftarrow \frac{\eta_{st}}{\sum_{t'} \eta_{st'}}$. Section 5.1 describes results for this model on a data set with more than 243 million parameters (i.e., distinct co-occurring word pairs).

2.2. Topic Modeling

We present experiments in topic modeling via the Latent Dirichlet Allocation (Blei et al., 2003) topic model (see figure 2). In LDA, we fix a finite number of topics T and assume a closed vocabulary of size V. We assume that each topic t has a multinomial distribution $\theta_{t\cdot} \sim \text{Dirichlet}(\text{Unif}(V), \psi)$. Each document draws a topic distribution $\phi \sim \text{Dirichlet}(\text{Unif}(T), \gamma)$. For each word position in a document, we draw an unobserved topic index z from ϕ and then draw a word from $\theta_{z\cdot}$.

Our goal is to find the MAP estimate of θ for the observed likelihood where the latent topic indicators and document topic distributions ϕ have been integrated out. In this setting, we can not perform an exact E-Step because of the coupling of latent variables through the integral over parameters. Instead,

we use a variational approximation of the posterior as outlined in Blei et al. (2003), where all parameters and latent variables are marginally independent. The relevant expected sufficient statistics for θ are the expected counts η_{tw} over topic t and word w pairings under the approximate variational distribution. The M-Step, as in the case of our word alignment model in section 2.1, consists of normalizing these counts: $\theta_{tw} = \frac{\eta_{tw}}{\sum_{w'} \eta_{tw'}}$. Section 5.2 describes results for this model. We note that the number of parameters in this model is a linear function of the number of topics T.

3. Distributing EM

Given the amount of data and number of parameters in many EM applications, it is worthwhile to distribute the algorithm across many machines. We will consider the setting in which our data set $\mathcal{D}$ has been divided into k splits $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$.

3.1. Distributing the E-Step

Distributing the E-Step is relatively straightforward, since the expected sufficient statistics for each datum can be computed independently given a current estimate of the parameters. Each of k nodes computes expected sufficient statistics for one split of the data,

$$\eta^{(i)} = \mathbf{E}_\theta \left[\eta | \mathcal{D}_i\right] \qquad \text{[Distributed E-Step]}$$

where we use the superscript (i) to emphasize that these counts are partial and reflect only the contributions from split $\mathcal{D}_i$ and not contributions from other partitions. We will also write α_i for the set of sufficient statistic *indices* that have nonzero count in $\eta^{(i)}$, and use $\eta[\alpha_i]$ to indicate the projection of η onto the subspace consisting of just those statistics in α_i.

In order to complete the E-Step, we must aggregate expected counts from all partitions in order to re-estimate parameters. This step involves distributed communication of a potentially large number of statistics. We name this phase the C-Step and will examine how to efficiently perform it in section 4. For the moment, we assume that there is a single computing node which accumulates all partial sufficient statistics,

$$\eta = \sum_{i=1}^{k} \eta^{(i)}[\alpha_i] \qquad \text{[C-Step]}$$

where we write $\eta^{(i)}[\alpha_i]$ to indicate that we only communicate non-zero counts. This is a simple and effective way to achieve near-linear speedup in the E-Step; previous work has utilized it effectively (Blei et al., 2003; Chu et al., 2006; Nowak, 2003).

3.2. Distributing the M-Step

A further possibility, which to our knowledge has not been fully exploited, is distributing the M-Step. Often in EM, it is the case that only a subset of parameters may ever be relevant to a split $\mathcal{D}_i$ of the data. For instance, in the word alignment model of section 2.1, if a word pairing (s, t) is not observed in some $\mathcal{D}_i$, node i will never need the parameter θ_{st}. For our full word alignment data set, when $k = 20$, less than 30 million of the 243 million total parameters are relevant to each node.

We will use β_i to refer to the subset of parameter indices relevant for $\mathcal{D}_i$. In order to distribute the M-Step, each node must receive all expected counts necessary to re-estimate all relevant parameters $\theta[\beta_i]$. In section 4, we develop different schemes for how nodes should communicate their partial expected counts, and show that this choice of C-Step topology can dramatically affect the efficiency of distributed EM.

One difficulty in distributing the M-Step lies in the fact that re-estimating $\theta[\beta_i]$ may require counts not found in $\eta[\alpha_i]$. In the case of the word alignment model, θ_{st} requires the counts $\eta_{st'}$ for all t' appearing with s in a sentence pair, even if t' did not occur in $\mathcal{D}_i$. Often these non-local statistics enter the computation only via normalization terms. This is the case for the word alignment and LDA models explored here. This observation suggests an easy way to get around the problem presented above in the case of discrete latent variables: we simply *augment* the set of sufficient statistics η with a set of redundant *sum* terms that provide the missing information needed to normalize parameter estimates. For the word alignment model, we would include a sufficient statistic $\eta_{s\cdot}$ to represent the sum $\sum_{t:(s,t)\in\mathcal{D}} \eta_{st}$. Then the re-estimated value of θ_{st} would simply be $\frac{\eta_{st}}{\eta_{s\cdot}}$. With these augmented statistics, estimating $\theta[\beta_i]$ requires only η_{st} and $\eta_{s\cdot}$ for all $(s, t) \in \mathcal{D}_i$. It might seem counterintuitive, but adding these extra statistics actually *decreases* the total necessary amount of communication, by trading a large number of sparse statistics for a few dense ones.

4. Topologies for Distributed EM

This section will consider techniques for performing the C-Step of distributed EM, in which a node i obtains the necessary sufficient statistics $\eta[\alpha_i]$ to estimate parameters $\theta[\beta_i]$. We assume that the sets of relevant count indices α_i have been augmented as discussed at the end of section 3 so that $\eta[\alpha_i]$ is sufficient to re-estimate $\theta[\beta_i]$.

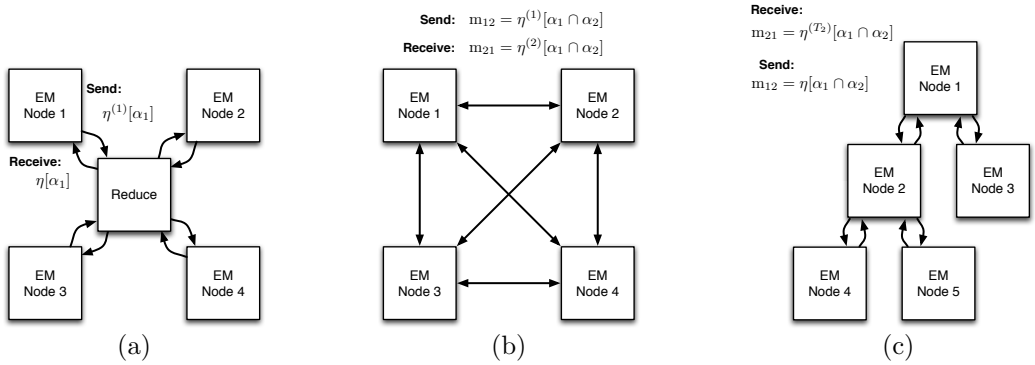


Figure 3: (a) MapReduce: Each node computes partial statistics in a local E-Step, sends these to a central "Reduce" node, and receives back completed statistics relevant for completing its local M-Step. (b) AllPairs: Each node communicates to each other node only the relevant partial sufficient statistics. For many applications, these intersections will be small. (c) JunctionTree: The network topology is a tree, chosen heuristically to optimize any desired criteria (e.g., bandwidth).

4.1. MapReduce Topology

A straightforward way to implement the C-Step is to have each node send its non-zero partial counts $\eta^{(i)}[\alpha_i]$ to a central "Reduce" node for accumulation into η. This central node then returns only the relevant completed counts $\eta[\alpha_i]$ to the nodes so that they can independently perform their local M-Steps. This approach, depicted in figure 3(a), is roughly analogous to the topology used in the MapReduce framework (Dean & Ghemawat, 2004). When parameters are numerous, this will already be more bandwidth-efficient than a naive MapReduce approach, in which the Reduce node would perform a global M-Step and then send *all* of the new parameters θ back to all nodes for the next iteration. To enable sending only relevant counts $\eta[\alpha_i]$, the actual iterations are preceded by a setup phase in which each node constructs an array of relevant count indices α_i and sends this to the Reduce node. This array also fixes an ordering on relevant statistics, so that later messages of counts can be densely encoded.

This MapReduce topology[3] may be a good choice for the C-Step when nodes share most of the same statistics. On the other hand, if sufficient statistics are sparse and numerous, the central reduce node can be a significant bandwidth and memory bottleneck in the distributed EM algorithm. Indeed, in practice, with either Model 1 or LDA, available amounts of training data can and do easily cause the sufficient statistics vectors to exceed the memory of any single node. The MapReduce topology for estimation of LDA has been discussed in related work, notably Newman et al. (2008), though they do not consider the sparse distribution of the M-step, which is necessary for very large data sets.

[3]For the remainder of this paper we will use MapReduce to refer to the *topology* used by the MapReduce system (Dean & Ghemawat, 2004). While the particular details of our implementation will differ substantially from the MapReduce system (e.g., we use a single reduce node), many key results should hold more generally (e.g., the MapReduce approach uses unnecessarily high bandwidth).

4.2. AllPairs Topology

MapReduce takes a completely *centralized* approach to implementing the C-Step, in which the accumulation of η at the Reduce node can be slow or even infeasible. This suggests a *decentralized* approach, in which nodes directly pass relevant counts to one another and no single node need store all of η or θ. This section describes one such approach, AllPairs, which in a sense represents the opposite extreme from MapReduce. In AllPairs, the network graph is a clique on the k nodes, and each node i passes a message $m_{ij} = \eta^{(i)}[\alpha_i \cap \alpha_j]$ to each other node j containing precisely the statistics j needs and nothing more (see figure 3(b)). Each node j then computes its completed set of sufficient statistics with a simple summation:

$$\begin{aligned}\eta[\alpha_i] &= \eta^{(i)} + \sum_{j \neq i} m_{ji} \\ &= \eta^{(i)} + \sum_{j \neq i} \eta^{(j)}[\alpha_i \cap \alpha_j]\end{aligned}$$

AllPairs requires a more complicated setup phase, where each node i calculates, for roughly half of the other nodes, the intersection $\alpha_i \cap \alpha_j$ of its parameters with the other node j's.[4] Node i then sends the contents of this intersection to j.

In each iteration, message passing proceeds asynchronously, and each node begins its local M-Step as

[4]Note that the C-Step time is now sensitive to how our data is partitioned. An interesting area for future work is intelligently partitioning the data so that data split intersections are small.

soon as it has finished sending and receiving the necessary counts. An important point is that, to avoid double counting, a received count cannot be folded into a node's local statistics until the local copy of that count has been incorporated into all outgoing messages.

AllPairs is attractive because it lacks the bandwidth bottleneck of MapReduce, all paths of communication are only one hop long, and each node need only be concerned with precisely those statistics relevant for its local E- and M-steps. On the down side, AllPairs needs a full crossbar connection between nodes, and requires unnecessarily high bandwidth for dense sufficient statistics that are relevant to datums on many nodes. In particular, a statistic that is relevant to k' nodes must be passed $k'(k'-1)$ times, as compared to an optimal value of $2(k'-1)$ (see section 4.3).

4.3. JunctionTree Topology

A tree-based topology related to the junction tree approach used for belief propagation in graphical models (Pearl, 1988) can avoid the bandwidth bottleneck of MapReduce and the bandwidth explosion of AllPairs. In this approach, the k nodes are embedded in an arbitrary tree structure T, and messages are passed along the edges in both directions (see figure 3(c)). We are certainly not the first to exploit such structures for distributing computation; see particularly Paskin et al. (2004), who use it for inference rather than estimation.

We first describe the most bandwidth-efficient method for communicating partial results about a *single* statistic, and then show how this can be extended to produce an algorithm that works for the entire C-Step. Consider a single sufficient statistic η_x (e.g., some η_{st} for Model 1) which is only relevant to E- and M-Steps on some subset of machines S. Before the C-Step, each node has $\eta_x^{(i)}$, and after communication each node should have $\eta_x = \sum_{i \in S} \eta_x^{(i)}$. We cannot hope to accomplish this goal by passing fewer than $2(|S|-1)$ pairwise messages; clearly, it must take at least $|S|-1$ messages before *any* node completes its counts, and then another $|S|-1$ messages for each of the other $|S|-1$ nodes to complete theirs too. This is fewer messages than either MapReduce or AllPairs passes.

This theoretical minimum bandwidth can be achieved by embedding the nodes of S in a tree. After designating an arbitrary node as the root, each node accumulates a partial sum from its *subtree* and then passes it up towards the root. Once the root has accumulated the completed sum η_x, it is recursively passed back down the tree until all nodes have received the completed count, for a total of $2(|S|-1)$ messages.

Of course, each node must obtain a *set* of complete relevant statistics $\eta[\alpha_i]$ rather than a single statistic η_x. One possibility is to pass messages for each sufficient statistic on a separate tree; while this represents the *bandwidth-optimal* solution for the entire C-step, in practice the overhead of managing 240 million different message trees would likely outweigh the benefits.

Instead, we can simply force all statistics to share the *same* global tree T. In each iteration we proceed much as before, designating an arbitrary root node and passing messages up and then down, except that now the message m_{ij} from node i to j conveys the intersection of their relevant statistics $\alpha_i \cap \alpha_j$ rather than a single number. For this to work properly, we require that T has the following *running intersection* property: for each sufficient statistic, all concerned nodes form a *connected subtree* of T. In other words, for all triples of nodes (i, x, j) where x is on the path from i to j, we must have $(\alpha_i \cap \alpha_j) \subseteq \alpha_x$. We can assume that this property holds, by augmenting sets of statistics at interior nodes if necessary.

When the running intersection property holds, the message contents can be expressed as

$$\begin{aligned} m_{ij} &= \eta^{(T_i)}[\alpha_i \cap \alpha_j] && \text{towards root} \\ m_{ji} &= \eta[\alpha_i \cap \alpha_j] && \text{away from root} \end{aligned}$$

where T_i is used to represent the subtree rooted at i, and $\eta^{(T_i)}$ is the sum of statistics from nodes in this subtree. Thus, the single global message passing phase can be thought of as $|\alpha|$ separate single-statistic message passing operations proceeding in parallel, where the root of each such sub-phase is the node in its subtree closest to the global root, and irrelevant operations involving other nodes and statistics can be ignored. In our actual implementation, we instead use an asynchronous message-passing protocol common in probabilistic reasoning systems (Pearl, 1988), which avoids the need to designate a root node in advance.

The setup phase for JunctionTree proceeds as follows: (1) All pairwise intersections of statistics are computed and saved to shared disk. (2) An arbitrary node chooses and broadcasts a directed, rooted tree T on the nodes which optimizes some criterion. (3) Each node (except the root) constructs the set of statistics that must lie on its incoming edge, by taking the union of the intersections of statistics (which can be reread from disk) for all pairs of nodes on opposite sides of the edge.[5] (4) Each node passes the constructed edge set along its incoming edge, fixing future message structures in the process. (5) Each node augments its α_i to

[5] More efficient algorithms are possible, but they require more memory.

include all statistics in local outgoing messages, thus enforcing the running intersection property.

To choose a heuristically good topology, we use the maximum spanning tree (MST) with edge weights equal to the sizes of the intersections $|\alpha_i \cap \alpha_j|$, so that nodes with more shared statistics tend to be closer together. This heuristic has been successfully used in the graphical models literature (Pearl, 1988) to construct junction trees. However, in general one can imagine much better heuristics that also consider, e.g., max degree, tree diameter or underlying network structure.

If statistics tend to be well-clustered within and between nodes, we can expect this MST to require less bandwidth than either alternate topology, and (like AllPairs) there should be no central bandwidth bottleneck. On the other hand, if statistics tend to be shared between only a few nodes and this sharing is not appropriately clustered, bandwidth and memory may increase because many statistics will have to be added to enforce the running intersection property.[6] Furthermore, if the diameter of the tree is large, latency may become an issue as many sequential message sending and incorporation steps will have to be performed. Finally, the setup phase takes longer because choosing the tree topology and enforcing the running intersection property may be expensive. Despite these potential drawbacks, we will see that MST generally performs best of the three topologies investigated here in terms of both bandwidth and total running time.

As a final note, if T is a "hub and spoke" graph, and the hub's statistics are augmented to contain all of η, a MapReduce variant is recovered as a special case of JunctionTree. This is the version of MapReduce we actually implemented; it differs from the version described in section 4.1 only in that the role of reduce node is assigned to one of the workers rather than a separate node, which reduces bandwidth usage.

5. Experiments

We performed experiments using the word alignment model from section 2.1 and the LDA topic model from section 2.2. For each of these models, we compared the network topologies used to perform the C-Step and how they affect the overall efficiency of EM. We implemented the following topologies (described in section 4): MapReduce, AllPairs, and JunctionTree. Although our implementation was done in Java, every reasonable care was taken to be time and memory efficient in our choice of data structures and in network socket communication. All experiments were performed on a cluster of identical, load-free 3.0 GHz 32-bit Intel machines. Running times per iteration represent the median over 10 runs of the maximum time on any node. We also examine the bandwidth of each topology, measured by the number of counts communicated across the network per iteration.

5.1. Word Alignment Results

We performed Model 1 (see section 2.1) experiments on the UN Arabic English Parallel Text TIDES Version 2 corpus, which consists of about 3 million sentences of translated UN proceedings from 1994 until 2001.[7] For the full data set, there are more than 243 million distinct parameters.

In table 1(a), we present results where the number of sentence-pair datums per node is held constant at 145K and the number of nodes (and thus total training data) is varied. For 10 or more nodes, the MapReduce topology runs out of memory due to the number of statistics that must be stored in memory at the Reduce node.[8] In contrast, both AllPairs and JunctionTree complete training for the full data set distributed on 20 nodes.

We also experimented with the setting where we fix the total amount of data at 200K sentences, but add more nodes to distribute the work. Figure 4 gives iteration times for all three topologies broken down according to E-, C-, and M-Steps. The MapReduce graph (figure 4(a)) shows that the C-Step begins dominating run time as the number of nodes increases. This effect reduces the benefit from distributing EM for larger numbers of nodes. Both AllPairs and JunctionTree have substantially smaller C-Steps, which contributes to much faster per-iteration times and also allows larger numbers of nodes to be effective.

On the full dataset, JunctionTree outperforms AllPairs, but not by a substantial margin. Although the two topologies have roughly comparable running times, they have different network behaviors. Figure 5, which compares bandwidth usage in billions of counts transferred over the network per iteration, shows that AllPairs uses substantially more bandwidth than either MapReduce or JunctionTree. This is due to the $O(k^2)$ number of messages sent per iteration. In contrast, JunctionTree typically has a higher la-

[6] This could be avoided by using different trees for different sets of statistics; we leave this for future work.

[7] LDC catalog #LDC2004E13. See `http://projects.ldc.upenn.edu/TIDES/index.html`.

[8] This issue could be sidestepped by using multiple Reduce nodes as in the MapReduce system; however, the fundamental inefficiency of the MapReduce topology would remain.

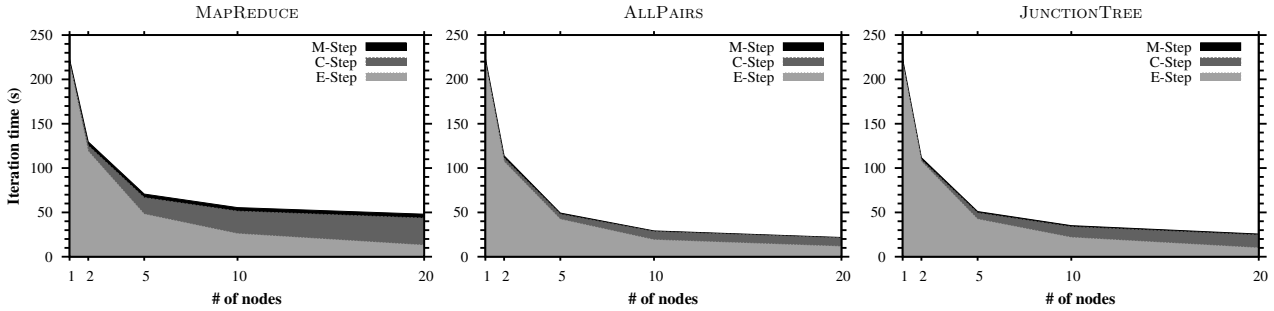


Figure 4: Speedup of median iteration time for three topologies as a function of the number of nodes, training Model 1 on 200k total sentence pairs. Time for each iteration is broken down into E-, C-, and M-Step time. The M-Step is present but difficult to see due to its brevity.

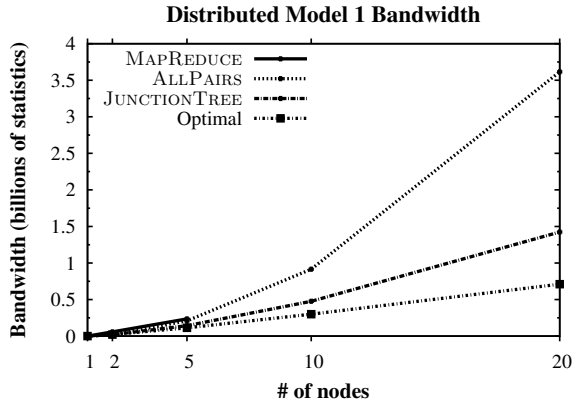


Figure 5: Bandwidth usage for three topologies compared to optimal, as a function of the number of nodes, training on Model 1 with 145k sentence pairs per node. MapReduce ran out of memory when run on more than 5 nodes.

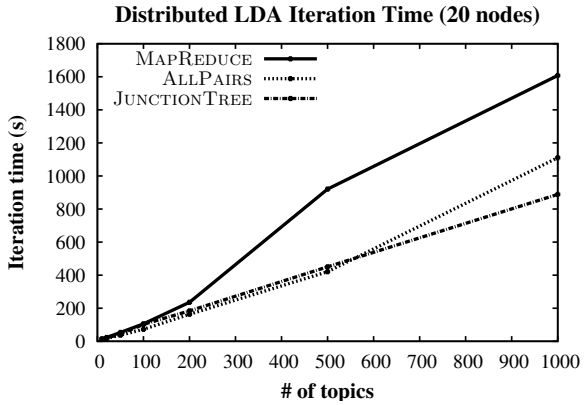


Figure 6: Median iteration time for three topologies, as a function of the number of topics, training on LDA with 20 nodes and all 804k documents.

tency due to the fact that nodes must wait to receive messages before they can send their own. AllPairs and JunctionTree with the MST heuristic represent a bandwidth and latency tradeoff, and the choice of which to use depends on the properties of the particular network.

5.2. Topic Modeling Results

We present results for the variational EM LDA topic model presented in section 2.2. Our results are on the Reuters Corpus Volume 1 (Lewis et al., 2004). This corpus consists of 804,414 newswire documents, where all tokens have been stemmed and stopwords removed.[9] There are approximately 116,000 unique word types after pre-processing. The number of parameters of interest is therefore 116,000T, where T is the number of topics that we specify.

We experimented with this model on the entire corpus and varied the number of topics. The largest number of topics we used was $T = 1{,}000$, which yields 116 million unique parameters. Our results on iteration time are presented in figure 6. Note that the number of parameters depends linearly on the number of topics, which can roughly be seen in figure 6. This figure demonstrates that the efficiency of the AllPairs and JunctionTree topologies as the number of parameters increases. We see that JunctionTree edges out AllPairs for a larger number of topics.

Table 1(b) shows detailed results for the experiment depicted in figure 6. Besides the difference in iteration times for the three algorithms as the number of topics (and statistics) grows, there are at least two other salient points. First, while the number of total statistics grows similarly to in the word alignment experiments, here the number of unique statistics is significantly smaller (i.e., each statistic, on average, is relevant to more nodes). This leads to significantly worse performance, especially in terms of bandwidth, for AllPairs. A second point is that setup times are much lower than for word alignment, because sets of relevant *words* can be determined first, and only then expanded to $(word, topic)$ pairs.

[9] We used the processed version of the corpus provided by Lewis et al. (2004).

Model 1, 145k sentence pairs per node

# nodes	1	2	5	10	20
# Unique Stats (in M)	29.37	47.84	90.58	147.65	243.01
# Total Stats (in M)	29.37	58.18	146.96	297.30	597.95
Opt Bandwidth (M of stats)	0.00	20.68	112.76	299.31	709.88
MapReduce					
Setup Time (s)	138.37	185.01	458.72	*	*
E-Step Time (s)	149.66	177.73	196.45	*	*
C-Step Time (s)	0.002	8.41	282.43	*	*
M-Step Time (s)	3.18	5.48	10.65	*	*
Iteration Time (s)	152.85	191.62	489.54	*	*
Max Hops	0	1	2	*	*
Bandwidth (M of stats)	0.00	58.75	233.18	*	*
Bottleneck (M of stats)	0.00	58.75	233.18	*	*
AllPairs					
Setup Time (s)	138.37	262.98	332.52	584.08	1003.11
E-Step Time (s)	149.66	163.37	166.99	168.66	204.63
C-Step Time (s)	0.002	2.91	17.64	56.51	594.18
M-Step Time (s)	3.18	3.43	3.53	3.49	3.61
Iteration Time (s)	152.85	169.71	188.16	228.66	802.43
Max Hops	0	1	1	1	1
Bandwidth (M of stats)	0.00	20.68	207.64	915.35	3615.97
Bottleneck (M of stats)	0.00	10.34	42.13	93.68	189.04
JunctionTree					
Setup Time (s)	138.37	262.98	393.77	868.22	2392.72
E-Step Time (s)	149.66	163.37	167.32	196.00	222.14
C-Step Time (s)	0.002	2.91	24.73	51.89	536.80
M-Step Time (s)	3.18	3.43	4.20	6.05	8.85
Iteration Time (s)	152.85	169.71	196.25	253.94	767.79
Max Hops	0	1	3	6	13
Bandwidth (M of stats)	0.00	20.68	142.51	475.82	1424.26
Bottleneck (M of stats)	0.00	10.34	54.50	92.84	171.12

(a)

LDA, all 804k documents, 20 nodes

# topics	10	50	100	500	1000
# Unique Stats (in M)	1.16	5.82	11.64	58.18	116.36
# Total Stats (in M)	5.03	25.17	50.34	251.71	503.43
Opt Bandwidth (M of stats)	7.74	38.71	77.41	387.07	774.15
MapReduce					
Setup Time (s)	3.90	14.17	23.58	96.50	225.85
E-Step Time (s)	9.36	24.65	47.16	260.44	524.09
C-Step Time (s)	5.18	26.37	51.91	599.32	993.60
M-Step Time (s)	0.20	2.69	6.51	39.19	89.88
Iteration Time (s)	14.73	53.72	105.58	898.95	1607.56
Max Hops	2	2	2	2	2
Bandwidth (M of stats)	9.52	47.60	95.20	475.99	951.98
Bottleneck (M of stats)	9.52	47.60	95.20	475.99	951.98
AllPairs					
Setup Time (s)	20.44	29.72	35.19	213.49	549.89
E-Step Time (s)	9.15	23.19	46.97	265.74	518.71
C-Step Time (s)	2.62	13.09	24.23	146.24	572.00
M-Step Time (s)	0.05	0.49	1.45	8.85	20.01
Iteration Time (s)	11.82	36.78	72.65	420.83	1110.72
Max Hops	1	1	1	1	1
Bandwidth (M of stats)	52.29	261.43	522.87	2614.33	5228.65
Bottleneck (M of stats)	2.68	13.40	26.80	134.00	268.01
JunctionTree					
Setup Time (s)	22.92	25.15	25.16	67.54	124.36
E-Step Time (s)	8.99	23.25	68.59	256.60	514.02
C-Step Time (s)	3.81	19.10	30.58	173.23	330.98
M-Step Time (s)	0.11	1.18	3.13	20.66	43.62
Iteration Time (s)	12.91	43.53	102.30	450.49	888.62
Max Hops	14	14	14	14	14
Bandwidth (M of stats)	12.85	64.23	128.46	642.30	1284.60
Bottleneck (M of stats)	1.39	6.93	13.87	69.33	138.67

(b)

Table 1: (a) Results for scaling up number of nodes, training Model 1 with 145k sentence pairs per node. (b) Results for scaling up number of topics, training LDA with all 804k documents on 20 nodes. All times are measured in seconds, statistics are counted in millions, and bandwidths are measured in millions of statistics passed per iteration. # unique stats measures $|\alpha|$, whereas # total stats measures $\sum_i |\alpha_i|$. Opt bandwidth is theoretically optimal bandwidth (see section 4.3). Setup time includes all time until all nodes started the first E-Step. Median total time per iteration is given, as well as a breakdown into E-, C-, and M-Steps. Max hops is the diameter of the graph. Bottleneck is maximum bandwidth in and out of any single node. (*) indicates an out-of-memory error.

We note that the total bandwidth is actually *lower* for MapReduce than JunctionTree since the MST only heuristically minimizes the *number* of disconnected statistic components, rather than the true cost of enforcing the running intersection property. Despite this, the bandwidth bottleneck for JunctionTree is still much lower than for MapReduce.

6. Conclusion

We have demonstrated theoretically and empirically that a distributed EM system can function successfully, allowing for both significant speedup and scaling up to computations that would be too large to fit in the memory of a single machine. Future work will consider applications to other machine learning methods, alternative junction tree heuristics, and more general graph topologies.

Acknowledgments

The authors of this work were supported (respectively) by DARPA IPTO contract FA8750-05-2-0249, a Microsoft Research Fellowship, and a Microsoft Research New Faculty Fellowship.

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *JMLR*, *3*, 993–1022.

Brown, P. F., Pietra, S. D., Pietra, V. J. D., & Mercer, R. L. (1994). The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, *19*, 263–311.

Chu, C.-T., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., & Olukotum, K. (2006). Map-Reduce for Machine Learning on Multicore. *NIPS*.

Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. *Sixth Symposium on Operating System Design and Implementation*.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A New Benchmark Collection for Text Categorization Research. *JMLR*.

Newman, D., Asuncion, A., Smyth, P., & Welling, M. (2008). Distributed Inference for Latent Dirichlet Allocation. *NIPS*.

Nowak, R. (2003). Distributed EM algorithms for density estimation and clustering in sensor networks. *IEEE Transactions on Signal Processing*, *51*, 2245–2253.

Paskin, M., Guestrin, C., & McFadden, J. (2004). Robust Probabilistic Inference in Distributed Systems. *UAI*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman.

Listwise Approach to Learning to Rank - Theory and Algorithm

Fen Xia* FEN.XIA@IA.AC.CN
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, P. R. China.

Tie-Yan Liu TYLIU@MICROSOFT.COM
Microsoft Research Asia, Sigma Center, No.49 Zhichun Road, Haidian District, Beijing, 100190, P. R. China.

Jue Wang JUE.WANG@IA.AC.CN
Wensheng Zhang WENSHENG.ZHANG@IA.AC.CN
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, P. R. China.

Hang Li HANGLI@MICROSOFT.COM
Microsoft Research Asia, Sigma Center, No.49 Zhichun Road, Haidian District, Beijing, 100190, P. R. China.

Abstract

This paper aims to conduct a study on the listwise approach to learning to rank. The listwise approach learns a ranking function by taking individual lists as instances and minimizing a loss function defined on the predicted list and the ground-truth list. Existing work on the approach mainly focused on the development of new algorithms; methods such as RankCosine and ListNet have been proposed and good performances by them have been observed. Unfortunately, the underlying theory was not sufficiently studied so far. To amend the problem, this paper proposes conducting theoretical analysis of learning to rank algorithms through investigations on the properties of the loss functions, including consistency, soundness, continuity, differentiability, convexity, and efficiency. A sufficient condition on consistency for ranking is given, which seems to be the first such result obtained in related research. The paper then conducts analysis on three loss functions: likelihood loss, cosine loss, and cross entropy loss. The latter two were used in RankCosine and ListNet. The use of the likelihood loss leads to the development of a new listwise method called ListMLE, whose loss function offers better properties, and also leads to better experimental results.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.
*The work was performed when the first author was an intern at Microsoft Research Asia.

1. Introduction

Ranking, which is to sort objects based on certain factors, is the central problem of applications such as information retrieval (IR) and information filtering. Recently machine learning technologies called 'learning to rank' have been successfully applied to ranking, and several approaches have been proposed, including the pointwise, pairwise, and listwise approaches.

The listwise approach addresses the ranking problem in the following way. In learning, it takes ranked lists of objects (e.g., ranked lists of documents in IR) as instances and trains a ranking function through the minimization of a listwise loss function defined on the predicted list and the ground truth list. The listwise approach captures the ranking problems, particularly those in IR in a conceptually more natural way than previous work. Several methods such as RankCosine and ListNet have been proposed. Previous experiments demonstrate that the listwise approach usually performs better than the other approaches (Cao et al., 2007)(Qin et al., 2007).

Existing work on the listwise approach mainly focused on the development of new algorithms, such as RankCosine and ListNet. However, there was no sufficient theoretical foundation laid down. Furthermore, the strength and limitation of the algorithms, and the relations between the proposed algorithms were still

not clear. This largely prevented us from deeply understanding the approach, more critically, from devising more advanced algorithms.

In this paper, we aim to conduct an investigation on the listwise approach.

First, we give a formal definition of the listwise approach. In ranking, the input is a set of objects, the output is a permutation of the objects[1], and the model is a ranking function which maps a given input to an output. In learning, the training data is drawn i.i.d. according to an unknown but fixed joint probability distribution between input and output. Ideally we would minimize the expected $0-1$ loss defined on the predicted list and the ground truth list. Practically we instead manage to minimize an empirical surrogate loss with respect to the training data.

Second, we evaluate a surrogate loss function from four aspects: (a) consistency, (b) soundness, (c) mathematical properties of continuity, differentiability, and convexity, and (d) computational efficiency in learning. We give analysis on three loss functions: likelihood loss, cosine loss, and cross entropy loss. The first one is newly proposed in this paper, and the last two were used in RankCosine and ListNet, respectively.

Third, we propose a novel method for the listwise approach, which we call ListMLE. ListMLE formalizes learning to rank as a problem of minimizing the likelihood loss function, equivalently maximizing the likelihood function of a probability model. Due to the nice properties of the loss function, ListMLE stands to be more effective than RankCosine and ListNet.

Finally, we have experimentally verified the correctness of the theoretical findings. We have also found that ListMLE can significantly outperform RankCosine and ListNet.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 gives a formal definition to the listwise approach. Section 4 conducts theoretical analysis of listwise loss functions. Section 5 introduces the ListMLE method. Experimental results are reported in Section 6 and the conclusion and future work are given in the last section.

2. Related Work

Existing methods for learning to rank fall into three categories. The pointwise approach (Nallapati, 2004) transforms ranking into regression or classification on single objects. The pairwise approach (Herbrich et al., 1999) (Freund et al., 1998) (Burges et al., 2005) transforms ranking into classification on object pairs. The advantage for these two approaches is that existing theories and algorithms on regression or classification can be directly applied, but the problem is that they do not model the ranking problem in a straightforward fashion. The listwise approach can overcome the drawback of the aforementioned two approaches by tackling the ranking problem directly, as explained below.

For instance, Cao et al. (2007) proposed one of the first listwise methods, called ListNet. In ListNet, the listwise loss function is defined as cross entropy between two parameterized probability distributions of permutations; one is obtained from the predicted result and the other is from the ground truth. Qin et al. (2007) proposed another method called RankCosine. In the method, the listwise loss function is defined on the basis of cosine similarity between two score vectors from the predicted result and the ground truth[2]. Experimental results show that the listwise approach usually outperforms the pointwise and pariwise approaches.

In this paper, we aim to investigate the listwise approach to learning to rank, particularly from the viewpoint of loss functions. Actually similar investigations have also been conducted for classification. For instance, in classification, consistency and soundness of loss functions are well studied. Consistency forms the basis for the success of a loss function. It is known that if a loss function is consistent, then the learned classifier can achieve the optimal Bayes error rate in the large sample limit. Many well known loss functions such as hinge loss, exponential loss, and logistic loss are all consistent (cf., (Zhang, 2004)(Bartlett et al., 2003)(Lin, 2002)). Soundness of a loss function guarantees that the loss can represent well the targeted learning problem. That is, an incorrect prediction should receive a larger penalty than a correct prediction, and the penalty should reflect the confidence of prediction. For example, hinge loss, exponential loss, and logistic loss are sound for classification. In contrast, square loss is sound for regression but not for classification (Hastie et al., 2001).

3. Listwise Approach

We give a formal definition of the listwise approach to learning to rank. Let X be the input space whose

[1] In this paper, we use permutation and ranked list interchangeably.

[2] In a broad sense, methods directly optimizing evaluation measures, such as SVM-MAP (Yue et al., 2007) and AdaRank (Xu & Li, 2007) can also be regarded as listwise algorithms. We will, however, limit our discussions in this paper on algorithms like ListNet and RankCosine.

elements are sets of objects to be ranked, Y be the output space whose elements are permutations of objects, and P_{XY} be an unknown but fixed joint probability distribution of X and Y. Let $\mathbf{h} : X \to Y$ be a ranking function, and H be the corresponding function space (i.e., $\mathbf{h} \in H$). Let $\mathbf{x} \in X$ and $\mathbf{y} \in Y$, and let $y(i)$ be the index of object which is ranked at position i. The task is to learn a ranking function that can minimize the expected loss $R(\mathbf{h})$, defined as:

$$R(\mathbf{h}) = \int_{X \times Y} l(\mathbf{h}(\mathbf{x}), \mathbf{y}) dP(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $l(\mathbf{h}(\mathbf{x}), \mathbf{y})$ is the $0-1$ loss function such that

$$l(\mathbf{h}(\mathbf{x}), \mathbf{y}) = \begin{cases} 1, & \text{if } \mathbf{h}(\mathbf{x}) \neq \mathbf{y} \\ 0, & \text{if } \mathbf{h}(\mathbf{x}) = \mathbf{y}, \end{cases} \tag{2}$$

That is to say, we formalize the ranking problem as a new 'classification' problem on permutations. If the permutation of the predicted result is the same as the ground truth, then we have zero loss; otherwise we have one loss. In real ranking applications, the loss can be cost-sensitive, i.e., depending on the positions of the incorrectly ranked objects. We will leave this as our future work and focus on the $0-1$ loss in this paper first. Actually, in the literature of classification, people also studied the $0-1$ loss first, before they eventually moved onto the cost-sensitive case.

It is easy to see that the optimal ranking function which can minimize the expected loss $R(\mathbf{h}_B) = \inf R(\mathbf{h})$ is given by the Bayes rule,

$$\mathbf{h}_B(\mathbf{x}) = \arg\max_{\mathbf{y} \in Y} P(\mathbf{y}|\mathbf{x}), \tag{3}$$

Since P_{XY} is unknown, formula (1) cannot be directly solved and thus $\mathbf{h}_B(\mathbf{x})$ cannot be easily obtained. In practice, we are given independently and identically distributed (i.i.d) samples $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m \sim P_{XY}$, we instead try to obtain a ranking function $\mathbf{h} \in H$ that minimizes the empirical loss.

$$R_S(\mathbf{h}) = \frac{1}{m} \sum_{i=1}^{m} l(\mathbf{h}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}). \tag{4}$$

Note that for efficiency consideration, in practice the ranking function usually works on individual objects. It assigns a score to each object (by employing a scoring function g), sorts the objects in descending order of the scores, and finally creates the ranked list. That is to say, $\mathbf{h}(\mathbf{x}^{(i)})$ is decomposable with respect to objects. It is defined as

$$\mathbf{h}(\mathbf{x}^{(i)}) = sort(g(x_1^{(i)}), \ldots, g(x_{n_i}^{(i)})). \tag{5}$$

where $x_j^{(i)} \in \mathbf{x}^{(i)}$, n_i denotes the number of objects in $\mathbf{x}^{(i)}$, $g(\cdot)$ denotes the scoring function, and $sort(\cdot)$ denotes the sorting function. As a result, (4) becomes:

$$R_S(\mathbf{g}) = \frac{1}{m} \sum_{i=1}^{m} l(sort(g(x_1^{(i)}), \ldots, g(x_{n_i}^{(i)})), \mathbf{y}^{(i)}). \tag{6}$$

Due to the nature of the sorting function and the $0-1$ loss function, the empirical loss in (6) is inherently non-differentiable with respect to g, which poses a challenge to the optimization of it. To tackle this problem, we can introduce a surrogate loss as an approximation of (6), following a common practice in machine learning.

$$R_S^{\phi}(\mathbf{g}) = \frac{1}{m} \sum_{i=1}^{m} \phi(\mathbf{g}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}), \tag{7}$$

where ϕ is a surrogate loss function and $\mathbf{g}(\mathbf{x}^{(i)}) = (g(x_1^{(i)}), \ldots, g(x_{n_i}^{(i)}))$. For convenience in notation, in the following sections, we sometimes write $\phi_{\mathbf{y}}(\mathbf{g})$ for $\phi(\mathbf{g}(\mathbf{x}), \mathbf{y})$ and use bold symbols such as $\mathbf{g}$ to denote vectors since for a given $\mathbf{x}$, $\mathbf{g}(\mathbf{x})$ becomes a vector.

4. Theoretical Analysis

4.1. Properties of Loss Function

We analyze the listwise approach from the viewpoint of surrogate loss function. Specifically, we look at the following properties [3] of it: (a) consistency , (b) soundness, (c) continuity, differentiability, and convexity, and (d) computational efficiency in learning.

Consistency is about whether the obtained ranking function can converge to the optimal one through the minimization of the empirical surrogate loss (7), when the training sample size goes to infinity. It is a necessary condition for a surrogate loss function to be a good one for a learning algorithm (cf., Zhang (2004)).

Soundness is about whether the loss function can indeed represent loss in ranking. For example, an incorrect ranking should receive a larger penalty than a correct ranking, and the penalty should reflect the confidence of the ranking. This property is particularly important when the size of training data is small, because it can directly affect the training results.

4.2. Consistency

We conduct analysis on learning to rank algorithms from the viewpoint of consistency. As far as we know,

[3] In addition, convergence rate is another issue to consider. We leave it as future work.

this is the first work discussing the consistency issue for ranking.

In the large sample limit, minimizing the empirical surrogate loss (7) amounts to minimizing the following expected surrogate loss

$$R^{\phi}(\mathbf{g}) = E_{X,Y}\{\phi_{\mathbf{y}}(\mathbf{g}(\mathbf{x}))\} = E_X\{Q(\mathbf{g}(\mathbf{x}))\} \quad (8)$$
$$\text{where } Q(\mathbf{g}(\mathbf{x})) = \sum_{\mathbf{y}\in Y} P(\mathbf{y}|\mathbf{x})\phi_{\mathbf{y}}(\mathbf{g}(\mathbf{x})).$$

Here we assume $\mathbf{g}(\mathbf{x})$ is chosen from a vector Borel measurable function set, whose elements can take any value from $\Omega \subset R^n$.

When the minimization of (8) can lead to the minimization of the expected $0-1$ loss (1), we say the surrogate loss function is consistent. A equivalent definition can be found in Definition 2. Actually this equivalence relationship has been discussed in related work on the consistency of classification (Zhang, 2004).

Definition 1. *We define $\Lambda_{\mathbf{y}}$ as the space of all possible probabilities on the permutation space Y, i.e., $\Lambda_Y \triangleq \{\mathbf{p} \in R^{|Y|} : \sum_{\mathbf{y}\in Y} p_{\mathbf{y}} = 1, p_{\mathbf{y}} \geq 0\}$.*

Definition 2. *The loss $\phi_{\mathbf{y}}(\mathbf{g})$ is consistent on a set $\Omega \subset R^n$ with respect to the ranking loss (1), if the following conditions hold: $\forall \mathbf{p} \in \Lambda_Y$, assume $\mathbf{y}^* = \arg max_{\mathbf{y}\in Y} p_{\mathbf{y}}$ and $Y^c_{\mathbf{y}^*}$ denotes the space of permutations after removing $\mathbf{y}^*$, we have*

$$\inf_{\mathbf{g}\in\Omega} Q(\mathbf{g}) < \inf_{\mathbf{g}\in\Omega, sort(\mathbf{g})\in Y^c_{\mathbf{y}^*}} Q(\mathbf{g})$$

We next give sufficient conditions of consistency in ranking.

Definition 3. *A permutation probability space Λ_Y is order preserving with respect to object i and j, if the following conditions hold: $\forall \mathbf{y} \in Y_{i,j} \triangleq \{\mathbf{y} \in Y : y^{-1}(i) < y^{-1}(j)\}$ where $y^{-1}(i)$ denotes the position for object i in $\mathbf{y}$, denote $\sigma^{-1}\mathbf{y}$ as the permutation which exchanges the positions of object i and j while hold others unchanged for $\mathbf{y}$, we have $p_{\mathbf{y}} > p_{\sigma^{-1}\mathbf{y}}$.*

Definition 4. *The loss $\phi_{\mathbf{y}}(\mathbf{g})$ is order sensitive on a set $\Omega \subset R^n$, if $\phi_{\mathbf{y}}(\mathbf{g})$ is a non-negative differentiable function and the following two conditions hold:*

1. *$\forall \mathbf{y} \in Y$, $\forall i < j$, denote $\sigma\mathbf{y}$ as the permutation which exchanges the object on position i and that on position j while holds others unchanged for $\mathbf{y}$, if $g_{y(i)} < g_{y(j)}$, then $\phi_{\mathbf{y}}(\mathbf{g}) \geq \phi_{\sigma\mathbf{y}}(\mathbf{g})$ and with at least one $\mathbf{y}$, the strict inequality holds.*
2. *If $g_i = g_j$, then either $\forall \mathbf{y} \in Y_{i,j}$, $\frac{\partial\phi_{\mathbf{y}}(\mathbf{g})}{\partial g_i} \leq \frac{\partial\phi_{\mathbf{y}}(\mathbf{g})}{\partial g_j}$, or $\forall \mathbf{y} \in Y_{i,j}$, $\frac{\partial\phi_{\mathbf{y}}(\mathbf{g})}{\partial g_i} \geq \frac{\partial\phi_{\mathbf{y}}(\mathbf{g})}{\partial g_j}$, and with at least one $\mathbf{y}$, the strict inequality holds.*

Theorem 5. *Let $\phi_{\mathbf{y}}(\mathbf{g})$ be an order sensitive loss function on $\Omega \subset R^n$. $\forall n$ objects, if its permutation probability space is order preserving with respect to $n-1$ objective pairs $(j_1, j_2), (j_2, j_3), \cdots, (j_n - 1, j_n)$. Then the loss $\phi_{\mathbf{y}}(\mathbf{g})$ is consistent with respect to (1).*

Due to space limitations, we only give the proof sketch. First, we can show if the permutation probability space is order preserving with respect to $n-1$ objective pairs $(j_1, j_2), (j_2, j_3), \cdots, (j_n - 1, j_n)$, then the permutation with the maximum probability is $\mathbf{y}^* = (j_1, j_2, \cdots, j_n)$. Second, for an order sensitive loss function, for any order preserving object pairs (j_1, j_2), the vector $\mathbf{g}$ which minimizes $Q(\mathbf{g})$ in (8) should assign a larger score to j_1 than to j_2. This can be proven by the change of loss due to exchanging the scores of j_1 and j_2. Given all these results and Definition 2, we can prove Theorem 5 by means of contradiction.

Theorem 5 gives sufficient conditions for a surrogate loss function to be consistent: the permutation probability space should be order preserving and the function should be order sensitive. Actually, the assumption of order preserving has already been made when we use the scoring function and sorting function for ranking. The property of order preserving has also been explicitly or implicitly used in previous work, such as Cossock and Zhang (2006). The property of order sensitive shows that starting with a ground truth permutation, the loss will increase if we exchange the positions of two objects in it, and the speed of increase in loss is sensitive to the positions of objects.

4.3. Case Studies

We look at the four properties of three loss functions.

4.3.1. Likelihood Loss

We introduce a new loss function for listwise approach, which we call likelihood loss. The likelihood loss function is defined as:

$$\phi(\mathbf{g}(\mathbf{x}), \mathbf{y}) = -\log P(\mathbf{y}|\mathbf{x}; \mathbf{g}) \quad (9)$$
$$\text{where } P(\mathbf{y}|\mathbf{x}; \mathbf{g}) = \prod_{i=1}^{n} \frac{\exp(g(x_{y(i)}))}{\sum_{k=i}^{n} \exp(g(x_{y(k)}))}.$$

Note that we actually define a parameterized exponential probability distribution over all the permutations given the predicted result (by the ranking function), and define the loss function as the negative log likelihood of the ground truth list. The probability distribution turns out to be a Plackett-Luce model (Marden, 1995).

The likelihood loss function has the nice properties as below.

First, the likelihood loss is consistent. The following proposition shows that the likelihood loss is order sensitive. Therefore, according to Theorem 5, it is consistent. Due to the space limitations, we omit the proof.

Proposition 6. *The likelihood loss (9) is order sensitive on* $\Omega \subset R^n$.

Second, the likelihood loss function is sound. For simplicity, suppose that there are two objects to be ranked (similar argument can be made when there are more objects). The two objects receive scores of g_1 and g_2 from a ranking function. Figure 1(a) shows the scores, and the point $\mathbf{g} = (g_1, g_2)$. Suppose that the first object is ranked below the second object in the ground truth. Then the upper left area above line $g_2 = g_1$ corresponds to correct ranking; and the lower right area incorrect ranking. According to the definition of likelihood loss, all the points on the line $g_2 = g_1 + d$ has the same loss. Therefore, we say the likelihood loss only depends on d. Figure 1(b) shows the relation between the loss function and d. We can see the loss function decreases monotonously as d increases. It penalizes negative values of d more heavily than positive ones. This will make the learning algorithm focus more on avoiding incorrect rankings. In this regard, the loss function is a good approximation of the $0-1$ loss.

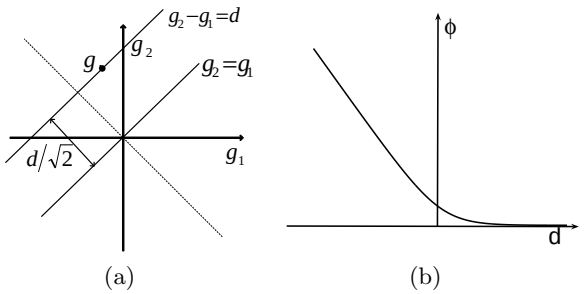


Figure 1. (a) Ranking scores of predicted result; (b) Loss ϕ v.s. d for the likelihood loss.

Third, it is easy to verify that the likelihood loss is continuous, differentiable, and convex (Boyd & Vandenberghe, 2004). Furthermore, the loss can be computed efficiently, with time complexity of linear order to the number of objects.

With the above good properties, a learning algorithm which optimizes the likelihood loss will become powerful for creating a ranking function.

4.3.2. Cosine Loss

The cosine loss is the loss function used in RankCosine (Qin et al., 2007), a listwise method. It is defined on the basis of the cosine similarity between the score vector of the ground truth and that of the predicted result.

$$\phi(\mathbf{g}(\mathbf{x}), \mathbf{y}) = \frac{1}{2}(1 - \frac{\psi_{\mathbf{y}}(\mathbf{x})^T \mathbf{g}(\mathbf{x})}{\|\psi_{\mathbf{y}}(\mathbf{x})\| \|\mathbf{g}(\mathbf{x})\|}). \tag{10}$$

The score vector of the ground truth is produced by a mapping $\psi_{\mathbf{y}}(\cdot) : R^d \to R$, which retains the order in a permutation, i.e, $\psi_{\mathbf{y}}(x_{y(1)}) > \cdots > \psi_{\mathbf{y}}(x_{y(n)})$.

First, we can prove that the cosine loss is consistent, given the following proposition. Due to space limitations, we omit the proof.

Proposition 7. *The cosine loss (10) is order sensitive on* $\Omega \subset R^n$.

Second, the cosine loss is not very sound. Let us again consider the case of ranking two objects. Figure 2(a) shows point $\mathbf{g} = (g_1, g_2)$ representing the scores of the predicted result and point $\mathbf{g}_\psi$ representing the ground truth (which depends on the mapping function ψ). We denote the angle from point $\mathbf{g}$ to line $g_2 = g_1$ as α, and the angle from $\mathbf{g}_\psi$ to line $g_2 = g_1$ as $\alpha_{\mathbf{g}_\psi}$. We investigate the relation between the loss and the angle α. Figure 2(b) shows the cosine loss as a function of α. From this figure, we can see that the cosine loss is not a monotonously decreasing function of α. When $\alpha > \alpha_{\mathbf{g}_\psi}$, it increases quickly, which means that it can heavily penalize correct rankings. Furthermore, the mapping function and thus $\alpha_{\mathbf{g}_\psi}$ can also affect the loss function. Specifically, the curve of the loss function can shift from left to right with different values of $\alpha_{\mathbf{g}_\psi}$. Only when $\alpha_{\mathbf{g}_\psi} = \pi/2$, it becomes a relatively satisfactory representation of loss for the learning problem.

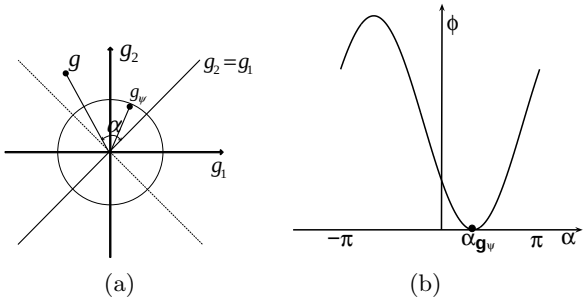


Figure 2. (a) Ranking scores of predicted result and ground truth; (b) Loss ϕ v.s. angle α for the cosine loss.

Third, it is easy to see that the cosine loss is continuous, differentiable, but not convex. It can also be computed in an efficient manner with a time complexity linear to the number of objects.

4.3.3. Cross Entropy Loss

The cross entropy loss is the loss function used in ListNet (Cao et al., 2007), another listwise method. The cross entropy loss function is defined as:

$$\phi(\mathbf{g}(\mathbf{x}), \mathbf{y}) = D(P(\pi|\mathbf{x}; \psi_{\mathbf{y}})||P(\pi|\mathbf{x}; \mathbf{g})) \quad (11)$$

$$\text{where} \quad P(\pi|\mathbf{x}; \psi_{\mathbf{y}}) = \prod_{i=1}^{n} \frac{\exp(\psi_{\mathbf{y}}(x_{\pi(i)}))}{\sum_{k=i}^{n} \exp(\psi_{\mathbf{y}}(x_{\pi(k)}))}$$

$$P(\pi|\mathbf{x}; \mathbf{g}) = \prod_{i=1}^{n} \frac{\exp(g(x_{\pi(i)}))}{\sum_{k=i}^{n} \exp(g(x_{\pi(k)}))}$$

where ψ is a mapping function whose definition is similar to that in RankCosine.

First, we can prove that the cross entropy loss is consistent, given the following proposition. Due to space limitations, we omit the proof.

Proposition 8. *The cross entropy loss (11) is order sensitive on $\Omega \subset R^n$.*

Second, the cross entropy loss is not very sound. Again, we look at the case of ranking two objects. $\mathbf{g} = (g_1, g_2)$ denotes the ranking scores of the predicted result. $\mathbf{g}_\psi$ denotes the ranking scores of the ground truth (depending on the mapping function). Similar to the discussions in the likelihood loss, the cross entropy loss only depends on the quantity d. Figure 3(a) illustrates the relation between $\mathbf{g}$, $\mathbf{g}_\psi$, and d. Figure 3(b) shows the cross entropy loss as a function of d. As can be seen that the loss function achieves its minimum at point $d_{\mathbf{g}_\psi}$, and then increases as d increases. That means it can heavily penalize those correct rankings with higher confidence. Note that the mapping function also affects the penalization. According to mapping functions, the penalization on correct rankings can be even larger than that on incorrect rankings.

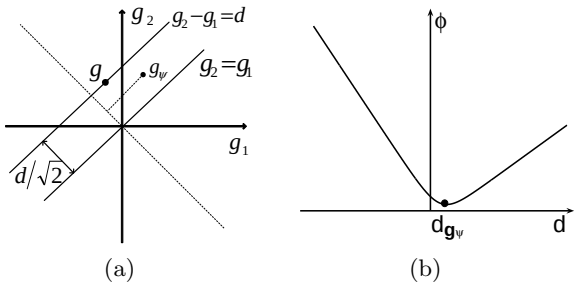


Figure 3. (a) Ranking scores of predicted result and ground truth; (b) Loss ϕ v.s. d for the cross entropy loss.

Third, it is easy to see that the cross entropy loss is continuous and differentiable. It is also convex because the log of a convex function is still convex, and the set of convex function is closed under addition (Boyd & Vandenberghe, 2004). However, it cannot be computed in an efficient manner. The time complexity is of exponential order to the number of objects.

Algorithm 1 ListMLE Algorithm

Input: training data$\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$
Parameter: learning rate η, tolerance rate ϵ
Initialize parameter ω
repeat
  for $i = 1$ **to** m **do**
    Input $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ to Neural Network and compute gradient $\triangle\omega$ with current ω
    Update $\omega = \omega - \eta \times \triangle\omega$
  end for
  calculate likelihood loss on the training set
until change of likelihood loss is below ϵ
Output: Neural Network model ω

Table 1 gives a summary of the properties of the loss functions. All the three loss functions as aforementioned are consistent, as well as continuous and differentiable. The likelihood loss is better than the cosine loss in terms of convexity and soundness, and is better than the cross entropy loss in terms of time complexity and soundness.

5. ListMLE

We propose a novel listwise method referred to as ListMLE. In learning of ListMLE, we employ the likelihood loss as the surrogate loss function, since it is proven to have all the nice properties as a surrogate loss. On the training data, we actually maximize the sum of the likelihood function with respect to all the training queries.

$$\sum_{i=1}^{m} \log P(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}; \mathbf{g}). \quad (12)$$

We choose Stochastic Gradient Descent (SGD) as the algorithm for conducting the minimization. As ranking model, we choose linear Neural Network (parameterized by ω). Algorithm 1 shows the learning algorithm based on SGD.

6. Experimental Results

We conducted two experiments to verify the correctness of the theoretical findings. One data set is synthetic data, and the other is the LETOR benchmark data for learning to rank (Liu et al., 2007).

Table 1. Comparison between different surrogate losses.

Loss	Consistency	Soundness	Continuity	Differentiability	Convexity	Complexity
Likelihood	√	√	√	√	√	$O(n)$
Cosine	√	×	√	√	×	$O(n)$
Cross entropy	√	×	√	√	√	$O(n! \cdot n)$

6.1. Experiment on Synthetic Data

We conducted an experiment using a synthetic data set. We created the data as follows. First, we randomly sample a point according to the uniform distribution on the square area $[0, 1] \times [0, 1]$. Then we assign to the point a score using the following rule, $y = x_1 + 10x_2 + \epsilon$ where ϵ denotes a random variable normally distributed with mean of zero and standard deviation of 0.005. In total, we generate 15 points and their scores in this way, and create a permutation on the points based on their scores, which forms an instance of ranking. We repeat the process and make 100 training instances, 100 validation instances, and 100 testing instances. We applied RankCosine, ListNet[4], and ListMLE to the data.

We tried different score mapping functions for RankCosine and ListNet, and used five most representative ones, i.e., $log(15-r)$, $\sqrt{15-r}$, $15-r$, $(15-r)^2$ and $\exp(15-r)$, where r denotes the positions of objects. We denote the mapping functions as *log*, *sqrt*, *l*, *q*, and *exp* for simplicity. The experiments were repeated 20 times with different initial values of parameters in the Neural Network model. Table 2 shows the means and standard deviations of the accuracies and Mean Average Precision (MAP)(Baeza-Yates & Ribeiro-Neto, 1999) of the three algorithms. The accuracy measures the proportion of correctly ranked instances and MAP[5] is a commonly used measure in IR.

As shown in the table, ListMLE achieves the best performance among all the algorithms in terms of both accuracy and MAP, owing to good properties of its loss function. The accuracies of RankCosine and ListNet vary according to the mapping functions. Especially, RankCosine achieves an accuracy of only 0.047 when using the mapping function *exp* while 0.917 when using the mapping function *l*. This result indicates that the performances of the cosine loss and the cross entropy loss depend on the mapping functions, while finding a suitable mapping function is not easy. Furthermore, RankCosine has a larger variance than ListMLE and ListNet. The likely explanation is that RankCosine's performance is sensitive to the initial values of parameters due to the non-convexity of its loss function.

[4]The top-1 version of the cross entropy loss was employed as in the original work (Cao et al., 2007).

[5]When calculating MAP, we treated the top-1 items as relevant and the other as irrelevant.

Table 2. The performance of three algorithms on the synthetic data set.

Algorithm	Accuracy	MAP
ListMLE	0.92 ± 0.011	0.999 ± 0.002
ListNet-log	0.905 ± 0.010	0.999 ± 0.002
ListNet-sqrt	0.917 ± 0.009	0.999 ± 0.002
ListNet-l	0.767 ± 0.021	0.995 ± 0.003
ListNet-q	0.868 ± 0.028	0.999 ± 0.002
ListNet-exp	0.832 ± 0.074	0.997 ± 0.004
RankCosine-log	0.180 ± 0.217	0.948 ± 0.034
RankCosine-sqrt	0.080 ± 0.159	0.886 ± 0.056
RankCosine-l	0.917 ± 0.112	0.999 ± 0.002
RankCosine-q	0.102 ± 0.161	0.890 ± 0.060
RankCosine-exp	0.047 ± 0.163	0.746 ± 0.136

6.2. Experiment on OHSUMED Data

We also conducted an experiment on OHSUMED, a benchmark data set for learning to rank provided in LETOR. There are in total 106 queries, and 16,140 query-document pairs upon which relevance judgments are made. The relevance judgments are either definitely relevant, possibly relevant, or not relevant. The data was in the form of feature vector and relevance label. There are in total 25 features. We used the data split provided in LETOR to conduct five-fold cross validation experiments. In evaluation, besides MAP, we adopted another measures commonly used in IR: Normalized Discounted Cumulative Gain (NDCG) (Jarvelin & Kekanainen, 2000).

Note that here the ground truth in the data is given as partial ranking, while the methods need to use total ranking (permutation) in training. To bridge the gap, for RankCosine and ListNet, we adopted the methods proposed in the papers (Cao et al., 2007) (Qin et al., 2007). For ListMLE we randomly selected one perfect permutation for each query from among the possible perfect permutations based on the ground truth.

We applied RankCosine, ListNet, and ListMLE to the data. The results reported below are those averaged over five trials. As shown in Figure 4, ListMLE achieves the best performance among all the algorithms. Especially, on NDCG@1, it has more than

5-point gains over RankCosine which is at the second place. We also conducted the t-test on the improvements of ListMLE over the other two algorithms. The results show that the improvements are statistically significant for NDCG@5, NDCG@7, NDCG@8, NDCG@9, and NDCG@10 (p-value < 0.05).

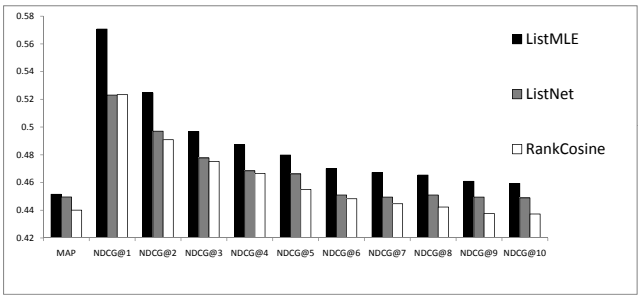


Figure 4. Ranking performance on OHSUMED data.

7. Conclusion

In this paper, we have investigated the theory and algorithms of the listwise approach to learning to rank. We have pointed out that to understand the effectiveness of a learning to rank algorithm, it is necessary to conduct theoretical analysis on its loss function. We propose investigating a loss function from the viewpoints of (a) consistency, (b) soundness, (c) continuity, differentiability, convexity, and (d) efficiency. We have obtained some theoretical results on consistency of ranking. We have conducted analysis on the likelihood loss, cosine loss, and cross entropy loss. The result indicates that the likelihood loss has better properties than the other two losses. We have then developed a new learning algorithm using the likelihood loss, called ListMLE and demonstrated its effectiveness through experiments.

There are several directions which we can further explore. (1) We want to conduct more theoretical analysis on the properties of loss functions, for example, weaker conditions for consistency and the rates of convergence. (2) We plan to study the case where cost-sensitive loss function is used instead of the $0-1$ loss function in defining the expected loss. (3) We plan to investigate other surrogate loss functions with the tools we have developed in this paper.

References

Baeza-Yates, R., & Ribeiro-Neto, B. (Eds.). (1999). *Modern information retrieval*. Addison Wesley.

Bartlett, P. L., Jordan, M. I., & McAuliffe, J. D. (2003). *Convexity, classification, and risk bounds* (Technical Report 638). Statistics Department, University of California, Berkeley.

Boyd, S., & Vandenberghe, L. (Eds.). (2004). *Convex optimization*. Cambridge University.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *Proceedings of ICML 2005* (pp. 89–96).

Cao, Z., Qin, T., Liu, T. Y., Tsai, M. F., & Li, H. (2007). Learning to rank: From pairwise approach to listwise approach. *Proceedings of the 24th International Conference on Machine Learning* (pp. 129–136). Corvallis, OR.

Cossock, D., & Zhang, T. (2006). Subset ranking using regression. *COLT* (pp. 605–619).

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. *Proceedings of ICML* (pp. 170–178).

Hastie, T., Tibshirani, R., & Friedman, J. H. (Eds.). (2001). *The elements of statistical learning: Data mining, inference and prediction*. Springer.

Herbrich, R., Graepel, T., & Obermayer, K. (1999). Support vector vector learning for ordinal regression. *Proceedings of ICANN* (pp. 97–102).

Jarvelin, K., & Kekanainen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. *Proceedings of SIGIR* (pp. 41–48).

Lin, Y. (2002). Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery*, 259–275.

Liu, T. Y., Qin, T., Xu, J., Xiong, W. Y., & Li, H. (2007). Letor: Benchmark dataset for research on learning to rank for information retrieval. *Proceedings of SIGIR*.

Marden, J. I. (Ed.). (1995). *Analyzing and modeling rank data*. London: Chapman and Hall.

Nallapati, R. (2004). Discriminative models for information retrieval. *Proceedings of SIGIR* (pp. 64–71).

Qin, T., Zhang, X.-D., Tsai, M.-F., Wang, D.-S., Liu, T.-Y., & Li, H. (2007). Query-level loss functions for information retrieval. *Information processing and management*.

Xu, J., & Li, H. (2007). Adarank: a boosting algorithm for information retrieval. *Proceedings of SIGIR* (pp. 391–398).

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimization average precision. *Proceedings of SIGIR* (pp. 271–278).

Zhang, T. (2004). Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research, 5*, 1225–1251.

Democratic Approximation of Lexicographic Preference Models

Fusun Yaman FUSUN@CS.UMBC.EDU
University of Maryland Baltimore County, Computer Science and Electrical Eng. Department, Baltimore, MD 21250 USA

Thomas J. Walsh THOMASWA@CS.RUTGERS.EDU
Michael L. Littman MLITTMAN@CS.RUTGERS.EDU
Rutgers University, Department of Computer Science, Piscataway, NJ 08854 USA

Marie desJardins MARIEDJ@CS.UMBC.EDU
University of Maryland Baltimore County, Computer Science and Electrical Eng. Department, Baltimore, MD 21250 USA

Abstract

Previous algorithms for learning lexicographic preference models (LPMs) produce a "best guess" LPM that is consistent with the observations. Our approach is more democratic: we do not commit to a single LPM. Instead, we approximate the target using the votes of a *collection* of consistent LPMs. We present two variations of this method—*variable voting* and *model voting*—and empirically show that these democratic algorithms outperform the existing methods. We also introduce an intuitive yet powerful learning bias to prune some of the possible LPMs. We demonstrate how this learning bias can be used with variable and model voting and show that the learning bias improves the learning curve significantly, especially when the number of observations is small.

1. Introduction

Lexicographic preference models (LPMs) are one of the simplest preference representations. An LPM defines an order of importance on the variables that describe the objects in a domain and uses this order to make preference decisions. For example, the meal preference of a vegetarian with a weak stomach could be represented by an LPM such that a vegetarian dish is always preferred over a non-vegetarian dish, and among vegetarian or non-vegetarian items, mild dishes are preferred to spicy ones. Previous work on learning LPMs from a set of preference observations has been limited to autocratic approaches: one of many possible LPMs is picked heuristically and used for future decisions. However, it is highly likely that autocratic methods will produce poor approximations of the target when there are few observations.

In this paper, we present a *democratic* approach to LPM learning, which does not commit to a single LPM. Instead, we approximate a target preference using the votes of a collection of consistent LPMs. We present two variations of this method: *variable voting* and *model voting*. Variable voting operates on the variable level and samples the consistent LPMs implicitly. The learning algorithm based on variable voting learns a partial order on the variables where all linearizations correspond to an LPM consistent with the observations. Model voting explicitly samples the consistent LPMs and employs weighted voting where the weights are computed using Bayesian priors. The additional complexity of voting-based algorithms is tolerable: both algorithms have low-order polynomial time complexity. Our experiments show that these democratic algorithms outperform more than half of the LPMs that can be produced by an autocratic algorithm, greatly increasing the chance of a positive outcome.

To further improve the performance of the learning algorithms when the number of observations is small, we introduce an intuitive yet powerful learning bias. The bias defines equivalence classes on the variables, indicating the most important set of variables, the second most important set, and so on. We demonstrate how this learning bias can be used with variable and model voting and show that the learning bias improves the learning curve significantly on appropriate problems, especially when the number of observations is small.

In the rest of the paper, we give some background on LPMs, then introduce our voting-based methods. We then introduce the learning bias and show how we can generalize the

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

voting methods to utilize such a bias. Finally, we present the results of our experiments, followed by related work and concluding remarks.

2. Lexicographic Decision Models

In this section, we briefly introduce the lexicographic preference model (LPM) and summarize previous results on learning LPMs. Before going into the definition of an LPM, we state that we only consider binary variables whose domain is $\{0, 1\}$.[1] Like others before us, we assume that the preferred value of each variable is known. Without loss of generality, we will assume that 1 is always preferred to 0.

Given a set of variables, $X = \{X_1 \dots X_n\}$, an object A over X is a vector of the form $[x_1, \dots, x_n]$. We use the notation $A(X_i)$ to refer the value of X_i in the object A. A *lexicographic preference model* $\mathcal{L}$ on X is a total order on a subset R of X. We denote this total order with $\sqsubset_{\mathcal{L}}$. Any variable in R is *relevant* with respect to $\mathcal{L}$; similarly, any variable in $X - R$ is *irrelevant* with respect to $\mathcal{L}$. If A and B are two objects, then the preferred object given $\mathcal{L}$ is determined as follows:

- Find the smallest variable X^* in $\sqsubset_{\mathcal{L}}$ such that X^* has different values in A and B. The object that has the value 1 for X^* is the most preferred.
- If all relevant variables in $\mathcal{L}$ have the same value in A and B, then the objects are equally preferred (a tie).

Example 1 *Suppose $X_1 < X_2 < X_3$ is the total order defined by an LPM $\mathcal{L}$, and consider objects $A = [1, 0, 1, 1]$, $B = [0, 1, 0, 0]$, $C = [0, 0, 1, 1]$ and $D = [0, 0, 1, 0]$. A is preferred over B because $A(X_1) = 1$, and X_1 is the most important variable in $\mathcal{L}$. B is preferred over C because $B(X_2) = 1$ and both objects have the same value for X_1. Finally, C and D are equally preferred because they have the same values for the relevant variables.*

An *observation* $o = (A, B)$ is an ordered pair of objects, connoting that A is preferred to B. In many practical applications, however, preference observations are gathered from demonstration of an expert who breaks ties arbitrarily. Thus, for some observations, A and B may actually be tied. An LPM $\mathcal{L}$ is *consistent* with an observation (A, B) iff $\mathcal{L}$ implies that A is preferred to B or that A and B are equally preferred.

The problem of learning an LPM is defined as follows. Given a set of observations, find an LPM $\mathcal{L}$ that is consistent with the observations. Previous work on learning LPMs was limited to the case where all variables are relevant. This assumption entails that, in every observation (A, B), A is strictly preferred to B, since ties can only happen when there are irrelevant attributes.

[1] The representation can easily be generalized to monotonic preferences with ordinal variables such that 1 corresponds to a preference on the increasing order and 0 on decreasing order.

Algorithm 1 *greedyPermutation*

Require: A set of variables X and a set of observations O.
Ensure: An LPM that is consistent with O, if one exists.
1: **for** $i = 1, \dots, n$ **do**
2: Arbitrarily pick one of $X_j \in X$ such that *MISS*$(X_j, O) = \min_{X_k \in X}$ *MISS*(X_k, O)
3: $\pi(X_j) := i$, assign the rank i to X_j
4: Remove X_j from X
5: Remove all observations (A, B) from O such that $A(X_j) \neq B(X_j)$
6: Return the total order $\sqsubset$ on X such that $X_i < X_j$ iff $\pi(X_i) < \pi(X_j)$

Schmitt and Martignon (2006) proposed a greedy algorithm that is guaranteed to find one of the LPMs that is consistent with the observations if one exists. They have also shown that for the noisy data case, finding an LPM that does not violate more than a constant number of the observations is NP-complete. Algorithm 1 is Schmitt and Martignon's greedy variable-permutation algorithm, which we use as a performance baseline. The algorithm refers to a function *MISS*(X_i, O), which is defined as $|\{(A, B) \in O : A(X_i) < B(X_i)\}|$; that is, the number of observations violated in O if the most important variable is selected as X_i. Basically, the algorithm greedily constructs a total order by choosing the variable at each step that causes the minimum number of inconsistencies with the observations. If multiple variables have the same minimum, then one of them is chosen arbitrarily. The algorithm runs in polynomial time, specifically $O(n^2 m)$, where n is the number of variables and m is the number of observations.

Dombi et al. (2007) have shown that if there are n variables, all of which are relevant, then $O(n \log n)$ queries to an oracle suffice to learn an LPM. Furthermore, it is possible to learn any LPM with $O(n^2)$ observations if all pairs differ in only two variables. They proposed an algorithm that can find the unique LPM induced by the observations. In case of noise due to irrelevant attributes the algorithm does not return an answer.

In this paper, we investigate the following problem: Given a set of observations with no noise, but possibly with arbitrarily broken ties, find a rule for predicting preferences that agrees with the target LPM that produced the observations.

3. Voting Algorithms

We propose a democratic approach for approximating the target LPM that produced a set of observations. Instead of finding just one of the consistent LPMs, it reasons with a collection of LPMs that are consistent with the observations. Given two objects, such an approach prefers the one

that a majority of its models prefer. A naive implementation of a voting algorithm would enumerate all LPMs that are consistent with a set of observations. However, since the number of models consistent with a set of observations can be exponential, the naive implementation is infeasible.

In this section, we describe two methods—*variable voting* and *model voting*—that sample the set of consistent LPMs and use voting to predict the preferred object. Unlike existing algorithms that learn LPMs, these methods do not require all variables to be relevant or observations to be tie-free. The following subsections explain the variable voting and model voting methods and summarize some of our theoretical results.

3.1. Variable Voting

Variable voting uses a generalization of the LPM representation. Instead of a total order on the variables, variable voting reasons with a *partial* order ($\preceq$) to find the preferred object in a given pair. Among the variables that are different in both objects, the ones that have the smallest rank (and are hence the most salient) in the partial order vote to choose the preferred object. The object that has the most "1" values for the voting variables is declared to be the preferred one. If the votes are equal, then the objects are equally preferred.

Definition 1 (Variable Voting) *Suppose X is a set of variables and $\preceq$ is a partial order on X. Given two objects, A and B, the variable voting process with respect to $\preceq$ for determining which of the two objects is preferred is:*

- *Define D, the set of variables that differ in A and B.*
- *Define D^*, the set of variables in D that have the smallest rank among D with respect to $\preceq$.*
- *Define N_A as the number of variables in D^* that favor A (i.e., that have value 1 in A and 0 in B) and N_B, as the number of variables in D^* that favor B.*
- *If $N_A > N_B$, then A is preferred. If $N_A < N_B$, then B is preferred. Otherwise, they are equally preferred.*

Example 2 *Suppose $\preceq$ is the partial order $\{X_2, X_3\} < \{X_1\} < \{X_4, X_5\}$. Consider objects $A = [0, 1, 1, 0, 0]$ and $B = [0, 0, 1, 0, 1]$. D is $\{X_2, X_5\}$. D^* is $\{X_2\}$ because X_2 is the smallest ranking variable in D with respect to $\preceq$. X_2 favors A because $A(X_2) = 1$. Thus, variable voting with $\preceq$ prefers A over B.*

Algorithm 2 presents the algorithm *learnVariableRank*, which learns a partial order $\preceq$ on the variables from a set of observations such that variable voting with respect to $\preceq$ will correctly predict the preferred objects in the observations. Specifically, it finds partial orders that define equivalence classes on the set of variables. The algorithm maintains the minimum possible rank for every variable that does not violate an observation with respect to variable voting. Initially, all variables are considered equally important (rank of 1). The algorithm loops over the set of observations until the ranks converge. At every iteration and for every pair, variable voting predicts a winner. If it is correct, then the ranks stay the same. Otherwise, the ranks of the variables that voted for the wrong object are incremented, thus reducing their importance [2]. Finally, the algorithm builds a partial order $\preceq$ based on the ranks such that $x \preceq y$ if and only if x has a lower rank than y.

Algorithm 2 *learnVariableRank*

Require: A set of X of variables, and a set O of observations
Ensure: A partial order on X.
1: $\Pi(x) = 1, \forall\, x \in X$
2: **while** Π can change **do**
3: **for** Every observation $(A, B) \in O$ **do**
4: Let D be the variables that differ in A and B
5: $D^* = \{x \in D | \forall y \in D, \Pi(x) \leq \Pi(y)\}$
6: V_A is the set of variables in D^* that are 1 in A.
7: V_B is the set of variables in D^* that are 1 in B.
8: **if** $|V_B| \geq |V_A|$ **then**
9: **for** $x \in V_B$ such that $\Pi(x) < |X|$ **do**
10: $\Pi(x) = \Pi(x) + 1$;
11: Return partial order $\preceq$ on X such that $x \preceq y$ iff $\Pi(x) < \Pi(y)$.

Table 1. The rank of the variables after each iteration of the for-loop in line 3 of the algorithm *learnVariableRank*.

Observations	X_1	X_2	X_3	X_4	X_5
$Initially$	1	1	1	1	1
$[0, 1, 1, 0, 0], [1, 1, 0, 1, 1]$	2	1	1	2	2
$[0, 1, 1, 0, 1], [1, 0, 0, 1, 0]$	2	1	1	2	2
$[1, 0, 1, 0, 0], [0, 0, 1, 1, 1]$	2	1	1	3	3

Example 3 *Suppose $X = \{X_1, X_2, X_3, X_4, X_5\}$ and O consists of $([0, 1, 1, 0, 0], [1, 1, 0, 1, 1])$, $([0, 1, 1, 0, 1], [1, 0, 0, 1, 0])$ and $([1, 0, 1, 0, 0], [0, 0, 1, 1, 1])$. Table 1 illustrates the ranks of every variable in X after each iteration of the for-loop in line 3 of the algorithm learnVariableRank. The ranks of the variables stay the same during the second iteration of the while-loop, thus, the loop terminates. The partial order $\preceq$ based on ranks of the variables is the same as the order given in Example 2.*

We next summarize our theoretical results about the algorithm *learnVariableRank*.

Correctness: Suppose $\preceq$ is a partial order returned by $learnVariableRank(X, O)$. It can be shown that any LPM $\mathcal{L}$ such that $\sqsubset_{\mathcal{L}}$ is a topological sort of $\preceq$ is consistent with

[2] In our empirical results, we also update the ranks when the prediction was correct but not unanimous. This produces a heuristic speed-up without detracting from the worst case guarantees.

O. Furthermore, $learnVariableRank$ never increments the ranks of the relevant variables beyond their actual rank in the target LPM. The ranks of the irrelevant variables can be incremented as far as the number of variables.

Convergence: *learnVariableRank* has a mistake-bound of $O(n^2)$, where n is the number of variables, because each mistake increases the sum of the potential ranks by at least 1 and the sum of the ranks the target LPM induces is $O(n^2)$. This bound guarantees that given enough observations (as described in the background section), *learnVariableRank* will converge to a partial order $\preceq$ such that every topological sort of $\preceq$ has the same prefix as the total order induced by the target LPM. If all variables are relevant, then $\preceq$ will converge to the total order induced by the target LPM.

Complexity: A very loose upper bound on the time complexity of *learnVariableRank* is $O(n^3m)$, where n is the number of variables and m is the number of observations. This bound holds because the while-loop on line 2 runs at most $O(n^2)$ times and the for-loop in line 3, runs for m observations. The time complexity of one iteration of the for-loop is $O(n)$; therefore, the overall complexity is $O(n^3m)$. We leave the investigation of tighter bounds and the average case analysis for future work.

3.2. Model Voting

The second method we present employs a Bayesian approach. This method randomly generates a sample set, S, of distinct LPMs, that are consistent with the observations. When a pair of objects is presented, the preferred one is predicted using weighted voting. That is, each $\mathcal{L} \in S$ casts a vote for the object it prefers, and this vote is weighted according to its posterior probability $P(\mathcal{L}|S)$.

Definition 2 (Model Voting) *Let U be the set of all LPMs, O be a set of observations, and $S \subset U$ be a set of LPMs that are consistent with O. Given two objects A and B, model voting prefers A over B with respect to S if*

$$\sum_{\mathcal{L}\in U} P(\mathcal{L}|S)V^{\mathcal{L}}_{(A>B)} > \sum_{\mathcal{L}\in U} P(\mathcal{L}|S)V^{\mathcal{L}}_{(B>A)}, \quad (1)$$

where $V^{\mathcal{L}}_{(A>B)}$ is 1 if A is preferred with respect to $\mathcal{L}$, and 0 otherwise. $V^{\mathcal{L}}_{(B>A)}$ is defined analogously. $P(\mathcal{L}|S)$ is the posterior probability of $\mathcal{L}$ being the target LPM given S, calculated as discussed below.

We first assume that all LPMs are equally likely *a priori*. In this case, given a sample S of size k, the posterior probability of an LPM $\mathcal{L}$ will be $1/k$ if and only if $\mathcal{L} \in S$, and 0 otherwise. Note that if S is maximal this case degenerates into the naive voting algorithm. However, it is generally not feasible to have all consistent LPMs—in practice, the sample has to be small enough to be feasible and large enough to be representative.

Algorithm 3 *sampleModels*

Require: A set of variables X, a set of observations O, and rulePrefix, an LPM to be extended.
Ensure: An LPM (possibly aggregated) consistent with O.
1: *candidates* is the set of variables $\{Y : Y \notin rulePrefix \mid \forall(A,B) \in O, A(Y) = 1 \; or A(Y) = B(Y)\}$.
2: **while** *candidates* $\neq \emptyset$ **do**
3: **if** $O = \emptyset$ **then**
4: return (*rulePrefix*, $*$).
5: Randomly remove a variable Z from *candidates* .
6: Remove any observation (C, D) from O such that $C(Z) \neq D(Z)$.
7: Extend *rulePrefix*: *rulePrefix* $=$ (*rulePrefix*, Z).
8: Recompute *candidates*.
9: return *rulePrefix*

In constructing S, we exploit the fact that many consistent LPMs share prefixes in the total order that they define on the variables. We wish to discover and compactly represent such LPMs. To this end, we introduce the idea of *aggregated LPMs*. An aggregated LPM, $(X_1, X_2 \ldots, X_k, *)$, represents a set of LPMs that define a total order with the prefix $X_1 < X_2 < \ldots < X_k$. Intuitively, an aggregated LPM states that any possible completion of the prefix is consistent with the observations. The algorithm *sampleModels* in Algorithm 3 implements a "smart sampling" approach by constructing an LPM that is consistent with the given observations, returning an aggregated LPM when possible. We start with an arbitrary consistent LPM (such as the empty set, which is always consistent) and add more variable orderings extending the input LPM. We first identify the variables that can be used in extending the prefix—that is, all variables X_i such that in every observation, either X_i is 1 in the preferred object or is the same in both objects. We then select one of those variables randomly and extend the prefix. Finally, we remove the observations that are explained with this selection and continue with the rest of the observations. If at any point, no observations remain, then we return the aggregated form of the prefix, since every completion of the prefix will be consistent with the null observation. Running *sampleModels* several times and eliminating duplicates will produce a set of (possibly aggregated) LPMs.

Example 4 *Consider the same set of observations O as in Example 3. Then, the LPMs that are consistent with O are as follows: $()$, (X_2), (X_2, X_3), $(X_2, X_3, X_1, *)$, (X_3), $(X_3, X_1, *)$, (X_3, X_2) and $(X_3, X_2, X_1, *)$. To illustrate the set of LPMs that an aggregate LPM represents, consider $(X_2, X_3, X_1, *)$, which has a total of 5 extensions: (X_2, X_3, X_1), (X_2, X_3, X_1, X_4), (X_2, X_3, X_1, X_5), $(X_2, X_3, X_1, X_4, X_5)$, $(X_2, X_3, X_1, X_5, X_4)$. Every*

time the algorithm sampleModels runs, it will randomly generate one of the aggregated LPMs: $(X_2, X_3, X_1, *)$, $(X_3, X_1, *)$, *or* $(X_3, X_2, X_1, *)$. *Note that the shorter models that are not produced by sampleModels are all sub-prefixes of the aggregated LPMs and it is easy to modify sampleModels to return those models as well.*

An aggregate LPM in a sample saves us from enumerating all possible extensions of a prefix, but it also introduces complications in computing the weights (posteriors) of the LPMs, as well as their votes. For example, when comparing two objects A and B, some extensions of an aggregate LPM might vote for A and some for B. Thus, we need to find the total number of LPMs that an aggregate LPM represents and determine what proportion of them favor A over B (or vice versa), without enumerating all extensions. Suppose there are n variables and $\mathcal{L}$ is an aggregated LPM with a prefix of length k. Then, the number of extensions of $\mathcal{L}$ is denoted by $F_{\mathcal{L}}$ and is equal to f_{n-k}, where f_m is defined to be:

$$f_m = \sum_{i=0}^{m} \binom{m}{i} \times i! = \sum_{i=0}^{m} \frac{(m)!}{(m-i)!}. \qquad (2)$$

Intuitively, f_m counts every possible permutation with at most m items. Note that f_m can be computed efficiently and that the number of all possible LPMs when there are n variables is given by f_n.

Consider a pair of objects A and B. We wish to determine how many extensions of an aggregate LPM $\mathcal{L} = (X_1, X_2, \ldots, X_k, *)$ would vote for one of the objects. We will call the variables $X_1 \ldots X_k$ the *prefix variables*. If A and B have different values for at least one prefix variable, then all extensions will vote in accordance with the smallest such variable. Suppose all prefix variables are tied and m is the set of all non-prefix variables. Then, m is composed of three disjoint sets a, b, and w, such that a is the set of variables that favor A, b is the set of variables that favor B, and w is the set of variables that are neutral (that is, that have the same value in A and B).

An extension $\mathcal{L}'$ of $\mathcal{L}$ will produce a tie iff all variables in a and b are irrelevant in $\mathcal{L}'$. The number of such extensions is $f_{|w|}$. The number of extensions that favor A over B is directly proportional to $|a|/(|a| + |b|)$. The number of extensions of $\mathcal{L}$ that will vote for A over B is denoted by $N^{\mathcal{L}}_{A>B}$, which is given by:

$$N^{\mathcal{L}}_{A>B} = \frac{|a|}{|b| + |a|} \times (f_m - f_{|w|}). \qquad (3)$$

The number of extensions of $\mathcal{L}$ that will vote for B over A is computed similarly. Note that the computation of $N^{\mathcal{L}}_{A>B}$, $N^{\mathcal{L}}_{B>A}$, and $F_{\mathcal{L}}$ can be done in linear time by caching the recurrent values.

Table 2. The posterior probabilities and number of votes of all LPMs in Example 5.

LPMs	$P(L\|S_1)$	$P(L\|S_2)$	$N^{\mathcal{L}}_{A>B}$	$N^{\mathcal{L}}_{B>A}$
$()$	1/31	0	0	0
(X_2)	1/31	0	1	0
(X_2, X_3)	1/31	0	1	0
$(X_2, X_3, X_1, *)$	5/31	5/26	5	0
(X_3)	1/31	0	0	0
$(X_3, X_1, *)$	16/31	16/26	7	7
(X_3, X_2)	1/31	0	1	0
$(X_3, X_2, X_1, *)$	5/31	5/26	5	0

Algorithm 4 *modelVote*

Require: A set of LPMs, S, and two objects, A and B.
Ensure: Returns either one of A or B or tie.

```
Initialize sampleSize to the number of non-aggregated LPMs in S.
for every aggregated LPM L ∈ S do
    sampleSize+=F_L.
Vote(A) = 0; Vote(B) = 0;
for every LPM L ∈ S do
    if L is not an aggregate rule then
        winner is the object L prefers among A and B.
        Increment Vote(winner) by 1/sampleSize.
    else
        if A and B differ in at least one prefix variable of L then
            L* is an extension of L referring only the prefix.
            winner is the object L* prefers among A and B
            Vote(winner) += F_L/sampleSize.
        else
            Vote(A) += N^L_{A>B}/sampleSize.
            Vote(B) += N^L_{B>A}/sampleSize.
if Vote(A) = Vote(B) then
    Return a tie
else
    Return the object obj with the highest Vote(obj).
```

Example 5 *Suppose* X *and* O *are as defined in Example 3. The first column of Table 2 lists all LPMs that are consistent with O. The second column gives the posterior probabilities of these models given the sample* S_1*, which is the set of all consistent LPMs. The third column is the posterior probability of the models given the sample* $S_2 = \{(X_2, X_3, X_1, *), (X_3, X_1, *), (X_3, X_2, X_1, *)\}$. *Given two objects* $A = [0, 1, 1, 0, 0]$ *and* $B = [0, 0, 1, 0, 1]$*, the number of votes for each object based on each LPM is given in the last two columns. Note that the total number of votes for* A *and* B *does not add up to the total number of extensions of* $(X_3, X_1, *)$ *because two of its extensions—*(X_3, X_1) *and* (X_3, X_1, X_4)*—prefer* A *and* B *equally.*

Algorithm 4 describes *modelVote*, which takes a sample of consistent LPMs and a pair of objects as input, and predicts the preferred object using the weighted votes of the LPMs in the sample.

Returning to Example 5, the reader can verify that model voting will prefer A over B. Next, we present our theoretical results on the *sampleModels* and *modelVote* algorithms.

Complexity: The time complexity of *sampleModels* is bounded by $O(n^2m)$, where n is the number of variables and m is the number of observations: the while-loop in line 2 runs at most n times; at each iteration, we have to process every observation, each time performing computations in $O(n)$ time. If we call *sampleModels* s times, then the total complexity of sampling is $O(sn^2m)$. For constant s, this bound is still polynomial. Similarly, the complexity of *modelVote* is $O(sn)$ because it considers each of the s rules in the sample, counting the votes of each rule, which can be done in $O(n)$ time.

Comparison to variable voting: The set of LPMs that is sampled via *learnVariableRank* is a subset of the LPMs that *sampleModels* can produce. The running example in the paper demonstrates that *sampleModels* can generate the LPM $(X_3, X_1, *)$; however, none of its extensions is consistent with the partial order *learnVariableRank* returns.

4. Introducing Bias

In general, when there are not many training examples for a learning algorithm, the space of consistent LPMs is large. In this case, it is not possible to find a good approximation of the target model. To overcome this problem, we can introduce bias (domain knowledge), indicating that certain solutions should be favored over the others. In this section, we propose a bias in the form of equivalence classes over the set of attributes. These equivalence classes indicate the set of most important attributes, second most important attributes, and so on. For example, when buying a used car, most people consider the most important attributes of a car to be the mileage, the year, and the make of the car. The second most important set of attributes is the color, number of doors, and body type. Finally, perhaps the least important properties are the interior color and the wheel covers. We now formally define a learning bias and what it means for an LPM to be consistent with a learning bias.

Definition 3 (Learning Bias) *A learning bias $\mathcal{B}$ for learning a lexicographic preference model on a set of variables X is a total order on a partition of X. $\mathcal{B}$ has the form $E_1 < E_2 < \ldots < E_k$, where $\cup_i E_i = X$. Intuitively, B defines a partial order on X such that for any two variables $x \in E_i$ and $y \in E_j$, $x < y$ iff $E_i < E_j$. We denote this partial order by $\preceq_B$.*

Definition 4 *Suppose that $X = \{X_1, \ldots X_n\}$ is a set of variables, $\mathcal{B}$ a learning bias, and $\mathcal{L}$ an LPM. $\mathcal{L}$ is consistent with $\mathcal{B}$ iff the total order $\sqsubset_{\mathcal{L}}$ is consistent with the partial order $\preceq_{\mathcal{B}}$.*

Intuitively, an LPM that is *consistent* with a learning bias respects the variable orderings induced by the learning bias. The learning bias prunes the space of possible LPMs. The size of the partition determines the strength of the bias; for example, if there is a single variable per set, then the bias defines a specific LPM. In general, the number of LPMs that is consistent with a learning bias of the form $E_1 < E_2 < \ldots < E_k$ can be computed with the following recursive formula:

$$G([e_1, \ldots e_k,]) = f_{e_1} + e_1! \times (G([e_2, \ldots e_k]) - 1), \quad (4)$$

where $e_i = |E_i|$ and the base case for the recursion is $G([]) = 1$. The first term in the formula counts the number of possible LPMs using only the variables in E_1, which are the most important variables. The definition of consistency entails that a variable can appear in $\sqsubset_{\mathcal{L}}$ iff all of the more important variables are already in $\sqsubset_{\mathcal{L}}$, hence the term $e_1!$. Note that the recursion on G is limited to the number of sets in the partition, which is bounded by the number of variables; therefore, it can also be computed in linear time by caching precomputed values of f.

To illustrate the power of a learning bias, consider a learning problem with nine variables. Without a bias, the total number of LPMs is 905,970. If a learning bias partitions the variables into three sets, each with three elements, then the number of LPMs consistent with the bias is only 646. A bias with four sets, where the first set has three variables and the rest have two, limits the number to 190.

We can easily generalize the *learnVariableRank* algorithm to utilize the learning bias, by changing only the first line of *learnVariableRank* which initializes the ranks of the variables. Given a bias of the form $S_1 < \ldots < S_k$, the generalized algorithm assigns the rank 1 (most important rank) to the variables in S_1, rank $|S_1| + 1$ to those in S_2, and so forth. This initialization ensures that an observation (A, B) is used for learning the order of variables in a class S_i only when A and B have the same values for all variables in classes $S_1 \ldots S_{i-1}$ and have different values for at least one variable in S_i.

The algorithm *modelVote* can also be generalized to use a learning bias $\mathcal{B}$. In the sample generation phase, we use *sampleModels* as presented earlier, and then eliminate all rules whose prefixes are not consistent with the bias. Note that even if the prefix of an aggregated LPM $\mathcal{L}$ is consistent with a bias, this may not be the case for every extension of $\mathcal{L}$. Thus, in the algorithm *modelVote*, we need to change any references to $F_{\mathcal{L}}$ and $N^{\mathcal{L}}_{A<B}$ (or $N^{\mathcal{L}}_{B<A}$) with $F^{\mathcal{B}}_{\mathcal{L}}$ and $N^{\mathcal{L},\mathcal{B}}_{A<B}$ (or $N^{\mathcal{L},\mathcal{B}}_{B<A}$), respectively, where:

- $F^{\mathcal{B}}_{\mathcal{L}}$ is the number of extensions of $\mathcal{L}$ that are consistent with $\mathcal{B}$, and

- $N^{\mathcal{L},\mathcal{B}}_{A<B}$ is the number of extensions of $\mathcal{L}$ that are consistent with $\mathcal{B}$ and prefer A. ($N^{\mathcal{L},\mathcal{B}}_{B<A}$ is similar.)

Suppose that $\mathcal{B}$ is a learning bias $E_1 < \ldots < E_m$. Let Y denote the prefix variables of an aggregate LPM $\mathcal{L}$ and E_k be the first set such that at least one variable in E_k is not in Y. Then, $F^{\mathcal{B}}_{\mathcal{L}} = G([|E_k - Y|, |E_{k+1} - Y|, \ldots |E_m - Y|])$.

When counting the number of extensions of $\mathcal{L}$ that are consistent with $\mathcal{B}$ and prefer A, we again need to examine the case where the prefix variables equally prefer the objects. Suppose Y is as defined as above and D_i denotes the set difference between E_i and Y. Let D_j be the first non-empty set and D_k be the first set such that at least one variable in D_k has different values in the two objects. Obviously, only the variables in D_k will influence the prediction of the preferred object. If

- $d_i = |D_i|$, the cardinality of D_i, and
- a is the set of variables in D_k that favor A, b is the set of variables in D_k that favor B, and w is the set of variables in D_k that are neutral,

then $N^{\mathcal{L},\mathcal{B}}_{A>B}$, the number of extensions of $\mathcal{L}$ that are consistent with $\mathcal{B}$ and prefer A, can be computed as follows:

$$N^{\mathcal{L},\mathcal{B}}_{A>B} = \frac{|a|}{|a| + |b|} \times (F^{\mathcal{B}}_{\mathcal{L}} - G([d_j \ldots d_{k-1}, |w|])). \quad (5)$$

5. Experiments

In this section, we explain our experimental methodology and discuss the results of our empirical evaluations. We define the *prediction performance* of an algorithm P with respect to a set of test observations T as:

$$\textit{performance}(P,T) = \frac{\textit{Correct}(P,T) + 0.5 \times \textit{Tie}(P,T)}{|T|} \quad (6)$$

where $\textit{Correct}(P,T)$ is the number of observations in T that are predicted correctly by P and $\textit{Tie}(P,T)$ is the number of observations in T that P predicted as a tie. Note that an LPM returned by *greedyPermutation* never returns a tie. In contrast, variable voting with respect to a partial order in which every variable is equally important will only return ties, so the overall performance will be 0.5, which is no better than randomly selecting the preferred objects. We will use MV, VV, and G to denote the model voting, variable voting, and the greedy approximations of an LPM.

Given sets of training and test observations, (O,T), we measure the *average* and *worst* performances of VV, MV and G. When combined with *learnVariableRank*, VV is a deterministic algorithm, so the average and worst performances of VV are the same. However, this is not the case

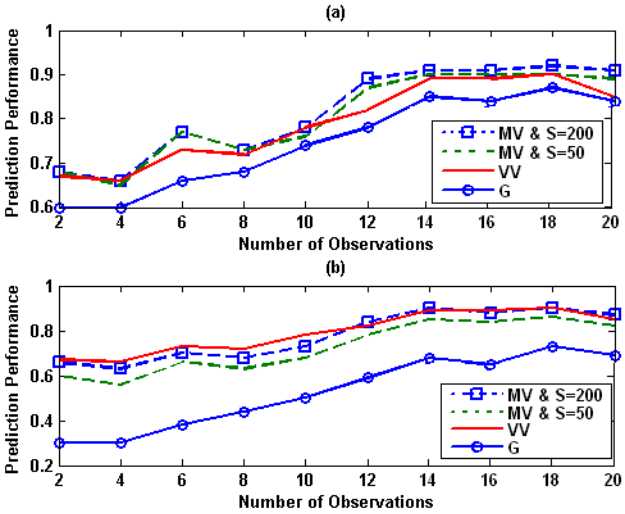


Figure 1. The average and worst prediction performance of the greedy algorithm, variable voting and model voting.

for MV with sampling, because *sampleModels* is randomized. Even for the same training and test data (O,T), the performance of MV can vary. To mitigate this, we ran MV 10 times for each (O,T) pair, and called *sampleModels* S times on each run (thus the sample size is at most S), recording the average and worst of its performance. The greedy algorithm G is also randomized (in line 2, one variable is picked arbitrarily), so we ran G 200 times for every (O,T), recording its average and worst performance.

For our experiments, the control variables are R, the number of relevant variables in the target LPM; I, the number of irrelevant variables; N_O, the number of training observations; and N_T, the number of test observations. For MV experiments we used sample sizes (S) of 50 and 200. Larger sample sizes (e.g. 800) slightly improved performance, but are omitted for space. For fixed values of R and I, an LPM $\mathcal{L}$ is randomly generated. (If a bias B is given, then $\mathcal{L}$ is also consistent with B.) We randomly generated N_O and N_T pairs of objects, each with $I + R$ variables. Finally, we labeled the preferred objects according to $\mathcal{L}$.

Figure 1a shows the average performance of G, MV with two different sample sizes and VV for $R = 15$, $I = 0$, and $N_T = 20$, as N_O ranges from 2 to 20. Figure 1b shows the worst performance for each algorithm. In these figures, the data points are averages over 20 different pairs of training and test sets (O,T). The average performance of VV and MV is better than the average performance of G, and the difference is significant at every data point. Also, note that the worst case performance of G after seeing two observations is around 0.3, which suggests a very poor approximation of the target. VV and MV's worst case performances are much better than the worst case performance of G, justifying the additional complexity of the algorithms MV

and VV. We have observed the same behavior for other values of R and I, and have also witnessed a significant performance advantage for MV over VV in the presence of irrelevant variables when training data is scarce. Space limitations prevent us from presenting these results.

Figure 2 shows the positive effect of learning bias on the performance of voting algorithms for $R = 10$, $I = 0$, and $N_T = 20$, as N_O ranges from 2 to 20. In addition, this experiment aims to show that bias does not undermine the advantage voting algorithms held over the greedy algorithm in the unbiased case. To this end we have trivially generalized G to produce LPMs that are consistent with a given bias. The data points are averages over 20 different pairs of training and test sets (O, T). We have arbitrarily picked two biases: B_1 : $\{X_1, X_2, X_3, X_4, X_5\} < \{X_6, X_7, X_8, X_9, X_{10}\}$ and B_2 : $\{X_1, X_2, X_3\} < \{X_4, X_5\} < \{X_6, X_7, X_8\} < \{X_9, X_{10}\}$. The performance of VV improved greatly with the introduction of learning biases. B_2 is a stronger bias than B_1 and prunes the space of consistent LPMs more than B_1. As a result, the performance gain due to B_2 is greater than that due to B_1. The difference between the bias curves and the non-bias curve is statistically significant except at the last point. Note that the biases are particularly effective when the number of training observations is small. The worst case performance of G with biases B_1 and B_2 are also shown in Figure 2. For both biases, the worst case performance of G is significantly lower than the performance of VV with the corresponding bias. We obtained very similar results with MV but due to space constraints we can not include them in this paper.

6. Related Work

Lexicographic orders and other preference models have been utilized in several research areas, including multicriteria optimization (Bertsekas & Tsitsiklis, 1997), linear programming , and game theory (Quesada, 2003). The lexicographic model and its applications were surveyed by Fishburn (1974). The most relevant existing work for learning and/or approximating LPMs is by Schmitt and Martignon (2006) and Dombi et al. (2007), which were summarized in Section 2. Another analogy, described by Schmitt and Martignon (2006), is between LPMs and decision lists (Rivest, 1987). Specifically, it was shown that LPMs are a special case of 2-decision lists, and that the algorithms for learning these two classes of models are not directly applicable to each other.

7. Conclusions and Future Work

In this paper, we presented democratic approximation methods for learning a lexicographic preference model (LPM) given a set of preference observations. Instead of committing to just one of the consistent LPMs, we maintain a set of models and predict based on the majority of votes. We described two such methods: *variable voting* and *model voting*. We showed that both methods can be implemented in polynomial time and exhibit much better worst- and average-case performance than the existing methods. Finally, we have defined a learning bias that can improve performance when the number of observations is small and incorporated this bias into the voting-based methods, significantly improving their empirical performance.

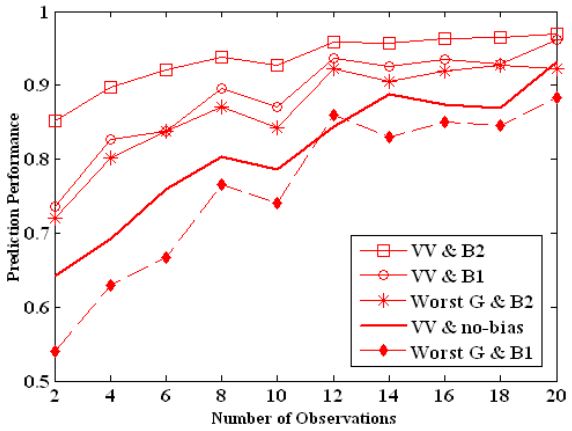


Figure 2. The effect of bias on VV and G.

The future directions of this work are twofold. First, we plan to generalize our algorithms to learn the preferred values of a variable as well as the total order on the variables. Second, we intend to develop democratic approximation techniques for other kinds of preference models.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency and the U. S. Air Force through BBN Technologies Corp. under contract number FA8650-06-C-7606.

References

Bertsekas, D., & Tsitsiklis, J. (1997). *Parallel and distributed computation: numerical methods*. Athena Scientific.

Dombi, J., Imreh, C., & Vincze, N. (2007). Learning lexicographic orders. *European Journal of Operational Research*, *183*, 748–756.

Fishburn, P. (1974). Lexicographic Orders, Utilities and Decision Rules: A Survey. *Management Science*, *20*, 1442–1471.

Quesada, A. (2003). Negative results in the theory of games with lexicographic utilities. *Economics Bulletin*, *3*, 1–7.

Rivest, R. (1987). Learning decision lists. *Machine Learning*, *2*, 229–246.

Schmitt, M., & Martignon, L. (2006). On the complexity of learning lexicographic strategies. *Journal of Machine Learning Research*, *7*, 55–83.

Preconditioned Temporal Difference Learning

Hengshuai Yao HENGSHUAI@GMAIL.COM
Zhi-Qiang Liu ZQ.LIU@CITYU.EDU.HK
School of Creative Media, City University of Hong Kong, Hong Kong, China

Abstract

This paper extends many of the recent popular policy evaluation algorithms to a generalized framework that includes least-squares temporal difference (LSTD) learning, least-squares policy evaluation (LSPE) and a variant of incremental LSTD (iLSTD). The basis of this extension is a preconditioning technique that solves a stochastic model equation. This paper also studies three significant issues of the new framework: it presents a new rule of step-size that can be computed online, provides an iterative way to apply preconditioning, and reduces the complexity of related algorithms to near that of temporal difference (TD) learning.

1. Introduction

In Reinforcement Learning (RL), a primary concern is how to reuse experience in an intelligent and fast way. To achieve this we must consider two major issues, namely, the data efficiency and the computational efficiency. Recently these two issues were widely studied by the research on temporal difference (TD) leaning. TD is a classical algorithm well suited for policy evaluation (Sutton, 1988), and achieves great success for its wide applications in control and AI games (Sutton & Barto, 1998). One of its significant advantages is its superior computational efficiency. If the feature vector has K components, TD requires $O(K)$ complexity. However, previous research shows that TD uses experience inefficiently (Lin & Mitchell, 1992)(Geramifard et al., 2006a). The reason is that TD throws the transition information away after using it for one update of weights. One way to reuse this information is to accumulate it into a data set once it has been experienced. Then TD methods are repeatedly applied to the data set. This pattern is known as experience replay (Lin & Mitchell, 1992), and has a high efficiency in using experience because each transition is exploited to the maximum extent. However, this method may be inefficient to perform online because the data set is possible to grow extremely large if the exploration process runs for a long time[1]. Another way is to extract some data structure from the sequence of experience and update the weights with the help of this structure. This way is more desirable because the data structure requires much smaller size of memory than the data set, and leads to recent popular algorithms such as least-squares temporal difference (LSTD) (Boyan, 1999) and least-squares policy evaluation (LSPE) (Nedić & Bertsekas, 2003). Compared to TD, the two algorithms are more data efficient, but similar to the experience replay, are still computationally expensive.

LSTD inverts some accumulated matrix per time step, and generally requires $O(K^3)$. Recursive LSTD (RLSTD) computes the inversion of LSTD's matrix iteratively using Sherman-Morison formula and reduces the complexity to $O(K^2)$ (Bradtke & Barto, 1996)(Xu et al., 2002). LSPE is similar to LSTD and will be examined later. Recently incremental LSTD (iLSTD) was proposed to strike a balance between LSTD and TD (Geramifard et al., 2006a)(Geramifard et al., 2006b): its data efficiency is almost as good as LSTD, but its computational cost is very near to that of TD. However, iLSTD still requires tuning the step-size manually as TD. In contrast, LSTD has no parameter to tune.

The aim of this paper is to explore the relations among recent popular policy evaluation algorithms. A framework of policy evaluation algorithms called the *preconditioned temporal difference* (PTD) learning is introduced, which includes LSTD, LSPE and a variant of iLSTD, *etc.* Furthermore, we maintain LSTD's prop-

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] Lin avoided this problem by using only a window of most recent experience (Lin & Mitchell, 1992). This, however, results in loss of information and it is in general difficult to prespecify the window size.

erty of ease of tuning. This paper proposes an adaptive step-size that can be computed online by all PTD algorithms.

To reduce computational complexity $O(K^2)$ of the PTD algorithms due to the inversion of the preconditioner matrix, we develop an efficient incremental process to apply preconditioning, which leads to a set of incremental PTD algorithms. Incremental PTD algorithms take advantage of the sparse nature of RL tasks, storing and manipulating only the valid experience by a *condensed* structure every time step. We also present an efficient adaptive step-size for incremental PTD algorithms. Results on Boyan chain example show that PTD algorithms using the adaptive step-size gives much faster convergence than using previous rule of step-size; incremental PTD algorithms via condensed implementation have a high efficiency in using data while a low complexity similar to iLSTD.

1.1. Stationary Model Equation

Given a state space $\mathcal{S} = \{1, 2, \ldots, N\}$, the problem of *policy evaluation* is to predict the long-term optimal reward of a policy for each state s:

$$J(s) = \sum_{t=0}^{\infty} \gamma^t r(s_t, s_{t+1}), \quad s_0 = s, \quad 0 < \gamma \leq 1,$$

where γ is the discount factor, and $r(s_t, s_{t+1})$ is the reward received by the agent at time t. Given $K (K \leq N)$ feature functions $\varphi_k(\cdot) : \mathcal{S} \mapsto \mathcal{R}$, $k = 1, \ldots, K$, the feature of state s_t is $\phi_t = [\varphi_1(s_t), \varphi_2(s_t), \ldots, \varphi_K(s_t)]'$. The optimal reward vector J can now be approximated by $\hat{J} = \Phi w$, where w is the weight vector, and Φ is the feature matrix whose entries are $\Phi(j, k) = \varphi_k(j)$, $k = 1, \ldots, K$; $j = 1, \ldots, N$.

For an ergodic Markov chain that has steady-state probabilities $\pi(1), \pi(2), \ldots, \pi(N)$, (Tsitsiklis & Van Roy, 1997) proved that TD(λ) eventually finds a weight vector w^* that satisfies a linear system

$$Aw^* = -b, \tag{1}$$

where A and b are defined by

$$A = \Phi' D(\gamma P - I) \sum_{k=0}^{\infty} (\lambda\gamma P)^k \Phi, \quad b = \Phi' D \sum_{k=0}^{\infty} (\lambda\gamma P)^k \bar{r},$$

D is the diagonal matrix with diagonal entries $\pi(i)$, $i = 1, \ldots, N$; $\lambda \in [0, 1]$ is the eligibility trace factor; P is the transition probability matrix; I is the identity matrix; and $\bar{r}$ is the vector with components $\bar{r}_i = \sum_{j=1}^{N} P_{i,j} r(i, j)$, $i = 1, \ldots, N$. For each $\lambda \in [0, 1]$, w^* is also the limit point of LSTD(λ), LSPE(λ) and iLSTD(λ). Equation (1) is only useful for analysis but not applicable in practice and will be called the stationary model equation.

1.2. Law of Large Numbers

A common structure grown by LSTD(λ), LSPE(λ) and iLSTD(λ) is updated incrementally. If the current transition from s_t to s_{t+1} incurs a reward r_t, then a matrix and a vector are updated by

$$\tilde{A}_{t+1} = \tilde{A}_t + z_t(\gamma\phi_{t+1} - \phi_t)'; \quad \tilde{b}_{t+1} = \tilde{b}_t + z_t r_t,$$

where z_t is the eligibility trace, computed recursively by $z_{t+1} = \lambda\gamma z_t + \phi_{t+1}$. Because the components of $\tilde{A}_{t+1}$ and $\tilde{b}_{t+1}$ can get to infinity it is better to use some well-defined term. For infinite-horizon problems, (Tadić, 2001)(Nedić & Bertsekas, 2003) used the following structure

$$A_{t+1} = \frac{1}{t+1}\tilde{A}_{t+1}; \quad b_{t+1} = \frac{1}{t+1}\tilde{b}_{t+1}, \tag{2}$$

which satisfies the law of large numbers. However, (2) are no longer consistent estimations of A and b for absorbing Markov chains such as Boyan chain example. In Section 2.1, such an extension is proposed.

1.3. Related Algorithms

At time t, the rule of LSTD(λ) for updating weights can be specified as $w_{t+1} = -\tilde{A}_{t+1}^{-1}\tilde{b}_{t+1}$. In practice, $\tilde{A}_t$ can be singular and a perturbation which sets $\tilde{A}_0$ to $\delta_0^- I$ ($\delta_0^- < 0$) should be used.

LSPE(λ) was proposed for infinite-horizon problem (Nedić & Bertsekas, 2003). If the current step-size is α_t, LSPE updates w by

$$w_{t+1} = w_t + \alpha_t (D_{t+1})^{-1} (A_{t+1} w_t + b_{t+1}), \tag{3}$$

where

$$D_{t+1} = \frac{1}{t+1}\left(\delta_0^+ I + \sum_{k=0}^{t} \phi_k \phi_k'\right), \quad \delta_0^+ > 0.$$

In the long run, D_{t+1} converges to $\Phi' D \Phi$.

1.4. Preconditioning

Generally, solutions to a linear system like (1) can be categorized into two classes. The first is the direct methods (Saad, 2003), which factorize A into easily invertible matrices, including Gaussian elimination and LU/QR factorization, *etc.* However, the complexity involved in factorizing A is not practical for large scale systems. The second class, known as the iterative solutions, scales well with the problem size and is very efficient for large and sparse linear systems.

According to the literature of iterative solutions, preconditioning is especially effective for symmetric system (Saad, 2003), but usually for RL tasks, matrix A is not symmetric. Therefore, the original stationary model equation is first transformed into the following symmetric form

$$A'Aw^* = -A'b, \tag{4}$$

which can be solved by Richardson's iteration (Saad, 2003)

$$w_{\tau+1} = w_\tau - \alpha A' \left(Aw_\tau + b\right), \tag{5}$$

where α is some positive scalar that should satisfy

$$\rho(I - \alpha A'A) < 1.$$

The technique of preconditioning refers to a general technique which preconditions a system before solving it. For example, preconditioning (4) gives us the preconditioned symmetric model equation

$$C^{-1}A'Aw^* = -C^{-1}A'b,$$

where C is an invertible matrix, usually called the *preconditioner*. Then the model equation is solved by the iteration

$$w_{\tau+1} = w_\tau - C^{-1}A' \left(Aw_\tau + b\right). \tag{6}$$

Convergence rate of (6) is governed by the spectral radius of $I - C^{-1}A'A$: the smaller the radius is, the faster the solution will be (Saad, 2003). Therefore a good preconditioner should make the preconditioned radius smaller than the original radius, *i.e.*, $\rho(I - C^{-1}A'A) < \rho(I - \alpha A'A)$.

2. The Generalized Framework

We first give consistent estimations of A and b for absorbing Markov chains, and then we show how to apply them together with preconditioning to policy evaluation.

2.1. Robbins-Monro for Absorbing Chains

A trajectory of an absorbing Markov chain is a finite sequence $s_0, \ldots, s_q$, where s_q is the absorbing state. Given trajectories $1, \ldots, M$, where the mth trajectory has length L_m, the consistent estimations of A and b are

$$A_M = \frac{1}{T}\sum_{m=1}^{M}\sum_{t=0}^{L_m} z_t(\gamma\phi_{t+1} - \phi_t)', \tag{7}$$

and

$$b_M = \frac{1}{T}\sum_{m=1}^{M}\sum_{t=0}^{L_m} z_t r_t, \tag{8}$$

where T is the number of all observed state visits in M trajectories, and z_t is the eligibility trace. Similarly, for absorbing Markov chain, LSPE should use the following structure to estimate $\Phi' D \Phi$:

$$D_M = \frac{1}{T}\sum_{m=1}^{M}\sum_{t=0}^{L_m} \phi_t\phi_t'. \tag{9}$$

On a transition from s_t to s_{t+1}, estimations (7), (8) and (9) can be updated incrementally, which is achieved by a Robbins-Monro (RM) procedure:

$$A_{t+1} = A_t + \frac{1}{T}(z_t(\gamma\phi_{t+1} - \phi_t)' - A_t), \tag{10}$$

$$b_{t+1} = b_t + \frac{1}{T}(z_t r_t - b_t), \tag{11}$$

and

$$D_{t+1} = D_t + \frac{1}{T}(\phi_t\phi_t' - D_t), \tag{12}$$

where T is updated by $T = T + 1$ after the three estimations. The convergence of RM procedure can be proved in similar manner to the case of infinite-horizon problems (Tadić, 2001)(Nedić & Bertsekas, 2003).

2.2. The Framework

Given A_{t+1} and b_{t+1} estimated by RM, we can define a *stochastic model equation*

$$A_{t+1}w = -b_{t+1}.$$

Because RM estimations have some error, the stochastic model equation is not satisfied, and there exists a nonzero residual vector

$$e_{t+1}(w) = A_{t+1}w + b_{t+1}. \tag{13}$$

A natural idea is that the current weights can be improved by minimizing the residual error $||e_{t+1}(w)||^2$, which produces a gradient descent algorithm

$$w_{t+1} = w_t - \alpha_t A'_{t+1}(A_{t+1}w_t + b_{t+1}),$$

where α_t is a positive step-size. Gradient descent algorithm is a stochastic form of the iteration (5).

The general preconditioned temporal difference (PTD) learning applies the technique of preconditioning to improve the convergence rate of gradient descent. Assume C_{t+1} is a chosen preconditioner, the rule of PTD can be cast as

$$w_{t+1} = w_t - \alpha_t C_{t+1}^{-1} A'_{t+1}(A_{t+1}w_t + b_{t+1}), \tag{14}$$

where α_t is some scalar but not necessarily positive. With the rule proposed in Section 3, the step-size guarantees the convergence of PTD algorithms and makes

them more flexible in stochastic environments than (6).

The choice of preconditioner is a key issue. Generally, preconditioner should decrease the spectral radius of gradient descent:

$$\rho(I - \alpha_t C_{t+1}^{-1} A'_{t+1} A_{t+1}) < \rho(I - \alpha_t A'_{t+1} A_{t+1}).$$

Gradient descent makes no preconditioning because it chooses the identity matrix as preconditioner. Good examples of preconditioner can be found in recent popular policy evaluation algorithms.

2.3. Relations to Previous Algorithms

LSTD, LSPE and iLSTD are all special forms of applying preconditioner to gradient descent algorithm:

$C_{t+1} = -A'_{t+1} D_{t+1}$, where D_{t+1} is defined in (12). One can easily verify that this is a variant of LSPE(λ).

$C_{t+1} = A'_{t+1} A_{t+1}$. This is an extended form of LSTD: $w_{t+1} = (1 - \alpha_t) w_t + \alpha_t(-A_{t+1}^{-1} b_{t+1})$. Using 1 as the step-size, we get exactly LSTD(λ). Later we will see that LSTD is optimal in choosing its step-size because certain residual error is minimized.

$C_{t+1} = -A'_{t+1}$. This approach is a variant of iLSTD(λ) (Yao & Liu, 2008).

3. The Rule of Step-size

This section presents an adaptive process to compute the step-size online for gradient descent algorithm, LSTD, iLSTD and LSPE. The four algorithms all use a preconditioner in the form of $A'_{t+1} B_{t+1}$, which is assumed to be used by general PTD algorithms.

The update direction of PTD is provided by a *preconditioned residual* (p-residual) vector

$$\delta_t = B_{t+1}^{-1} e_{t+1}(w_t), \tag{15}$$

where e_{t+1} is the residual vector defined in (13). This p-residual is an "old" one, because it is obtained before the weight update. After the weight update, the p-residual vector changes to

$$\theta_{t+1} = B_{t+1}^{-1} e_{t+1}(w_{t+1}).$$

From (14) and (15), θ_{t+1} can be rewritten as

$$\begin{aligned} \theta_{t+1} &= B_{t+1}^{-1}(A_{t+1}(w_t - \alpha_t \delta_t) + b_{t+1}) \\ &= \delta_t - \alpha_t B_{t+1}^{-1} A_{t+1} \delta_t. \end{aligned}$$

Because θ_{t+1} stands for an improved difference between the two sides of the preconditioned stochastic model equation, naturally we hope that the new p-residual error is smaller than the old one: $||\theta_{t+1}||^2 < ||\delta_t||^2$. This can be guaranteed by requiring that θ_{t+1} be orthogonal to $\theta_{t+1} - \delta_t$. Accordingly, we obtain a new rule of step-size

$$\alpha_t = \frac{\delta'_t B_{t+1}^{-1} A_{t+1} \delta_t}{(B_{t+1}^{-1} A_{t+1} \delta_t)'(B_{t+1}^{-1} A_{t+1} \delta_t)}. \tag{16}$$

This step-size is the optimal value that minimizes the new p-residual error over $\alpha \in \mathcal{R}$, *i.e.*, $\alpha_t = \arg\min ||\theta_{t+1}||^2$. Obviously the step-size is positive when $B_{t+1}^{-1} A_{t+1}$ is positive definite, which is true for gradient descent algorithm, iLSTD, and LSTD.

It is interesting that in (16), if $B_{t+1} = A_{t+1}$, then the step-size is equal to 1. This indicates that LSTD's choice of step-size is optimal in the sense that the p-residual error $||(1 - \alpha_t)(w_t + A_{t+1}^{-1} b_{t+1})||^2$ is minimized. When $B_{t+1} = -I$, the residual error $||(I + \alpha_t A_{t+1}) e_{t+1}(w_t)||^2$ is minimized; for ease of later comparisons with previous step-size of iLSTD, this variant will be called the Minimal Residual (MR) algorithm.

To compute p-residual vector and step-size, PTD algorithms have to carry out matrix inversion, which requires $O(K^3)$. Sherman-Morison formula is a solution to reduce this complexity to $O(K^2)$. Another efficient solution is to apply preconditioning incrementally and take advantage of the sparse nature of RL tasks.

4. Incremental PTD

The key of incremental preconditioning is to approximate the p-residual and the adaptive step-size in an iterative way.

4.1. Iterative P-residual and Approximated Step-size

Let κ_t be the error caused by the residual and the current iterative p-residual, defined by

$$\kappa_t = e_{t+1}(w_t) - B_{t+1} \hat{\delta}_t. \tag{17}$$

The new estimation of δ_t can be improved by

$$\hat{\delta}_{t+1} = \hat{\delta}_t - \beta_t \kappa_t, \tag{18}$$

where β_t is computed by

$$\beta_t = -\frac{\kappa'_t (B_{t+1} \kappa_t)}{(B_{t+1} \kappa_t)'(B_{t+1} \kappa_t)}. \tag{19}$$

Substituting the p-residual δ_t in (14) with the iterative p-residual $\hat{\delta}_t$, we get the general form of incremental PTD algorithms

$$w_{t+1} = w_t - \hat{\alpha}_t \hat{\delta}_t, \tag{20}$$

Algorithm 1 Efficient matrix-vector multiplication using CSR.

Input: $Z_Q^{t+1}(a, c, d)$ and a vector β_t
Output: A vector $o_t = Q_{t+1}\beta_t$
for $k = 1$ **to** K **do**
  $k_1 = d_{t+1}(k)$
  $k_2 = d_{t+1}(k+1) - 1$
  $o_t(k) = a_{t+1}(k_1 : k_2)'\beta_t(c_{t+1}(k_1 : k_2))$
end for

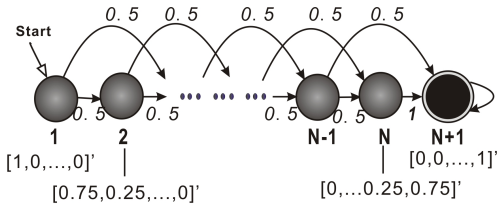


Figure 1. Boyan chain example with $N + 1$ states. The transition probabilities are marked on the arch.

where $\hat{\alpha}_t$ is computed by the following steps.

Given the iterative p-residual $\hat{\delta}_t$, steps (21a)–(21d) compute a vector v, which is an approximation of $B_{t+1}^{-1}A_{t+1}\delta_t$; then the approximated step-size is computed by (21e):

$$\xi_t = A_{t+1}\hat{\delta}_t, \tag{21a}$$

$$\chi_t = \xi_t - B_{t+1}v_t, \tag{21b}$$

$$\eta_t = -\frac{\chi_t'(B_{t+1}\chi_t)}{(B_{t+1}\chi_t)'(B_{t+1}\chi_t)}, \tag{21c}$$

$$v_{t+1} = v_t - \eta_t\chi_t, \tag{21d}$$

$$\hat{\alpha}_t = \frac{\hat{\delta}_t'v_{t+1}}{v_{t+1}'v_{t+1}}. \tag{21e}$$

If $B_{t+1} = -I$, iterative p-residual reproduces p-residual exactly and we get MR(λ); If $B_{t+1} = -D_{t+1}$, we get an incremental form of LSPE(λ) (iLSPE(λ)) that applies preconditioning via iterative p-residual.

4.2. Incremental PTD Using CSR

If the function approximation used is sparse, then matrix Φ is sparse. While this seems a restrictive condition, several popular linear function approximation schemes such as lookup table, Boyan's linear interpolation approximation and tile coding (Sutton & Barto, 1998), are indeed sparse. If the transition matrix is also sparse, matrices A_{t+1} and B_{t+1} will both have many zero entries, implying that "no experience is available for the states related to these entries". Therefore, it is better to remove the void experience and store only the valid experience by a *condensed* structure. Here the Compressed Sparse Row (CSR) format (Saad, 2003) is used. Let Q_t stand for A_t or B_t. The CSR format is a triplet $Z_Q^t(a_t, c_t, d_t)$, where a_t is a real array containing all the real values of the nonzero elements of Q_t; c_t is an integer array containing the column indices of the elements stored in a_t; and d_t is an integer array containing the pointers to the beginning of each row in a_t and c_t.

When Q_{t+1} is sparse, the need for fast matrix-vector multiplication offers a place where CSR fits in. The details are shown by Algorithm 1, whose complexity is $O(l_{t+1})$, where l_{t+1} is the number of nonzero entries in Q_{t+1}. Now (17), (19), (21a), (21b) and (21c) can make a call to Algorithm 1, and the complexity of incremental PTD is given by the following theorem which is proved in (Yao & Liu, 2008).

Theorem 4.2.1 (Complexity of incremental PTD)**.** *The per-time-step complexity of incremental PTD using CSR is $O(qK)$, where q is a small positive real related to the sparsity of matrix A.*

5. Boyan Chain Example

Boyan chain and the features are shown in Figure 1. Transition from N to $N + 1$ incurs a reward -2; transition from $N + 1$ incurs 0; the other transitions incur -3. The discount factor γ is set to 1.

The first experiment is another implementation of experience replay. As RM procedure is able to extract compressed experience information by estimations of A and b, it is natural to ask whether experience can be well replayed by repeatedly presenting RM's estimations to PTD algorithms. Two questions arise for this approach. Will it lead to convergence? What is the role of λ for PTD(λ)?

RM(λ) were first run and averaged over 10 sets of 10000 trajectories, and then their estimated structures were repeatedly presented to Gradient descent(GRAD(λ)), iLSTD(λ), LSPE(λ) and LSTD(λ). All compared algorithms used the adaptive step-size derived in Section 3, and converged to satisfactory solutions for a variety of problem sizes with all $\lambda \in [0, 1]$. The case of $N = 12$ is shown in Figure 2. It is very interesting that for all PTD algorithms smaller λ gives better performance; $\lambda = 0$ performs best and $\lambda = 1$ performs worst, —exactly the same role with that for TD(λ) under repeated presentation training paradigm (Sutton, 1988). Explanation can be given if we view TD as a model exploration process: although TD does not use the model A and b explicitly, its learn-

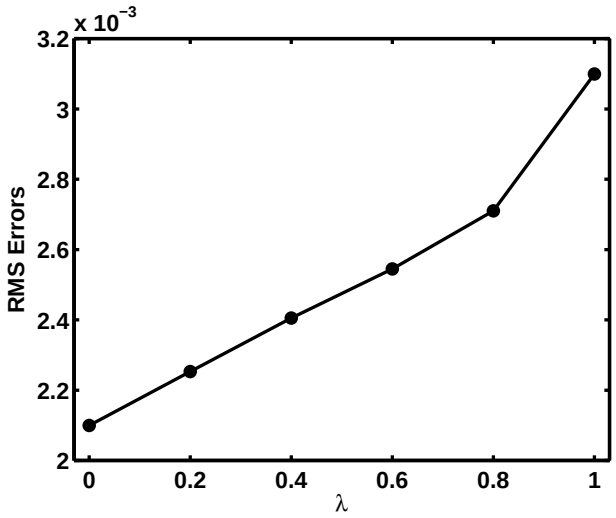


Figure 2. Effects of λ: the (same) RMS errors by GRAD(λ), MR(λ), LSPE(λ) and LSTD(λ).

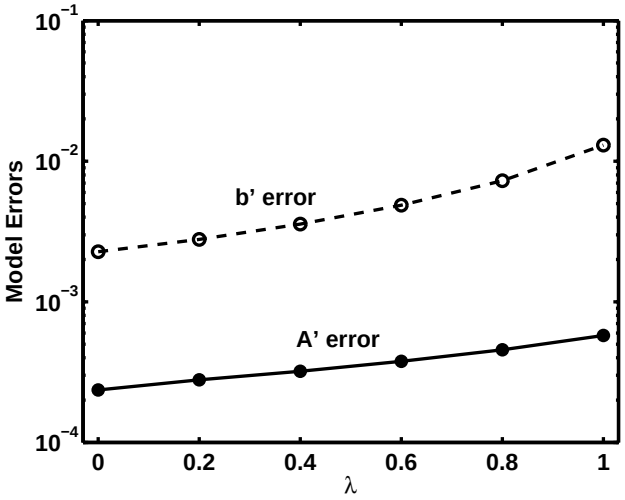


Figure 3. Model errors by RM(λ) procedures.

ing requires exploring and sampling temporal values of the model. It appears that both TD and PTD algorithms rely on the model data reflected by the sets of trajectories.

To explore λ's effect for algorithms, we only have to study its role for the model data, which is extracted by RM procedure. Results are shown in Figure 3, where the model errors are measured by $||A_T - A||_2$ and $||b_T - b||_2$, averaged over 10 sets of 10000 trajectories. It can be observed that smaller λ has smaller modeling errors for A and b, —the role of λ for RM(λ) is just what should be consistent with that for TD(λ) and PTD(λ).

Although all PTD algorithms converge to the same solution, their rates of convergence are quite different. The case of $\lambda = 1$ is shown in Figure 4. MR, LSPE and LSTD are faster than Gradient descent because they make use of preconditioning and their spectral radii are smaller than that of Gradient descent. Figure 5 compares the spectral radii $\rho(I - \alpha_t C_t^{-1} A_t' A_t)$ of different algorithms.

Algorithms were also compared under the same learning paradigm as Boyan (Boyan, 1999), where weights were updated immediately after RM estimations at each time step. All compared algorithms used the adaptive rule except that iLSPE used the approximate step-size developed in Section 4.1. For both adaptive step-size and approximated step-size, a satisfactory convergence was obtained. Results are shown in Figure 6 and Figure 7, where each point was the averaged RMS error over 10 sets of data. It is clear that some intermediate value of λ performs best in both learning error and convergence rate for all algorithms. Generally, four preconditioned algorithms learns faster than Gradient descent algorithm. However, the convergence rate advantages of MR(λ), iLSPE(λ), LSPE(λ) and LSTD(λ) over Gradient descent are becoming smaller as λ increases. The reason may be that larger λ causes larger model error and deteriorates the effects of preconditioning.

Experiment was also run to compare the adaptive step-size with the rule used by (Geramifard et al., 2006a), which takes $\alpha_t = \frac{c_0(c_1+1)}{traj\#+c_1}$, where c_0 was chosen from $\{0.01, 0.1, 1\}$, and c_1 was chosen from $\{100, 1000, 10^6\}$. The best performance of all the nine combinations of the two constants was experimentally chosen for iLSTD. Figure 8 shows that RMS error of MR (adaptive step-size) is faster to decrease than that of iLSTD. From Figure 8, we can also observe that PTD's predictions (such as those given by LSTD and LSPE) have larger variations than incremental PTD's (such as those given by MR and iLSPE). The reason is that PTD algorithms are based on the inversion of preconditioner, which is not well conditioned at the beginning stage of learning; while incremental PTD algorithms avoid numerical instability via iterative p-residual.

Table 1 compares the complexity of PTD and incremental PTD algorithms, where CSR are used for MR (one CSR for A_t) and iLSPE (one CSR for A_t and one CSR for D_t). We can see that incremental PTD algorithms have a clear computational advantage over

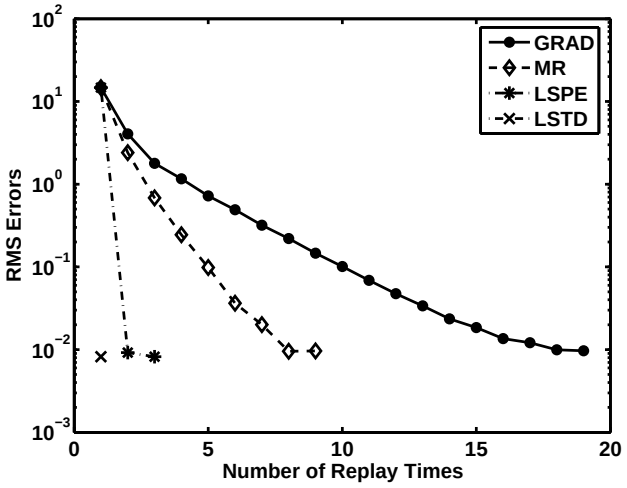


Figure 4. The role of preconditioner ($\lambda = 1$). Algorithms are stopped if RMS error is smaller than 0.01.

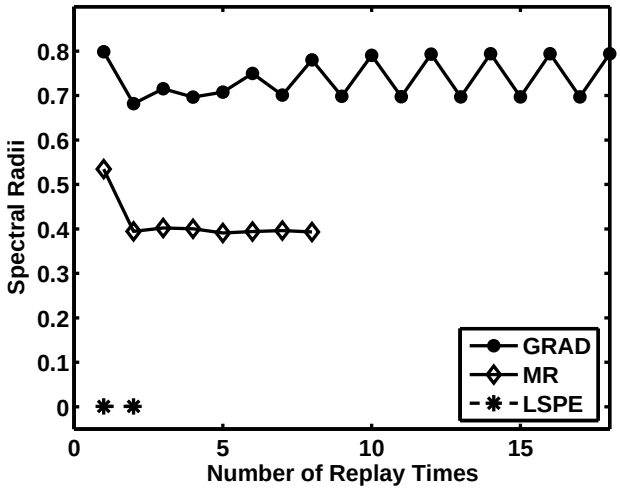


Figure 5. Spectral radius comparisons ($\lambda = 1$). LSTD's spectral radius is 0 permanently, thus not shown.

Table 1. Comparison of per-time-step running time (ms) of PTD and incremental PTD algorithms on $K = 401$ ($\lambda = 0$). The machine used is Pentium(R) 4 PC (CPU 3.00GHZ; RAM 1.00GB).

LSTD	LSPE	iLSTD	MR	iLSPE
72.3	120.3	5.9	10.6	21.9

Table 2. Comparison of memory requirements for A using CSR and full matrix for a variety of problem sizes ($\lambda = 0$).

N	12	100	400	800	1200	1600
$\frac{l}{K^2}$	0.75	0.1479	0.04	0.0198	0.0132	0.01

PTD algorithms. Reason lies in that CSR enables incremental PTD to manipulate much smaller size of data than PTD. Table 2 shows the relative memory requirements of CSR and full matrix for a variety of problem sizes by the ratio l/K^2, where l is the nonzero entries of A. We can observe that the larger the size of state space is, the more advantages will be gained by using CSR.

6. Conclusion

In this paper we proposed two general frameworks, PTD and incremental PTD, which are more data efficient than TD. Generally PTD approaches such as LSTD and LSPE are computationally expensive, whereas incremental PTD algorithms such as MR and iLSPE can take advantage of sparse nature of RL tasks, and have complexity near to that of iLSTD and TD. We also develop an adaptive process for computing the step-size online for PTD algorithms, and an approximated process for computing the step-size for incremental PTD algorithms.

Acknowledgement

We are thankful to Lihong Li, George Konidaris and Andrew Barto for helpful discussions with a draft of this paper. We appreciate for the suggestions by the four reviewers that help improve this paper in many aspects. This research has been partially supported by RGC CERG grant No. CityU 1178/06 (9041147) from Hong Kong UGC.

References

Boyan, J. A. (1999). Least-squares temporal difference learning. *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 49–56). Morgan Kaufmann.

Bradtke, S., & Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, *22*, 33–57.

Geramifard, A., Bowling, M., & Sutton, R. S. (2006a). Incremental least-squares temporal difference learning. *Twenty-First National Conference on Artificial Intelligence (AAAI-06)* (pp. 356–361). AAAI Press.

Geramifard, A., Bowling, M., Zinkevich, M., & Sutton, R. S. (2006b). iLSTD: Eligibility traces and

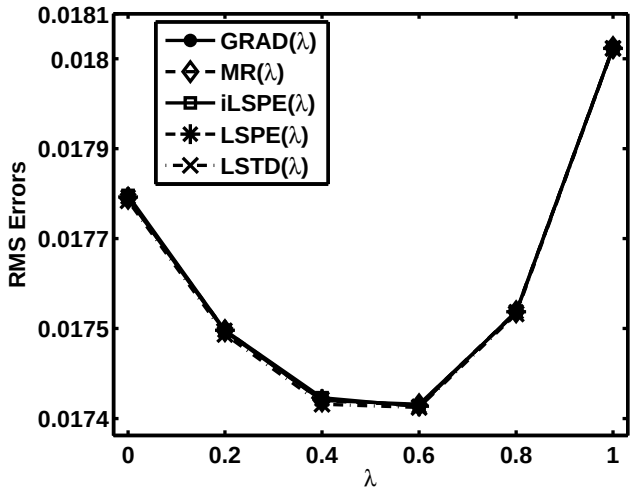


Figure 6. The RMS errors at 90000th visit by GRAD(λ), MR(λ), iLSPE(λ), LSPE(λ) and LSTD(λ).

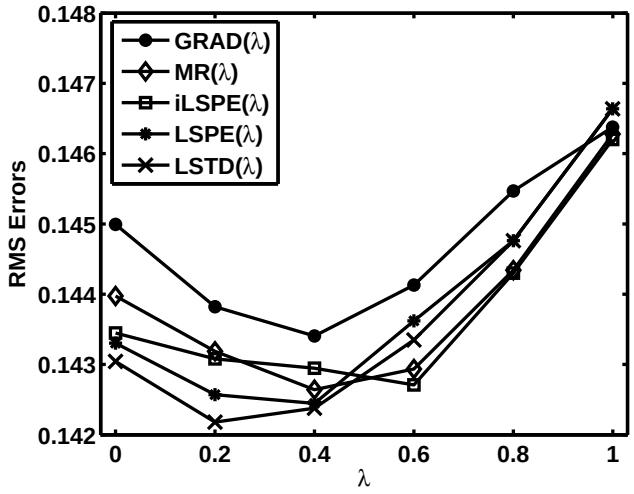


Figure 7. Comparison of convergence rate in terms of RMS errors at 900th state visit.

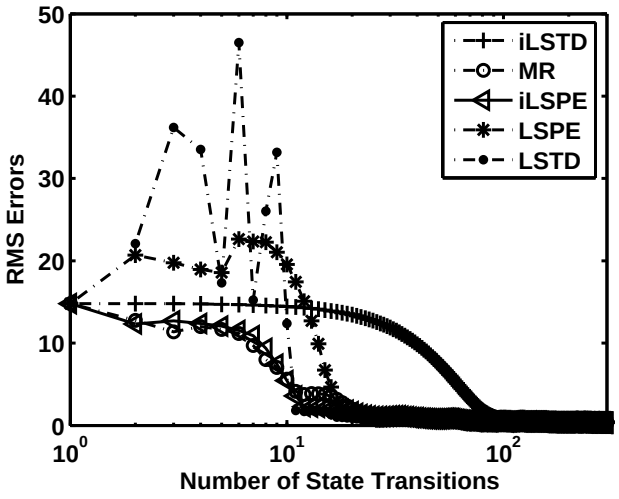


Figure 8. Averaged RMS errors (over 10×10000 trajectories) of iLSTD, MR, iLSPE, LSPE and LSTD in Boyan's setting ($\lambda = 0$). Perturbation factors of LSTD and LSPE were -0.01 and 0.01 respectively. Weights for all algorithms were initialized to $[0.1, 0.1, 0.1, 0.1]'$ except for LSTD.

convergence analysis. *Advances in Neural Information Processing Systems 19* (pp. 441–448).

Lin, L.-J., & Mitchell, T. M. (1992). *Memory approaches to reinforcement learning in non-markovian domains* (Technical Report CMU-CS-92-138). Carnegie Mellon University, Pittsburgh, PA 15213.

Nedić, A., & Bertsekas, D. P. (2003). Least-squares policy evaluation algorithms with linear function approximation. *Journal of Discrete Event Systems*, *13*, 79–110.

Saad, Y. (2003). *Iterative methods for sparse linear systems*. SIAM.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.

Tadić, V. (2001). On the convergence of temporal-difference learning with linear function approximation. *Machine Learning*, *42*, 241–267.

Tsitsiklis, J. N., & Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, *42*, 674–690.

Xu, X., He, H., & Hu, D. (2002). Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Intelligence Research*, *16*, 259–292.

Yao, H., & Liu, Z. (2008). *Preconditioned temporal difference learning* (Technical Report CityU-SCM-MCG-0408). City University of Hong Kong.

A Quasi-Newton Approach to Nonsmooth Convex Optimization

Jin Yu JIN.YU@ANU.EDU.AU
S.V. N. Vishwanathan SVN.VISHWANATHAN@NICTA.COM.AU
Simon Günter GUENTER_SIMON@HOTMAIL.COM
Nicol N. Schraudolph NIC@SCHRAUDOLPH.ORG
Canberra Research Laboratory, NICTA, Canberra, Australia
Research School of Information Sciences & Engineering, Australian National University, Canberra, Australia

Abstract

We extend the well-known BFGS quasi-Newton method and its limited-memory variant LBFGS to the optimization of nonsmooth convex objectives. This is done in a rigorous fashion by generalizing three components of BFGS to subdifferentials: The local quadratic model, the identification of a descent direction, and the Wolfe line search conditions. We apply the resulting subLBFGS algorithm to L_2-regularized risk minimization with binary hinge loss, and its direction-finding component to L_1-regularized risk minimization with logistic loss. In both settings our generic algorithms perform comparable to or better than their counterparts in specialized state-of-the-art solvers.

1. Introduction

The (L)BFGS quasi-Newton method (Nocedal and Wright, 1999) is widely regarded as the workhorse of smooth nonlinear optimization due to its combination of computational efficiency with good asymptotic convergence. Given a smooth objective function $J : \mathbb{R}^d \to \mathbb{R}$ and a current iterate $\boldsymbol{w}_t \in \mathbb{R}^d$, BFGS forms a local quadratic model of J:

$$Q_t(\boldsymbol{p}) \;:=\; J(\boldsymbol{w}_t) + \tfrac{1}{2}\,\boldsymbol{p}^\top \boldsymbol{B}_t^{-1}\boldsymbol{p} + \nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}, \quad (1)$$

where $\boldsymbol{B}_t \succ 0$ is a positive-definite estimate of the inverse Hessian of J. Minimizing $Q_t(\boldsymbol{p})$ gives the quasi-Newton direction

$$\boldsymbol{p}_t := -\boldsymbol{B}_t \nabla J(\boldsymbol{w}_t), \quad (2)$$

which is used for the parameter update:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta_t \boldsymbol{p}_t. \quad (3)$$

The step size $\eta_t \in \mathbb{R}_+$ is normally determined by a line search obeying the Wolfe conditions:

$$\begin{aligned} J(\boldsymbol{w}_{t+1}) \;&\leq\; J(\boldsymbol{w}_t) + c_1 \eta_t \nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}_t \\ \text{and} \quad \nabla J(\boldsymbol{w}_{t+1})^\top \boldsymbol{p}_t \;&\geq\; c_2 \nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}_t, \end{aligned} \quad (4)$$

with $0 < c_1 < c_2 < 1$. The matrix $\boldsymbol{B}_t$ is then modified via the incremental rank-two update

$$\boldsymbol{B}_{t+1} = (\boldsymbol{I} - \varrho_t \boldsymbol{s}_t \boldsymbol{y}_t^\top)\boldsymbol{B}_t(\boldsymbol{I} - \varrho_t \boldsymbol{y}_t \boldsymbol{s}_t^\top) + \varrho_t \boldsymbol{s}_t \boldsymbol{s}_t^\top, \quad (5)$$

where $\boldsymbol{s}_t := \boldsymbol{w}_{t+1} - \boldsymbol{w}_t$ and $\boldsymbol{y}_t := \nabla J(\boldsymbol{w}_{t+1}) - \nabla J(\boldsymbol{w}_t)$ denote the most recent step along the optimization trajectory in parameter and gradient space, respectively, and $\varrho_t := (\boldsymbol{y}_t^\top \boldsymbol{s}_t)^{-1}$. Given a descent direction $\boldsymbol{p}_t$, the Wolfe conditions ensure that $(\forall t)\ \boldsymbol{s}_t^\top \boldsymbol{y}_t > 0$ and hence $\boldsymbol{B}_0 \succ 0 \implies (\forall t)\ \boldsymbol{B}_t \succ 0$.

Limited-memory BFGS (LBFGS) is a variant of BFGS designed for solving large-scale optimization problems where the $O(d^2)$ cost of storing and updating $\boldsymbol{B}_t$ would be prohibitively expensive. LBFGS approximates the quasi-Newton direction directly from the last m pairs of $\boldsymbol{s}_t$ and $\boldsymbol{y}_t$ via a matrix-free approach. This reduces the cost to $O(md)$ space and time per iteration, with m freely chosen (Nocedal and Wright, 1999).

Smoothness of the objective function is essential for standard (L)BFGS because both the local quadratic model (1) and the Wolfe conditions (4) require the existence of the gradient ∇J at every point. Even though nonsmooth convex functions are differentiable everywhere except on a set of Lebesgue measure zero (Hiriart-Urruty and Lemaréchal, 1993), in practice (L)BFGS often fails to converge on such problems (Lukšan and Vlček, 1999; Haarala, 2004). Various subgradient-based approaches, such as subgradient descent (Nedich and Bertsekas, 2000) or bundle methods (Teo et al., 2007), are therefore preferred.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

Although a convex function might not be differentiable everywhere, a subgradient always exists. Let $\boldsymbol{w}$ be a point where a convex function J is finite. Then a subgradient is the normal vector of any tangential supporting hyperplane of J at $\boldsymbol{w}$. Formally, $\boldsymbol{g}$ is called a subgradient of J at $\boldsymbol{w}$ if and only if

$$J(\boldsymbol{w}') \;\geq\; J(\boldsymbol{w}) + (\boldsymbol{w}' - \boldsymbol{w})^\top \boldsymbol{g} \quad \forall \boldsymbol{w}'. \tag{6}$$

The set of all subgradients at a point is called the subdifferential, and is denoted by $\partial J(\boldsymbol{w})$. If this set is not empty then J is said to be *subdifferentiable at* $\boldsymbol{w}$. If it contains exactly one element, *i.e.,* $\partial J(\boldsymbol{w}) = \{\nabla J(\boldsymbol{w})\}$, then J is *differentiable* at $\boldsymbol{w}$.

In this paper we systematically modify the standard (L)BFGS algorithm so as to make it amenable to subgradients. This results in sub(L)BFGS, a new subgradient quasi-Newton method which is applicable to a wide variety of nonsmooth convex optimization problems encountered in machine learning.

In the next section we describe our new algorithm generically, before we discuss its application to L_2-regularized risk minimization with hinge loss in Section 3. Section 4 compares and contrasts our work with other recent efforts in this area. Encouraging experimental results are reported in Section 5. We conclude with an outlook and discussion in Section 6.

2. Subgradient BFGS Method

We modify the standard BFGS algorithm to derive our new algorithm (subBFGS, Algorithm 1) for nonsmooth convex optimization. These modifications can be grouped into three areas, which we elaborate on in turn: generalizing the local quadratic model, finding a descent direction, and finding a step size that obeys a subgradient reformulation of the Wolfe conditions.

2.1. Generalizing the Local Quadratic Model

Recall that BFGS assumes the objective function J is differentiable everywhere, so that at the current iterate $\boldsymbol{w}_t$ we can construct a local quadratic model (1) of $J(\boldsymbol{w}_t)$. For a nonsmooth objective function, such a model becomes ambiguous at non-differentiable points (Figure 1). To resolve the ambiguity, we could simply replace the gradient $\nabla J(\boldsymbol{w}_t)$ in (1) with some subgradient $\boldsymbol{g}_t \in \partial J(\boldsymbol{w}_t)$. However, as will be discussed later, the resulting quasi-Newton direction $\boldsymbol{p}_t := -\boldsymbol{B}_t \boldsymbol{g}_t$ is not necessarily a descent direction. To address this fundamental modeling problem, we first generalize the

Algorithm 1 SUBGRADIENT BFGS (SUBBFGS)

Initialize: $t := 0, \boldsymbol{w}_0 = \boldsymbol{0}, \boldsymbol{B}_0 = \boldsymbol{I}$;
Set direction-finding stopping tolerances $\epsilon, k_{\max} \in \mathbb{R}_+$;
Compute subgradient $\boldsymbol{g}_0 \in \partial J(\boldsymbol{w}_0)$;
while not converged **do**
  $\boldsymbol{p}_t = \texttt{descentDirection}(\boldsymbol{g}_t, \epsilon, k_{\max})$; (Algorithm 2)
  if $\boldsymbol{p}_t =$ failure **then**
    Return $\boldsymbol{w}_t$;
  end if
  Find η_t that obeys (14); (*e.g.*, Algorithm 3)
  $\boldsymbol{s}_t = \eta_t \boldsymbol{p}_t$;
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{s}_t$;
  Compute subgradient $\boldsymbol{g}_{t+1} \in \partial J(\boldsymbol{w}_{t+1})$;
  $\boldsymbol{y}_t = \boldsymbol{g}_{t+1} - \boldsymbol{g}_t$;
  Update $\boldsymbol{B}_{t+1}$ via (5);
  $t := t + 1$;
end while

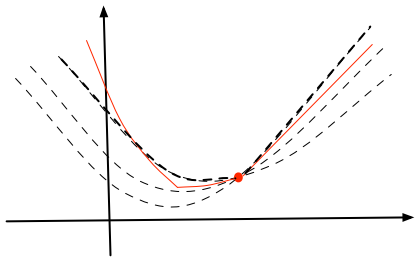


Figure 1. Quadratic models (dashed) *vs.* tightest pseudo-quadratic fit (7) (bold dashes) to the objective function (solid line) at a subdifferentiable point (solid disk).

local quadratic model as follows:

$$\begin{aligned} Q_t(\boldsymbol{p}) \;&:=\; J(\boldsymbol{w}_t) + M_t(\boldsymbol{p}), \quad \text{where} \\ M_t(\boldsymbol{p}) \;&:=\; \tfrac{1}{2}\boldsymbol{p}^\top \boldsymbol{B}_t^{-1} \boldsymbol{p} \;+\; \sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}. \end{aligned} \tag{7}$$

Note that where J is differentiable, (7) reduces to the familiar BFGS quadratic model (1). At non-differentiable points, however, the model is no longer quadratic, as the supremum may be attained at different elements of $\partial J(\boldsymbol{w}_t)$ for different directions $\boldsymbol{p}$. Instead it can be viewed as the tightest pseudo-quadratic fit to J at $\boldsymbol{w}_t$ (Figure 1).

Ideally, we would like to minimize $Q_t(\boldsymbol{p})$, or equivalently $M_t(\boldsymbol{p})$, in (7) to obtain the best search direction,

$$\boldsymbol{p}^* := \operatorname*{arginf}_{\boldsymbol{p} \in \mathbb{R}^d} M_t(\boldsymbol{p}). \tag{8}$$

This is generally intractable due to the presence of a supremum over the entire subdifferential set $\partial J(\boldsymbol{w}_t)$. In many machine learning problems, however, the set $\partial J(\boldsymbol{w}_t)$ has some special structure that simplifies calculation of the supremum in (7). In what follows, we develop an iteration that is guaranteed to find a quasi-Newton descent direction, assuming an oracle that supplies $\operatorname{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}$ for a given direction

Algorithm 2 $\boldsymbol{p}_t = \texttt{descentDirection}(\boldsymbol{g}^{(1)}, \epsilon, k_{\max})$

input subgradient $\boldsymbol{g}^{(1)} \in \partial J(\boldsymbol{w}_t)$,
tolerance $\epsilon \in \mathbb{R}_+$, iteration limit $k_{\max}$;
output descent direction $\boldsymbol{p}_t$;

1: Initialize: $i := 1$, $\bar{\boldsymbol{g}}^{(1)} = \boldsymbol{g}^{(1)}$, $\boldsymbol{p}^{(1)} = -\boldsymbol{B}_t \boldsymbol{g}^{(1)}$;
2: $\boldsymbol{g}^{(2)} = \operatorname{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}^{(1)}$;
3: Calculate $\epsilon^{(1)}$ via (13);
4: **while** $(\boldsymbol{g}^{(i+1)\top} \boldsymbol{p}^{(i)} > 0$ or $\epsilon^{(i)} > \epsilon)$ and $i < k_{\max}$ **do**
5: $\quad \mu^* := \min\left[1, \frac{(\boldsymbol{g}^{(i+1)} - \bar{\boldsymbol{g}}^{(i)})^\top \boldsymbol{p}^{(i)}}{(\boldsymbol{g}^{(i+1)} - \bar{\boldsymbol{g}}^{(i)})^\top \boldsymbol{B}_t (\boldsymbol{g}^{(i+1)} - \bar{\boldsymbol{g}}^{(i)})}\right]$;
6: $\quad \bar{\boldsymbol{g}}^{(i+1)} = (1-\mu^*)\bar{\boldsymbol{g}}^{(i)} + \mu^* \boldsymbol{g}^{(i+1)}$;
7: $\quad \boldsymbol{p}^{(i+1)} = (1-\mu^*)\boldsymbol{p}^{(i)} - \mu^* \boldsymbol{B}_t \boldsymbol{g}^{(i+1)}$;
8: $\quad \boldsymbol{g}^{(i+2)} = \operatorname{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}^{(i+1)}$;
9: $\quad$ Calculate $\epsilon^{(i+1)}$ via (13);
10: $\quad i := i + 1$;
11: **end while**
12: **if** $\boldsymbol{g}^{(i+1)\top} \boldsymbol{p}^{(i)} > 0$ **then**
13: $\quad$ **return** failure;
14: **else**
15: $\quad$ **return** $\operatorname{argmin}_{j \le i} M_t(\boldsymbol{p}^{(j)})$.
16: **end if**

$\boldsymbol{p} \in \mathbb{R}^d$. In Section 3.1 we provide an efficient implementation of such an oracle for L_2-regularized risk minimization with the hinge loss.

2.2. Finding a Descent Direction

A direction $\boldsymbol{p}_t$ is a descent direction if and only if $\boldsymbol{g}^\top \boldsymbol{p}_t < 0 \; \forall \boldsymbol{g} \in \partial J(\boldsymbol{w}_t)$ (Belloni, 2005), or equivalently

$$\sup_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t \; < \; 0. \tag{9}$$

In particular, for a smooth convex function the quasi-Newton direction (2) is always a descent direction because $\nabla J(\boldsymbol{w}_t)^\top \boldsymbol{p}_t = -\nabla J(\boldsymbol{w}_t)^\top \boldsymbol{B}_t \nabla J(\boldsymbol{w}_t) < 0$ holds due to the positivity of $\boldsymbol{B}_t$.

For nonsmooth functions, however, the quasi-Newton direction $\boldsymbol{p}_t := -\boldsymbol{B}_t \boldsymbol{g}_t$ for a given $\boldsymbol{g}_t \in \partial J(\boldsymbol{w}_t)$ may not fulfill the descent condition (9), making it impossible to find a step that obeys (4), thus causing a failure of the line search. We now present an iterative approach to finding a quasi-Newton *descent* direction.

Inspired by bundle methods (Teo et al., 2007), we build the following convex lower bound on $M_t(\boldsymbol{p})$:

$$M_t^{(i)}(\boldsymbol{p}) \; := \; \tfrac{1}{2} \boldsymbol{p}^\top \boldsymbol{B}_t^{-1} \boldsymbol{p} + \sup_{j \le i} \boldsymbol{g}^{(j)\top} \boldsymbol{p}, \tag{10}$$

where $i, j \in \mathbb{N}$. Given a $\boldsymbol{p}^{(i)} \in \mathbb{R}^d$ the lower bound (10) is successively tightened by computing

$$\boldsymbol{g}^{(i+1)} := \operatorname*{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}^{(i)}, \tag{11}$$

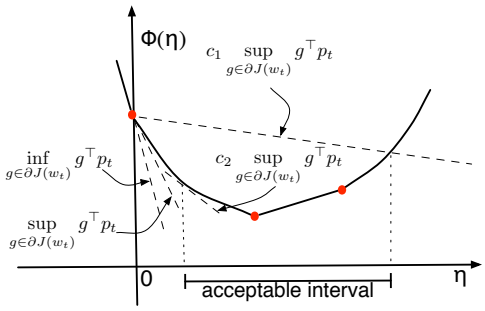


Figure 2. Geometric interpretation of the subgradient Wolfe conditions (14). Solid disks are subdifferentiable points; the slopes of dashed lines are indicated.

such that $M_t^{(i)}(\boldsymbol{p}) \le M_t^{(i+1)}(\boldsymbol{p}) \le M_t(\boldsymbol{p}) \; \forall \boldsymbol{p} \in \mathbb{R}^d$. Here we set $\boldsymbol{g}^{(1)} \in \partial J(\boldsymbol{w}_t)$, and assume that $\boldsymbol{g}^{(i+1)}$ is provided by an oracle. To solve $\inf_{\boldsymbol{p} \in \mathbb{R}^d} M_t^{(i)}(\boldsymbol{p})$, we rewrite it as a constrained optimization problem:

$$\inf_{\boldsymbol{p}, \xi} \left(\tfrac{1}{2} \boldsymbol{p}^\top \boldsymbol{B}_t^{-1} \boldsymbol{p} + \xi\right) \text{ s.t. } \boldsymbol{g}^{(j)\top} \boldsymbol{p} \le \xi \;\; \forall j \le i. \tag{12}$$

This problem can be solved exactly via quadratic programming, but doing so may incur substantial computational expense. Instead we adopt an alternative approach (Algorithm 2) which does not solve $\inf_{\boldsymbol{p} \in \mathbb{R}^d} M_t^{(i)}(\boldsymbol{p})$ to optimality. The key idea is to write the proposed descent direction at iteration $i+1$ as a convex combination of $\boldsymbol{p}^{(i)}$ and $-\boldsymbol{B}_t \boldsymbol{g}^{(i+1)}$. The optimal combination coefficient μ^* can be computed exactly (Step 5 of Algorithm 2) using an argument based on maximizing dual progress. Finally, to derive an implementable stopping criterion, we define $\epsilon^{(i)}$ to be

$$\min_{j \le i} \left[\boldsymbol{p}^{(j)\top} \boldsymbol{g}^{(j+1)} - \tfrac{1}{2}(\boldsymbol{p}^{(j)\top} \bar{\boldsymbol{g}}^{(j)} + \boldsymbol{p}^{(i)\top} \bar{\boldsymbol{g}}^{(i)})\right], \tag{13}$$

where $\bar{\boldsymbol{g}}^{(i)}$ is an aggregated subgradient (Step 6 of Algorithm 2) which lies in the convex hull of $\boldsymbol{g}^{(j)} \in \partial J(\boldsymbol{w}_t) \; \forall j \le i$. $\epsilon^{(i)}$ is monotonically decreasing, and upper bounds the distance from the optimal value of the dual of $M_t(\boldsymbol{p})$, leading us to a practical stopping criterion (Step 4 of Algorithm 2) for our direction-finding procedure. Yu et al. (2008) provide details, and prove that Algorithm 2 converges to the optimal dual objective value with precision ϵ at an $O(1/\epsilon)$ rate.

2.3. Subgradient Line Search

Given the current iterate $\boldsymbol{w}_t$ and a search direction $\boldsymbol{p}_t$, the task of a line search is to find a step size $\eta \in \mathbb{R}_+$ which decreases the objective function along the line $\boldsymbol{w}_t + \eta \boldsymbol{p}_t$, *i.e.*, $J(\boldsymbol{w}_t + \eta \boldsymbol{p}_t) =: \Phi(\eta)$. The Wolfe conditions (4) are used in line search routines to enforce a sufficient decrease in the objective value, and

to exclude unnecessarily small step sizes (Nocedal and Wright, 1999). However, the original Wolfe conditions require the objective function to be smooth. To extend them to nonsmooth convex problems, we propose the following subgradient reformulation:

$$J(\boldsymbol{w}_{t+1}) \;\leq\; J(\boldsymbol{w}_t) \;+\; c_1\eta_t \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t$$
$$\text{and} \sup_{\boldsymbol{g}'\in\partial J(\boldsymbol{w}_{t+1})} \boldsymbol{g}'^\top \boldsymbol{p}_t \;\geq\; c_2 \sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}_t, \qquad (14)$$

where $0 < c_1 < c_2 < 1$. Figure 2 illustrates how these conditions enforce acceptance of non-trivial step sizes that decrease the objective value. Yu et al. (2008) formally show that for any given descent direction we can always find a positive step size that satisfies (14).

2.4. Limited-Memory Subgradient BFGS

It is straightforward to implement an LBFGS variant of our subBFGS algorithm: We simply modify Algorithms 1 and 2 to compute all products of $\boldsymbol{B}_t$ with a vector by means of the standard LBFGS matrix-free scheme (Nocedal and Wright, 1999).

3. sub(L)BFGS Implementation for L_2-Regularized Risk Minimization

Many machine learning algorithms can be viewed as minimizing the L_2-regularized risk

$$J(\boldsymbol{w}) \;:=\; \frac{c}{2}\|\boldsymbol{w}\|^2 \;+\; \frac{1}{n}\sum_{i=1}^{n} l(\boldsymbol{w}^\top \boldsymbol{x}_i, z_i), \qquad (15)$$

where $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ are the training instances, $z_i \in \mathcal{Z} \subseteq \mathbb{R}$ the corresponding labels, and the loss l is a non-negative convex function of $\boldsymbol{w}$ which measures the discrepancy between z_i and the predictions arising from $\boldsymbol{w}$ via $\boldsymbol{w}^\top \boldsymbol{x}_i$. A loss function commonly used for binary classification is the hinge loss

$$l(\boldsymbol{w}^\top \boldsymbol{x}, z) \;:=\; \max(0, 1 - z\,\boldsymbol{w}^\top \boldsymbol{x}), \qquad (16)$$

where $z \in \{\pm 1\}$. L_2-regularized risk minimization with binary hinge loss is a convex but nonsmooth optimization problem; in this section we show how sub(L)BFGS (Algorithm 1) can be applied to it.

Differentiating (15) after plugging in (16) yields

$$\partial J(\boldsymbol{w}) = c\,\boldsymbol{w} - \frac{1}{n}\sum_{i=1}^{n} \beta_i z_i \boldsymbol{x}_i = \bar{\boldsymbol{w}} - \frac{1}{n}\sum_{i\in\mathcal{M}} \beta_i z_i \boldsymbol{x}_i, \qquad (17)$$

where $\bar{\boldsymbol{w}} := c\,\boldsymbol{w} - \frac{1}{n}\sum_{i\in\mathcal{E}} z_i \boldsymbol{x}_i$ and

$$\beta_i := \begin{cases} 1 & \text{if } i \in \mathcal{E}, \quad \mathcal{E} := \{i : 1 - z_i \boldsymbol{w}^\top \boldsymbol{x}_i > 0\}, \\ [0,1] & \text{if } i \in \mathcal{M}, \quad \mathcal{M} := \{i : 1 - z_i \boldsymbol{w}^\top \boldsymbol{x}_i = 0\}, \\ 0 & \text{if } i \in \mathcal{W}, \quad \mathcal{W} := \{i : 1 - z_i \boldsymbol{w}^\top \boldsymbol{x}_i < 0\}. \end{cases}$$

$\mathcal{E}$, $\mathcal{M}$, and $\mathcal{W}$ index the set of points which are in error, on the margin, and well-classified, respectively.

3.1. Realizing the Direction-Finding Method

Recall that our sub(L)BFGS algorithm requires an oracle that provides $\operatorname{argsup}_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}$ for a given direction $\boldsymbol{p}$. For L_2-regularized risk minimization with binary hinge loss we can implement such an oracle at computational cost linear in the number $|\,\mathcal{M}_t\,|$ of current marginal points. (Normally $|\,\mathcal{M}_t\,| \ll n$.) Towards this end we use (17) to obtain

$$\sup_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p} \;=\; \sup_{\beta_i, i\in\mathcal{M}_t} \left(\bar{\boldsymbol{w}}_t - \frac{1}{n}\sum_{i\in\mathcal{M}_t} \beta_i z_i \boldsymbol{x}_i\right)^\top \boldsymbol{p}$$
$$= \;\bar{\boldsymbol{w}}_t^\top \boldsymbol{p} - \frac{1}{n}\sum_{i\in\mathcal{M}_t} \inf_{\beta_i\in[0,1]} \beta_i z_i \boldsymbol{x}_i^\top \boldsymbol{p}. \qquad (18)$$

Since for a given $\boldsymbol{p}$ the first term of the right-hand side of (18) is a constant, the supremum is attained when we set $\beta_i\ \forall i \in \mathcal{M}_t$ via the following strategy:

$$\beta_i := \begin{cases} 0 & \text{if } z_i \boldsymbol{x}_i^\top \boldsymbol{p}_t \geq 0, \\ 1 & \text{if } z_i \boldsymbol{x}_i^\top \boldsymbol{p}_t < 0. \end{cases} \qquad (19)$$

3.2. Implementing the Line Search

The one-dimensional convex function Φ obtained by restricting (15) to a line can be evaluated efficiently. To see this, rewrite the objective as

$$J(\boldsymbol{w}) \;:=\; \frac{c}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\mathbf{1}^\top \max(\mathbf{0}, \mathbf{1} - \boldsymbol{z}\cdot\boldsymbol{X}\boldsymbol{w}), \qquad (20)$$

where $\mathbf{0}$ and $\mathbf{1}$ are column vectors of zeros and ones, respectively, $\cdot$ denotes the Hadamard (componentwise) product, and $\boldsymbol{z} \in \mathbb{R}^n$ collects correct labels corresponding to each row of data in $\boldsymbol{X} := [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n\times d}$. Given a search direction $\boldsymbol{p}_t$ at an iterate $\boldsymbol{w}_t$, this allows us to write

$$\Phi(\eta) := J(\boldsymbol{w}_t + \eta\boldsymbol{p}_t) = \frac{1}{n}\boldsymbol{\delta}(\eta)^\top[\mathbf{1} - (\boldsymbol{f} + \eta\Delta\boldsymbol{f})] \qquad (21)$$
$$+ \frac{c}{2}\|\boldsymbol{w}_t\|^2 + c\eta\boldsymbol{w}_t^\top \boldsymbol{p}_t + \frac{c\eta^2}{2}\|\boldsymbol{p}_t\|^2$$

where $\boldsymbol{f} := \boldsymbol{z}\cdot\boldsymbol{X}\boldsymbol{w}_t$, $\Delta\boldsymbol{f} := \boldsymbol{z}\cdot\boldsymbol{X}\boldsymbol{p}_t$, and

$$\delta_i(\eta) := \begin{cases} 1 & \text{if } f_i + \eta\,\Delta f_i < 1, \\ [0,1] & \text{if } f_i + \eta\,\Delta f_i = 1, \\ 0 & \text{if } f_i + \eta\,\Delta f_i > 1 \end{cases} \qquad (22)$$

for $1 \leq i \leq n$. We cache $\boldsymbol{f}$ and $\Delta\boldsymbol{f}$, expending $O(nd)$ computational effort. We also cache $\frac{c}{2}\|\boldsymbol{w}_t\|^2$, $c\boldsymbol{w}_t^\top \boldsymbol{p}_t$, and $\frac{c}{2}\|\boldsymbol{p}_t\|^2$, each of which requires $O(n)$ work. The

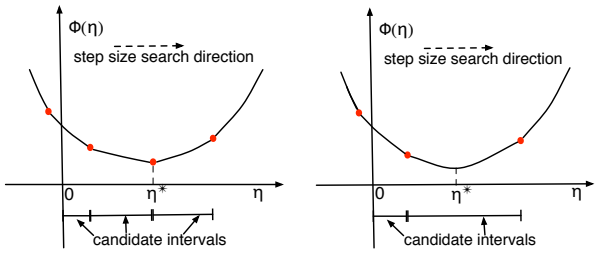


Figure 3. Nonsmooth convex function Φ of step size η. Solid disks are subdifferentiable points; the optimal η^* falls on such a point (left), or between two such points (right).

evaluation of $\boldsymbol{\delta}(\eta)$ and its inner product with $\mathbf{1} - (\boldsymbol{f} + \eta\,\Delta\boldsymbol{f})$ both take $O(n)$ effort. All other terms in (21) can be computed in constant time, thus reducing the amortized cost of evaluating $\Phi(\eta)$ to $O(n)$. We are now in a position to introduce an exact line search which takes advantage of this scheme.

3.2.1. Exact Line Search

Differentiating (21) with respect to η and setting the gradient to zero shows that $\eta^* := \operatorname{argmin}_\eta \Phi(\eta)$ satisfies $\eta^* = (\boldsymbol{\delta}(\eta^*)^\top \Delta\boldsymbol{f}/n - c\,\boldsymbol{w}_t^\top \boldsymbol{p}_t)/(c\,\|\boldsymbol{p}_t\|^2)$. It is easy to verify that $\Phi(\eta)$ is piecewise quadratic, and differentiable everywhere except at $\eta_i := (1-f_i)/\Delta f_i$, where it becomes subdifferentiable. At these points an element of the indicator function $\boldsymbol{\delta}(\eta)$ (22) changes from 0 to 1 or vice versa; otherwise $\boldsymbol{\delta}(\eta)$ remains constant. Thus for a smooth interval (η_a, η_b) between subdifferentiable points η_a and η_b (*cf.* Figure 3), if the candidate step

$$\eta^*_{a,b} = \frac{\boldsymbol{\delta}(\eta')^\top \Delta\boldsymbol{f}/n - c\,\boldsymbol{w}_t^\top \boldsymbol{p}_t}{c\,\|\boldsymbol{p}_t\|^2}, \quad \eta' \in (\eta_a, \eta_b) \qquad (23)$$

lies in the interval, it is optimal: $\eta^*_{a,b} \in [\eta_a, \eta_b] \Rightarrow \eta^* = \eta^*_{a,b}$. Otherwise, the interval boundary is optimal if its subdifferential contains zero: $0 \in \partial\,\Phi(\eta_a) \Rightarrow \eta^* = \eta_a$. Sorting the subdifferentiable points η_i facilitates efficient search over intervals; see Algorithm 3 for details.

Algorithm 3 $\eta = \texttt{linesearch}(\boldsymbol{w}_t, \boldsymbol{p}_t, c, \boldsymbol{f}, \Delta\boldsymbol{f})$

input $\boldsymbol{w}_t, \boldsymbol{p}_t, c, \boldsymbol{f}$, and $\Delta\boldsymbol{f}$ as in (21);
output step size η;

$b = c\,\boldsymbol{w}_t^\top \boldsymbol{p}_t$, $h = c\,\|\boldsymbol{p}_t\|^2$;
$n = \texttt{length}(\boldsymbol{f})$, $j := 1$;
$\boldsymbol{\alpha} := [(\mathbf{1} - \boldsymbol{f})/\Delta\boldsymbol{f}, 0]$; (subdifferentiable points)
$\boldsymbol{\pi} = \texttt{sort}(\boldsymbol{\alpha})$; (index vector)
while $\alpha_{\pi_j} \le 0$ **do**
  $j := j + 1$;
end while
$\eta := \alpha_{\pi_j}/2$;
for $i := 1$ to n **do**
  $\delta_i := \begin{cases} 1 & \text{if } f_i + \eta\,\Delta f_i < 1; \\ 0 & \text{otherwise;} \end{cases}$
end for
$\varrho := \boldsymbol{\delta}^\top \Delta\boldsymbol{f}/n$;
while $j \le \texttt{length}(\boldsymbol{\pi})$ **do**
  $\eta := (\varrho - b)/h$; (candidate step)
  if $\eta \in [\alpha_{\pi_{j-1}}, \alpha_{\pi_j}]$ **then**
    return η;
  else if $\eta < \alpha_{\pi_{j-1}}$ **then**
    return $\eta := \alpha_{\pi_{j-1}}$;
  else
    repeat
      $\varrho := \begin{cases} \varrho - \Delta f_{\pi_j}/n & \text{if } \delta_{\pi_j} = 1, \\ \varrho + \Delta f_{\pi_j}/n & \text{otherwise;} \end{cases}$
      $j := j + 1$;
    until $\alpha_{\pi_j} \ne \alpha_{\pi_{j-1}}$
  end if
end while

4. Related Work

Lukšan and Vlček (1999) propose an extension of BFGS to nonsmooth convex problems. Their algorithm samples gradients around non-differentiable points in order to obtain a descent direction. In many machine learning problems evaluating the objective function and its gradient is very expensive. Therefore, our direction-finding algorithm (Algorithm 2) repeatedly samples subgradients from the set $\partial J(\boldsymbol{w})$ via the oracle, which is computationally more efficient.

Recently, Andrew and Gao (2007) introduced a variant of nonsmooth BFGS, the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) algorithm, suitable for optimizing L_1-regularized log-linear models:

$$J(\boldsymbol{w}) := c\|\boldsymbol{w}\|_1 + \frac{1}{n}\sum_{i=1}^{n} \ln(1 + e^{-z_i \boldsymbol{w}^\top \boldsymbol{x}_i}), \qquad (24)$$

where the logistic loss is smooth, but the regularizer is only subdifferentiable at points where $\boldsymbol{w}$ has zero elements. From the optimization viewpoint this objective is very similar to the L_2-regularized hinge loss; the direction finding and line search methods that we discussed in Sections 3.1 and 3.2, respectively, can be applied to this problem with slight modifications.

OWL-QN is based on the observation that the L_1 regularizer is linear within any given orthant. Therefore, it maintains an approximation $\boldsymbol{B}^{\text{ow}}$ to the inverse Hessian of the logistic loss, and uses an efficient scheme to select orthants for optimization. In fact, its success greatly depends on its direction-finding subroutine, which demands a specially chosen subgradient $\boldsymbol{g}^{\text{ow}}$ (Andrew and Gao, 2007, Equation 4) to produce the quasi-Newton direction, $\boldsymbol{p}^{\text{ow}} = \pi(\boldsymbol{p}, \boldsymbol{g}^{\text{ow}})$, where $\boldsymbol{p} := -\boldsymbol{B}^{\text{ow}}\boldsymbol{g}^{\text{ow}}$ and the projection π returns a search direction by setting the i^{th} element of $\boldsymbol{p}$ to zero whenever $p_i g_i^{\text{ow}} > 0$. As shown in Section 5, the direction-

Table 1. Datasets, regularization constants c, direction-finding convergence criterion ϵ, and the overall number k of direction-finding iterations for L_1-regularized logistic loss and L_2-regularized hinge loss minimization tasks, respectively.

Dataset	Tr./Test Data	Dimen.	Density	c_{L_1}	ϵ_{L_1}	k_{L_1}	$k_{L_1\text{rand}}$	c_{L_2}	ϵ_{L_2}	k_{L_2}
Covertype	522911/58101	54	22.22%	10^{-6}	10^{-5}	0	0	10^{-6}	10^{-8}	44
CCAT	781265/23149	47236	0.16%	10^{-6}	10^{-5}	356	467	10^{-4}	10^{-8}	66
Astro	29882/32487	99757	0.077%	10^{-5}	10^{-3}	1668	2840	$5 \cdot 10^{-5}$	10^{-8}	17
MNIST	60000/10000	780	19.22%	10^{-4}	10^{-5}	60	102	$1.4286 \cdot 10^{-6}$	10^{-8}	244

finding subroutine of OWL-QN can be replaced by Algorithm 2, which in turn makes the algorithm more robust to the choice of subgradients.

Many optimization techniques use past gradients to build a model of the objective function. Bundle method solvers like SVMStruct (Joachims, 2006) and BMRM (Teo et al., 2007) use them to lower-bound the objective by a piecewise linear function which is minimized to obtain the next iterate. This fundamentally differs from the BFGS approach of using past gradients to approximate the (inverse) Hessian, hence building a quadratic model of the objective function.

Vojtěch and Sonnenburg (2007) speed up the convergence of a bundle method solver for the L_2-regularized binary hinge loss. Their main idea is to perform a line search along the line connecting two successive iterates of a bundle method solver. Although developed independently, their line search algorithm is very reminiscent of the method we describe in Section 3.2.1.

5. Experiments

We now evaluate the performance of our subLBFGS algorithm, and compare it to other state-of-the-art nonsmooth optimization methods on L_2-regularized hinge loss minimization. We also compare a variant of OWL-QN that uses our direction-finding routine to the original on L_1-regularized logistic loss minimization.

Our experiments used four datasets: the Covertype dataset of Blackard, Jock & Dean, CCAT from the Reuters RCV1 collection, the Astro-physics dataset of abstracts of scientific papers from the Physics ArXiv (Joachims, 2006), and the MNIST dataset of handwritten digits with two classes: even and odd digits. We used subLBFGS with a buffer of size $m = 15$ throughout. Table 1 summarizes our parameter settings, and reports the overall number of direction-finding iterations for all experiments. We followed the choices of Vojtěch and Sonnenburg (2007) for the L_2 regularization constants; for L_1 they were chosen from the set $10^{\{-6,-5,\cdots,-1\}}$ to achieve the lowest test error.

On convex problems such as these every convergent optimizer will reach the same solution; comparing generalisation performance is therefore pointless. We combined training and test datasets to evaluate the convergence of each algorithm in terms of the objective function value *vs.* CPU seconds. All experiments were carried out on a Linux machine with dual 2.8 GHz Xeon processors with 4GB RAM.

5.1. L_2-Regularized Hinge Loss

For our first set of experiments, we applied subLBFGS together with our exact line search (Algorithm 3) to the task of L_2-regularized hinge loss minimization. Our control methods are the bundle method solver BMRM (Teo et al., 2007) and an optimized cutting plane algorithm, OCAS version 0.6.0 (Vojtěch and Sonnenburg, 2007), both of which demonstrated strong results on the L_2-regularized hinge loss minimization in their corresponding papers.

Figure 4 shows that subLBFGS (solid) reaches the neighbourhood of the optimum (less than 10^{-3} away) noticeably (up to 7 times) faster than BMRM (dashed). As BMRM's approximation to the objective function improves over the course of optimization, it gradually catches up with subLBFGS, though ultimately subLBFGS still converges faster on 3 out of 4 datasets. The performance of subLBFGS and OCAS (dash-dotted) are very similar: OCAS converges slightly faster than subLBFGS on the Astrophysics dataset but is outperformed by subLBFGS on the MNIST dataset.

5.2. L_1-Regularized Logistic Loss

To demonstrate the utility of our direction-finding routine (Algorithm 2) in its own right, we plugged it into the OWL-QN algorithm (Andrew and Gao, 2007) as an alternative direction-finding method, such that $\boldsymbol{p}^{\text{ow}} = \texttt{descentDirection}(\boldsymbol{g}^{\text{ow}}, \epsilon, k_{\max})$,[1] and compared this variant (denoted by OWL-QN*) with the original on L_1-regularized logistic loss minimization.

Using the stopping criterion suggested by Andrew and Gao (2007), we run experiments until the averaged rel-

[1] Note for the objective (24) it is trivial to construct an oracle that supplies $\text{argsup}_{\boldsymbol{g} \in \partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}$.

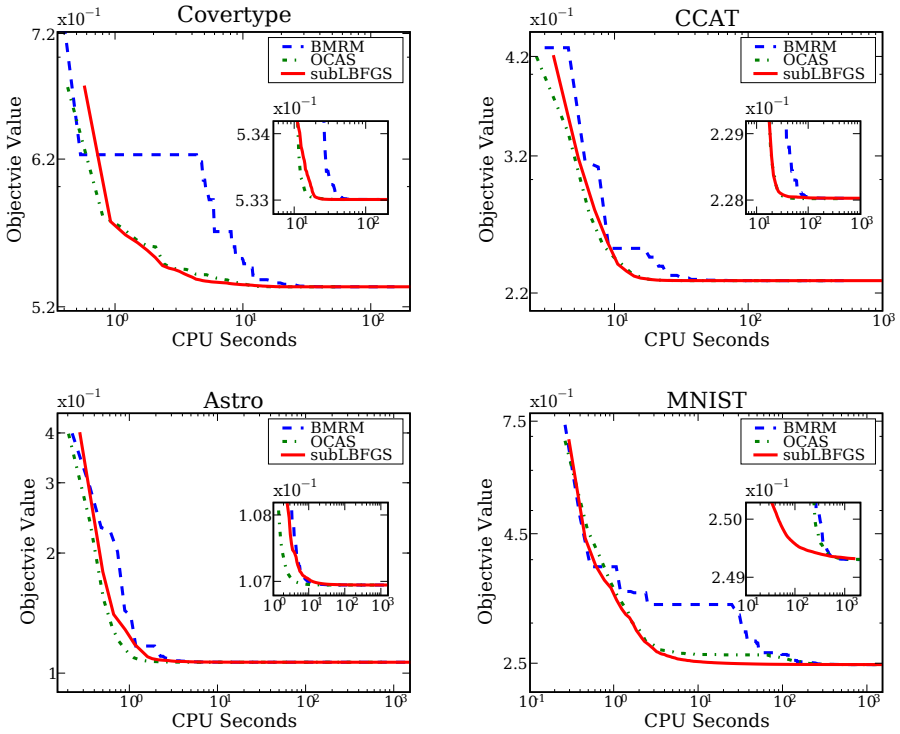


Figure 4. Objective function value *vs.* CPU seconds on L_2-regularized hinge loss minimization tasks.

ative change in the objective value over the previous 5 iterations falls below 10^{-5}. Figure 5 shows only minor differences in convergence between the two algorithms.

To examine the algorithms' sensitivity to the choice of subgradients, we also ran them with random subgradients (as opposed to the specially chosen $\boldsymbol{g}^{\text{ow}}$ used before) fed to their corresponding direction-finding routines. OWL-QN relies heavily on its particular choice of subgradients, hence breaks down completely under these conditions: The only dataset where we could even plot its (poor) performance was Covertype (dotted OWL-QN(2) line in Figure 5). Our direction-finding routine, by contrast, is self-correcting and thus not affected by this manipulation: The curves for OWL-QN*(2) (plotted for Covertype in Figure 5) lie virtually on top of those for OWL-QN*. Table 1 shows that in this case more direction-finding iterations are needed, *i.e.*, $k_{L_1\text{rand}} \geq k_{L_1}$. This empirically confirms that as long as $\operatorname{argsup}_{\boldsymbol{g}\in\partial J(\boldsymbol{w}_t)} \boldsymbol{g}^\top \boldsymbol{p}$ is given, Algorithm 2 can indeed be used as a canned quasi-Newton direction-finding routine.

6. Outlook and Discussion

We proposed an extension of BFGS suitable for handling nonsmooth problems often encountered in the machine learning context. As our experiments show, our algorithms are versatile and applicable to many problems, while their performance is comparable to if not better than that of their counterparts in custom-built solvers.

In some experiments we observe that subLBFGS initially makes rapid progress towards the solution but slows down closer to the optimum. We hypothesize that initially its quadratic model allows subLBFGS to make rapid progress, but closer to the optimum it is no longer an accurate model of an objective function dominated by the nonsmooth hinges. We are therefore contemplating hybrid solvers which seamlessly switch between sub(L)BFGS and bundle solvers.

In this paper we applied subLBFGS to L_2-regularized risk minimization with binary hinge loss. It can also be extended to deal with generalizations of the hinge loss, such as multi-class, multi-category, and ordinal regression problems; this is part of our ongoing research.

Finally, to put our contributions in perspective, recall that we modified three aspects of the standard BFGS algorithm, namely the quadratic model (Section 2.1), the descent direction finding (Section 2.2), and the line search (Section 2.3). Each of these modifications is versatile enough to be used as a component in other nonsmooth optimization algorithms. This not only offers the promise of improving existing algorithms, but

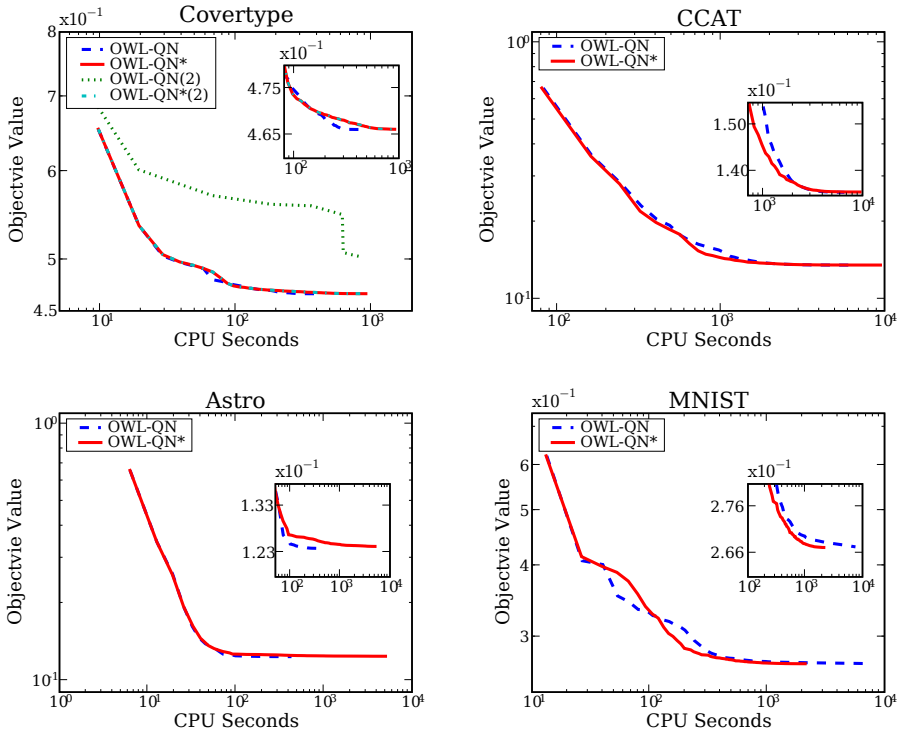


Figure 5. Objective function value *vs.* CPU seconds on L_1-regularized logistic loss minimization tasks.

may also help clarify connections between them. We hope that this will focus attention on those core subroutines that need to be made more efficient in order to handle larger and larger datasets.

Acknowledgement

NICTA is funded by the Australian Government's Backing Australia's Ability and the Centre of Excellence programs. This work is also supported by the IST Program of the European Community, under the FP7 Network of Excellence, ICT-216886-NOE.

References

G. Andrew and J. Gao. Scalable training of l_1-regularized log-linear models. In *Proc. Intl. Conf. Machine Learning*, pages 33–40, New York, NY, USA, 2007. ACM.

A. Belloni. Introduction to bundle methods. Technical report, Operation Research Center, M.I.T., 2005.

M. Haarala. *Large-Scale Nonsmooth Optimization.* PhD thesis, University of Jyväskylä, 2004.

J. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms, I and II*, volume 305 and 306. Springer-Verlag, 1993.

T. Joachims. Training linear SVMs in linear time. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD)*. ACM, 2006.

L. Lukšan and J. Vlček. Globally convergent variable metric method for convex nonsmooth unconstrained minimization. *Journal of Optimization Theory and Applications*, 102(3):593–613, 1999.

A. Nedich and D. P. Bertsekas. Convergence rate of incremental subgradient algorithms. In S. Uryasev and P. M. Pardalos, editors, *Stochastic Optimization: Algorithms and Applications*, pages 263–304. Kluwer Academic Publishers, 2000.

J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer, 1999.

C. Teo, Q. Le, A. Smola, and S. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (KDD)*. ACM, 2007.

F. Vojtěch and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. Technical Report 1, Fraunhofer Institute FIRST, December 2007. `http://publica.fraunhofer.de/documents/N-66225.html`.

J. Yu, S. V. N. Vishwanathan, S. Günter, and N. N. Schraudolph. A quasi-Newton approach to nonsmooth convex optimization. Technical Report arXiv:0804.3835, April 2008. `http://arxiv.org/pdf/0804.3835`.

Predicting Diverse Subsets Using Structural SVMs

Yisong Yue YYUE@CS.CORNELL.EDU
Thorsten Joachims TJ@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

In many retrieval tasks, one important goal involves retrieving a diverse set of results (e.g., documents covering a wide range of topics for a search query). First of all, this reduces redundancy, effectively showing more information with the presented results. Secondly, queries are often ambiguous at some level. For example, the query "Jaguar" can refer to many different topics (such as the car or feline). A set of documents with high topic diversity ensures that fewer users abandon the query because no results are relevant to them. Unlike existing approaches to learning retrieval functions, we present a method that explicitly trains to diversify results. In particular, we formulate the learning problem of predicting diverse subsets and derive a training method based on structural SVMs.

1. Introduction

State of the art information retrieval systems commonly use machine learning techniques to learn ranking functions (Burges et al., 2006; Chapelle et al., 2007). Existing machine learning approaches typically optimize for ranking performance measures such as mean average precision or normalized discounted cumulative gain. Unfortunately, these approaches do not consider diversity, and also (often implicitly) assume that a document's relevance can be evaluated independently from other documents.

Indeed, several recent studies in information retrieval have emphasized the need to optimize for diversity (Zhai et al., 2003; Carbonell & Goldstein, 1998; Chen & Karger, 2006; Zhang et al., 2005; Swaminathan et al., 2008). In particular, they stressed the need to model inter-document dependencies. However, none of these approaches addressed the learning problem, and thus either use a limited feature space or require extensive tuning for different retrieval settings. In contrast, we present a method which can automatically learn a good retrieval function using a rich feature space.

In this paper we formulate the task of diversified retrieval as the problem of predicting diverse subsets. Specifically, we formulate a discriminant based on maximizing word coverage, and perform training using the structural SVM framework (Tsochantaridis et al., 2005). For our experiments, diversity is measured using subtopic coverage on manually labeled data. However, our approach can incorporate other forms of training data such as clickthrough results. To the best of our knowledge, our method is the first approach that can directly train for subtopic diversity. We have also made available a publicly downloadable implementation of our algorithm[1].

For the rest of this paper, we first provide a brief survey of recent related work. We then present our model and describe the prediction and training algorithms. We finish by presenting experiments on labeled query data from the TREC 6-8 Interactive Track as well as a synthetic dataset. Our method compares favorably to conventional methods which do not perform learning.

2. Related Work

Our prediction method is most closely related to the Essential Pages method (Swaminathan et al., 2008), since both methods select documents to maximize weighted word coverage. Documents are iteratively selected to maximize the marginal gain, which is also similar to approaches considered by (Zhai et al., 2003; Carbonell & Goldstein, 1998; Chen & Karger, 2006; Zhang et al., 2005). However, none of these previous approaches addressed the learning problem.

Learning to rank is a well-studied problem in machine learning. Existing approaches typically consider the one-dimensional ranking problem, e.g., (Burges et al.,

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

[1] `http://projects.yisongyue.com/svmdiv/`

2006; Yue et al., 2007; Chapelle et al., 2007; Zheng et al., 2007; Li et al., 2007). These approaches maximize commonly used measures such as mean average precision and normalized discounted cumulative gain, and generalize well to new queries. However, diversity is not considered. These approaches also evaluate each document independently of other documents.

From an online learning approach, Kleinberg et al. (2008) used a multi-armed bandit method to minimize abandonment (maximizing clickthrough) for a single query. While abandonment is provably minimized, their approach cannot generalize to new queries.

The diversity problem can also be treated as learning preferences for sets, which is the approach taken by the DD-PREF modeling language (desJardins et al., 2006; Wagstaff et al., 2007). In their case, diversity is measured on a per feature basis. Since subtopics cannot be treated as features (it is only given in the training data), their method cannot be directly applied to maximizing subtopic diversity. Our model does not need to derive diversity directly from individual features, but does require richer forms of training data (i.e., subtopics explicitly labeled).

Another approach uses a global class hierarchy over queries and/or documents, which can be leveraged to classify new documents and queries (Cai & Hofmann, 2004; Broder et al., 2007). While previous studies on hierarchical classification did not focus on diversity, one might consider diversity by mapping subtopics onto the class hierarchy. However, it is difficult for such hierarchies to achieve the granularity required to measure diversity for individual queries (see beginning of Section 6 for a description of subtopics used in our experiments). Using a large global hierarchy also introduces other complications such as how to generate a comprehensive set of topics and how to assign documents to topics. It seems more efficient to collect labeled training data containing query-specific subtopics (e.g., TREC Interactive Track).

3. The Learning Problem

For each query, we assume that we are given a set of candidate documents $\mathbf{x} = \{x_1, \ldots, x_n\}$. In order to measure diversity, we assume that each query spans a set of topics (which may be distinct to that query). We define $\mathbb{T} = \{T_1, \ldots, T_n\}$, where topic set T_i contains the subtopics covered by document $x_i \in \mathbf{x}$. Topic sets may overlap. Our goal is to select a subset $\mathbf{y}$ of K documents from $\mathbf{x}$ which maximizes topic coverage.

If the topic sets $\mathbb{T}$ were known, a good solution could be computed via straightforward greedy subset selection, which has a $(1 - 1/e)$-approximation bound (Khuller et al., 1997). Finding the globally optimal subset takes n choose K time, which we consider intractable for even reasonably small values of K. However, the topic sets of a candidate set are not known, nor is the set of all possible topics known. We merely assume to have a set of training examples of the form $(\mathbf{x}^{(i)}, \mathbb{T}^{(i)})$, and must find a good function for predicting $\mathbf{y}$ in the absence of $\mathbb{T}$. This in essence is the learning problem.

Let $\mathcal{X}$ denote the space of possible candidate sets $\mathbf{x}$, $\mathcal{T}$ the space of topic sets $\mathbb{T}$, and $\mathcal{Y}$ the space of predicted subsets $\mathbf{y}$. Following the standard machine learning setup, we formulate our task as learning a hypothesis function $h : \mathcal{X} \rightarrow \mathcal{Y}$ to predict a $\mathbf{y}$ when given $\mathbf{x}$. We quantify the quality of a prediction by considering a loss function $\Delta : \mathcal{T} \times \mathcal{Y} \rightarrow \Re$ which measures the penalty of choosing $\mathbf{y}$ when the topics to be covered are those in $\mathbb{T}$.

We restrict ourselves to the supervised learning scenario, where training examples $(\mathbf{x}, \mathbb{T})$ consist of both the candidate set of documents and the subtopics. Given a set of training examples, $S = \{(\mathbf{x}^{(i)}, \mathbb{T}^{(i)}) \in \mathcal{X} \times \mathcal{T} : i = 1, \ldots, N\}$, the strategy is to find a function h which minimizes the empirical risk,

$$R_S^{\Delta}(h) = \frac{1}{N} \sum_{i=1}^{N} \Delta(\mathbb{T}^{(i)}, h(\mathbf{x}^{(i)})).$$

We encourage diversity by defining our loss function $\Delta(\mathbb{T}, \mathbf{y})$ to be the weighted percentage of distinct subtopics in $\mathbb{T}$ not covered by $\mathbf{y}$, although other formulations are possible, which we discuss in Section 8.

We focus on hypothesis functions which are parameterized by a weight vector $\mathbf{w}$, and thus wish to find $\mathbf{w}$ to minimize the empirical risk, $R_S^{\Delta}(\mathbf{w}) \equiv R_S^{\Delta}(h(\cdot; \mathbf{w}))$. We use a discriminant $\mathcal{F} : \mathcal{X} \times \mathcal{Y} \rightarrow \Re$ to compute how well predicting $\mathbf{y}$ fits for $\mathbf{x}$. The hypothesis then predicts the $\mathbf{y}$ which maximizes $\mathcal{F}$:

$$h(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathcal{F}(\mathbf{x}, \mathbf{y}; \mathbf{w}). \quad (1)$$

We assume our discriminant to be linear in a joint feature space $\Psi : \mathcal{X} \times \mathcal{Y} \rightarrow \Re^m$, which we can write as

$$\mathcal{F}(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y}). \quad (2)$$

The feature representation Ψ must enable meaningful discrimination between high quality and low quality predictions. As such, different feature representations may be appropriate for different retrieval settings. We discuss some possible extensions in Section 8.

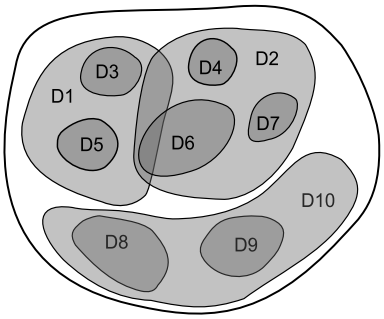


Figure 1. Visualization of Documents Covering Subtopics

This word appears ...
... in a document in $\mathbf{y}$.
... at least 5 times in a document in $\mathbf{y}$.
... with frequency at least 5% in a document in $\mathbf{y}$.
... in the title of a document in $\mathbf{y}$.
... within the top 5 TFIDF of a document in $\mathbf{y}$.

Table 1. Examples of Importance Criteria

The word v has ...
... a $\lvert D_1(v)\rvert/n$ ratio of at least 40%
... a $\lvert D_2(v)\rvert/n$ ratio of at least 50%
... a $\lvert D_\ell(v)\rvert/n$ ratio of at least 25%

Table 2. Examples of Document Frequency Features

4. Maximizing Word Coverage

Figure 1 depicts an abstract visualization of our prediction problem. The sets represent candidate documents $\mathbf{x}$ of a query, and the area covered by each set is the "information" (represented as subtopics $\mathbb{T}$) covered by that document. If $\mathbb{T}$ were known, we could use a greedy method to find a solution with high subtopic diversity. For $K = 3$, the optimal solution in Figure 1 is $\mathbf{y} = \{D1, D2, D10\}$. In general however, the subtopics are unknown. We instead assume that the candidate set contains discriminating features which separates subtopics from each other, and these are primarily based on word frequencies.

As a proxy for explicitly covering subtopics, we formulate our discriminant Ψ based on weighted word coverage. Intuitively, covering more (distinct) words should result in covering more subtopics. The relative importance of covering any word can be modeled using features describing various aspects of word frequencies within documents in $\mathbf{x}$. We make no claims regarding any generative models relating topics to words, but rather simply assume that word frequency features are highly discriminative of subtopics within $\mathbf{x}$.

We now present a simple example of Ψ from (2). Let $V(\mathbf{y})$ denote the union of words contained in the documents of the predicted subset $\mathbf{y}$, and let $\phi(v, \mathbf{x})$ denote the feature vector describing the frequency of word v amongst documents in $\mathbf{x}$. We then write Ψ as

$$\Psi(\mathbf{x}, \mathbf{y}) = \sum_{v \in V(\mathbf{y})} \phi(v, \mathbf{x}). \tag{3}$$

Given a model vector $\mathbf{w}$, the benefit of covering word v in candidate set x is $\mathbf{w}^T \phi(v, \mathbf{x})$. This benefit is realized when a document in $\mathbf{y}$ contains v, i.e., $v \in V(\mathbf{y})$. We use the same model weights for all words. A prediction is made by choosing $\mathbf{y}$ to maximize (2).

This formulation yields two properties which enable optimizing for diversity. First, covering a word twice provides no additional benefit. Second, the feature vector $\phi(v, \mathbf{x})$ is computed using other documents in the candidate set. Thus, diversity is measured locally rather than relative to the whole corpus. Both properties are absent from conventional ranking methods which evaluate each document individually.

In practical applications, a more sophisticated Ψ may be more appropriate. We develop our discriminant by addressing two criteria: how well a document covers a word, and how important it is to cover a word in $\mathbf{x}$.

4.1. How well a document covers a word

In our simple example (3), a single word set $V(\mathbf{y})$ is used, and all words that appear at least once in $\mathbf{y}$ are included. However, documents do not cover all words equally well, which is something not captured in (3). For example, a document which contains 5 instances of the word "lion" might cover the word better than another document which only contains 2 instances.

Instead of using only one $V(\mathbf{y})$, we can use L such word sets $V_1(\mathbf{y}), \ldots, V_L(\mathbf{y})$. Each word set $V_\ell(\mathbf{y})$ contains only words satisfying certain importance criteria. These importance criteria can be based on properties such as appearance in the title, the term frequency in the document, and having a high TFIDF value in the document (Salton & Buckley, 1988). Table 1 contains examples of importance criteria that we considered. For example, if importance criterion ℓ requires appearing at least 5 times in a document, then $V_\ell(\mathbf{y})$ will be the set of words which appear at least 5 times in some document in $\mathbf{y}$. The most basic criterion simply requires appearance in a document, and using only this criterion will result in (3).

We use a separate feature vector $\phi_\ell(v, \mathbf{x})$ for each importance level. We will describe ϕ_ℓ in greater detail in Section 4.2. We define Ψ from (2) to be the vector

Algorithm 1 Greedy subset selection by maximizing weighted word coverage

Input: $\mathbf{w}$, $\mathbf{x}$
Initialize solution $\hat{\mathbf{y}} \leftarrow \emptyset$
for $k = 1, \ldots, K$ **do**
  $\hat{x} \leftarrow \text{argmax}_{x:x \notin \hat{\mathbf{y}}} \mathbf{w}^T \Psi(\mathbf{x}, \hat{\mathbf{y}} \cup \{d\})$
  $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} \cup \{\hat{x}\}$
end for
return $\hat{\mathbf{y}}$

composition of all the ϕ_ℓ vectors,

$$\Psi(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \sum_{v \in V_1(\mathbf{y})} \phi_1(v, \mathbf{x}) \\ \vdots \\ \sum_{v \in V_L(\mathbf{y})} \phi_L(v, \mathbf{x}) \\ \sum_{i=1}^{n} y_i \psi(x_i, \mathbf{x}) \end{bmatrix}. \quad (4)$$

We can also include a feature vector $\psi(x, \mathbf{x})$ to encode any salient document properties which are not captured at the word level (e.g., "this document received a high score with an existing ranking function").

4.2. The importance of covering a word

In this section, we describe our formulation for the feature vectors $\phi_1(v, \mathbf{x}), \ldots, \phi_L(v, \mathbf{x})$. These features encode the benefit of covering a word, and are based primarily on document frequency in $\mathbf{x}$.

Using the importance criteria defined in Section 4.1, let $D_\ell(v)$ denote the set of documents in $\mathbf{x}$ which cover word v at importance level ℓ. For example, if the importance criterion is "appears at least 5 times in the document", then $D_\ell(v)$ is the set of documents that have at least 5 copies of v. This is, in a sense, a complementary definition to $V_\ell(\mathbf{y})$.

We use thresholds on the ratio $|D_\ell(v)|/n$ to define feature values of $\phi_\ell(v, \mathbf{x})$ that describe word v at different importance levels. Table 2 describes examples of features that we considered.

4.3. Making Predictions

Putting the formulation together, $\mathbf{w}_\ell^T \phi_\ell(v, \mathbf{x})$ denotes the benefit of covering word v at importance level ℓ, where $\mathbf{w}_\ell$ is the sub-vector of $\mathbf{w}$ which corresponds to ϕ_ℓ in (4). A word is only covered at importance level ℓ if it appears in $V_\ell(\mathbf{y})$. The goal then is to select K documents which maximize the aggregate benefit.

Selecting the K documents which maximizes (2) takes n choose K time, which quickly becomes intractable for even small values of K. Algorithm 1 describes a greedy algorithm which iteratively selects the document with highest marginal gain. Our prediction problem is a special case of the Budgeted Max Coverage problem (Khuller et al., 1997), and the greedy algorithm is known to have a $(1 - 1/e)$-approximation bound. During prediction, the weight vector $\mathbf{w}$ is assumed to be already learned.

5. Training with Structural SVMs

SVMs have been shown to be a robust and effective approach to complex learning problems in information retrieval (Yue et al., 2007; Chapelle et al., 2007). For a given training set $S = \{(\mathbb{T}^{(i)}, \mathbf{x}^{(i)})\}_{i=1}^N$, we use the structural SVM formulation, presented in Optimization Problem 1, to learn a weight vector $\mathbf{w}$.

Optimization Problem 1. (STRUCTURAL SVM)

$$\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N} \xi_i \quad (5)$$

s.t. $\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}^{(i)}$:

$$\mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \geq \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}) + \Delta(\mathbb{T}^{(i)}, \mathbf{y}) - \xi_i \quad (6)$$

The objective function (5) is a tradeoff between model complexity, $\|\mathbf{w}\|^2$, and a hinge loss relaxation of the training loss for each training example, $\sum \xi_i$, and the tradeoff is controlled by the parameter C. The $\mathbf{y}^{(i)}$ in the constraints (6) is the prediction which minimizes $\Delta(\mathbb{T}^{(i)}, \mathbf{y}^{(i)})$, and can be chosen via greedy selection.

The formulation of Ψ in (4) is very similar to learning a straightforward linear model. The key difference is that each training example is now a set of documents $\mathbf{x}$ as opposed to a single document. For each training example, each "suboptimal" labeling is associated with a constraint (6). There are now an immense number of constraints to define for SVM training.

Despite the large number of constraints, we can use Algorithm 2 to solve OP 1 efficiently. Algorithm 2 is a cutting plane algorithm, iteratively adding constraints until we have solved the original problem within a desired tolerance ϵ (Tsochantaridis et al., 2005). The algorithm starts with no constraints, and iteratively finds for each example $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ the $\hat{\mathbf{y}}$ which encodes the most violated constraint. If the corresponding constraint is violated by more than ϵ we add $\hat{\mathbf{y}}$ into the working set $\mathcal{W}_i$ of active constraints for example i, and re-solve (5) using the updated $\mathcal{W}$. Algorithm 2's outer loop is guaranteed to halt within a polynomial number of iterations for any desired precision ϵ.

Theorem 1. *Let* $\bar{R} = \max_i \max_{\mathbf{y}} \|\Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \Psi(\mathbf{x}^{(i)}, \mathbf{y})\|$, $\bar{\Delta} = \max_i \max_{\mathbf{y}} \Delta(\mathbb{T}^{(i)}, \mathbf{y})$, *and for any*

Algorithm 2 Cutting plane algorithm for solving OP 1 within tolerance ϵ.

Input: $(\mathbf{x}^{(1)}, \mathbb{T}^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbb{T}^{(N)})$, C, ϵ
$\mathcal{W}_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
repeat
  for $i = 1, \ldots, n$ **do**
    $H(\mathbf{y}; \mathbf{w}) \equiv \Delta(\mathbb{T}^{(i)}, \mathbf{y}) + \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}) - \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}_i)$
    compute $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y}; \mathbf{w})$
    compute $\xi_i = \max\{0, \max_{\mathbf{y} \in \mathcal{W}_i} H(\mathbf{y}; \mathbf{w})\}$
    if $H(\hat{\mathbf{y}}; \mathbf{w}) > \xi_i + \epsilon$ **then**
      $\mathcal{W}_i \leftarrow \mathcal{W}_i \cup \{\hat{\mathbf{y}}\}$
      $\mathbf{w} \leftarrow$ optimize (5) over $\mathcal{W} = \bigcup_i \mathcal{W}_i$
    end if
  end for
until no $\mathcal{W}_i$ has changed during iteration

$\epsilon > 0$, Algorithm 2 terminates after adding at most

$$\max\left\{\frac{2n\bar{\Delta}}{\epsilon}, \frac{8C\bar{\Delta}\bar{R}^2}{\epsilon^2}\right\}$$

constraints to the working set $\mathcal{W}$. See (Tsochantaridis et al., 2005) for proof.

However, each iteration of the inner loop of Algorithm 2 must compute $\text{argmax}_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y}; \mathbf{w})$, or equivalently,

$$\underset{\mathbf{y} \in \mathcal{Y}}{\text{argmax}}\, \Delta(\mathbb{T}^{(i)}, \mathbf{y}) + \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}), \tag{7}$$

since $\mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ is constant with respect to $\mathbf{y}$. Though closely related to prediction, this has an additional complication with the $\Delta(\mathbb{T}^{(i)}, \mathbf{y})$ term. As such, a constraint generation oracle is required.

5.1. Finding Most Violated Constraint

The constraint generation oracle must efficiently solve (7). Unfortunately, solving (7) exactly is intractable since exactly solving the prediction task,

$$\underset{\mathbf{y} \in \mathcal{Y}}{\text{argmax}}\, \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}),$$

is intractable. An approximate method must be used. The greedy inference method in Algorithm 1 can be easily modified for this purpose. Since constraint generation is also a special case of the Budgeted Max Coverage Problem, the $(1-1/e)$-approximation bound still holds. Despite using an approximate constraint generation oracle, SVM training is still known to terminate in a polynomial number of iterations (Finley & Joachims, 2008). Furthermore in practice, training typically converges much faster than the worst case considered by the theoretical bounds.

Intuitively, a small set of the constraints can approximate to ϵ precision the feasible space defined by the intractably many constraints. When constraint generation is approximate however, the ϵ precision guarantee no longer holds. Nonetheless, using approximate constraint generation can still offer good performance, which we will evaluate empirically.

6. Experiment Setup

We tested the effectiveness of our method using the TREC 6-8 Interactive Track Queries[2]. Relevant documents are labeled using subtopics. For example, query 392 asked human judges to identify different applications of robotics in the world today, and they identified 36 subtopics among the results such as nanorobots and using robots for space missions.

The 17 queries we used are 307, 322, 326, 347, 352, 353, 357, 362, 366, 387, 392, 408, 414, 428, 431, 438, and 446. Three of the original 20 queries were discarded due to having small candidate sets, making them uninteresting for our experiments. Following the setup in (Zhai et al., 2003), candidate sets only include documents which are relevant to at least one subtopic. This decouples the diversity problem, which is the focus of our study, from the relevance problem. In practice, approaches like ours might be used to post-process the results of a commercial search engine. We also performed Porter stemming and stop-word removal.

We used a 12/4/1 split for our training, validation and test sets, respectively. We trained our SVM using C values varying from 1e-5 to 1e3. The best C value is then chosen on the validation set, and evaluated on the test query. We permuted our train/validation/test splits until all 17 queries were chosen once for the test set. Candidate sets contain on average 45 documents, 20 subtopics, and 300 words per document. We set the retrieval size to $K = 5$ since some candidate sets contained as few as 16 documents.

We compared our method against Okapi (Robertson et al., 1994), and Essential Pages (Swaminathan et al., 2008). Okapi is a conventional retrieval function which evaluates the relevance of each document individually and does not optimize for diversity. Like our method, Essential Pages also optimizes for diversity by selecting documents to maximize weighted word coverage (but based on a fixed, rather than a learned, model). In their model, the benefit of document x_i covering a word v is defined to be

$$TF(v, x_i) \log\left(\frac{1}{DF(v, \mathbf{x})}\right),$$

[2] `http://trec.nist.gov/`

Method	Loss
Random	0.469
Okapi	0.472
Unweighted Model	0.471
Essential Pages	0.434
$\text{SVM}^{\Delta}_{div}$	0.349
$\text{SVM}^{\Delta}_{div2}$	0.382

Table 3. Performance on TREC ($K = 5$)

where $TF(v, x_i)$ is the term frequency of v in x_i and $DF(v, \mathbf{x})$ is the document frequency of v in $\mathbf{x}$.

We define our loss function to be the weighted percentage of subtopics not covered. For a given candidate set, each subtopic's weight is proportional to the number of documents that cover that subtopic. This is attractive since it assigns a high penalty to not covering a popular subtopic. It is also compatible with our discriminant since frequencies of important words will vary based on the distribution of subtopics.

The small quantity of TREC queries makes some evaluations difficult, so we also generated a larger synthetic dataset of 100 candidate sets. Each candidate set has 100 documents covering up to 25 subtopics. Each document samples 300 words independently from a multinomial distribution over 5000 words. Each document's word distribution is a mixture of its subtopics' distributions. We used this dataset to evaluate how performance changes with retrieval size K. We used a 15/10/75 split for training, validation, and test sets.

7. Experiment Results

Let $\text{SVM}^{\Delta}_{div}$ denote our method which uses term frequencies and title words to define importance criteria (how well a document covers a word), and let $\text{SVM}^{\Delta}_{div2}$ denote our method which in addition also uses TFIDF. $\text{SVM}^{\Delta}_{div}$ and $\text{SVM}^{\Delta}_{div2}$ use roughly 200 and 300 features, respectively. Table 1 contains examples of importance criteria that could be used.

Table 3 shows the performance results on TREC queries. We also included the performance of randomly selecting 5 documents as well as an unweighted word coverage model (all words give equal benefit when covered). Only Essential Pages, $\text{SVM}^{\Delta}_{div}$ and $\text{SVM}^{\Delta}_{div2}$ performed better than random.

Table 4 shows the per query comparisons between $\text{SVM}^{\Delta}_{div}$, $\text{SVM}^{\Delta}_{div2}$ and Essential Pages. Two stars indicate 95% significance using the Wilcoxon signed rank test. While the comparison is not completely fair since Essential Pages was designed for a slightly different setting, it demonstrates the benefit of automatically fitting a retrieval function to the specific task at hand.

Method Comparison	Win / Tie / Lose
$\text{SVM}^{\Delta}_{div}$ vs Essential Pages	14 / 0 / 3 **
$\text{SVM}^{\Delta}_{div2}$ vs Essential Pages	13 / 0 / 4
$\text{SVM}^{\Delta}_{div}$ vs $\text{SVM}^{\Delta}_{div2}$	9 / 6 / 2

Table 4. Per Query Comparison on TREC ($K = 5$)

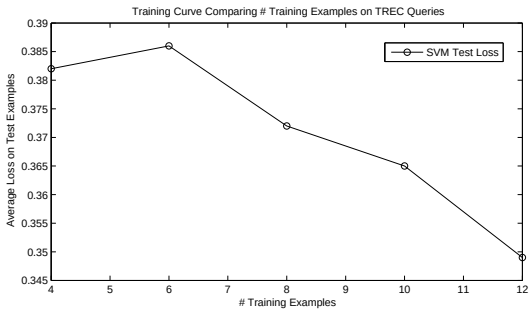


Figure 2. Comparing Training Size on TREC ($K = 5$)

Despite having a richer feature space, $\text{SVM}^{\Delta}_{div2}$ performs worse than $\text{SVM}^{\Delta}_{div}$. We conjecture that the top TFIDF words do not discriminate between subtopics. These words are usually very descriptive of the query as a whole, and thus will appear in all subtopics.

Figure 2 shows the average test performance of $\text{SVM}^{\Delta}_{div}$ as the number of training examples is varied. We see a substantial improvement in performance as training set size increases. It appears that more training data would further improve performance.

7.1. Approximate Constraint Generation

Using appoximate constraint generation might compromise our model's ability to (over-)fit the data. We addressed this concern by examining the training loss as the C parameter is varied. The training curve of $\text{SVM}^{\Delta}_{div}$ is shown in Figure 3. Greedy optimal refers to the loss incurred by a greedy method with knowledge of subtopics. As we increase C (favoring low training loss over low model complexity), our model is able to fit the training data almost perfectly. This indicates that approximate constraint generation is acceptable for our training purposes.

7.2. Varying Predicted Subset Size

We used the synthetic dataset to evaluate the behavior of our method as we vary the retrieval size K. It is difficult to perform this evaluation on the TREC queries – since some candidate sets have very few documents

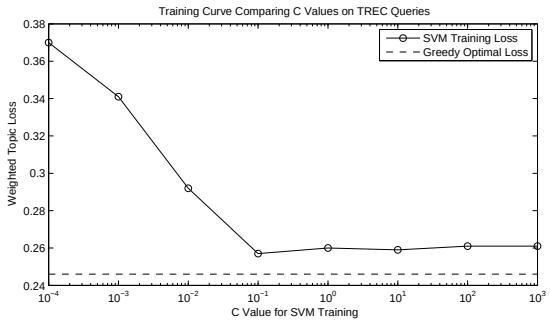


Figure 3. Comparing C Values on TREC ($K = 5$)

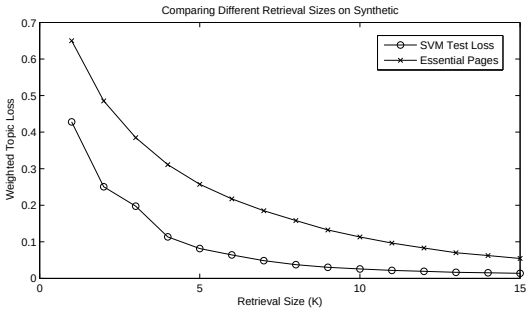


Figure 4. Varying Retrieval Size on Synthetic

or subtopics, using higher K would force us to discard more queries. Figure 4 shows that the test performance of $\text{SVM}^{\Delta}_{div}$ consistently outperforms Essential Pages at all levels of K.

7.3. Running Time

Predicting takes linear time. During training, Algorithm 2 loops for 10 to 100 iterations. For ease of development, we used a Python interface[3] to SVM^{struct}. Even with our unoptimized code, most models trained within an hour, with the slowest finishing in only a few hours. We expect our method to easily accomodate much more data since training scales linearly with dataset size (Joachims et al., to appear).

8. Extensions

8.1. Alternative Discriminants

Maximizing word coverage might not be suitable for other types of retrieval tasks. Our method is a general framework which can incorporate other discriminant formulations. One possible alternative is to maximize the pairwise distance of items in the predicted subset. Learning a weight vector for (2) would then amount to finding a distance function for a specific retrieval task.

[3] `http://www.cs.cornell.edu/~tomf/svmpython2/`

Any discriminant can be used so long as it captures the salient properties of the retrieval task, is linear in a joint feature space (2), and has effective inference and constraint generation methods.

8.2. Alternative Loss Functions

Our method is not restricted to using subtopics to measure diversity. Only our loss function $\Delta(\mathbb{T}, \mathbf{y})$ makes use of subtopics during SVM training. We can also incorporate loss functions which can penalize other types of diversity criteria and also use other forms of training data, such as clickthrough logs. The only requirement is that it must be computationally compatible with the constraint generation oracle (7).

8.3. Additional Word Features

Our choice of features is based almost exclusively on word frequencies. The sole exception is using title words as an importance criterion. The goal of these features is to describe how well a document covers a word and the importance of covering a word in a candidate set. Other types of word features might prove useful, such as anchor text, URL, and any meta information contained in the documents.

9. Conclusion

In this paper we have presented a general machine learning approach to predicting diverse subsets. Our method compares favorably to methods which do not perform learning, demonstrating the usefulness of training feature rich models for specific retrieval tasks. To the best of our knowledge, our method is the first approach which directly trains for subtopic diversity. Our method is also efficient since it makes predictions in linear time and has training time that scales linearly in the number of queries.

In this paper we separated the diversity problem from the relevance problem. An interesting direction for future work would be to jointly model both relevance and diversity. This is a more challenging problem since it requires balancing a tradeoff for presenting both novel and relevant information.

The non-synthetic TREC dataset is also admittedly small. Generating larger (and publicly available) labeled datasets which encode diversity information is another important direction for future work.

Acknowledgements

The work was funded under NSF Award IIS-0713483, NSF CAREER Award 0237381, and a gift from Ya-

hoo! Research. The first author is also partly funded by a Microsoft Research Fellowship and a Yahoo! Key Technical Challenge Grant. The authors also thank Darko Kirovski for initial discussions regarding his work on Essential Pages.

References

Broder, A., Fontoura, M., Gabrilovich, E., Joshi, A., Josifovski, V., & Zhang, T. (2007). Robust classification of rare queries using web knowledge. *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR).*

Burges, C. J. C., Ragno, R., & Le, Q. (2006). Learning to rank with non-smooth cost functions. *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS).*

Cai, L., & Hofmann, T. (2004). Hierarchical document categorization with support vector machines. *In Proceedings of the ACM Conference on Information and Knowledge Management (CIKM).*

Carbonell, J., & Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and reproducing summaries. *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR).*

Chapelle, O., Le, Q., & Smola, A. (2007). Large margin optimization of ranking measures. *NIPS workshop on Machine Learning for Web Search.*

Chen, H., & Karger, D. (2006). Less is more: Probabilistic models for retrieving fewer relevant documents. *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR).*

desJardins, M., Eaton, E., & Wagstaff, K. (2006). Learning user preferences for sets of objects. *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 273–280). ACM.

Finley, T., & Joachims, T. (2008). Training structural svms when exact inference is intractable. *Proceedings of the International Conference on Machine Learning (ICML).*

Joachims, T., Finley, T., & Yu, C. (to appear). Cutting-plane training of structural svms. *Machine Learning.*

Khuller, S., Moss, A., & Naor, J. (1997). The budgeted maximum coverage problem. *Information Processing Letters, 70(1)*, 39–45.

Kleinberg, R., Radlinski, F., & Joachims, T. (2008). Learning diverse rankings with multi-armed bandits. *Proceedings of the International Conference on Machine Learning (ICML).*

Li, P., Burges, C., & Wu, Q. (2007). Learning to rank using classification and gradient boosting. *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS).*

Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M., & Gatford, M. (1994). Okapi at TREC-3. *Proceedings of TREC-3.*

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management, 24(5)*, 513–523.

Swaminathan, A., Mathew, C., & Kirovski, D. (2008). *Essential pages* (Technical Report MSR-TR-2008-015). Microsoft Research.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR), 6(Sep)*, 1453–1484.

Wagstaff, K., desJardins, M., Eaton, E., & Montminy, J. (2007). Learning and visualizing user preferences over sets. *American Association for Artificial Intelligence (AAAI).*

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimizing average precision. *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR).*

Zhai, C., Cohen, W. W., & Lafferty, J. (2003). Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR).*

Zhang, B., Li, H., Liu, Y., Ji, L., Xi, W., Fan, W., Chen, Z., & Ma, W. (2005). Improving web search results using affinity graph. *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR).*

Zheng, Z., Zha, H., Zhang., T., Chapelle, O., Chen, K., & Sun, G. (2007). A general boosting method and its application to learning ranking functions for web search. *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS).*

Improved Nyström Low-Rank Approximation and Error Analysis

Kai Zhang TWINSEN@CSE.UST.HK
Ivor W. Tsang IVOR@CSE.UST.HK
James T. Kwok JAMESK@CSE.UST.HK
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong

Abstract

Low-rank matrix approximation is an effective tool in alleviating the memory and computational burdens of kernel methods and sampling, as the mainstream of such algorithms, has drawn considerable attention in both theory and practice. This paper presents detailed studies on the Nyström sampling scheme and in particular, an error analysis that directly relates the Nyström approximation quality with the encoding powers of the landmark points in summarizing the data. The resultant error bound suggests a simple and efficient sampling scheme, the k-means clustering algorithm, for Nyström low-rank approximation. We compare it with state-of-the-art approaches that range from greedy schemes to probabilistic sampling. Our algorithm achieves significant performance gains in a number of supervised/unsupervised learning tasks including kernel PCA and least squares SVM.

1. Introduction

Kernel methods play a central role in machine learning and have demonstrated huge success in modelling real-world data with highly complex, nonlinear structures. Examples include the support vector machine, kernel Fisher discriminant analysis and kernel principal component analysis. The key element of kernel methods is to map the data into a kernel-induced Hilbert space $\varphi(\cdot)$ where dot product between points can be computed equivalently through the kernel evaluation $\langle \varphi(x_i), \varphi(x_j) \rangle = K(x_i, x_j)$. Given n sample points, this necessitates the calculation of an $n \times n$ symmetric, positive (semi-)definite kernel matrix. The resultant complexities in terms of both space (quadratic) and time (usually cubic) can be quite demanding for large problems, posing a big challenge on practical applications.

Appearing in *Proceedings of the 25^{th} International Conference on Machine Learning*, Helsinki, Finland, 2008.

A useful way to alleviate the memory and computational burdens of kernel methods is to utilize the rapid decaying spectra of the kernel matrices (Williams & Seeger, 2000) and perform low-rank approximation in the form of $K = GG'$, where $G \in \mathbb{R}^{n \times m}$ with $m \ll n$. However, the optimal (eigenvalue) decomposition takes $O(n^3)$ time and efficient alternatives have to be sought. In the following, we give a brief review on efficient techniques for low-rank decompositions of symmetric, positive (semi-)definite kernel matrices.

Greedy approaches have been applied in several fast algorithms for approximating the kernel matrix. In (Smola & Schölkopf, 2000), the kernel matrix K is approximated by the subspace spanned by a subset of its columns. The basis vectors are chosen incrementally to minimize an upper bound of the approximation error. The algorithm takes $O(m^2nl)$ time using a probabilistic heuristic, where l is the random subset size. In (Ouimet & Bengio, 2005), a greedy sampling scheme is proposed based on how well a sample point can be represented by a (constrained) linear combination of the current subspace basis in the feature space. Their algorithm scales as $O(m^2n)$. Another well-known greedy approach for low-rank approximation of positive semi-definite matrices is the incomplete Cholesky decomposition (Fine & Scheinberg, 2001; Bach & Jordan, 2005; Bach & Jordan, 2002). It is a variant of the Cholesky decomposition that skip pivots below a certain threshold, and factorizes the kernel matrix K as $K = GG'$ where $G \in \mathbb{R}^{n \times m}$ is a lower triangular matrix.

Another class of low-rank approximation algorithms stem from the Nyström method. The Nyström method was originally designed to solve integral equations (Baker, 1977). Given a kernel matrix K, the Nyström method can be deemed as choosing a subset of m columns (hence rows) $E \in \mathbb{R}^{n \times m}$, and reconstructing the complete kernel matrix by $K \simeq EW^{-1}E'$, where W is the intersection of the selected rows and columns of K. The most popular sampling scheme for Nyström method is random sampling, which leads to fast versions of kernel machines (Williams & Seeger, 2001; Lawrence & Herbrich, 2003) and spectral

clustering (Fowlkes et al., 2004). In (Platt, 2005), several variants of multidimensional scaling are all shown to be related to the Nyström approximation.

There are also a large body of randomized algorithms for low-rank decomposition of arbitrary matrices (Frieze et al., 1998; Achlioptas & McSherry, 2001; Drineas et al., 2003), where the goal is to design column/row sampling probabilities that achieve provable probabilistic bounds. These algorithms are designed for a more general purpose and will not be the focus of this paper. However, we note that one of these randomized algorithms has been recently revised for efficient low-rank approximation of the symmetric Gram matrix (Drineas & Mahoney, 2005). Therefore we will use it as a representative of randomized algorithms in our empirical evaluations. The basic idea of (Drineas & Mahoney, 2005) is to sample the columns of the kernel matrix based on a pre-computed distribution using the norms of the columns. The reconstruction of the kernel matrix is also normalized by the sampling distribution.

In terms of efficiency, greedy approaches usually take $O(m^2n)$ time for sampling, while the random scheme only needs $O(n)$ and is much more efficient. Probabilistic approaches, or randomized algorithms in general, are usually more expensive in that the sampling distributions have to be computed based on the original matrix, which require at least $O(n^2)$. In terms of memory, note that the matrices (E and W) needed in the Nyström method with random sampling can be simply computed on demand. This greatly reduces the memory requirement for very large-scale problems. In contrast, the intermediate matrices for greedy approaches have to be incrementally updated and stored.

Although the Nyström method possesses desirable scaling properties and has been applied with success in various machine learning problems, analysis on its key step of choosing the landmark set is relatively limited. In (Drineas & Mahoney, 2005), a probabilistic error bound is provided on the Nyström low-rank approximation. However, the error bound only applies to the specially designed sampling scheme, which needs to compute the norms of all the rows/columns of the kernel matrix and is hence quite expensive. In (Zhang & Kwok, 2006), a block quantization scheme is proposed for fast spectral embedding. The kernel eigen-system is approximated by first computing a block-wise constant kernel matrix and then extrapolating its eigenvectors through the weighted Nyström extension. However, the error analysis is only on the block-quantization step, and how the Nyström method affects the approximation quality in general remains unclear. Thus, the motivation of this paper is to provide a more concrete analysis on how the sampling scheme (or the choice of the landmark points) in general influences the Nyström low-rank approximation, and to improve the sampling strategy while still preserving its computational efficiency.

Our key finding is that the Nyström low-rank approximation depends crucially on the quantization error induced by encoding the sample set with the landmark points. This suggests that, instead of applying the greedy or probabilistic sampling, the landmark points can be simply chosen as the k-means cluster centers, which finds a local minimum of the quantization error. To the best of our knowledge, the k-means has not been applied in the Nyström low-rank approximation. The complexity of k-means is only linear in the sample size and dimension and, as our analysis expected, it demonstrates very encouraging performance that is consistently better than all known variants of Nyström. We also compare it with the greedy approach of incomplete Cholesky decomposition and again obtain positive results.

The rest of the paper is organized as follows. In Section 2, we give a brief introduction of the Nyström method. In Section 3, we present an error analysis on how the Nyström low-rank approximation is affected by the chosen landmark points, and propose the k-means algorithm for the sampling step. In Section 4, we compare our approach with a number of state-of-the-art low-rank decomposition techniques (including both greedy and probabilistic sampling approaches). The last section gives concluding remarks.

2. Nyström Method

The Nyström method is originated from the numerical treatment of integral equations of the form

$$\int p(y)k(x,y)\phi_i(y)dy = \lambda_i\phi_i(x), \tag{1}$$

where $p(\cdot)$ is the probability density function, k is a positive definite kernel function, and $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ and $\phi_1, \phi_2, \ldots$ are the eigenvalues and eigenfunctions of the integral equation, respectively. Given a set of i.i.d. samples $\{x_1, x_2, \ldots, x_q\}$ drawn from $p(\cdot)$, the basic idea is to approximate the integral in (1) by the empirical average:

$$\frac{1}{q}\sum_{j=1}^{q} k(x,x_j)\phi_i(x_j) \simeq \lambda_i\phi_i(x). \tag{2}$$

Choosing x in (2) from $\{x_1, x_2, \ldots, x_q\}$ leads to a standard eigenvalue decomposition $K^{(q)}U^{(q)} = U^{(q)}\Lambda^{(q)}$, where $K^{(q)}_{ij} = k(x_i, x_j)$ for $i, j = 1, 2, \ldots, q$, $U^{(q)} \in \mathbb{R}^{q\times q}$ has orthonormal columns and $\Lambda^{(q)} \in \mathbb{R}^{q\times q}$ is a diagonal matrix. The eigenfunctions ϕ_i's and eigenvalues λ_i's in (1) can be approximated by $U^{(q)}$ and $\Lambda^{(q)}$, as (Williams & Seeger, 2001):

$$\phi_i(x_j) \simeq \sqrt{q}U^{(q)}_{ji}, \quad \lambda_i \simeq \lambda^{(q)}_i/q. \tag{3}$$

This means, the Nyström method using different subset sizes q's are all approximations to λ_i and ϕ_i in the inte-

gral equation (1). As a result, the Nyström method using a small q can also be deemed as approximating the Nyström method using a large q. Suppose the sample set $\mathcal{X} = \{x_i\}_{i=1}^n$, with the corresponding $n \times n$ kernel matrix K. Then the Nyström method that randomly chooses a subset $\mathcal{Z} = \{z_i\}_{i=1}^m$ of m landmark points will approximate the eigen-system of the full kernel matrix $K\Phi_K = \Phi_K\Lambda_K$ by (Williams & Seeger, 2001)

$$\Phi_K \simeq \sqrt{\frac{m}{n}} E\Phi_{\mathcal{Z}}\Lambda_{\mathcal{Z}}^{-1}, \;\; \Lambda_K \simeq \frac{n}{m}\Lambda_{\mathcal{Z}}. \tag{4}$$

Here, $E \in \mathbb{R}^{n\times m}$ with $E_{ij} = k(x_i, z_j)$, and $\Phi_{\mathcal{Z}}, \Lambda_{\mathcal{Z}} \in \mathbb{R}^{m\times m}$ contain the eigenvectors and eigenvalues of $W \in \mathbb{R}^{m\times m}$ where $W_{ij} = k(z_i, z_j)$. Using the approximations in (4), K can be reconstructed as

$$\begin{aligned} K &\simeq \left(\sqrt{\frac{m}{n}} E\Phi_{\mathcal{Z}}\Lambda_{\mathcal{Z}}^{-1}\right)\left(\frac{n}{m}\Lambda_{\mathcal{Z}}\right)\left(\sqrt{\frac{m}{n}} E\Phi_{\mathcal{Z}}\Lambda_{\mathcal{Z}}^{-1}\right)' \\ &= EW^{-1}E'. \end{aligned} \tag{5}$$

Equation (5) is the basis for Nyström low-rank approximation of the kernel matrix (Williams & Seeger, 2001; Fowlkes et al., 2004).

3. Error Analysis of the Nyström Method

In this section we analyze how the Nyström approximation error depends on the choice of landmark points. We first provide an important observation (Section 3.1), and then derive the error bound in more general settings based on a "clustered" data model (Section 3.2-3.4). The error bound gives important insights on the design of efficient sampling schemes for accurate low-rank approximation (Section 3.5).

3.1. Observation

Proposition 1. *Given the data set $\mathcal{X} = \{x_i\}_{i=1}^n$, and the landmark point set $\mathcal{Z} = \{z_j\}_{j=1}^m$. Then the Nyström reconstruction of the kernel entry $K(x_i, x_j)$ will be exact if there exist two landmark points such that $z_p = x_i$, and $z_q = x_j$.*

Proof. Let $K_{x_k,\mathcal{Z}} \in \mathbb{R}^{1\times m}$ be the similarity between x_k and the landmark points $\mathcal{Z}$. Then the Nyström reconstruction of the kernel entry will be $K_{x_i,\mathcal{Z}}W^{-1}K'_{x_j,\mathcal{Z}}$, where $W \in R^{m\times m}$ is the kernel matrix defined on the landmark set $\mathcal{Z}$. Let $W^{(k)}$ be the kth row of W, then we have $K_{x_i,\mathcal{Z}} = W^{(p)}$ and $K_{x_j,\mathcal{Z}} = W^{(q)}$ since $x_i = z_p$, and $x_j = z_q$. As a result, the reconstructed entry will be $W^{(p)}W^{-1}(W^{(q)})' = W_{pq} = K(z_p, z_q) = K(x_i, x_j)$. □

Proposition 1 indicates that the landmark points should be chosen to overlap sufficiently with the original data. However, it is often impossible to use a small landmark set to represent every sample point accurately.

3.2. Approximation Error of Sub-Kernel Matrix

In this section we apply a "clustered" data model to analyze the quality of Nyström low rank approximation. Here, the data clusters can be naturally obtained by assigning each sample to the closest landmark point. As will be seen, this model allows us to derive an explicit error bound for the Nyström approximation.

Again, suppose that the landmark set is $\mathcal{Z} = \{z_i\}_{i=1}^m$, and the whole sample set $\mathcal{X}$ is partitioned into m disjoint clusters $\mathcal{S}_k$'s. Let $c(i)$ be the function that maps each sample $x_i \in \mathcal{X}$ to the closest landmark point $z_{c(i)} \in \mathcal{Z}$, i.e., $c(i) = \arg\min_{j=1,2,\cdots,m} \|x_i - z_j\|$. Our goal is to study the approximation error in (5):

$$\mathcal{E} = \left\|K - EW^{-1}E'\right\|_F, \tag{6}$$

where $\|\cdot\|_F$ denotes the matrix Frobenious norm.

First, we consider the simpler notion of *partial approximation error* defined as follows.

Definition 1. *Suppose each cluster has T samples*[1]*. Repeat the following sampling process T times: at each time t, pick one sample from each cluster, and denote the set of samples chosen at time t as $\mathcal{X}_{\mathcal{I}_t}$. Then $\mathcal{X} = \{\mathcal{X}_{\mathcal{I}_1} \cup \mathcal{X}_{\mathcal{I}_1} \cup ... \cup \mathcal{X}_{\mathcal{I}_T}\}$, and the whole kernel matrix will be correspondingly decomposed into T^2 blocks, each of size $m \times m$. Let $K_{\mathcal{I}_i,\mathcal{I}_j}$, and $E_{\mathcal{I}_i,\mathcal{Z}}$ be the $m\times m$ similarity matrices defined on $(\mathcal{X}_{\mathcal{I}_i}, \mathcal{X}_{\mathcal{I}_i})$ and $(\mathcal{X}_{\mathcal{I}_i}, \mathcal{Z})$, respectively, and $W \in \mathbb{R}^{m\times m}$ the kernel matrix defined on $\mathcal{Z}$. The* partial approximation error *is the difference between $K_{\mathcal{I}_i,\mathcal{I}_j}$ and its Nyström approximation under the Frobenius norm*

$$\mathcal{E}_{\mathcal{I}_i,\mathcal{I}_j} = \|K_{\mathcal{I}_i,\mathcal{I}_j} - E_{\mathcal{I}_i,\mathcal{Z}}W^{-1}E'_{\mathcal{I}_j,\mathcal{Z}}\|_F. \tag{7}$$

We assume the kernel k satisfies the following property:

$$(k(a,b) - k(c,d))^2 \le C_{\mathcal{X}}^k\left(\|a-c\|^2 + \|b-d\|^2\right), \forall a,b,c,d \tag{8}$$

where $C_{\mathcal{X}}^k$ is a constant depending on k and the sample set $\mathcal{X}$. The validity of this assumption on a number of commonly used kernels will be proved in Section 3.4.

Proposition 2. *For kernel k satisfying property (8), the partial approximation error $\mathcal{E}_{\mathcal{I}_i,\mathcal{I}_j}$ is bounded by*

$$\begin{aligned} \mathcal{E}_{\mathcal{I}_i,\mathcal{I}_j} \le\; & \sqrt{2mC_{\mathcal{X}}^k(e_{\mathcal{I}_i} + e_{\mathcal{I}_j})} + \sqrt{mC_{\mathcal{X}}^k e_{\mathcal{I}_i}} \\ & + \sqrt{mC_{\mathcal{X}}^k e_{\mathcal{I}_j}} + mC_{\mathcal{X}}^k\sqrt{e_{\mathcal{I}_i}e_{\mathcal{I}_j}}\|W^{-1}\|_F. \end{aligned} \tag{9}$$

where $e_{\mathcal{I}_i}$ is the quantization error induced by coding each sample in $\mathcal{X}_{\mathcal{I}_i}$ by the closest landmark point in $\mathcal{Z}$, i.e.,

$$e_{\mathcal{I}_i} = \textstyle\sum_{x_i\in\mathcal{X}_{\mathcal{I}_i}} \left\|x_i - z_{c(i)}\right\|^2. \tag{10}$$

[1] If cluster sizes differ, add "virtual samples" to each cluster such that all the clusters have the same size (which is equal to $T = \max_k |\mathcal{S}_k|$). The virtual samples added to cluster $\mathcal{S}_k$ are chosen as the landmark point z_k for that cluster, so they will not induce extra quantization errors but will loosen the bound.

Proof. We will first define the following matrices

$$A_{\mathcal{I}_i,\mathcal{I}_j} = K_{\mathcal{I}_i,\mathcal{I}_j} - W; B_{\mathcal{I}_i,\mathcal{Z}} = E_{\mathcal{I}_i,\mathcal{Z}} - W; \\ C_{\mathcal{I}_j,\mathcal{Z}} = E_{\mathcal{I}_j,\mathcal{Z}} - W, \tag{11}$$

and then show that they have bounded Frobenius norms. Without loss of generality, we specify the indices as follows: $K_{\mathcal{I}_i,\mathcal{I}_j}(p,q) = k(x_{\mathcal{I}_i(p)}, x_{\mathcal{I}_j(q)})$; $E_{\mathcal{I}_i,\mathcal{Z}}(p,q) = k(x_{\mathcal{I}_i(p)}, z_q)$; $E_{\mathcal{I}_j,\mathcal{Z}}(p,q) = k(x_{\mathcal{I}_j(p)}, z_q)$; and $W(p,q) = k(z_p, z_q)$. With property (8), we have

$$\begin{aligned} \| A_{\mathcal{I}_i,\mathcal{I}_j} \|_F^2 &= \sum_{p,q=1}^{m} \left(k(x_{\mathcal{I}_i(p)}, x_{\mathcal{I}_j(q)}) - k(z_p, z_q) \right)^2 \\ &\leq C_{\mathcal{X}}^k \sum_{p,q=1}^{m} \left(\left\| x_{\mathcal{I}_i(p)} - z_p \right\|^2 + \left\| x_{\mathcal{I}_i(q)} - z_q \right\|^2 \right) \\ &= mC_{\mathcal{X}}^k \left(\sum_{p=1}^{m} \left\| x_{\mathcal{I}_i(p)} - z_p \right\|^2 + \sum_{q=1}^{m} \left\| x_{\mathcal{I}_j(q)} - z_q \right\|^2 \right) \\ &= 2mC_{\mathcal{X}}^k \left(e_{\mathcal{I}_i} + e_{\mathcal{I}_j} \right), \end{aligned}$$

where $e_{\mathcal{I}_i}$ is the same as that in (10) since $c(\mathcal{I}(q)) = q$.

For matrix $B_{\mathcal{I}_i,\mathcal{Z}}$, we have

$$\begin{aligned} \|B_{\mathcal{I}_i,\mathcal{Z}}\|_F^2 &= \sum_{p,q} \left(k(x_{\mathcal{I}_i(p)}, z_q) - k(z_p, z_q) \right)^2 \\ &\leq mC_{\mathcal{X}}^k \sum_{p=1}^{m} \left\| x_{\mathcal{I}_i(p)} - z_p \right\|^2 = mC_{\mathcal{X}}^k e_{\mathcal{I}_i}, \end{aligned}$$

and similarly for matrix $C_{\mathcal{I}_j,\mathcal{Z}}$ $\|C_{\mathcal{I}_j,\mathcal{Z}}\|_F^2 \leq mC_{\mathcal{X}}^k e_{\mathcal{I}_j}$.

Note that the partial approximation error $\mathcal{E}_{\mathcal{I}_i,\mathcal{I}_j}$ (7) can be re-written as follows using (11).

$$\begin{aligned} &\| \mathcal{E}_{\mathcal{I}_i,\mathcal{I}_j} \|_F = \left\| W + A_{\mathcal{I}_i,\mathcal{I}_j} - (W + B_{\mathcal{I}_i,\mathcal{Z}}) W^{-1} (W + C_{\mathcal{I}_j,\mathcal{Z}})' \right\|_F \\ &= \left\| W + A_{\mathcal{I}_i,\mathcal{I}_j} - W' - C'_{\mathcal{I}_j,\mathcal{Z}} - B_{\mathcal{I}_i,\mathcal{Z}} - B_{\mathcal{I}_i,\mathcal{Z}} W^{-1} C'_{\mathcal{I}_j,\mathcal{Z}} \right\|_F \\ &\leq \|A_{\mathcal{I}_i,\mathcal{I}_j}\|_F + \|B_{\mathcal{I}_i,\mathcal{Z}}\|_F + \|C_{\mathcal{I}_j,\mathcal{Z}}\|_F + \|B_{\mathcal{I}_i,\mathcal{Z}}\|_F \|C_{\mathcal{I}_j,\mathcal{Z}}\|_F \|W^{-1}\|_F \end{aligned}$$

Using the bounds on $\|A_{\mathcal{I}_i,\mathcal{I}_j}\|$, $\|B_{\mathcal{I}_i,\mathcal{Z}}\|$, $\|C_{\mathcal{I}_j,\mathcal{Z}}\|$ together with the definition in (11), we have Proposition 2 □

3.3. Approximation Error of Complete Kernel Matrix

With the estimated partial approximation error, we can now obtain a bound on the complete error for Nyström approximation (6). The basic idea is to sum up the partial errors $\mathcal{E}_{\mathcal{I}_i,\mathcal{I}_j}$ over all $i, j = 1, 2, ..., T$.

Proposition 3. *The error of the Nyström approximation (6) is bounded by*

$$\mathcal{E} \leq 4T\sqrt{mC_{\mathcal{X}}^k eT} + mC_{\mathcal{X}}^k Te \|W^{-1}\|_F \tag{12}$$

where $T = \max_k |\mathcal{S}_k|$*, and* $e = \sum_{i=1}^{n} \left\| x_i - z_{c(i)} \right\|^2$ *is the total quantization error of coding each sample* $x_i \in \mathcal{X}$ *with the closest landmark point* $z_j \in \mathcal{Z}$*.*

Proof. Here we sum up the terms in (9) separately.

$$\begin{aligned} &\sum_{i,j=1}^{T} \sqrt{2mC_{\mathcal{X}}^k (e_{\mathcal{I}_i} + e_{\mathcal{I}_j})} = \sqrt{2mC_{\mathcal{X}}^k} \sum_{i=1}^{T} \left(\sum_{j=1}^{T} \sqrt{e_{\mathcal{I}_i} + e_{\mathcal{I}_j}} \right) \\ &\leq \sqrt{2mC_{\mathcal{X}}^k} \sum_{i=1}^{T} \left(\sqrt{T} \sqrt{Te_{\mathcal{I}_i} + \sum_{j=1}^{T} e_{\mathcal{I}_j}} \right) \leq 2T\sqrt{mC_{\mathcal{X}}^k Te} \end{aligned}$$

where $e = \sum_{j=1}^{T} e_{\mathcal{I}_j} = \sum_{x_i \in \mathcal{X}} \|x_i - z_{c(i)}\|^2$ is the same as defined in proposition 3. Similarly, the second term (and the third term) in (9) can be summarized as

$$\sum_{i,j=1}^{T} \sqrt{mC_{\mathcal{X}}^k e_{\mathcal{I}_i}} = \sqrt{mC_{\mathcal{X}}^k} \sum_{j=1}^{T} \left(\sum_{i=1}^{T} \sqrt{e_{\mathcal{I}_i}} \right) \leq T\sqrt{mC_{\mathcal{X}}^k eT}$$

The last term in (9) can be summarized as

$$\begin{aligned} \sum_{i,j=1}^{T} mC_{\mathcal{X}}^k \sqrt{e_{\mathcal{I}_i} e_{\mathcal{I}_j}} \|W^{-1}\|_F &= mC_{\mathcal{X}}^k \|W^{-1}\|_F \left(\sum_{i=1}^{T} \sqrt{e_{\mathcal{I}_i}} \right)^2 \\ &\leq mC_{\mathcal{X}}^k \|W^{-1}\|_F Te \end{aligned}$$

By combining all these terms, we arrive at Proposition 3. □

3.4. $C_{\mathcal{X}}^k$ Under Different Kernels

In this section, we show that many commonly used kernel functions satisfy the property in (8). Consider the stationary kernel $k(x,y) = \kappa\left(\left\|\frac{x-y}{\sigma}\right\|\right)$, including the Gaussian kernel $\kappa(\alpha) = \exp(-\alpha^2)$, Laplacian kernel $\kappa(\alpha) = \exp(-\alpha)$, and inverse distance kernel $\kappa(\alpha) = (\alpha + \epsilon)^{-1}$. By using the mean value theorem and triangular inequality, we have, for any $a, b, c, d \in R^d$,

$$\begin{aligned} (k(a,b) - k(c,d))^2 &= \left(\kappa\left(\|a-b\|/\sigma \right) - \kappa\left(\|c-d\|/\sigma \right) \right)^2 \\ &= \left[\kappa'(\xi)/\sigma \right]^2 \left(\|a-b\| - \|c-d\| \right)^2. \end{aligned}$$

Let $v_1 = a - c$ and $v_2 = b - d$. Note that we have $\|c-d\| \leq \|a-b\| + \|v_1\| + \|v_2\|$ and similarly $\|a-b\| \leq \|c-d\| + \|v_1\| + \|v_2\|$. So $\|a-b\| - \|c-d\|$ is always bounded by

$$\begin{aligned} \left(\|a-b\| - \|c-d\| \right)^2 &\leq \left(\|a-c\| + \|b-d\| \right)^2 \\ &\leq 2\left(\|a-c\|^2 + \|b-d\|^2 \right). \end{aligned}$$

So $C_{\mathcal{X}}^k$ can be chosen as $\max[2\kappa'(\xi)/\sigma]^2$ which is often bounded ($C_{\mathcal{X}}^k$ is $\frac{1}{2\sigma^2}$ for the Gaussian, $\frac{1}{\sigma^2}$ for the Laplacian, and $\frac{1}{\sigma^2\epsilon^4}$ for the inverse distance). Similarly, for polynomial kernels of the form $k(x,y) = (\langle x, y \rangle + \epsilon)^d$,

$$\begin{aligned} &(k(a,b) - k(c,d))^2 = \left((a'b + \epsilon)^d - (c'd + \epsilon)^d \right)^2 \\ &= \left(p'(\xi)(a'b - c'd) \right)^2 = \left(p'(\xi) \left((a-c)'b + (b-d)'c \right) \right)^2 \\ &\leq \left[2p'(\xi) \right]^2 \left(\|(a-c)'b\|^2 + \|(b-d)'c\|^2 \right) \\ &\leq \left[2p'(\xi) R \right]^2 \left(\|a-c\|^2 + \|b-d\|^2 \right), \end{aligned}$$

where R is the larger one of the two quantities: the maximum pairwise distance between samples, and maximum distance between samples and the origin point; and $p(z) = z^d$. So $C_{\mathcal{X}}^k$ can be chosen as $\max[\kappa'(\xi)R]^2 = d^2R^d$.

3.5. Sampling Procedure

The error bound in Proposition 3 provides important insights on how to choose the landmark points in the Nyström method. As can be seen, consistently, that for a number of commonly used kernels, the most important factor that influences the approximation quality is e, the error of quantizing each of the samples in $\mathcal{X}$ with the closest landmark in $\mathcal{Z}$. If this quantization error is zero, the Nyström low-rank approximation of the kernel matrix will also be exact. This agrees well with the ideal case discussed in Section 3.1.

Motivated by this observation and the fact that k-means clustering can find a local minimum of the quantization error (Gersho & Gray, 1992), we propose to use the centers obtained from the k-means as the landmark points. Here, k is the desired number of landmark points in $\mathcal{Z}$. The larger the k, the more accurate the approximation though at the cost of higher computations. Despite its simplicity, the k-means procedure can greatly improve the approximation quality compared to other sampling schemes, as will be demonstrated empirically in Section 4. Recent advances in speeding up the k-means algorithms (Elkan, 2003; Kanungo et al., 2001) also make this k-means-based sampling strategy particularly suitable for large-scale problems.

4. Experiments

This section presents empirical evaluations of the various low-rank approximation schemes. First, we discuss how the low rank approximation fits into different applications. One is to solve linear systems of the form $(K + \sigma I)x = a$, where K is the kernel matrix, $\sigma \geq 0$ is a regularization parameter and I is the $n \times n$ identity matrix. Given the low-rank approximation $K \simeq GG'$, the following holds (Williams & Seeger, 2001) by the Woodbury formula

$$(K + \sigma I)^{-1} \simeq \frac{1}{\sigma}\left(I - G(\sigma I + G'G)^{-1}G'\right), \qquad (13)$$

which only needs $O(m^2n)$ time and $O(mn)$ memory. Therefore, it can be used in speeding up the Gaussian processes (Williams & Seeger, 2001) and least-squares SVM (LS-SVM) (Suykens & Vandewalle, 1999).

The second application is to reconstruct the eigen-system of a matrix approximated by its low-rank decomposition.

Proposition 4. *Given the low-rank approximation $K \approx GG'$, where $G \in \mathbb{R}^{n\times m}$ and $m \ll n$, the top m eigenvectors U of K can be obtained as $U \approx GV\Lambda^{-1/2}$ in $O(m^2n)$ time, where $V, \Lambda \in \mathbb{R}^{m\times m}$ are from the eigenvalue decomposition of the $m \times m$ matrix $S = G'G = V\Lambda V'$.*

Proof can be found in (Fowlkes et al., 2004). Therefore low-rank approximation is useful for algorithms that rely on eigenvectors of the kernel matrix, such as kernel PCA (Schölkopf et al., 1998), Laplacian eigenmap (Belkin & Niyogi, 2002) and normalized cut.

Note that the Nyström method, when designed originally to solve integral equations, did not provide orthogonal approximations to the kernel eigenfunctions. Thanks to the matrix completion view (5) (Fowlkes et al., 2004; Williams & Seeger, 2001), the Nyström method can be utilized for obtaining orthogonal eigenvectors (Proposition 4), though the time complexity increases from the simple Nyström extension (4) of $O(mn)$ to $O(m^2n)$. In the experiments we focus on the orthogonalized eigenvector approximation.

Table 1. Complexities of basis selection for the different methods.

	Ours	Nyström	Drineas	ICD
time	$O(mn)$	$O(n)$	$O(n^2)$	$O(m^2n)$
space	$O(mn)$	$O(mn)$	$O(mn)$	$O(mn)$

We compare altogether five low-rank approximation algorithms, including: 1. incomplete Cholesky decomposition (ICD)[2]; 2. Nyström method (with random sampling); 3. the method in (Drineas & Mahoney, 2005); 4. our method (for simplicity, the maximum number of k-means iterations is restricted to 10); 5. SVD. Note that SVD (or eigenvalue decomposition in our context) provides the best low-rank approximation in terms of both the Frobenius norm and spectral norm (Golub & Van Loan, 1996). The complexities of basis selection (i.e., choosing E and W in Nyström, or sampling the columns in (Drineas & Mahoney, 2005) and ICD) in the different algorithms are listed in Table 1. Evaluations are performed in the contexts of kernel matrix approximation (Section 4.1), kernel PCA (Section 4.2), and LS-SVM classification (Section 4.3). We use core(TM)-dual PC with 2.13GHz CPU and the codes are in matlab.

4.1. Approximating the Kernel Matrix

We first examine the performance of the low-rank approximation schemes by measuring their approximation errors (in terms of the Frobenius norm) on the kernel matrix. We choose a number of benchmark data sets from the LIBSVM archive[3], summarized in Table 2. Note that our approximation error bound in Proposition 3 applies to most kernel functions (Section 3.4), and preliminary experimental results with these kernels have shown the superiority of our sampling scheme compared with other low-rank approximation methods. However, due to lack of space, we

[2] http://www.di.ens.fr/∼fbach/kernel-ica/index.htm

[3] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

Table 2. A summary of data sets.

data	german	splice	adult1a	dna
size	1000	1000	1605	2000
dimension	24	60	123	180
data	segment	w1a	svmgd1a	satimage
size	2310	2477	3089	4435
dimension	19	300	4	36

will only report results for the Gaussian kernel $K(x, y) = \exp(-\|x - y\|^2/\gamma)$. Here, γ is chosen as the average squared distance between data points and the mean of each data set. We gradually increase the subset size m from 1% to 10% of the data size. To reduce statistical variability, results of methods 2, 3, and 4 are based on averages over 20 repetitions.

The approximation errors are plotted in Figure 1. As can be seen, our algorithm is only inferior to SVD on most data sets. Moreover, though the method in (Drineas & Mahoney, 2005) involves a more complicated probabilistic sampling scheme, its performance is only comparable or sometimes even worse than the Nyström method with simple random sampling. Similar observations have also been reported in the context of SVD (Drineas et al., 2003). ICD seems to be inferior on several data sets. However, for data sets whose kernel spectra decay rapidly to zero[4] (such as the segment, svmguide1a and satimage), ICD can also quickly attain performance comparable to others.

We also examine empirically the relationship between $\mathcal{E}$ and e under different sampling schemes. Figure 2 reports the results on the german data, where $m = 100$ and each sampling scheme is repeated 100 times. As can be seen, there is a strong, positive correlation between $\mathcal{E}$ and e. This is observed on most data and agrees with our error analysis.

4.2. Kernel PCA

In kernel PCA, the key step is to obtain eigenvectors of the centered kernel matrix HKH, where $H = I - \frac{1}{n}11' \in \mathbb{R}^{n\times n}$. Following Proposition 2 of (Ouimet & Bengio, 2005), with the low-rank decomposition $K \simeq GG'$, the centered kernel matrix can be written as $(HG)(HG)'$ or $(G - \bar{G})(G - \bar{G})'$, where $\bar{G} \in R^{n\times m}$ and all its rows equal to the mean of rows in G. Hence the top m eigenvectors can be obtained in $O(m^2 n)$ time according to Proposition 4.

We evaluate the low rank approximation schemes by the embedding onto the top 3 principal directions. We align the approximate embeddings ($\tilde{U}$) with the standard KPCA embedding (U) through a linear transform, and report the minimum misalignment error: $\min_{A\in\mathbb{R}^{3\times 3}} \|U - \tilde{U}A\|_F$. The parameter setting is the same as in Section 4.1, except that we fix $m = 0.05n$ for all the low-rank decomposition algorithms. Again, results of methods 2, 3, 4 in Table 3 are averaged over 20 repetitions. As we can seen, our algorithm is the best on most data sets, next comes the standard Nyström and the method by (Drineas & Mahoney, 2005). The time consumptions of all low-rank approximation schemes are significantly lower than SVD.

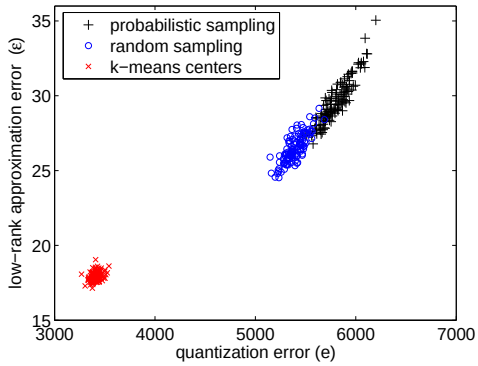


Figure 2. Low-rank approximation error versus quantization error for different sampling schemes.

4.3. Least Squares SVM

Given the kernel matrix K, the training labels $y \in \{\pm 1\}^{n\times 1}$, and the regularization parameter $C > 0$, the LS-SVM classifier $f(x) = \sum_{i=1}^n \alpha_i \phi(x, x_i) + b$ is solved by $b = y'M^{-1}1/y'M^{-1}y$, and $\alpha = M^{-1}(1 - by)$, where 1 is a vector of all ones, and $M = Y(K + I/C)Y$ and $Y = \text{diag}(y)$. Note that $M^{-1} = Y(K + I/C)^{-1}Y$ can be computed efficiently using (13).

We evaluate different low-rank approximation schemes in LS-SVM, using some difficult pairs of the USPS digits[5]. We use Gaussian kernel $\exp(-\|x - y\|^2/\gamma)$ and $C = 0.5$. Table 4 reports the classification performance of the standard LS-SVM, and those with different low-rank approximation schemes, at $m = 0.05n$ and $0.1n$. Again, methods 2, 3, 4 are repeated 20 times. For $m = 0.05n$, our approach is significantly better than methods 1,2,3 with a confidence level that is at least 99.5%. For $m = 0.1n$, ours is also better with a confidence level that is at least 97.5% on the first 7 pairs. For the last 4 pairs, the differences between our approach and methods 1,2,3 are not statistically significant. Note, however, that the testing errors obtained by the various approximation algorithms on these 4 pairs are all close to those of the exact LS-SVM, i.e., all approximation algorithms have reached their possibly best performance.

[4]Note that the (squared) rank-m approximation error of SVD is $\sum_{i=m+1}^n \sigma_i^2$, where σ_i's are the singular values of K sorted in descending order (Golub & Van Loan, 1996). Therefore, if SVD's error in Figure 1 drops rapidly, so does the spectrum of K.

[5]ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data/

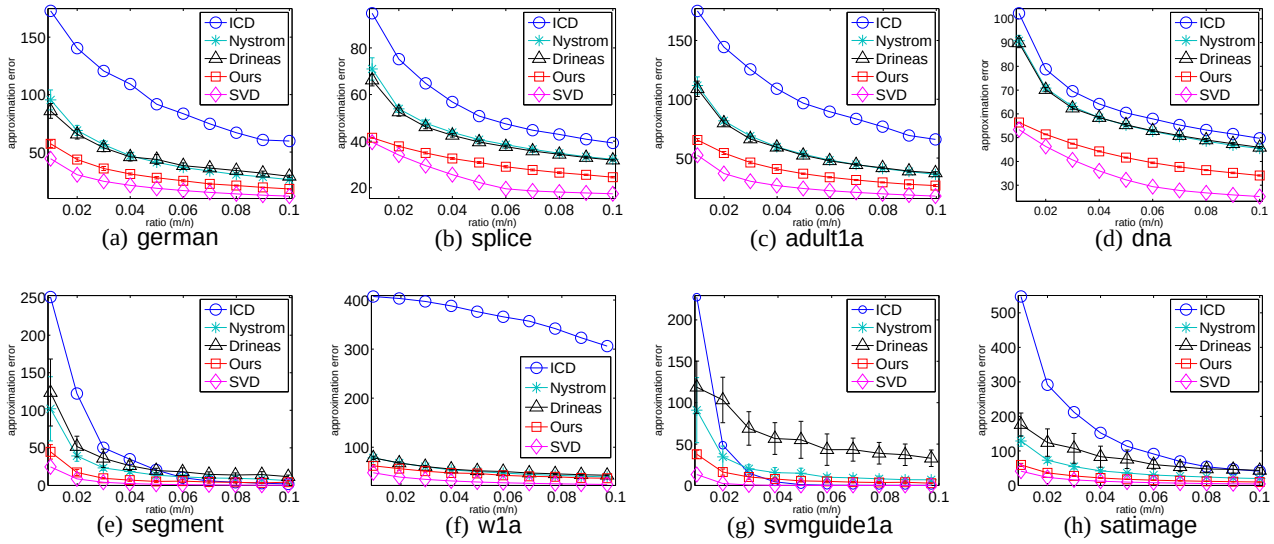


Figure 1. Approximation errors (in terms of the Frobenius norm) on the kernel matrix by different low-rank approximation schemes.

Table 3. Approximation errors and CPU time consumed for the different low-rank approximation schemes in the context of kernel PCA. Due to the lack of space, we do not show the standard deviation of the CPU time.

	approximation error				CPU time (seconds)				
data	Ours	Nyström	Drineas	ICD	SVD	Ours	Nyström	Drineas	ICD
german	$\mathbf{(4.40\pm0.58)}\times10^{-2}$	$(2.64\pm0.58)\times10^{-1}$	$(2.71\pm0.34)\times10^{-1}$	5.11×10^{-1}	27.6	0.8	0.03	0.3	0.09
splice	$\mathbf{(3.44\pm0.43)}\times10^{-1}$	$(1.06\pm0.11)\times10^{0}$	$(1.07\pm0.11)\times10^{0}$	1.27×10^{0}	24.2	0.9	0.05	0.6	0.1
adult1a	$\mathbf{(4.41\pm0.49)}\times10^{-2}$	$(2.86\pm0.42)\times10^{-1}$	$(2.84\pm0.66)\times10^{-1}$	6.19×10^{-1}	134.8	3.0	0.2	4.0	0.7
dna	$\mathbf{(1.88\pm0.21)}\times10^{-1}$	$(1.09\pm0.08)\times10^{0}$	$(1.01\pm0.14)\times10^{0}$	1.17×10^{0}	197.0	6.6	0.5	10.6	1.5
segment	$\mathbf{(7.87\pm4.43)}\times10^{-4}$	$(8.37\pm4.08)\times10^{-3}$	$(1.84\pm0.99)\times10^{-2}$	2.37×10^{-2}	322.8	4.2	0.3	1.8	1.0
w1a	$\mathbf{(1.55\pm0.78)}\times10^{-1}$	$(2.81\pm0.62)\times10^{-1}$	$(6.05\pm3.39)\times10^{-1}$	1.11×10^{0}	394.0	12.8	1.8	35.3	3.6
svmguide1a	$(5.16\pm2.12)\times10^{-4}$	$(3.71\pm2.26)\times10^{-3}$	$(2.78\pm1.60)\times10^{-2}$	$\mathbf{5.07}\times10^{-4}$	650.4	6.7	0.5	2.4	2.3
satimage	$\mathbf{(5.20\pm0.97)}\times10^{-4}$	$(6.19\pm0.28)\times10^{-3}$	$(6.80\pm1.01)\times10^{-2}$	2.47×10^{-2}	2762.8	16.1	1.5	15.9	7.5

5. Conclusion

The Nyström method is a useful technique for low-rank approximation. However, analysis on its key step of choosing the landmark points and especially that in terms of approximation quality is still limited. In this paper, we draw an intuitive but important connection between the Nyström approximation quality and the encoding capacities of landmark points. Our analysis suggests the k-means as a natural sampling scheme. Despite its simplicity, the k-means-based sampling gives encouraging performance when empirically compared with state-of-the-art low-rank approximation techniques. One future direction is to utilize label/side information for task-specific decomposition, where one excellent example is (Bach & Jordan, 2005) in the context of incomplete Cholesky decomposition.

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grant 614907.

References

Achlioptas, D., & McSherry, F. (2001). Fast computation of low rank matrix approximations. *Proceedings of the 23th Annual ACM Symposium on Theory of Computing* (pp. 611 – 618).

Bach, F., & Jordan, M. (2002). Kernel independent component analysis. *Journal of Machine Learning Research, 3*, 1–48.

Bach, F., & Jordan, M. (2005). Predictive low-rank decomposition for kernel methods. *Proceedings of the 22th International Conference on Machine Learning* (pp. 33 – 40).

Baker, C. (1977). *The numerical treatment of integral equations*. Oxford: Clarendon Press.

Belkin, M., & Niyogi, P. (2002). Laplacian eigenmaps and

Table 4. Testing errors (in %) on different pairs of the USPS digits.

		$m = 0.05n$					$m = 0.1n$				
data	LS-SVM	SVD	Ours	Nyström	Drineas	ICD	SVD	Ours	Nyström	Drineas	ICD
1-7	0.97	0.97	0.77±0.09	1.56±0.51	2.69±1.64	4.62	0.97	0.81±0.11	0.94±0.18	1.54±0.35	1.22
2-3	2.47	2.47	2.91±0.55	4.48±1.51	4.54±1.16	4.12	2.47	2.72±0.22	2.77±0.59	2.97±0.59	4.94
2-5	1.67	0.83	1.91±0.62	2.82±1.10	2.98±0.73	5.30	1.67	1.29±0.31	1.57±0.46	1.58±0.51	2.23
3-8	2.71	3.31	3.93±0.84	5.64±1.45	5.52±1.21	6.02	2.71	2.70±0.23	3.78± 0.53	3.97±0.55	4.21
5-8	2.76	2.14	3.38±0.65	4.34±1.52	4.28±1.38	4.29	2.14	2.66±0.34	2.82±0.53	3.12±0.58	4.60
6-8	0.59	0.59	1.04±0.35	2.69±1.34	2.24±0.98	5.65	0.89	0.61±0.25	1.03±0.45	1.18±0.44	5.05
8-9	1.45	1.16	1.04±0.50	2.79±2.00	2.52±0.85	2.33	1.45	1.13±0.23	1.22±0.51	1.79±0.37	2.91
2-7	1.44	0.86	0.90±0.10	1.47±0.64	2.45±1.25	3.76	0.86	0.96±0.17	0.87±0.09	1.08±0.39	2.61
3-5	4.91	4.91	6.53±1.18	7.39±1.39	7.03±1.25	7.36	3.98	5.25±0.64	5.22±0.77	5.49±0.83	5.52
4-7	2.88	2.59	2.54±0.40	3.28±0.96	3.30±1.03	8.06	2.59	2.60±0.26	2.41±0.36	2.43±0.45	5.47
5-6	1.21	0.60	1.57±0.63	2.79±1.25	2.52±1.33	3.93	1.21	1.27±0.32	1.25±0.41	1.48±0.29	2.12

spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems 14*.

Drineas, P., Drinea, E., & Huggins, P. (2003). An experimental evaluation of a Monte-Carlo algorithm for singular value decomposition. *Proceedings of 8th Panhellenic Conference on Informatics* (pp. 279–296).

Drineas, P., & Mahoney, M. W. (2005). On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research, 6*, 2153–2175.

Elkan, E. (2003). Using the triangular inequality to accelerate k-means. *Proceedings of the 21th International Conference on Machine Learning* (pp. 147 – 153).

Fine, S., & Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research, 2*, 243 – 264.

Fowlkes, C., Belongie, S., Chung, F., & Malik, J. (2004). Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*, 214–225.

Frieze, A., Kannan, R., & Vempala, S. (1998). Fast Monte-Carlo algorithms for finding low-rank approximations. *Proceedings of the 39th Annual Symposium on Foundations of Computer Science* (pp. 370 – 378).

Gersho, A., & Gray, R. (1992). *Vector quantization and signal compression*. Boston: Kluwer Academic Press.

Golub, G., & Van Loan, C. (1996). *Matrix computations*. Johns Hopkins University Press. 3rd edition.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2001). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*, 881 – 892.

Lawrence, N. Seeger, M., & Herbrich, R. (2003). Fast sparse Gaussian process methods: The informative vector machine. *Advances in Neural Information Processing Systems*. (pp. 625–632). MIT Press.

Ouimet, M., & Bengio, Y. (2005). Greedy spectral embedding. *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics* (pp. 253–260).

Platt, J. C. (2005). Fastmap, MetricMap, and Landmark MDS are all Nyström algorithms. *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics* (pp. 261–268).

Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation, 10*, 1299–1319.

Smola, A., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. *Proceedings of the 17th International Conference on Machine Learning* (pp. 911 – 918).

Suykens, J., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters, 9*, 293–300.

Williams, C., & Seeger, M. (2000). The effect of the input density distribution on kernel-based classifiers. *Proceedings of the 17th International Conference on Machine Learning* (pp. 1159–1166).

Williams, C., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems 13* (pp. 682 – 688).

Zhang, K., & Kwok, J. (2006). Block-quantized kernel matrix for fast spectral embedding. *Proceedings of the 23rd international conference on Machine learning* (pp. 1097 – 1104).

Estimating Local Optimums in EM Algorithm over Gaussian Mixture Model

Zhenjie Zhang ZHENJIE@COMP.NUS.EDU.SG
Bing Tian Dai DAIBINGT@COMP.NUS.EDU.SG
Anthony K.H. Tung ATUNG@COMP.NUS.EDU.SG
School of Computing, National University of Singapore, Computing 1, Law Link, 117590, Singapore

Abstract

EM algorithm is a very popular iteration-based method to estimate the parameters of Gaussian Mixture Model from a large observation set. However, in most cases, EM algorithm is not guaranteed to converge to the global optimum. Instead, it stops at some local optimums, which can be much worse than the global optimum. Therefore, it is usually required to run multiple procedures of EM algorithm with different initial configurations and return the best solution. To improve the efficiency of this scheme, we propose a new method which can estimate an upper bound on the logarithm likelihood of the local optimum, based on the current configuration after the latest EM iteration. This is accomplished by first deriving some region bounding the possible locations of local optimum, followed by some upper bound estimation on the maximum likelihood. With this estimation, we can terminate an EM algorithm procedure if the estimated local optimum is definitely worse than the best solution seen so far. Extensive experiments show that our method can effectively and efficiently accelerate conventional multiple restart EM algorithm.

1. Introduction

Gaussian Mixture Model (GMM) (McLachlan & Peel, 2000) is a powerful tool in unsupervised learning to model unlabelled data in a multi-dimensional space. However, given an observation data set, estimating the parameters of the underlying Gaussian Mixture Model of the data is not a trivial task, especially when the dimensionality or the number of components is large. Usually, this model estimation problem is transformed to a new problem, which try to find parameters maximizing the likelihood probability on the observations from the Gaussian distributions. In the past decades, EM algorithm (Dempster et al., 1977) has become the most widely method used in the problem of learning Gaussian Mixture Model (Ma et al., 2001; Jordan & Xu, 1995; McLachlan & Krishnan, 1996).

Although EM algorithm can converge in finite iterations, there is no guarantee on the convergence to global optimum. Instead, it usually stops at some local optimum, which can be arbitrarily worse than the global optimum. Although there have been extensive studies on how to avoid bad local optimums, it is still required to run EM algorithm with different random initial configurations and the best local optimum is returned as final result. This leads to a great waste of computation resource since most of the calculations do not have any contribution to the final result.

In this paper, we propose a fast stopping method to overcome the problem of trapping into bad local optimums. Given any current configuration after an EM iteration, our method can estimate an upper bound on the final likelihood of the local optimum current configuration is leading to. Therefore, if the estimated local optimum is definitely not better than the best local optimum achieved in previous runs, current procedure can be terminated immediately.

To facilitate such local optimum estimation, we first prove that a region in the parameter space can definitely cover the unknown local optimum. If a region covers the current configuration and any configuration on the boundary of the region gives lower likelihood than the current one does, we can show that the local optimum is "trapped" in the region; and we call such region as a maximal region. In this paper, we adopt a

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

special type of maximal region, which can be computed efficiently. Since the best likelihood of any configuration in a maximal region can be estimated in relatively short time, it can be decided immediately on whether current procedure still has potential to achieve a better local optimum. In our experiments, such method is shown to greatly improve the efficiency of original EM algorithm for GMM, on both synthetic and real data sets.

The rest of the paper is organized as follows. We first introduce the definitions and related works on Gaussian Mixture Model and EM algorithm in Section 2 . Section 3 proves the local trapping property of EM algorithm on GMM; and Section 4 presents our study on maximal region of local optimum. We propose our algorithm on estimating the likelihood of a local optimum in Section 5 . Section 6 shows some experimental result. Finally, section 7 concludes this paper.

2. Model and Related Works

In this section, we review the basic models of Gaussian Mixture Model, EM algorithm, and some acceleration method proposed for a special type of Gaussian Mixture Model (K-Means Algorithm).

2.1. Gaussian Mixture Model

In GMM model (McLachlan & Peel, 2000), there exist k underlying components $\{\omega_1, \omega_2, \ldots, \omega_k\}$ in a d-dimensional data set. Each component follows some Gaussian distribution in the space. The parameters of the component ω_j include $\mathbf{\Theta}_j = \{\boldsymbol{\mu}_j, \mathbf{\Sigma}_j, \pi_j\}$, in which $\boldsymbol{\mu}_j = (\boldsymbol{\mu}_j[1], \ldots, \boldsymbol{\mu}_j[d])$ is the center of the Gaussian distribution, $\mathbf{\Sigma}_j$ is the covariance matrix of the distribution and π_j is the probability of the component ω_j. Based on the parameters, the probability of a point coming from component ω_j appearing at $\boldsymbol{x} = (x[1], \ldots, x[d])$ can be represented by

$$\Pr(\boldsymbol{x}|\mathbf{\Theta}_j) = \frac{|\mathbf{\Sigma}_j^{-1}|^{1/2}}{(2\pi)^{d/2}} \exp\left\{-\frac{1}{2}(x-\boldsymbol{\mu}_j)^T \mathbf{\Sigma}_j^{-1}(x-\boldsymbol{\mu}_j)\right\}$$

Thus, given the component parameter set $\mathbf{\Theta} = \{\mathbf{\Theta}_1, \mathbf{\Theta}_2, \ldots, \mathbf{\Theta}_k\}$ but without any component information on an observation point $\boldsymbol{x}$, the probability of observing $\boldsymbol{x}$ is estimated by

$$\Pr(\boldsymbol{x}|\mathbf{\Theta}) = \sum_{j=1}^{k} \Pr(\boldsymbol{x}|\mathbf{\Theta}_j)\pi_j$$

The problem of learning GMM is estimating the parameter set $\mathbf{\Theta}$ of the k component to maximize the likelihood of a set of observations $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, which is represented by

$$\Pr(D|\mathbf{\Theta}) = \prod_{i=1}^{n} \Pr(\boldsymbol{x}_i|\mathbf{\Theta}) \tag{1}$$

Based on the parameters of the GMM model, the posterior probability of $\boldsymbol{x}_i$ from component ω_j (or the weight of $\boldsymbol{x}_i$ in component j), τ_{ij}, can be calculated as follows.

$$\tau_{ij} = \frac{\Pr(\boldsymbol{x}_i|\mathbf{\Theta}_j)\pi_j}{\sum_{l=1}^{k} \Pr(\boldsymbol{x}_i|\mathbf{\Theta}_l)\pi_l} \tag{2}$$

To simplify the notations, we use $\mathbf{\Phi}$ to denote the set of all τ_{ij} for any pair of i, j, and use $\Psi(\mathbf{\Theta})$ to denote the corresponding $\mathbf{\Phi}$ based on current configuration $\mathbf{\Theta}$. For ease of analysis, the original optimization problem on equation (1), is usually transformed to an equal maximization problem on the following variable, called log likelihood.

$$\begin{aligned} L(\mathbf{\Theta}, \mathbf{\Phi}) \quad = \quad & \sum_{i=1}^{n}\sum_{j=1}^{k} \tau_{ij}(\ln\frac{\pi_j}{\tau_{ij}} + \frac{\ln|\mathbf{\Sigma}_j^{-1}|}{2} - \\ & \frac{(\boldsymbol{x}_i-\boldsymbol{\mu}_j)^T\mathbf{\Sigma}_j^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)}{2}) \end{aligned} \tag{3}$$

L is actually a function over $\mathbf{\Theta}$ and $\mathbf{\Phi}$, the latter of which is usually optimized according to $\mathbf{\Theta}$. Thus, the problem of learning GMM is finding an optimal parameter set $\mathbf{\Theta}^*$ which can maximize the function $L(\mathbf{\Theta}^*, \Psi(\mathbf{\Theta}^*))$.

2.2. EM Algorithm

EM algorithm (Dempster et al., 1977) is a widely used technique for probabilistic parameter estimation. To estimate $\mathbf{\Theta} = \{\mathbf{\Theta}_1, \ldots, \mathbf{\Theta}_k\}$, it starts with a randomly chosen initial parameter configuration $\mathbf{\Theta}^0$. Then, it keeps invoking iterations to recompute $\mathbf{\Theta}^{t+1}$ based on $\mathbf{\Theta}^t$. Every iteration consists of two steps, E-step and M-step. In E-step, the algorithm computes the expected value of τ_{ij} for each pair of i and j based on $\mathbf{\Theta}^t = \{\mathbf{\Theta}_1^t, \ldots, \mathbf{\Theta}_k^t\}$ and equation (2).

In M-step, the algorithm finds a new group of parameters $\mathbf{\Theta}^{t+1}$ to maximize L based on $\mathbf{\Phi}^t = \{\tau_{ij}^t\}$ and $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$. The details of the update process for $\boldsymbol{\mu}_j$, $\mathbf{\Sigma}_j$ and π_j are listed below.

$$\boldsymbol{\mu}_j^{t+1}[l] = \frac{\sum_{i=1}^{n} \tau_{ij}^t \boldsymbol{x}_i[l]}{\sum_{i=1}^{n} \tau_{ij}^t} \tag{4}$$

$$\boldsymbol{\Sigma}_j^{t+1} = \frac{\sum_{i=1}^n \tau_{ij}^t (\boldsymbol{x}_i - \boldsymbol{\mu}_j^{t+1})(\boldsymbol{x}_i - \boldsymbol{\mu}_j^{t+1})^T}{\sum_{i=1}^n \tau_{ij}^t} \quad (5)$$

$$\pi_j^{t+1} = \frac{\sum_{i=1}^n \tau_{ij}^t}{n} \quad (6)$$

The iteration process stops only when $\boldsymbol{\Theta}^N = \boldsymbol{\Theta}^{N-1}$ after N iterations. We note that both E-step and M-step always improve the objective function, i.e. $L(\boldsymbol{\Theta}^t, \boldsymbol{\Phi}^t) \geq L(\boldsymbol{\Theta}^t, \boldsymbol{\Phi}^{t-1}) \geq L(\boldsymbol{\Theta}^{t-1}, \boldsymbol{\Phi}^{t-1})$. Based on this property, EM-algorithm will definitely converge to some local optimum. The convergence properties of EM algorithm over GMM have been extensively studied in (Xu & Jordan, 1996; Ma et al., 2001).

2.3. K-Means Algorithm and Its Acceleration

K-Means algorithm can be considered as a special problem of GMM learning with several constraints. First, the covariance matrix for each component must be identity matrix. Second, the posterior probability τ_{ij} can only be 0 or 1. Therefore, in E-step of the algorithm, each point is assigned to the closest center under Euclidean distance; whereas in M-step, the set of geometric center of each cluster is used to replace the old set.

With the problem simplification from GMM to K-Means, there have been many methods proposed to accelerate the multiple restart EM algorithm for K-Means. In (Kanungo et al., 2002), for example, Kanungo et al. applied indexing technique to achieve a much more efficient implementation of E-step. In (Elkan, 2003), Elkan accelerated both E-step and M-step by employing triangle inequality of Euclidean distance to reduce the time for distance computations. In (Zhang et al., 2006), Zhang et al. introduced a lower bound estimation on the k-means local optimums to efficiently cut the procedures not leading to good solutions. However, all these methods proposed for k-means algorithm cannot be directly extended to the general GMM. As far as we know, our paper is the first study on acceleration of the multiple restart EM algorithm with robustness guarantee.

To improve the readability of the paper, we summarize all notations in Table 1.

Table 1: Table of Notations

Notation	Description
n	number of points in data
d	dimensionality of data space
k	number of components
ω_j	component j
$\boldsymbol{\Theta}_j$	parameter set of ω_j
$\boldsymbol{\mu}_j$	center of ω_j
$\boldsymbol{\Sigma}_j$	covariance matrix of ω_j
π_j	probability of ω_j
$\boldsymbol{\Theta}$	configuration of all components
$\boldsymbol{x}_i$	ith point in the data
τ_{ij}	posterior probability $\Pr(\omega_j \vert \boldsymbol{x}_i)$
$\boldsymbol{\Phi}$	the set of all τ_{ij}
$\Psi(\boldsymbol{\Theta})$	the optimal $\boldsymbol{\Phi}$ with $\boldsymbol{\Theta}$
$\mathcal{S}$	solutions space for configurations
$L(\boldsymbol{\Theta}, \boldsymbol{\Phi})$	objective log likelihood function
Δ	a parameter for a maximal region

3. Local Trapping Property

In this section, we prove the local trapping property of EM algorithm on GMM. To derive the analysis, we first define a solution space $\mathcal{S}$, containing $(d^2 + d + 1)k$ dimensions where d is the dimensionality of the original data space. Any configuration $\boldsymbol{\Theta}$, either valid or invalid, can be represented by a point in $\mathcal{S}$. Without loss of generality, we use $\boldsymbol{\Theta}$ to denote the configuration as well as the corresponding point in solution space $\mathcal{S}$. The rest of the section will be spent to prove the following theorem.

Theorem 1 *Given a closed region R in the solution space $\mathcal{S}$ covering current configuration $\boldsymbol{\Theta}^t$, EM algorithm converges to a local optimum in R if every configuration $\boldsymbol{\Theta}$ on the boundary of R has $L(\boldsymbol{\Theta}, \Psi(\boldsymbol{\Theta})) < L(\boldsymbol{\Theta}^t, \boldsymbol{\Phi}^t)$*

Given two configurations $\boldsymbol{\Theta}^t$ and $\boldsymbol{\Theta}^{t+1}$ across one EM iteration, we define a path between $\boldsymbol{\Theta}^t$ and $\boldsymbol{\Theta}^{t+1}$ in $\mathcal{S}$ as follows. This path consists of two parts, called P^1 and P^2 respectively. P^1 starts at $\boldsymbol{\Theta}^t$ and ends at $\boldsymbol{\Theta}^\#$, where $\boldsymbol{\Theta}^\# = \{\boldsymbol{\Theta}_j^\#\}$. Here $\boldsymbol{\Theta}_j^\# = \{\boldsymbol{\mu}_j^t, \boldsymbol{\Sigma}_j^\#, \pi_j^t\}$, and $\boldsymbol{\Sigma}_j^\# = \sum_i \tau_{ij}^t (\boldsymbol{x}_i - \boldsymbol{\mu}_j^t)(\boldsymbol{x}_i - \boldsymbol{\mu}_j^t)^T / \sum_i \tau_{ij}^t$. An intermediate configuration between $\boldsymbol{\Theta}^t$ and $\boldsymbol{\Theta}^\#$ is defined as $\boldsymbol{\Theta}^\alpha$, in which $\boldsymbol{\mu}_j^t$ and π_j^t remain the same, while $\boldsymbol{\Sigma}_j^\alpha$ in $\boldsymbol{\Theta}^\alpha$ is $((1-\alpha)(\boldsymbol{\Sigma}^t)^{-1} + \alpha(\boldsymbol{\Sigma}^\#)^{-1})^{-1}$. When α increases from 0 to 1, we can move from $\boldsymbol{\Theta}^t$ to $\boldsymbol{\Theta}^\#$ in the solutions space $\mathcal{S}$. The second part of the path starts at $\boldsymbol{\Theta}^\#$ and ends at $\boldsymbol{\Theta}^{t+1}$. Any intermediate configuration $\boldsymbol{\Theta}^\beta = \{\boldsymbol{\Theta}_j^\beta\}$, where $\boldsymbol{\mu}_j^\beta = (1-\beta)\boldsymbol{\mu}_j^t + \beta\boldsymbol{\mu}_j^{t+1}$, $\boldsymbol{\Sigma}_j^\beta = \sum_i \tau_{ij}^t (\boldsymbol{x}_i - \boldsymbol{\mu}_j^\beta)(\boldsymbol{x}_i - \boldsymbol{\mu}_j^\beta)^T / \sum_i \tau_{ij}^t$, and $\pi_j^\beta = (\pi_j^t)^{1-\beta}(\pi_j^{t+1})^\beta$. Similarly, a continuous movement from $\boldsymbol{\Theta}^\#$ to $\boldsymbol{\Theta}^{t+1}$ can be made by increasing β from 0 to 1. The following lemmas prove that any intermediate configuration on the path is a better solution than $\boldsymbol{\Theta}^t$.

Lemma 1 *Given any intermediate configuration $\mathbf{\Theta}^\alpha$ between $\mathbf{\Theta}^t$ and $\mathbf{\Theta}^\#$, we have $L(\mathbf{\Theta}^\alpha, \Psi(\mathbf{\Theta}^\alpha)) \geq L(\mathbf{\Theta}^t, \mathbf{\Phi}^t)$.*

Proof: By the optimality property of $\Psi(\mathbf{\Theta}^\alpha)$, we have $L(\mathbf{\Theta}^\alpha, \Psi(\mathbf{\Theta}^\alpha)) \geq L(\mathbf{\Theta}^\alpha, \mathbf{\Phi}^t)$.

Since $\mathbf{\Sigma}_j^\# = \sum_i \tau_{ij}^t (\boldsymbol{x}_i - \boldsymbol{\mu}_j^t)(\boldsymbol{x}_i - \boldsymbol{\mu}_j^t)^T / \sum_i \tau_{ij}^t$ is the optimal choice for $\mathbf{\Sigma}_j$ if τ_{ij}^t, $\boldsymbol{\mu}_j^t$ and π_j^t are fixed, we also have $L(\mathbf{\Theta}^\#, \mathbf{\Phi}^t) \geq L(\mathbf{\Theta}^t, \mathbf{\Phi}^t)$.

By the definition of $\mathbf{\Theta}^\alpha$ and the property of $\mathbf{\Theta}^\#$ above, the following equations can be easily derived.

$$\begin{aligned} & L(\mathbf{\Theta}^\alpha, \mathbf{\Phi}^t) \\ = \; & (1-\alpha) L(\mathbf{\Theta}^t, \mathbf{\Phi}^t) + \alpha L(\mathbf{\Theta}^\#, \mathbf{\Phi}^t) \\ \geq \; & L(\mathbf{\Theta}^t, \mathbf{\Phi}^t) \end{aligned}$$

Therefore, it is straightforward to reach the conclusion that $L(\mathbf{\Theta}^\alpha, \Psi(\mathbf{\Theta}^\alpha)) \geq L(\mathbf{\Theta}^t, \mathbf{\Phi}^t)$. □

Lemma 2 *Given any intermediate configuration $\mathbf{\Theta}^\beta$ between $\mathbf{\Theta}^\#$ and $\mathbf{\Theta}^{t+1}$, we have $L(\mathbf{\Theta}^\beta, \Psi(\mathbf{\Theta}^\alpha)) \geq L(\mathbf{\Theta}^t, \mathbf{\Phi}^t)$.*

Proof: Again, the basic inequality $L(\mathbf{\Theta}^\beta, \Psi(\mathbf{\Theta}^\beta)) \geq L(\mathbf{\Theta}^\beta, \mathbf{\Theta}^t)$ holds. Based on this, we can prove the lemma by showing $L(\mathbf{\Theta}^\beta, \mathbf{\Phi}^t) \geq L(\mathbf{\Theta}^\#, \mathbf{\Phi}^t)$, since $L(\mathbf{\Theta}^\#, \mathbf{\Phi}^t) \geq L(\mathbf{\Theta}^t, \mathbf{\Phi}^t)$.

If $\mathbf{\Sigma}_j^\beta = \sum_i \tau_{ij}^t (\boldsymbol{x}_i - \boldsymbol{\mu}_j^\beta)(\boldsymbol{x}_i - \boldsymbol{\mu}_j^\beta)^T / \sum_i \tau_{ij}^t$, a very interesting result is that $\sum_j \sum_i \tau_{ij}^t (\boldsymbol{x}_i - \boldsymbol{\mu}_j^\beta)^T (\mathbf{\Sigma}_j^\beta)^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j^\beta)$ remains constant for any β, as is shown below.

$$\sum_j \sum_i \tau_{ij}^t (\boldsymbol{x}_i - \boldsymbol{\mu}_j^\beta)^T (\mathbf{\Sigma}_j^\beta)^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_j^\beta) = nd$$

Therefore, for any $\mathbf{\Theta}^\beta$, we only need to consider the sum $\sum_i \sum_j \tau_{ij}^t \left(\ln(\pi_j^\beta / \tau_{ij}^t) - \ln(|\mathbf{\Sigma}_j^\beta|)/2 \right)$.

By the definition of π_j^β, since $\pi_j^\beta = (\pi_j^t)^{1-\beta} (\pi_j^{t+1})^\beta$, we have $\ln \pi_j^\beta = (1-\beta) \ln \pi_j^t + \beta \ln \pi_j^{t+1}$. Then,

$$\sum_{i=1}^n \tau_{ij}^t \ln \frac{\pi_j^\beta}{\tau_{ij}^t} = (1-\beta) \sum_{i=1}^n \tau_{ij}^t \ln \frac{\pi_j^t}{\tau_{ij}^t} + \beta \sum_{i=1}^n \tau_{ij}^t \ln \frac{\pi_j^{t+1}}{\tau_{ij}^t} \tag{7}$$

Therefore, $\sum_i \sum_j \tau_{ij}^t \ln \frac{\pi_j^\beta}{\tau_{ij}^t} \geq \sum_i \sum_j \tau_{ij}^t \ln \frac{\pi_j^t}{\tau_{ij}^t}$, since $\sum_{i=1}^n \tau_{ij}^t \ln \frac{\pi_j^{t+1}}{\tau_{ij}^t} \geq \sum_{i=1}^n \tau_{ij}^t \ln \frac{\pi_j^t}{\tau_{ij}^t}$.

On the other hand, based on the definition of $\mathbf{\Sigma}_j^\beta$, we can prove that

$$\begin{aligned} \mathbf{\Sigma}_j^\beta \; = \; & \sum_{i=1}^n \tau_{i,j}^t (\boldsymbol{x}_i - \boldsymbol{\mu}_j^t)(\boldsymbol{x}_i - \boldsymbol{\mu}_j^t)^T + \\ & (\beta^2 - 2\beta) (\sum_{i=1}^n \tau_{i,j}^t) (\boldsymbol{\mu}_j^{t+1} - \boldsymbol{\mu}_j^t)(\boldsymbol{\mu}_j^{t+1} - \boldsymbol{\mu}_j^t)^T \end{aligned}$$

Since $\beta^2 - 2\beta \leq 0$ for any β between 0 and 1, $\ln |\mathbf{\Sigma}_j^\beta| \leq \ln |\mathbf{\Sigma}_j^\#|$. And thus, we have $-\ln |\mathbf{\Sigma}_j^\beta|/2 \geq -\ln |\mathbf{\Sigma}_j^\#|/2$.

Combing the results above, we reach the conclusion that $L(\mathbf{\Theta}^\beta, \mathbf{\Phi}^t) \geq L(\mathbf{\Theta}^\#, \mathbf{\Phi}^t)$, leading to the correctness of the lemma. □

Proof for Theorem 1

Proof: We prove the theorem by contradiction. If R satisfies the boundary condition but EM algorithm converges to some configuration out of R in $\mathcal{S}$, there is at least one pair of configurations $\{\mathbf{\Theta}^s, \mathbf{\Theta}^{s+1}\}$ that $\mathbf{\Theta}^s$ is in R but $\mathbf{\Theta}^{s+1}$ is not. By setting up the path $\{\mathbf{\Theta}^\alpha\} \cap \{\mathbf{\Theta}^\beta\}$ between $\mathbf{\Theta}^s$ and $\mathbf{\Theta}^{s+1}$ as defined above, we know there is at least one $\mathbf{\Theta}^\alpha$ ($\mathbf{\Theta}^\beta$) that $\mathbf{\Theta}^\alpha$ ($\mathbf{\Theta}^\beta$) is exactly on the boundary of R. By Lemma 1 (Lemma 2), we know $L(\mathbf{\Theta}^\alpha, \Psi(\mathbf{\Theta}^\alpha)) \geq L(\mathbf{\Theta}^s, \mathbf{\Phi}^s)$ ($L(\mathbf{\Theta}^\beta, \Psi(\mathbf{\Theta}^\beta)) \geq L(\mathbf{\Theta}^s, \mathbf{\Phi}^s)$). On the other hand, any $\mathbf{\Theta}^\alpha$ or $\mathbf{\Theta}^\beta$ is better than $\mathbf{\Theta}^t$ by the definition of R. This leads to the contradiction, since $L(\mathbf{\Theta}^s, \mathbf{\Phi}^s) > L(\mathbf{\Theta}^t, \mathbf{\Phi}^t)$. □

4. Maximal Region

Based on Theorem 1, we define the concept of *Maximal Region* in GMM as follows. Given the current configuration $\mathbf{\Theta}^t$, a region R in S is the maximal region for $\mathbf{\Theta}^t$, if (1) R covers $\mathbf{\Theta}^t$, and (2) any boundary configuration $\mathbf{\Theta}$ of R has $L(\mathbf{\Theta}, \Psi(\mathbf{\Theta})) < L(\mathbf{\Theta}^t, \mathbf{\Phi}^t)$, by Theorem 1, EM algorithm converges to some local optimum in R.

Given the current configuration $\mathbf{\Theta}^t$, there are infinite number of valid maximal regions in the solution space, most of which are hard to verify and manipulate. To facilitate efficient computation, we further propose a special class of maximal regions. Given $\mathbf{\Theta}^t$ and a positive real value $\Delta < 1$, we define a closed region $R(\mathbf{\Theta}^t, \Delta) \subseteq S$ as the union of any configuration $\mathbf{\Theta}$, each $\theta_j = \{\boldsymbol{\mu}_j, \mathbf{\Sigma}_j, \pi_j\}$ of which satisfies all of the conditions below: (1) $(1-\Delta)\pi_j^t \leq \pi_j \leq (1+\Delta)\pi_j^t$; (2) $-\Delta \leq tr((\mathbf{\Sigma}_j^t)^{-1}(\mathbf{\Sigma}_j^t) - I) \leq \Delta$; and (3) $(\boldsymbol{\mu}_j - \boldsymbol{\mu}_j^t)^T (\mathbf{\Sigma}_j^t)^{-1} (\boldsymbol{\mu}_j - \boldsymbol{\mu}_j^t) \leq \Delta^2$; where $tr(M)$ denotes the trace of the matrix M and I is the identity matrix of dimension d.

Theorem 2 *Any configuration* $\mathbf{\Theta}$ *on the boundary of* $R(\mathbf{\Theta}^t, \Delta)$ *has* $L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1}) \leq L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1}) - n \min_j \pi_j^t \Delta^2/6$.

Proof: Given any $R(\mathbf{\Theta}^t, \Delta)$, any configuration on the boundary must satisfy one of the following conditions for at least one j $(1 \leq j \leq k)$: (1) $(1-\Delta)\pi_j^t = \pi_j$; (2) $\pi_j^t = (1+\Delta)\pi_j^t$; (3) $|tr(\mathbf{\Sigma}_j^{-1}(\mathbf{\Sigma}_j^t) - I)| = \Delta$; and (4) $(\boldsymbol{\mu}_j - \boldsymbol{\mu}_j^t)^T(\mathbf{\Sigma}_j^t)^{-1}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_j^t) = \Delta^2$.

If $\mathbf{\Theta}$ satisfies condition (1) for some component l, $L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1})$ is maximized if $\boldsymbol{\mu}_j^t$ and $\mathbf{\Sigma}_j^t$ remain unchanged for all j, while $\pi_j = \frac{1-(1-\Delta)\pi_l^t}{1-\pi_l^t}\pi_j^t$ for all $j \neq l$. Therefore, we have the following upper bound.

$$\begin{aligned}
& L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1}) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1}) \\
\leq\ & n\pi_l^t \ln(1-\Delta) + n(1-\pi_l^t)\ln\frac{1-(1-\Delta)\pi_l^t}{1-\pi_l^t} \\
=\ & n\pi_l^t \ln(1-\Delta) + n(1-\pi_l^t)\ln\left(1+\frac{\Delta\pi_l^t}{1-\pi_l^t}\right) \\
\leq\ & n\pi_l^t\left(-\Delta - \frac{\Delta^2}{2}\right) + n(1-\pi_l^t)\frac{\Delta\pi_l^t}{1-\pi_l^t} \\
=\ & -\frac{n\pi_l^t\Delta^2}{2}
\end{aligned}$$

The second inequality from the bottom is achieved by applying Taylor expansion on $\ln(1-\Delta)$. By iterating l with all k components, we have $L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1}) \leq L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1}) - \min_j n\pi_l^t\Delta^2/2$.

If $\mathbf{\Theta}$ satisfies condition (2) for some component l, $L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1})$ can be maximized similarly. We have

$$\begin{aligned}
& L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1}) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1}) \\
\leq\ & n\pi_l^t \ln(1+\Delta) + n(1-\pi_l^t)\ln\frac{1-(1+\Delta)\pi_l^t}{1-\pi_l^t} \\
=\ & n\pi_l^t \ln(1+\Delta) + n(1-\pi_l^t)\ln\left(1-\frac{\Delta\pi_l^t}{1-\pi_l^t}\right) \\
\leq\ & n\pi_l^t\left(\Delta - \frac{\Delta^2}{2} + \frac{\Delta^3}{3}\right) + n(1-\pi_l^t)\frac{\Delta\pi_l^t}{1-\pi_l^t} \\
\leq\ & -\frac{n\pi_l^t\Delta^2}{6}
\end{aligned}$$

Again, the third inequality from the bottom is due to Taylor expansion of $\ln(1+\Delta)$. The last inequality is because $\Delta^3 \leq \Delta^2$ for any $0 \leq \Delta \leq 1$.

If $\mathbf{\Theta}$ satisfies condition (3) for some component l, L is maximized if all other parameters remain the same. Thus,

$$\begin{aligned}
& L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1}) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1}) \\
\leq\ & \frac{n\pi_l^t}{2}\left(\ln|\mathbf{\Sigma}_l^{-1}\mathbf{\Sigma}_l^t| - tr\left(\left(\mathbf{\Sigma}_l^{-1} - (\mathbf{\Sigma}_l^t)^{-1}\right)\mathbf{\Sigma}_l^t\right)\right) \\
=\ & \frac{n\pi_l^t}{2}\left(tr\left(\log\left(\mathbf{\Sigma}_l^{-1}\mathbf{\Sigma}_l^t\right)\right) - tr\left(\mathbf{\Sigma}_l^{-1}\mathbf{\Sigma}_l^t - I\right)\right) \\
\leq\ & \frac{n\pi_l^t}{2}\left(-\frac{tr(\mathbf{\Sigma}_l^{-1}\mathbf{\Sigma}_l^t - I)^2}{2}\right) \\
=\ & -\frac{n\pi_l^t\Delta^2}{4}
\end{aligned}$$

The fourth equality is derived by the definitions of $\mathbf{\Sigma}_l^t$ and π_l^t. And the second inequality from bottom is due to the taylor expansion on the logarithm matrix.

Finally, if $\mathbf{\Theta}$ satisfies condition (4) for some component l, L is maximized if $\mathbf{\Sigma}_l = \frac{\sum \tau_{il}(\boldsymbol{x}_i - \boldsymbol{\mu}_l)(\boldsymbol{x}_i - \boldsymbol{\mu}_l)^T}{\sum \tau_{il}}$. In this case, $\sum_i \tau_l^{t-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T\mathbf{\Sigma}_j^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j) = \sum_i \tau_l^{t-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j^t)^T(\mathbf{\Sigma}_j^t)^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_j^t) = n\pi_j d$. Thus, the only difference on the log likelihood function L stems from the change on the determinant of the covariance matrix.

$$\begin{aligned}
& L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1}) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1}) \\
\leq\ & \sum_{i=1}^n \frac{\tau_{il}}{2}\left(-\ln|\mathbf{\Sigma}_l| + \ln|\mathbf{\Sigma}_l^t|\right) \\
=\ & \sum_{i=1}^n \frac{\tau_{il}}{2}\left(-\ln|\mathbf{\Sigma}_l^t + (\boldsymbol{\mu}_l - \boldsymbol{\mu}_l^t)(\boldsymbol{\mu}_l - \boldsymbol{\mu}_l^t)^T)| + \ln|\mathbf{\Sigma}_l^t|\right) \\
\leq\ & \sum_{i=1}^n \frac{\tau_{il}}{2}\left(-\ln\left(|\mathbf{\Sigma}_l^t| + |(\boldsymbol{\mu}_l - \boldsymbol{\mu}_l^t)(\boldsymbol{\mu}_l - \boldsymbol{\mu}_l^t)^T|\right) + \ln|\mathbf{\Sigma}_l^t|\right) \\
\leq\ & \sum_{i=1}^n \frac{\tau_{il}}{2}\left(-\ln\left(|\mathbf{\Sigma}_l^t| + \Delta^2|\mathbf{\Sigma}_l^t|\right) + \ln|\mathbf{\Sigma}_l^t|\right) \\
=\ & -\sum_{i=1}^n \frac{\tau_{il}\ln(1+\Delta^2)}{2} \\
\leq\ & -\frac{n\pi_l^t\Delta^2}{2}
\end{aligned}$$

The fourth inequality applies the property of positive definite matrices that $|A+B| > |A| + |B|$ (Lutkepohl, 1996) .

In all of the four cases, the reduction on the likelihood function L is at least $-\frac{n\min_j \pi_j^t\Delta^2}{6}$. This completes the proof of the theorem. □

Last theorem implies that $\mathbf{\Theta}$ will reduce the log likelihood function by at least $n\min_j^t \pi_j\Delta^2/6$ if $\mathbf{\Phi}$ remains $\mathbf{\Phi}^{t-1}$. The following question is how much we can increase the likelihood if we use the optimal $\Psi(\mathbf{\Theta})$ instead of $\mathbf{\Phi}^{t-1}$.

Lemma 3 *Given* $\mathbf{\Theta} \in R(\mathbf{\Theta}^t, \Delta)$, $Pr(\boldsymbol{x}_i|\theta_j)\pi_j$ *is no larger than* $(1+\Delta)^{1.5}\frac{|(\mathbf{\Sigma}_j^t)^{-1}|^{1/2}}{(2\pi)^{d/2}}\exp\{-(1-\Delta)M(\boldsymbol{x},\mathbf{\Theta}_j)^2/2\}\pi_j^t$, *where* $M(\boldsymbol{x},\mathbf{\Theta}_j)$ *is*

$$\max\left\{\sqrt{((x-\boldsymbol{\mu}_j^t)^T(\mathbf{\Sigma}_j^t)^{-1}(x-\boldsymbol{\mu}_j^t))}-\Delta, 0\right\}$$

Lemma 4 *Given* $\mathbf{\Theta} \in R(\mathbf{\Theta}^t, \Delta)$, $Pr(\boldsymbol{x}_i|\theta_j)\pi_j$ *is no smaller than* $(1-\Delta)^{1.5}\frac{|(\mathbf{\Sigma}_j^t)^{-1}|^{1/2}}{(2\pi)^{d/2}}\exp\{-(1+\Delta)N(\boldsymbol{x},\mathbf{\Theta}_j)^2/2\}\pi_j^t$, *where* $N(\boldsymbol{x},\mathbf{\Theta}_j)$ *is*

$$\sqrt{((x-\boldsymbol{\mu}_j^t)^T(\mathbf{\Sigma}_j^t)^{-1}(x-\boldsymbol{\mu}_j^t))}+\Delta$$

.

The proofs of Lemma 3 and Lemma 4 are available in (Zhang et al., 2008).

Lemma 5 *Given a region* $R(\mathbf{\Theta}^t, \Delta)$ *as defined above, an upper bound,* U_{ij}, *on* $\tau_{ij} \in \Psi(\mathbf{\Theta})$ *for any* $\mathbf{\Theta} \in R(\mathbf{\Theta}^t, \Delta)$ *can be calculated in constant time.*

Proof: For any configuration $\mathbf{\Theta}$ on the boundary of $R(\mathbf{\Theta}^t, \Delta)$, the optimal value of τ_{ij} can be calculated by equation (2). By Lemma 3 and Lemma 4, we can compute $\max_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\omega_j)\pi_j$ and $\min_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\omega_j)\pi_j$. Therefore,

$$\tau_{ij} \le U_{ij} = \frac{\max_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\omega_j)\pi_j}{\sum_l \min_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\omega_l)\pi_j}$$

$$\tau_{ij} \ge L_{ij} = \frac{\min_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\omega_j)\pi_j}{\sum_l \max_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\omega_l)\pi_j}$$

The calculations can be finished in constant time with the two sums pre-computed. □

By Lemma 5, the increase upper bound from $L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1})$ to $L(\mathbf{\Theta}, \Psi(\mathbf{\Theta}))$ can be calculated by the following equation.

$$\begin{aligned} & L(\mathbf{\Theta}, \Psi(\mathbf{\Theta})) - L(\mathbf{\Theta}, \mathbf{\Phi}^{t-1}) \\ \le\ & \ln \sum\sum U_{ij} \max_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\omega_j)\pi_j - \\ & \ln \sum\sum \tau_{ij} \max_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\omega_j)\pi_j \end{aligned} \quad (8)$$

The following theorem gives a sufficient condition on a maximal region $R(\mathbf{\Theta}^t, \Delta)$ for some positive value Δ.

Theorem 3 $R(\mathbf{\Theta}^t, \Delta)$ *is a maximal region for* $\mathbf{\Theta}^t$ *if* $\ln \frac{\sum_i\sum_j U_{ij}\max_{\mathbf{\Theta}}\Pr(\boldsymbol{x}_i|\omega_j)\pi_j}{\sum_i\sum_j \tau_{ij}\max_{\mathbf{\Theta}}\Pr(\boldsymbol{x}_i|\omega_j)\pi_j} - n\min\pi_j^t\Delta^2/6 < L(\mathbf{\Theta}^t, \mathbf{\Phi}^t) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1})$

Proof: By the definition of L, we have

$$L(\mathbf{\Theta}, \Psi(\mathbf{\Theta})) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1}) \le \ln \frac{\sum_i\sum_j U_{ij}\Pr(\boldsymbol{x}_i|\omega_j)\pi_j}{\sum_i\sum_j \tau_{ij}\Pr(\boldsymbol{x}_i|\omega_j)\pi_j}$$

It is not hard to verify that the derivative of $L(\mathbf{\Theta}, \Psi(\mathbf{\Theta})) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1})$ to $\Pr(\boldsymbol{x}_i|\omega_j)\pi_j$ is always positive. Therefore, the equation above can be maximized if we employ the maximum value of $\Pr(\boldsymbol{x}_i|\omega_j)\pi_j$. Based on the analysis above, we know that

$$\begin{aligned} & L(\mathbf{\Theta}, \Psi(\mathbf{\Theta})) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1}) \\ \le\ & \ln \frac{\sum_i\sum_j U_{ij}\max_{\mathbf{\Theta}}\Pr(\boldsymbol{x}_i|\omega_j)\pi_j}{\sum_i\sum_j \tau_{ij}\max_{\mathbf{\Theta}}\Pr(\boldsymbol{x}_i|\omega_j)\pi_j} \end{aligned}$$

By Theorem 2, $L(\mathbf{\Theta}, \Phi^{t-1}) - L(\mathbf{\Theta}^t, \Phi^{t-1}) \le n\min_j \pi_j^t\Delta^2/6$. Therefore, by Theorem 1, $L(\mathbf{\Theta}, \Psi(\mathbf{\Theta})) < L(\mathbf{\Theta}^t, \mathbf{\Phi}^t)$ if the condition of the theorem is satisfied. □

For any local optimum $\mathbf{\Theta}^*$ in the maximal region $R(\mathbf{\Theta}^t, \Delta)$, the following theorem upper bound the likelihood function $L(\mathbf{\Theta}^*, \Psi(\mathbf{\Theta}^*))$.

Theorem 4 *Given a valid maximal region* $R(\mathbf{\Theta}^t, \Delta)$, *if EM algorithm converges to local optimum* $\mathbf{\Theta}^*$, $L(\mathbf{\Theta}^*, \Psi(\mathbf{\Theta}^*)) \le L(\mathbf{\Theta}^t, \mathbf{\Phi}^t) + n\min\pi_j^t\Delta^2/6$.

Proof: Since $\sum_i\sum_j(U_{ij} - \tau_{ij}^t)\max_{\mathbf{\Theta}}\Pr(\omega_j|\boldsymbol{x}_i) < n\min\pi_j^t\Delta^2/6$ by Theorem 3 and $L(\mathbf{\Theta}, \Psi(\mathbf{\Theta})) - L(\mathbf{\Theta}, \mathbf{\Phi}^t) \le \sum_i\sum_j(U_{ij} - \tau_{ij}^t)\max_{\mathbf{\Theta}}\Pr(\omega_j|\boldsymbol{x}_i)$, we have $L(\mathbf{\Theta}, \Psi(\mathbf{\Theta})) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^t) \le L(\mathbf{\Theta}, \Psi(\mathbf{\Theta})) - L(\mathbf{\Theta}, \mathbf{\Phi}^t) \le n\min\pi_j^t\Delta^2/6$. □

5. Algorithm

Theorem 3 provides an easy way to verify whether $R(\mathbf{\Theta}^t, \Delta)$ is a valid maximal region. On the other hand, Theorem 4 implies that a smaller Δ can lead to tighter bound on the likelihood function L. However, it is not necessary to get the tightest bound on local optimum in our algorithm, since the goal of our algorithm is estimating whether the current configuration can lead to better solution. Instead, we set Δ as $\min\left\{1, \sqrt{\frac{6(L^*-L(\mathbf{\Theta}^t,\mathbf{\Phi}^t))}{n\min\pi_j^t}}\right\}$, where L^* is the best result we have seen so far. This Δ is the maximal one of all Δ values, which are able to prune the current EM procedure by Theorem 4

The details of the algorithm are summarized in Algo 1. In this algorithm, conventional M-step and E-step

Algorithm 1 New Iteration(Data Set D, Current $\mathbf{\Theta}^{t-1}$, current $\mathbf{\Phi}^{t-1}$, component number k, sample number m, current best result L^*)

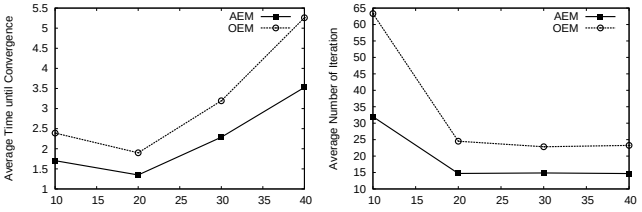
1: Compute new $\mathbf{\Theta}^t$ by M-Step.
2: Compute new $\mathbf{\Phi}^t$ by E-Step.
3: **if** $L(\mathbf{\Theta}^t, \mathbf{\Phi}^t) < L^*$ **then**
4: Let $\Delta = \min\left\{1, \sqrt{\frac{6(L^* - L(\mathbf{\Theta}^t, \mathbf{\Phi}^t))}{n \min \pi_j^t}}\right\}$
5: $S = X = 0$
6: **for** each $\boldsymbol{x}_i$ **do**
7: **for** each dimension j **do**
8: Get $l_{ij} = \max_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\theta_j)\pi_j$ by Lemma 3.
9: Get $s_{ij} = \min_{\mathbf{\Theta}} \Pr(\boldsymbol{x}_i|\theta_j)\pi_j$ by Lemma 4.
10: Get U_{ij} by Lemma 5.
11: $S+ = U_{ij} * l_{ij}$
12: $X+ = \tau_{ij}^{t-1} * l_{ij}$
13: **end for**
14: **end for**
15: **if** $\ln S - \ln X - n \min \pi_j \Delta^2/6 < L(\mathbf{\Theta}^t, \mathbf{\Phi}^t) - L(\mathbf{\Theta}^t, \mathbf{\Phi}^{t-1})$ **then**
16: Stop the current procedure of EM algorithm.
17: **end if**
18: **else**
19: Return $(\mathbf{\Theta}^t, \mathbf{\Phi}^t)$
20: **end if**

are invoked first. If the current configuration is better than the best solution we have seen before, there is no need to test the upper bound of the local optimum. Otherwise, the value of Δ is set according to $\min \pi_j^t$, L^* and $L(\mathbf{\Theta}^T, \mathbf{\Phi}^t)$. For each point and each component, l_{ij}, s_{ij} and U_{ij} are collected according to Lemma 3, Lemma 4 and Lemma 5 respectively. With the information collected from each point, the condition of Theorem 1 can be tested. If this condition is satisfied, we can assert that current local optimum can never be better than L^*, leading to the termination of the current procedure.

6. Experiments

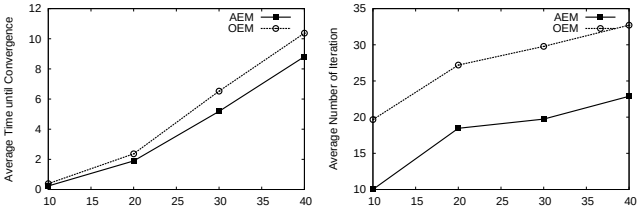
In this section, we report the experimental results on the comparison of our accelerated EM algorithm (AEM) and the conventional EM algorithm (OEM). We note that in our implementation, either AEM or OEM will be stopped if it does not converge after 100 iterations.

We employ both synthetic and real data sets in our empirical studies. The synthetic data sets are generated in a d-dimensional unit cube. There are k components in the space. Each component follows some Gaussian distribution. The center, size and covariance matrix



(a) CPU Time (b) Number of Iterations

Figure 1: Performance vs. varying dimensionality



(a) CPU Time (b) Number of Iterations

Figure 2: Performance vs. varying component number

of each component are randomly generated independently. Two real data sets are also tested, including *Cloud* and *Spam*, both of which are available on UCI Machine Learning Repository. The *Cloud* data set consists of 1024 points in 10-dimensional space, while *Spam* data set has 4601 points in 58 dimensions. Both of the real data set are normalized before being used in our experiments.

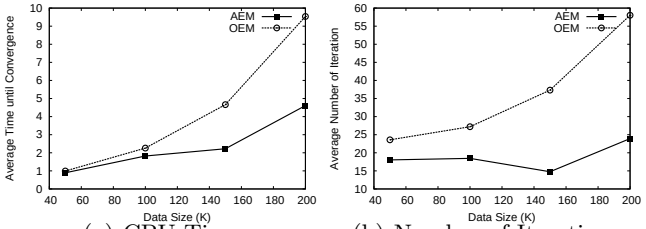
Two performance measurements are recorded in our experiments, including CPU time and number of iterations. An algorithm is supposed to be better if it spends less CPU time and invokes less time of iterations. All of the experiments are compiled and run on a Fedora Core 6 linux machine with 3.0 GHz Processor, 1GB of memory and GCC 4.1.2.

In the experiments on the data sets, we test the performances of the algorithms with varying dimensionality D, number of components k, and the number of points in the data S. The default setting of our experiments is $D = 20$, $k = 20$, and $S = 100$K. The time of EM restart is fixed at 100 in all tests. More experimental results are available in the technical report (Zhang et al., 2008).

6.1. Results on Synthetic Data

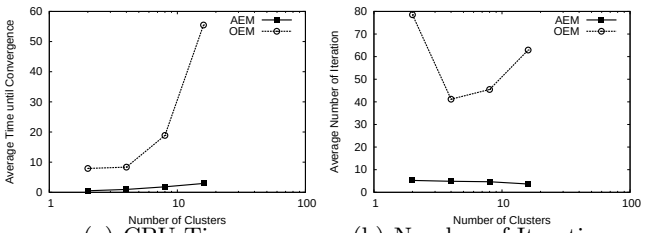
In Figure 1(a) and Figure 1(b), we present the experimental result by varying the dimensionality from 10 to 40. The results show that AEM is much more efficient than OEM. On data set with low dimensionality, AEM is almost two times faster than OEM, both on the CPU time and the number of iterations. The advantage is very obvious, even on high dimensional space.

The results of our experiments on varying component



(a) CPU Time (b) Number of Iterations

Figure 3: Performance vs. varying data size



(a) CPU Time (b) Number of Iterations

Figure 4: Performance vs. varying component number on *Spam* data

number are summarized in Figure 2(a) and Figure 2(b). From the figures, we can see the performance advantage of AEM is stable, with the increase of component number. The CPU time and number of iterations on AEM is only about half of those of OEM.

As is shown in Figure 3(a), Figure 3(b) AEM has much better performance than OEM when we increase the data size from 50K to 200K. AEM can detect those worse local optimums much earlier, if there are more data available. The number of iterations invoked by AEM is almost the same, even when the data has been doubled. The ratio of CPU time is more stable when the data size is larger.

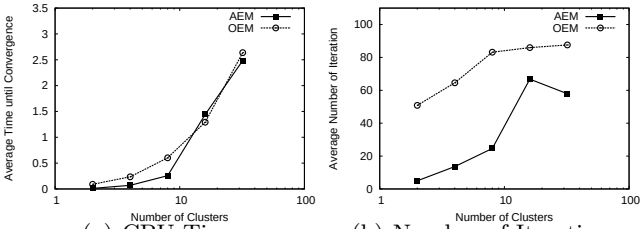
6.2. Results on Real Data

On *Spam* data set, AEM also show great advantage over OEM, on CPU time (Figure 4(a)) and on the number of iterations (Figure 4(b)). AEM is more efficient than OEM by one magnitude, independent to the number of components k.

However, the experiments on *Cloud* data set show quite different results than the pervious results, where AEM has very limited advantage. We believe the difference on the results stems from normalization problem.

7. Conclusion

In this paper, we propose a new acceleration method for multiple restart EM algorithm over Gaussian Mixture Model. We derive an upper bound on the local optimum of the likelihood function in the solution space. This upper bound computation turns out to be both efficient and effective in pruning un-promising procedures of EM algorithm.



(a) CPU Time (b) Number of Iterations

Figure 5: Performance vs. varying component number on *Cloud* data

Acknowledgement: The authors would like to acknowledge the valuable comments from the reviewers of ICML 2008.

References

Dempster, A. P., Laird, N. M., & Robin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm (with discussion). *Journal of Royal Statistical Society B*, *39*, 1–38.

Elkan, C. (2003). Using the triangle inequality to accelerate k-means. *ICML* (pp. 147–153).

Jordan, M. I., & Xu, L. (1995). Convergence results for the em approach to mixtures of experts architectures. *Neural Networks*, *8*, 1409–1431.

Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., & Wu, A. (2002). An efficient k-means clustering algorithm: analysis and implementation.

Lutkepohl, H. (1996). *Handbook of matrices.* John Wiley & Sons Ltd.

Ma, J., Xu, L., & Jordan, M. I. (2001). Asymptotic convergence rate of the em algorithm for gaussian mixtures. *Neural Computation*, *12*, 2881–2907.

McLachlan, G., & Krishnan, T. (1996). *The em algorithm and extensions.* Wiley-Interscience.

McLachlan, G., & Peel, D. (2000). *Finite mixture models.* Wiley-Interscience.

Xu, L., & Jordan, M. I. (1996). On convergence properties of the em algorithm for gaussian mixtures. *Neural Computation*, *8*, 129–151.

Zhang, Z., Dai, B. T., & Tung, A. K. H. (2006). On the lower bound of local optimums in k-means algorithm. *ICDM* (pp. 775–786).

Zhang, Z., Dai, B. T., & Tung, A. K. H. (2008). Estimating local optimums in em algorithm over gaussian mixture model. *http://www.comp.nus.edu.sg/∼zhangzh2/papers/em.pdf.*

Efficient MultiClass Maximum Margin Clustering

Bin Zhao ZHAOBINHERE@HOTMAIL.COM
Fei Wang FEIWANG03@MAILS.TSINGHUA.EDU.CN
Changshui Zhang ZCS@MAIL.TSINGHUA.EDU.CN
State Key Laboratory of Intelligent Technologies and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

Abstract

This paper presents a *cutting plane* algorithm for multiclass *maximum margin clustering* (*MMC*). The proposed algorithm constructs a nested sequence of successively tighter relaxations of the original *MMC* problem, and each optimization problem in this sequence could be efficiently solved using the *constrained concave-convex procedure* (*CCCP*). Experimental evaluations on several real world datasets show that our algorithm converges much faster than existing *MMC* methods with guaranteed accuracy, and can thus handle much larger datasets efficiently.

1. Introduction

Clustering (Duda et al., 2001; Shi & Malik, 2000; Ding et al., 2001) aims at dividing data into groups of similar objects, *i.e.* clusters. Recently, motivated by the success of large margin methods in supervised learning, (Xu et al., 2004) proposed *maximum margin clustering* (*MMC*), which borrows the idea from the *support vector machine* theory and aims at finding the maximum margin hyperplane which can separate the data from different classes in an unsupervised way.

Technically, what *MMC* does is to find a way to label the samples by running an *SVM* implicitly, and the *SVM* margin obtained would be maximized over all possible labelings (Xu et al., 2004). However, unlike supervised large margin methods which are usually formulated as *convex optimization* problems, *maximum margin clustering* is a *non-convex integer optimization* problem, which is much more difficult to solve.

Several attempts have been made to solve the *maximum margin clustering* problem in polynomial time. (Xu et al., 2004) and (Valizadegan & Jin, 2007) made several relaxations to the original *MMC* problem and reformulated it as a *semi-definite programming* (*SDP*) problem. However, even with the recent advances in interior point methods, solving *SDP*s is still computationally very expensive. Consequently, the algorithms can only handle very small datasets containing several hundreds of samples. More recently, *Zhang et al.* utilized alternating optimization techniques to solve the *MMC* problem (Zhang et al., 2007), in which the *maximum margin clustering* result is obtained by solving a series of *SVM* training problems. However, there is no guarantee on how fast it can converge and the algorithm is still time demanding on large scale datasets. Moreover, the methods described above can only handle binary clustering problems (Zhao et al., 2008), and there are significant complications to deriving an effective *maximum margin clustering* approach for the multiclass scenario[1]. Therefore, how to efficiently solve the *multiclass MMC* problem to make it capable of clustering large scale dataset is a very challenging research topic.

In this paper, we propose a *cutting plane multiclass maximum margin clustering* algorithm *CPM3C*. Specifically, the algorithm constructs a nested sequence of successively tighter relaxations of the original *multiclass MMC* problem, and each optimization problem in this sequence could be efficiently solved using the *constrained concave-convex procedure* (*CCCP*). Moreover, we show that the computational time of *CPM3C* scales roughly linearly with the dataset size. Our experimental evaluations on several real world datasets show that *CPM3C* performs better than existing *MMC* methods, both in efficiency and accuracy.

[1] It should be noted that (Xu & Schuurmans, 2005) proposed a multiclass extension for *MMC*, however, their algorithm has a time complexity of $O(n^7)$, which renders it impractical for real world datasets.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

The rest of this paper is organized as follows. Section 2 will show the *CPM3C* algorithm in detail, and the time complexity analysis of *CPM3C* will be presented in section 3. Section 4 presents the experimental results on several real world datasets, followed by the conclusions in section 5.

2. Cutting Plane Multiclass Maximum Margin Clustering

We will formally present the *cutting plane multiclass maximum margin clustering* (*CPM3C*) algorithm in this section.

2.1. Multiclass Maximum Margin Clustering

Maximum margin clustering (*MMC*) extends the theory of *support vector machine* (*SVM*) to the unsupervised scenario. Specifically, given a point set $\mathcal{X} = \{\mathbf{x}_1, \quad, \mathbf{x}_n\}$ and their labels $\mathbf{y} = (y_1, \ldots, y_n) \in \{1, \ldots, k\}^n$, *SVM* defines a weight vector $\mathbf{w}_p$ for each class $p \in \{1, \ldots, k\}$ and classifies sample $\mathbf{x}$ by $y^* = \arg\max_{y \in \{1,\ldots,k\}} \mathbf{w}_y^T \mathbf{x}$ with the weight vectors obtained as follows [2](Crammer & Singer, 2001)

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi} \quad & \frac{1}{2} \sum_{p=1}^{k} ||\mathbf{w}_p||^2 + \sum_{i=1}^{n} \xi_i \\ s.t. \quad & \forall i = 1, \ldots, n, r = 1, \ldots, k \\ & \mathbf{w}_{y_i}^T \mathbf{x}_i + \ _{y_i,r} \quad \mathbf{w}_r^T \mathbf{x}_i \quad 1 \quad \xi_i \end{aligned} \tag{1}$$

where the data samples in $\mathcal{X}$ are mapped into a high (possibly infinite) dimensional feature space, and by using the kernel trick, this mapping could be done implicitly. However, in those cases where kernel trick cannot be applied, it is possible to compute the coordinates of each sample in the *kernel PCA basis* (Schölkopf et al., 1999) according to kernel K. Therefore, throughout the rest of this paper, we use $\mathbf{x}_i$ to denote the sample mapped by the kernel function.

Instead of finding a large margin classifier given labels on the data as in *SVM*, *MMC* targets to find a labeling that would result in a large margin classifier (Xu et al., 2004). That is to say, if we subsequently run an *SVM* with the labeling obtained from *MMC*, the margin would be maximal among all possible labelings. *multiclass MMC* could be formulated as follows:

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi,\mathbf{y}} \quad & \frac{1}{2} \sum_{p=1}^{k} ||\mathbf{w}_p||^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \\ s.t. \quad & \forall i = 1, \ldots, n, r = 1, \ldots, k \\ & \mathbf{w}_{y_i}^T \mathbf{x}_i + \ _{y_i,r} \quad \mathbf{w}_r^T \mathbf{x}_i \quad 1 \quad \xi_i \end{aligned} \tag{2}$$

[2]Although we focus on the *multiclass SVM* formulation of (Crammer & Singer, 2001), our method can be directly applied to other *multiclass SVM* formulations.

where $\sum_{i=1}^{n} \xi_i$ is divided by n to better capture how the regularization parameter scales with the dataset size. Different from *SVM*, where the class labels are given and the only variables are $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$, *MMC* targets to find not only the optimal weight vectors, but also the optimal labeling vector $\mathbf{y}^*$.

2.2. Cutting Plane Algorithm

In this section, we will reformulate problem (2) to reduce the number of variables. Specifically,

Theorem 1 *Problem (2) is equivalent to*

$$\begin{aligned} \min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi} \quad & \frac{1}{2} \sum_{p=1}^{k} ||\mathbf{w}_p||^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \\ s.t. \quad & \forall i = 1, \ldots, n, r = 1, \ldots, k \\ & \sum_{p=1}^{k} \mathbf{w}_p^T \mathbf{x}_i \prod_{q=1,q\neq p}^{k} I_{(\mathbf{w}_p^T \mathbf{x}_i > \mathbf{w}_q^T \mathbf{x}_i)} \\ & + \prod_{q=1,q\neq r}^{k} I_{(\mathbf{w}_r^T \mathbf{x}_i > \mathbf{w}_q^T \mathbf{x}_i)} \quad \mathbf{w}_r^T \mathbf{x}_i \quad 1 \quad \xi_i \end{aligned} \tag{3}$$

where $I(\ )$ is the indicator function *and the label for sample $\mathbf{x}_i$ is determined as*

$$y_i = \sum_{p=1}^{k} p \prod_{q=1,q\neq p}^{k} I_{(\mathbf{w}_p^T \mathbf{x}_i > \mathbf{w}_q^T \mathbf{x}_i)} \tag{4}$$

Proof. We will show that for every $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$ the smallest feasible $\sum_{i=1}^{n} \xi_i$ are identical for problem (2) and problem (3), and their corresponding labeling vectors are the same. For a given $(\mathbf{w}_1, \ldots, \mathbf{w}_k)$, the ξ_i in problem (2) can be optimized individually. According to the constraint in problem (2),

$$\xi_i \quad 1 \quad (\mathbf{w}_{y_i}^T \mathbf{x}_i + \ _{y_i,r} \quad \mathbf{w}_r^T \mathbf{x}_i), \ \forall r = 1, \ldots, k \tag{5}$$

As the objective is to minimize $\frac{1}{n} \sum_{i=1}^{n} \xi_i$, the optimal value for ξ_i is

$$\xi_i^{(1)} = \min_{y_i=1,\ldots,k} \max_{r=1,\ldots,k} \{1 \quad (\mathbf{w}_{y_i}^T \mathbf{x}_i + \ _{y_i,r} \quad \mathbf{w}_r^T \mathbf{x}_i)\} \tag{6}$$

and we denote the corresponding class label by $y_i^{(1)}$. Without loss of generality, we assume the following relationship

$$\mathbf{w}_{i_1}^T \mathbf{x}_i \quad \mathbf{w}_{i_2}^T \mathbf{x}_i \quad \ldots \quad \mathbf{w}_{i_k}^T \mathbf{x}_i \tag{7}$$

where $(i_1, i_2, \ldots, i_k)$ is a permutation of $(1, 2, \ldots, k)$. For $y_i \neq i_1$, $\max_{r=1,\ldots,k}\{1 \quad (\mathbf{w}_{y_i}^T \mathbf{x}_i + \delta_{y_i,r} \quad \mathbf{w}_r^T \mathbf{x}_i)\}$ 1, while for $y_i = i_1$, $\max_{r=1,\ldots,k}\{1 \quad (\mathbf{w}_{y_i}^T \mathbf{x}_i + \delta_{y_i,r} \quad \mathbf{w}_r^T \mathbf{x}_i)\} \leq 1$, therefore, $y_i^{(1)} = i_1$ and

$$\begin{aligned} \xi_i^{(1)} &= \max_{r=1,\ldots,k} \{1 \quad (\mathbf{w}_{i_1}^T \mathbf{x}_i + \ _{i_1,r} \quad \mathbf{w}_r^T \mathbf{x}_i)\} \\ &= \max\{0, 1 \quad (\mathbf{w}_{i_1}^T \mathbf{x}_i \quad \mathbf{w}_{i_2}^T \mathbf{x}_i)\} \end{aligned} \tag{8}$$

Similarly, for problem (3), the optimal value for ξ_i is

$$\begin{aligned}\xi_i^{(2)} &= \max_{r=1,\ldots,k}\left\{1 \quad \left[\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i\prod_{q=1,q\neq p}^{k}I_{(\mathbf{w}_p^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)}\right.\right.\\ &\quad \left.\left.+\prod_{q=1,q\neq r}^{k}I_{(\mathbf{w}_r^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)} \quad \mathbf{w}_r^T\mathbf{x}_i\right]\right\} \qquad (9)\\ &= \max_{r=1,\ldots,k}\{1 \quad (\mathbf{w}_{i_1}^T\mathbf{x}_i + \ _{i_1,r} \quad \mathbf{w}_r^T\mathbf{x}_i)\}\\ &= \max\{0,1 \quad (\mathbf{w}_{i_1}^T\mathbf{x}_i \quad \mathbf{w}_{i_2}^T\mathbf{x}_i)\}\end{aligned}$$

and the class label could be calculated as

$$y_i^{(2)} = \sum_{p=1}^{k} p\prod_{q=1,q\neq p}^{k}I_{(\mathbf{w}_p^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)} = i_1 \qquad (10)$$

Therefore, the objective functions of both optimization problems are equivalent for any $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$ with the same optimal ξ_i, and consequently so are their optima. Moreover, their corresponding labeling vectors $\mathbf{y}$ are the same. Hence, we proved that problem (2) is equivalent to problem (3). □

By reformulating problem (2) as problem (3), the number of variables involved is reduced by n, but there are still n slack variables ξ_i in problem (3). Define $\mathbf{e}_p$ as the $k \quad 1$ vector with only the p-th element being 1 and others 0, $\mathbf{e}_0$ as the $k \quad 1$ zero vector and $\mathbf{e}$ as the all one vector. To further reduce the number of variables involved in the optimization problem, we have the following theorem

Theorem 2 *Problem (3) can be equivalently formulated as problem (11), with* $\xi^* = \frac{1}{n}\sum_{i=1}^{n}\xi_i^*$.

$$\begin{aligned}\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi\geq 0} \quad & \frac{1}{2} \quad \sum_{p=1}^{k}||\mathbf{w}_p||^2+\xi \qquad (11)\\ s.t. \quad & \forall \mathbf{c}_i \in \{\mathbf{e}_0,\mathbf{e}_1,\ldots,\mathbf{e}_k\},\ i=1,\ldots,n\\ & \frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}+\sum_{p=1}^{k}c_{ip}(z_{ip} \quad \mathbf{w}_p^T\mathbf{x}_i)\right\}\\ & \frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e} \quad \xi\end{aligned}$$

where $z_{ip} = \prod_{q=1,q\neq p}^{k} I_{(\mathbf{w}_p^T\mathbf{x}_i>\mathbf{w}_q^T\mathbf{x}_i)}\ \forall i = 1,\ldots,n; p = 1\ldots,k$ *and each constraint* $\mathbf{c}$ *is represented as a* $k \quad n$ *matrix* $\mathbf{c} = (\mathbf{c}_1,\ldots,\mathbf{c}_n)$.

Proof. To justify the above theorem, we will show that problem (3) and problem (11) have the same objective value and an equivalent set of constraints. Specifically, we will prove that for every $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$, the smallest feasible ξ and $\sum_{i=1}^{n}\xi_i$ are related by $\xi = \frac{1}{n}\sum_{i=1}^{n}\xi_i$. This means, with $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$ fixed, $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi)$ and $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi_i)$ are optimal solutions to problem (3) and (11) respectively, and they result in the same objective function value.

For any given $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$, the ξ_i in problem (3) can be optimized individually and the optimum is achieved as

$$\xi_i^{(2)} = \max\{0,1 \quad (\mathbf{w}_{i_1}^T\mathbf{x}_i \quad \mathbf{w}_{i_2}^T\mathbf{x}_i)\} \qquad (12)$$

where we assume the relation in (7) holds.

Similarly for problem (11), the optimal ξ is

$$\begin{aligned}\xi^{(3)} = \max_{\mathbf{c}_1,\ldots,\mathbf{c}_n\in\{\mathbf{e}_0,\ldots,\mathbf{e}_k\}}\left\{\frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e} \quad \frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}\right.\right. \qquad (13)\\ \left.\left.+\sum_{p=1}^{k}c_{ip}(z_{ip} \quad \mathbf{w}_p^T\mathbf{x}_i)\right]\right\}\end{aligned}$$

Since each $\mathbf{c}_i$ are independent in Eq.(13), they can be optimized individually. Therefore,

$$\begin{aligned}\xi^{(3)} &= \frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{c}_i}\{\mathbf{c}_i^T\mathbf{e} \quad \mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip} \quad \sum_{p=1}^{k}c_{ip}(z_{ip} \quad \mathbf{w}_p^T\mathbf{x}_i)\}\\ &= \frac{1}{n}\sum_{i=1}^{n}\max \quad 0, \max_{p=1,\ldots,k}[1 \quad (\mathbf{w}_{i_1}^T\mathbf{x}_i + \ _{i_1,p} \quad \mathbf{w}_p^T\mathbf{x}_i)]\\ &= \frac{1}{n}\sum_{i=1}^{n}\max\left\{0,\max[0,1 \quad (\mathbf{w}_{i_1}^T\mathbf{x}_i \quad \mathbf{w}_{i_2}^T\mathbf{x}_i)]\right\}\\ &= \frac{1}{n}\sum_{i=1}^{n}\max\{0,1 \quad (\mathbf{w}_{i_1}^T\mathbf{x}_i \quad \mathbf{w}_{i_2}^T\mathbf{x}_i)\} = \frac{1}{n}\sum_{i=1}^{n}\xi_i^{(2)}\end{aligned}$$

Hence, for any $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$, the objective functions for problem (3) and problem (11) have the same value given the optimal ξ and ξ_i. Therefore, the optima of the two optimization problems are the same. □

Putting theorems 1 and 2 together, we could therefore solve problem (11) instead to find the same *maximum margin clustering* solution, with the number of variables reduced by $2n \quad 1$. Although the number of variables in problem (11) is greatly reduced, the number of constraints increases from nk to $(k+1)^n$. The algorithm we propose in this paper targets to find a small subset of constraints from the whole set of constraints in problem (11) that ensures a sufficiently accurate solution. Specifically, we employ an adaptation of the *cutting plane* algorithm (Kelley, 1960) to solve problem (11), where we construct a nested sequence of successively tighter relaxations of problem (11). Moreover, we can prove theoretically (see section 3) that we can always find a polynomially sized subset of constraints, with which the solution of the relaxed problem fulfills all constraints from problem (11) up to a precision of . That is to say, the remaining exponential number of constraints are guaranteed to be violated by no more than , without the need for explicitly adding them to the optimization problem (Tsochantaridis et al., 2005). Specifically, the *CPM3C* algorithm keeps a subset Ω of working constraints and

computes the optimal solution to problem (11) subject to the constraints in Ω. The algorithm then adds the most violated constraint in problem (11) into Ω. In this way, a successively strengthening approximation of the original *MMC* problem is constructed by a cutting plane that cuts off the current optimal solution from the feasible set (Kelley, 1960). The algorithm stops when no constraint in (11) is violated by more than . Here, the feasibility of a constraint is measured by the corresponding value of ξ, therefore, the most violated constraint is the one that results in the largest ξ. Since each constraint in problem (11) is represented by a k n matrix $\mathbf{c}$, then we have

Theorem 3 *Define* $p^* = \arg\max_p(\mathbf{w}_p^T\mathbf{x}_i)$ *and* $r^* = \arg\max_{r\neq p^*}(\mathbf{w}_r^T\mathbf{x}_i)$ *for* $i = 1,\ldots,n$, *the most violated constraint could be calculated as follows*

$$\mathbf{c}_i = \begin{array}{ll} \mathbf{e}_{r^*} & if\ (\mathbf{w}_{p^*}^T\mathbf{x}_i \quad \mathbf{w}_{r^*}^T\mathbf{x}_i) < 1 \\ \mathbf{0} & otherwise \end{array},\ i = 1,\ldots,n \qquad (14)$$

Proof. The most violated constraint is the one that would result in the largest ξ. As each $\mathbf{c}_i$ in the constraint is independent, in order to fulfill all constraints in problem (11), the value of ξ is as follows

$$\begin{aligned}\xi^* &= \frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{c}_i}\{\mathbf{c}_i^T\mathbf{e} \quad \mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip} \quad \sum_{p=1}^{k}c_{ip}(z_{ip} \quad \mathbf{w}_p^T\mathbf{x}_i)\} \\ &= \frac{1}{n}\sum_{i=1}^{n}\max_{\mathbf{c}_i}\{\mathbf{c}_i^T[\mathbf{e} \quad \mathbf{w}_{p^*}^T\mathbf{x}_i\mathbf{e} \quad \mathbf{z}_i + \mathbf{t}_i]\}\end{aligned}$$

where $\mathbf{t}_i = (\mathbf{w}_1^T\mathbf{x}_i,\ldots,\mathbf{w}_k^T\mathbf{x}_i)^T$. Since $\mathbf{c}_i \in \{\mathbf{e}_0,\ldots,\mathbf{e}_k\}$, $\mathbf{c}_i$ selects the largest element of the vector $\mathbf{e}$ $\mathbf{w}_{p^*}^T\mathbf{x}_i\mathbf{e}$ $\mathbf{z}_i + \mathbf{t}_i$, which could be calculated as 1 $(\mathbf{w}_{p^*}^T\mathbf{x}_i \quad \mathbf{w}_{r^*}^T\mathbf{x}_i)$. Therefore, the most violated constraint $\mathbf{c}$ that results in the largest ξ^* could be calculated as in Eq.(14). □

The *CPM3C* algorithm iteratively selects the most violated constraint under the current weight vectors and adds it into the working constraint set Ω until no violation of constraints is detected. Moreover, if a point $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi)$ fulfills all constraints up to precision

$$\begin{aligned}&\forall \mathbf{c}_i \in \{\mathbf{e}_0,\mathbf{e}_1,\ldots,\mathbf{e}_k\}^n,\ i = 1,\ldots,n \qquad (15)\\ &\frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip} + \sum_{p=1}^{k}c_{ip}(z_{ip} \quad \mathbf{w}_p^T\mathbf{x}_i)\right\} \quad \frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e} \quad \xi\end{aligned}$$

then the point $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi + \ )$ is feasible. Furthermore, as in the objective function of problem (11), there is a single slack variable ξ that measures the clustering loss. Hence, we could simply select the stopping criterion as all samples satisfying the inequality (15). Then, the approximation accuracy of this approximate solution is directly related to the training loss.

2.3. Enforcing the Class Balance Constraint

In 2-class *maximum margin clustering*, a trivially "optimal" solution is to assign all patterns to the same class, and the resultant margin will be infinite (Xu et al., 2004). Similarly, for the multiclass scenario, a large margin can always be achieved by eliminating classes (Xu & Schuurmans, 2005). Therefore, we add the following class balance constraints to avoid the trivially "optimal" solutions

$$l \quad \sum_{i=1}^{n}\mathbf{w}_p^T\mathbf{x}_i \quad \sum_{i=1}^{n}\mathbf{w}_q^T\mathbf{x}_i \quad l,\ \forall p,q = 1,\ldots,k \qquad (16)$$

where l 0 controls the class imbalance. Therefore, *multiclass maximum margin clustering* with working constraint set Ω could be formulated as follows

$$\begin{aligned}\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi\geq 0} \quad & \frac{1}{2} \quad \sum_{p=1}^{k}||\mathbf{w}_p||^2 + \xi \qquad (17)\\ s.t. \quad & \frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip} + \sum_{p=1}^{k}c_{ip}(z_{ip} \quad \mathbf{w}_p^T\mathbf{x}_i)\right\}\\ & \frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e} \quad \xi,\ \forall[\mathbf{c}_1,\ldots,\mathbf{c}_n] \in \Omega\\ & l \quad \sum_{i=1}^{n}\mathbf{w}_p^T\mathbf{x}_i \quad \sum_{i=1}^{n}\mathbf{w}_q^T\mathbf{x}_i \quad l,\ \forall p,q = 1,\ldots,k\end{aligned}$$

Before getting into details of solving problem (17), we first present the *CPM3C* approach in Algorithm 1.

Algorithm 1 Cutting Plane Multiclass MMC

Initialize $\Omega = \phi$
repeat
  Solve problem (17) for $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$ under the current working constraint set Ω and select the most violated constraint $\mathbf{c}$ with Eq.(14). Set $\Omega = \Omega \cup \{\mathbf{c}\}$.
until $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$ satisfies $\mathbf{c}$ up to precision

2.4. Optimization via the *CCCP*

In each iteration of the *CPM3C* algorithm, we need to solve problem (17) to obtain the optimal classifying hyperplanes under the current working constraint set Ω. Although the objective function in (17) is convex, the constraints are not, and this makes problem (17) difficult to solve. Fortunately, the *constrained concave-convex procedure (CCCP)* is just designed to solve those optimization problems with a concave-convex objective function under concave-convex constraints (Smola et al., 2005). In the following, we will show how to utilize *CCCP* to solve problem (17).

The objective function in (17) and the second constraint are convex. Moreover, the first constraint is, although non-convex, the difference of two convex functions. Hence, we can solve (17) with *CCCP*. Notice that while $\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}+\sum_{p=1}^{k}c_{ip}z_{ip}\right]$ is convex, it is a non-smooth function of $(\mathbf{w}_1,\ldots,\mathbf{w}_k)$. To use *CCCP*, we need to calculate the *subgradients*:

$$\partial_{\mathbf{w}_r}\left\{\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}+\sum_{p=1}^{k}c_{ip}z_{ip}\right]\right\}_{\mathbf{w}=\mathbf{w}^{(t)}} \quad (18)$$
$$= \frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e}z_{ip}^{(t)}\mathbf{x}_i \qquad \forall r=1,\ldots,k$$

Given an initial point $(\mathbf{w}_1^{(0)},\ldots,\mathbf{w}_k^{(0)})$, *CCCP* computes $(\mathbf{w}_1^{(t+1)},\ldots,\mathbf{w}_k^{(t+1)})$ from $(\mathbf{w}_1^{(t)},\ldots,\mathbf{w}_k^{(t)})$ by replacing $\frac{1}{n}\sum_{i=1}^{n}\left[\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}+\sum_{p=1}^{k}c_{ip}z_{ip}\right]$ in the constraint with its first order Taylor expansion at $(\mathbf{w}_1^{(t)},\ldots,\mathbf{w}_k^{(t)})$, i.e.

$$\frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}{\mathbf{w}_p^{(t)}}^T\mathbf{x}_i z_{ip}^{(t)}+\sum_{p=1}^{k}c_{ip}z_{ip}^{(t)}\right\} \quad (19)$$
$$+\frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}(\mathbf{w}_p \quad \mathbf{w}_p^{(t)})^T\mathbf{x}_i z_{ip}^{(t)}$$
$$= \frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}^{(t)}+\sum_{p=1}^{k}c_{ip}z_{ip}^{(t)}\right\}$$

By substituting the above first-order Taylor expansion into problem (11), we obtain the following *quadratic programming (QP)* problem:

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi\geq 0} \ \frac{1}{2}\sum_{p=1}^{k}||\mathbf{w}_p||^2+\xi \quad (20)$$
$$s.t. \ \forall[\mathbf{c}_1,\ldots,\mathbf{c}_n]\in\Omega$$
$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{c}_i^T\mathbf{e} \quad \xi+\frac{1}{n}\sum_{i=1}^{n}\sum_{p=1}^{k}c_{ip}\mathbf{w}_p^T\mathbf{x}_i$$
$$\frac{1}{n}\sum_{i=1}^{n}\left\{\mathbf{c}_i^T\mathbf{e}\sum_{p=1}^{k}\mathbf{w}_p^T\mathbf{x}_i z_{ip}^{(t)}+\sum_{p=1}^{k}c_{ip}z_{ip}^{(t)}\right\} \quad 0$$
$$l \quad \sum_{i=1}^{n}\mathbf{w}_p^T\mathbf{x}_i \quad \sum_{i=1}^{n}\mathbf{w}_q^T\mathbf{x}_i \quad l, \forall p,q=1,\ldots,k$$

Moreover, the dual problem of (20) is a *QP* problem with $|\Omega|+2$ variables and could be solved in polynomial time, where $|\Omega|$ denotes the total number of constraints in Ω. Putting everything together, according to the formulation of the *CCCP* (Smola et al., 2005), we solve problem (17) with the approach presented in Algorithm 2, where we set the stopping criterion in *CCCP* as the difference between two iterations less than % and set % = 0.01, which means the current objective function is larger than 1 % of the objective function in last iteration, since *CCCP* decreases the objective function monotonically.

Algorithm 2 Solve problem (17) with CCCP

Initialize $\mathbf{w}_p=\mathbf{w}_p^0$ for $p=1,\ldots,k$.
repeat
  Find $(\mathbf{w}_1^{t+1},\ldots,\mathbf{w}_k^{t+1})$ as the solution to the *quadratic programming* problem (20).
  Set $\mathbf{w}_p=\mathbf{w}_p^{t+1}, p=1,\ldots,k$
until stopping criterion satisfied.

2.5. Theoretical Analysis

We provide the following theorem regarding the correctness of the *CPM3C* algorithm.

Theorem 4 *For any dataset* $\mathcal{X}=(\mathbf{x}_1,\ldots,\mathbf{x}_n)$ *and any* >0, *if* $(\mathbf{w}_1^*,\ldots,\mathbf{w}_k^*,\xi^*)$ *is the optimal solution to problem (11) with the class balance constraint, then our* CPM3C *algorithm returns a point* $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi)$ *for which* $(\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi+\ )$ *is feasible in problem (11) and satisfies the class balance constraint. Moreover, the corresponding objective value is better than the one corresponds to* $(\mathbf{w}_1^*,\ldots,\mathbf{w}_k^*,\xi^*)$.

Based on the above theorem, indicates how close one wants to be to the error rate of the best classifying hyperplanes and can thus be used as the stopping criterion (Joachims, 2006).

3. Time Complexity Analysis

In this section, we will provide analysis on the time complexity of *CPM3C*. For the high-dimensional *(say, d-dimensional)* sparse data commonly encountered in applications like text mining and bioinformatics, we assume each sample has only $s \ll d$ non-zero features, *i.e.*, s implies the sparsity, while for non-sparse data, by simply setting $s=d$, all our theorems still hold.

Theorem 5 *Each iteration of* CPM3C *takes time* $O(snk)$ *for a constant working set size* $|\Omega|$.

Moreover, for the binary clustering scenario, we have the following theorem

Theorem 6 *For any* >0, >0, *and any dataset* $\mathcal{X}=\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$ *with samples belonging to two different classes, the* CPM3C *algorithm terminates after adding at most* $\frac{R}{2}$ *constraints, where* R *is a constant number independent of* n *and* s.

It is true that the number of constraints can potentially explode for small values of , however, experi-

ence with *CPM3C* shows that relatively large values of are sufficient without loss of clustering accuracy. Since the number of iterations in *CPM3C* (with $k = 2$) is bounded by $\frac{R}{2}$, a constant independent of n and s, and each iteration of the algorithm takes time $O(snk)$ ($O(sn)$ for the binary clustering scenario), we arrive at the following theorem

Theorem 7 *For any dataset $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with n samples belonging to 2 classes and sparsity of s, and any fixed value of > 0 and > 0, the* CPM3C *algorithm takes time $O(sn)$ to converge.*

For the multiclass scenario, experimental results shown in section 4 also demonstrate that the computational time of *CPM3C* scales roughly linearly with the dataset size n.

4. Experiments

In this section, we will validate the accuracy and efficiency of the *CPM3C* algorithm on several real world datasets. Moreover, we will also analyze the scaling behavior of *CPM3C* with the dataset size and the sensitivity of *CPM3C* to , both in accuracy and efficiency. All the experiments are performed with MATLAB 7.0 on a 1.66GHZ Intel Core$^{\text{TM}}$2 Duo PC running Windows XP with 1.5GB main memory.

4.1. Datasets

We use eight datasets in our experiments, which are selected to cover a wide range of properties: **Digits, Letter** and **Satellite** from the **UCI** repository, **MNIST**[3], **20 newsgroup**[4], **WebKB**[5], **Cora** (McCallum et al., 2000) and **RCVI** (Lewis et al., 2004). In order to compare *CPM3C* with other *MMC* algorithms which can only perform binary clustering, we choose the first two classes from **Letter** and **Satellite**. For the **20 newsgroup** dataset, we choose the topic *rec* which contains *autos, motorcycles, baseball* and *hockey* from the version 20-news-18828. For **WebKB**, we select a subset consists of about 6000 web pages from computer science departments of four schools (Cornell, Texas, Washington, and Wisconsin). For **Cora**, we select a subset containing the research paper of subfield data structure (DS), hardware and architecture (HA), machine learning (ML), operating system (OS) and programming language (PL). For **RCVI**, we use the data samples with the highest four topic codes (CCAT, ECAT, GCAT and MCAT) in the "Topic Codes" hierarchy in the training set.

[3]http://yann.lecun.com/exdb/mnist/

[4]http://people.csail.mit.edu/jrennie/20Newsgroups/

[5]http://www.cs.cmu.edu/∼WebKB/

Table 1. Descriptions of the datasets.

Data	Size (n)	Feature (N)	Class	Sparsity
Letter	1555	16	2	98.9%
UCIDig	1797	64	10	51.07%
UCISat	2236	36	2	100%
MNIST	70000	784	10	19.14%
Cora-DS	751	6234	9	0.68%
Cora-HA	400	3989	7	1.1%
Cora-ML	1617	8329	7	0.58%
Cora-OS	1246	6737	4	0.75%
Cora-PL	1575	7949	9	0.56%
WK-CL	827	4134	7	2.32%
WK-TX	814	4029	7	1.97%
WK-WT	1166	4165	7	2.05%
WK-WC	1210	4189	7	2.16%
20-news	3970	8014	4	0.75%
RCVI	21251	47152	4	0.16%

4.2. Comparisons and Clustering Results

Besides our *CPM3C* algorithm, we also implements some other competitive algorithms and present their results for comparison. Specifically, we use **K-Means (KM)** and **Normalized Cut (NC)** as baselines, and also compared with **Maximum Margin Clustering (MMC)** (Xu et al., 2004), **Generalized Maximum Margin Clustering (GMC)** (Valizadegan & Jin, 2007) and **Iterative Support Vector Regression (SVR)** (Zhang et al., 2007) which all aim at clustering data with the maximum margin hyperplane. Technically, for **k-means**, the cluster centers are initialized randomly. For **NC**, the implementation is the same as in (Shi & Malik, 2000), and the width of the Gaussian kernel is set by exhaustive search from the grid $\{0.1\sigma_0, 0.2\sigma_0, \ldots, \sigma_0\}$, where σ_0 is the range of distance between any two data points in the dataset. Moreover, for **MMC** and **GMC**, the implementation is the same as in (Xu et al., 2004; Xu & Schuurmans, 2005) and (Valizadegan & Jin, 2007) respectively. Furthermore, the implementation code for **SVR** is downloaded from http://www.cse.ust.hk/ twinsen and the initialization is based on *k-means* with randomly selected initial data centers, and the width of the Gaussian kernel is set in the same way as in **NC**.

In the experiments, we set the number of clusters equal to the true number of classes k for all the clustering algorithms. To assess clustering accuracy, we follow the strategy used in (Xu et al., 2004) where we first take a set of labeled data, remove the labels for all data samples and run the clustering algorithms, then we label each of the resulting clusters with the majority class according to the original training labels, and finally measure the number of correct classifications made by each clustering. Moreover, we also calculate the *Rand Index* (Rand, 1971) for each clustering result. The *clustering accuracy* and *Rand index* results are summarized in table 2 and table 3 respectively,

Table 2. Clustering accuracy(%) comparisons.

Data	KM	NC	MMC	GMC	SVR	CPM3C
Dig 3-8	94.68	65.00	90.00	94.40	96.64	**96.92**
Dig 1-7	94.45	55.00	68.75	97.8	99.45	**100.0**
Dig 2-7	96.91	66.00	98.75	99.50	**100.0**	**100.0**
Dig 8-9	90.68	52.00	96.25	84.00	96.33	**97.74**
Letter	82.06	76.80	-	-	92.80	**94.47**
UCISat	95.93	95.79	-	-	96.82	**98.48**
Text-1	50.53	93.79	-	-	**96.82**	95.00
Text-2	50.38	91.35	-	-	93.99	**96.28**
UCIDig	96.38	97.57	-	-	98.18	**99.38**
MNIST	89.21	89.92	-	-	92.41	**95.71**
Dig 0689	42.23	93.13	94.83	-	-	**96.63**
Dig 1279	40.42	90.11	91.91	-	-	**94.01**
Cora-DS	28.24	36.88	-	-	-	**43.75**
Cora-HA	34.02	42.00	-	-	-	**59.75**
Cora-ML	27.08	31.05	-	-	-	**45.58**
Cora-OS	23.87	23.03	-	-	-	**58.89**
Cora-PL	33.80	33.97	-	-	-	**46.83**
WK-CL	55.71	61.43	-	-	-	**71.95**
WK-TX	45.05	35.38	-	-	-	**69.29**
WK-WT	53.52	32.85	-	-	-	**77.96**
WK-WC	49.53	33.31	-	-	-	**73.88**
20-news	35.27	41.89	-	-	-	**70.63**
RCVI	27.05	-	-	-	-	**61.97**

where the results for *k-means* and *iterative SVR* are averaged over 50 independent runs and '-' means the corresponding algorithm cannot handle the dataset in reasonable time. Since *GMC* and *iterative SVR* can only handle binary clustering problems, we also provide experiments on several 2-class problems: **Letters**, **Satellite**, **autos vs. motorcycles (Text-1)** and **baseball vs. hockey (Text-2)**. Moreover, for the **UCI-Digits** and **MNIST** datasets, we enumerate all 45 possible class pairs, and report the average clustering results. Furthermore, as the *MMC* and *GMC* algorithms can only handle datasets with no more than a few hundred samples, we perform experiments on **UCI Digits** and focus on those pairs (3 vs 8, 1 vs 7, 2 vs 7, 8 vs 9, 0689 and 1279) that are difficult to differentiate. From the tables we can clearly observe

Table 3. Rand Index comparisons.

Data	KM	NC	MMC	GMC	SVR	CPM3C
Dig 3-8	0.904	0.545	0.823	0.899	0.940	**0.945**
Dig 1-7	0.995	0.504	0.571	0.962	0.995	**1.00**
Dig 2-7	0.940	0.550	0.978	0.994	**1.00**	**1.00**
Dig 8-9	0.835	0.500	0.929	0.733	0.934	**0.956**
Letter	0.706	0.644	-	-	0.867	**0.897**
UCISat	0.922	0.919	-	-	0.939	**0.971**
Text-1	0.500	0.884	-	-	**0.939**	0.905
Text-2	0.500	0.842	-	-	0.887	**0.929**
UCIDig	0.933	0.956	-	-	0.967	**0.989**
MNIST	0.808	0.818	-	-	0.860	**0.921**
Dig 0689	0.696	0.939	0.941	-	-	**0.968**
Dig 1279	0.681	0.909	0.913	-	-	**0.943**
Cora-DS	0.589	**0.744**	-	-	-	0.735
Cora-HA	0.385	0.659	-	-	-	**0.692**
Cora-ML	0.514	0.720	-	-	-	**0.754**
Cora-OS	0.518	0.522	-	-	-	**0.721**
Cora-PL	0.643	0.675	-	-	-	**0.703**
WK-CL	0.603	0.602	-	-	-	**0.728**
WK-TX	0.604	0.514	-	-	-	**0.707**
WK-WT	0.616	0.581	-	-	-	**0.747**
WK-WC	0.581	0.509	-	-	-	**0.752**
20-news	0.581	0.496	-	-	-	**0.782**
RCVI	0.471	-	-	-	-	**0.698**

that our *CPM3C* algorithm can beat other competitive algorithms on almost all the datasets.

4.3. Speed of CPM3C

Table 4 compares the CPU-time of *CPM3C* with other competitive algorithms. According to the table, *CPM3C* is at least 18 times faster than *SVR*, 200 times faster than *GMC*. As reported in (Valizadegan & Jin, 2007), *GMC* is about 100 times faster than *MMC*. Hence, *CPM3C* is still faster than *MMC* by about four orders of magnitude. Moreover, as the sample size increases, the CPU-time of *CPM3C* grows much slower than that of *iterative SVR*, which indicates *CPM3C* has much better scaling property with the sample size than *SVR*. Finally, *CPM3C* also performs much faster than conventional *kmeans*, which is a very appealing result. As for the *Ncut* method, since the calculation of the similarity matrix is very time consuming and usually takes several hours on the text datasets, we do not report the time it spends here.

Table 4. CPU-time (seconds) comparisons.

Data	KM	GMC	SVR	CPM3C
Dig 3-8	0.51	276.16	19.72	1.10
Dig 1-7	0.54	289.53	20.49	0.95
Dig 2-7	0.50	304.81	19.69	0.75
Dig 8-9	0.49	277.26	19.41	0.85
Letter	0.08	-	2133	0.87
UCISat	0.19	-	6490	4.54
Text-1	66.09	-	930.0	19.75
Text-2	52.32	-	913.8	16.16
Dig 0689	34.28	-	-	9.66
Dig 1279	17.78	-	-	17.47
Cora-DS	839.67	-	-	35.31
Cora-HA	204.43	-	-	24.35
Cora-ML	22781	-	-	69.04
Cora-OS	47931	-	-	13.98
Cora-PL	7791.4	-	-	165.0
WK-CL	672.69	-	-	9.534
WK-TX	766.77	-	-	10.53
WK-WT	4135.2	-	-	10.67
WK-WC	1578.2	-	-	9.041
20-news	2387.8	-	-	215.6
RCVI	428770	-	-	587.9

4.4. Dataset size n vs. Speed

In the theoretical analysis section, we state that the computational time of *CPM3C* scales linearly with the number of samples. We present numerical demonstra-

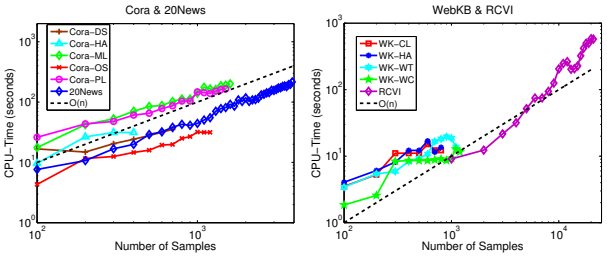


Figure 1. CPU-Time (seconds) of *CPM3C* as a function of dataset size n.

tion for this statement in figure 1, where a log-log plot of how computational time increases with the size of the data set is shown. Specifically, lines in the log-log plot correspond to polynomial growth $O(n^d)$, where d is the slope of the line. Figure 1 shows that the CPU-time of *CPM3C* scales roughly $O(n)$, which is consistent with the statement in section 3.

4.5. vs. Accuracy & Speed

Theorem 6 states that the total number of iterations involved in *CPM3C* is at most $\frac{R}{2}$, and this means with higher , the algorithm might converge fast. However, as is directly related to the training loss in *CPM3C*, we need to determine how small should be to guarantee sufficient accuracy. We present in figure 2 and figure 3 how clustering accuracy and computational time scale with . According to figure 2, = 0.01

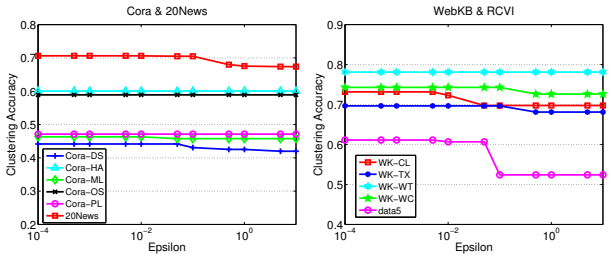


Figure 2. Clustering accuracy of *CPM3C* vs. .

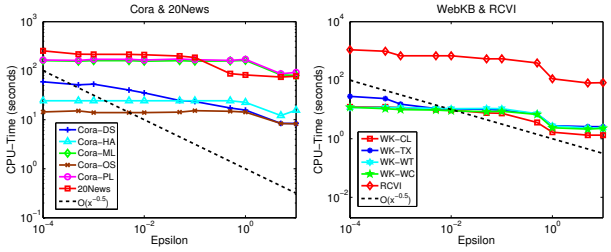


Figure 3. CPU-time (seconds) of *CPM3C* vs. .

is small enough to guarantee clustering accuracy. The log-log plot in figure 3 verifies that the CPU-time of *CPM3C* decreases as increases. Moreover, the empirical scaling of roughly $O(\frac{1}{0.5})$ is much better than $O(\frac{1}{2})$ in the bound from theorem 6.

5. Conclusions

We propose the *cutting plane multiclass maximum margin clustering* (*CPM3C*) algorithm in this paper, to cluster data samples with the maximum margin hyperplane. Preliminary theoretical analysis of the algorithm is provided, where we show that the computational time of *CPM3C* scales linearly with the sample size n with guaranteed accuracy. Moreover, experimental evaluations on several real world datasets show that *CPM3C* performs better than existing *MMC* methods, both in efficiency and accuracy.

Acknowledgments

This work is supported by the projects (60721003) and (60675009) of the National Natural Science Foundation of China.

References

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, *2*, 265–292.

Ding, C., He, X., Zha, H., Gu, M., & Simon, H. D. (2001). A min-max cut algorithm for graph partitioning and data mining. *ICDM* (pp. 107–114).

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classi cation*. John Wiley & Sons, Inc.

Joachims, T. (2006). Training linear svms in linear time. *SIGKDD 12*.

Kelley, J. E. (1960). The cutting-plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics*, *8*, 703–712.

Lewis, D. D., Yang, Y., Rose, T., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *JMLR*, *5*, 361–397.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the contruction of internet portals with machine learning. *Information Retrieval Journal*, *3*, 127–163.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *JASA*, *66*, 846–850.

Schölkopf, B., Smola, A. J., & Müller, K. R. (1999). Kernel principal component analysis. *Advances in kernel methods: support vector learning*, 327–352.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *PAMI*.

Smola, A. J., Vishwanathan, S., & Hofmann, T. (2005). Kernel methods for missing variables. *AISTATS 10*.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *JMLR*, *6*, 1453–1484.

Valizadegan, H., & Jin, R. (2007). Generalized maximum margin clustering and unsupervised kernel learning. *NIPS 19* (pp. 1417–1424).

Xu, L., Neufeld, J., Larson, B., & Schuurmans, D. (2004). Maximum margin clustering. *NIPS 17*.

Xu, L., & Schuurmans, D. (2005). Unsupervised and semi-supervised multi-class support vector machines. *AAAI*.

Zhang, K., Tsang, I. W., & Kowk, J. T. (2007). Maximum margin clustering made practical. *ICML 24*.

Zhao, B., Wang, F., & Zhang, C. (2008). E cient maximum margin clustering via cutting plane algorithm. *SDM* (pp. 751–762).

Laplace Maximum Margin Markov Networks

Jun Zhu[†*] JUN-ZHU@MAILS.TSINGHUA.EDU.CN
Eric P. Xing[†] EPXING@CS.CMU.EDU
Bo Zhang[*] DCSZB@MAIL.TSINGHUA.EDU.CN

[*]Department of Computer Science and Technology, Tsinghua University, Beijing 100084 China
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

We propose Laplace max-margin Markov networks (LapM^{3}N), and a general class of Bayesian M^{3}N (BM^{3}N) of which the LapM^{3}N is a special case with sparse structural bias, for robust structured prediction. BM^{3}N generalizes extant structured prediction rules based on point estimator to a Bayes-predictor using a learnt distribution of rules. We present a novel *Structured Maximum Entropy Discrimination* (SMED) formalism for combining Bayesian and max-margin learning of Markov networks for structured prediction, and our approach subsumes the conventional M^{3}N as a special case. An efficient learning algorithm based on variational inference and standard convex-optimization solvers for M^{3}N, and a generalization bound are offered. Our method outperforms competing ones on both synthetic and real OCR data.

1. Introduction

In recent years, log-linear models based on composite features that explicitly exploit the structural dependencies among elements in high-dimensional inputs (e.g., DNA strings, text sequences, image lattices) and structured interpretational outputs (e.g., gene segmentation, natural language parsing, scene description) have gained substantial popularity in learning structured predictions from complex data. Major instances of such models include the conditional random fields (CRFs) (Lafferty et al., 2001), Markov networks (MNs) (Taskar et al., 2003), and other specialized graphical models (Altun et al., 2003). Adding to the flexibilities and expressive power of such models, different learning paradigms have been explored, such as maximum likelihood estimation (Lafferty et al., 2001), and max-margin learning (Altun et al., 2003; Taskar et al., 2003; Tsochantaridis et al., 2004).

For domains with complex feature space, it is often desirable to pursue a "sparse" representation of the model that leaves out irrelevant features. Learning such a sparse model is key to reduce the rick of overfitting and achieve good generalizability. In likelihood-based estimation, sparse model fitting has been extensively studied. A commonly used strategy is to add an L_1-penalty to the likelihood function, which can also be viewed as a MAP estimation under a Laplace prior. Recent work along this line includes (Lee et al., 2006; Wainwright et al., 2006; Andrew & Gao, 2007).

This progress notwithstanding, little progress has been made so far on learning sparse MNs or log-linear models in general based on the max-margin principle, which is arguably a more desirable paradigm for training highly discriminative structured prediction models in a number of application contexts. While sparsity has been pursued in maximum margin learning of certain discriminative models such as SVM that are "unstructured" (i.e., with a univariate output), by using L_1-regularization (Bennett & Mangasarian, 1992) or by adding a cardinality constraint (Chan et al., 2007), generalization of these techniques to structured output space turns out to be extremely non-trivial. For example, although it appears possible to formulate sparse max-margin learning as a convex optimization problem as for SVM, both the primal and dual problems are hard to solve since there is no obvious way to exploit the conditional independence structures within a regularized MN to efficiently deal with the typically exponential number of margin constraints. Another empirical insight as we will show in this paper is that the L_1-regularized estimation is not so robust. Discarding the features that are not completely irrelevant can potentially hurt generalization ability.

In this paper, we propose a new formalism called *Structured Maximum Entropy Discrimination*

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

(SMED), which offers a general framework to combine Bayesian learning and max-margin learning of log-linear models for structured prediction. SMED is a generalization of the maximum entropy discrimination (Jaakkola et al., 1999) methods originally developed for classification to the broader problem of structured learning. It facilitates posterior inference of a full distribution of feature coefficients (i.e., weights), rather than a point-estimate as in the standard max-margin Markov network (M^3N) (Taskar et al., 2003), under a user-specified prior distribution of the coefficients and generalized maximum margin constraints. One can use the learned posterior distribution of coefficients to form a Bayesian max-margin Markov network (BM^3N) that is equivalent to a weighted sum of differentially parameterized M^3Ns, or one can obtain a MAP BM^3N. We show that, by using a Laplace prior for the feature coefficients, the resulting BM^3N is effectively a "sparse" max-margin Markov network, which we refer to as a Laplace M^3N ($LapM^3N$). But unlike the L_1-regularized maximum likelihood estimation, where sparsity is due to a hard threshold introduced by the Laplace prior (Kaban, 2007), the effect of Laplace prior in $LapM^3N$ is a biased posterior weighting of the parameters. Smaller parameters are shrunk more and thus robust estimation is achieved when the data have irrelevant features. The Bayesian formalism also makes the $LapM^3N$ less sensitive to regularization constants. Interestingly, a trivial assumption on the prior distribution of the coefficients, i.e., a standard (zero-mean and identity covariance) normal, reduces BM^3N to the standard M^3N, as shown in Theorem 3.

The paper is structured as follows. The next section reviews the basic structured prediction formalism and sets the stage for our model. Sec. 3 presents the SMED formalism and basic results on BM^3N. Sec. 4 presents $LapM^3N$ and a novel learning algorithm. Sec. 5 presents a generalization bound of BM^3N. Sec. 6 shows empirical results. Sec. 7 concludes this paper.

2. Preliminaries

Consider a structured prediction problem such as natural language parsing, image understanding, or DNA decoding. The objective is to learn a predictive function $h : \mathcal{X} \mapsto \mathcal{Y}$ from a structured input $\mathbf{x} \in \mathcal{X}$ (e.g., a sentence or an image) to a structured output $\mathbf{y} \in \mathcal{Y}$ (e.g., a sentence parsing or a scene annotation), where $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_l$ with $\mathcal{Y}_i = \{y_1, \ldots, y_{m_i}\}$ represents a combinatorial space of structured interpretations of multi-facet objects. For example, $\mathcal{Y}$ could correspond to the space of all possible instantiations of the part-of-speech (POS) tagging in the parse tree of a sentence, or the space of all possible ways of labeling entities over some segmentation of an image. The prediction $\mathbf{y} \equiv (y_1, \ldots, y_l)$ is *structured* because each individual label $y_i \in \mathcal{Y}_i$ within $\mathbf{y}$ must be determined in the context of other labels $y_{j \neq i}$, rather than independently as in a standard classification problem.

Let $F : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ represent a discriminant function over the input-output pairs from which one can define the predictive function h. A common choice of F is a linear model, which is based on a set of feature functions $f_k : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$ and their weights w_k, i.e., $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$. Given F, the prediction function h is typically defined in terms of an optimization problem that maximizes F over the response variable $\mathbf{y}$ given input $\mathbf{x}$:

$$h_0(\mathbf{x}; \mathbf{w}) = \arg \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} F(\mathbf{x}, \mathbf{y}; \mathbf{w}). \quad (1)$$

Depending on the specific choice of the objective function $C(\mathbf{w})$ for estimating the parameter $\mathbf{w}$ (e.g., likelihood, or margin), incarnations of the general structured prediction formalism described above can be seen in models such as the CRFs (Lafferty et al., 2001), where $C(\mathbf{w})$ is the conditional likelihood of the true structured label; and the M^3N (Taskar et al., 2003), where $C(\mathbf{w})$ is the margin between the true label and any other label. Recent advances in structured prediction has introduced regularizations of $C(\mathbf{w})$ in the CRF context, so that a *sparse* $\mathbf{w}$ can be learned (Andrew & Gao, 2007). To the best of our knowledge, existing max-margin structured prediction methods utilize a single discriminant function $F(\,\cdot\,; \mathbf{w})$ defined by the "optimum" estimate of $\mathbf{w}$, similar to a practice in Frequentist statistics. In this paper, we propose a Bayesian version of the predictive rule in Eq. (1) so that the prediction function h can be obtained from a posterior mean over multiple (indeed infinitely many) $F(\,\cdot\,; \mathbf{w})$; and we also propose a new formalism and objective $C(\mathbf{w})$ that lead to a **Bayesian M^3N**, which subsumes the standard M^3N as a special case, and can achieve a posterior shrinkage effect on $\mathbf{w}$ that resembles L_1-regulatiztion. To our knowledge, although sparse graphical model learning based on various likelihood-based principles has recently received substantial attention (Lee et al., 2006; Wainwright et al., 2006), learning sparse networks based on the maximum margin principle has not yet been successfully explored. Our proposed method represents an initial foray in this important direction.

Before dwelling into exposition of the proposed approach, we end this section with a brief recapitulation of the basic M^3N that motivates this work, and provides a useful baseline that grounds the proposed approach. Under a max-margin framework, given training data $\mathcal{D} = \{\langle \mathbf{x}^i, \mathbf{y}^i \rangle\}_{i=1}^N$, we obtain a point estimate

of the weight vector $\mathbf{w}$ by solving the following max-margin problem P0 (Taskar et al., 2003):

$$\text{P0 (M}^3\text{N)}: \quad \min_{\mathbf{w},\xi} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.t. } \forall i, \forall \mathbf{y} \neq \mathbf{y}^i: \quad \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i,\ \xi_i \geq 0\ ,$$

where $\Delta\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - \mathbf{f}(\mathbf{x}^i, \mathbf{y})$ and $\mathbf{w}^\top\Delta\mathbf{f}_i(\mathbf{y})$ is the "margin" between the true label $\mathbf{y}^i$ and a prediction $\mathbf{y}$, $\Delta\ell_i(\mathbf{y})$ is a loss function with respect to $\mathbf{y}^i$, and ξ_i is a slack variable that absorbs errors in the training data. Various loss functions have been proposed in the literature (Tsochantaridis et al., 2004). In this paper, we adopt the *hamming loss* used in (Taskar et al., 2003): $\Delta\ell_i(\mathbf{y}) = \sum_{j=1}^{|\mathbf{x}^i|}\mathbb{I}(y_j \neq y_j^i)$, where $\mathbb{I}(\cdot)$ is an indicator function that equals to one if the argument is true and zero otherwise. The optimization problem P0 is intractable because the feasible space for $\mathbf{w}$, $\mathcal{F}_0 = \{\mathbf{w} : \mathbf{w}^\top\Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i;\ \forall i, \forall\mathbf{y} \neq \mathbf{y}^i\}$, is defined by $O(N|\mathcal{Y}|)$ number of constraints, and $\mathcal{Y}$ itself is exponential to the size of the input $\mathbf{x}$. Exploring sparse dependencies among individual labels y_i in $\mathbf{y}$, as reflected in the specific design of the feature functions (e.g., based on pair-wise labeling potentials), and convex duality of the objective, efficient algorithms based on cutting-plane (Tsochantaridis et al., 2004) or message-passing (Taskar et al., 2003) have been proposed to obtain an approximate optimum solution. As described shortly, these algorithms can be directly employed as subroutines in solving our proposed model.

3. Bayesian Maximum Margin Markov Networks

In this paper, we take a Bayesian approach and learn a distribution $p(\mathbf{w})$, rather than a point estimate of $\mathbf{w}$, in a max-margin manner. For prediction, we take the average over all the possible models, that is:

$$h_1(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \int p(\mathbf{w})F(\mathbf{x},\mathbf{y};\mathbf{w})\,\mathrm{d}\mathbf{w}\,. \quad (2)$$

Now, the open question is how we can devise an appropriate objective function over $p(\mathbf{w})$, in a similar spirit as the L_2-norm cost over $\mathbf{w}$ in P0, that leads to an optimum estimate of $p(\mathbf{w})$. Below, we present a structured maximum entropy discrimination (SMED) framework that facilitates the estimation of a Bayesian M^{3}N defined by $p(\mathbf{w})$. As we show in the sequel, our Bayesian max-margin learning formalism offers several advantages like the PAC-Bayes generalization guarantee and estimation robustness.

3.1. SMED and the Bayesian M^{3}N

Given a training set $\mathcal{D}$, analogous to the feasible space $\mathcal{F}_0$ for weight vector $\mathbf{w}$ in an M^{3}N (i.e., problem P0), the feasible subspace $\mathcal{F}_1$ of weight distribution $p(\mathbf{w})$ is defined by a set of *expected* margin constraints:

$$\mathcal{F}_1 = \{p(\mathbf{w}) : \langle\Delta F_i(\mathbf{y};\mathbf{w}) - \Delta\ell_i(\mathbf{y})\rangle_{p(\mathbf{w})} \geq -\xi_i,\ \forall i, \mathbf{y} \neq \mathbf{y}^i\},$$

where $\Delta F_i(\mathbf{y};\mathbf{w}) = F(\mathbf{x}^i,\mathbf{y}^i;\mathbf{w}) - F(\mathbf{x}^i,\mathbf{y};\mathbf{w})$ and $\langle\cdot\rangle_p$ denotes the expectations with respect to p.

To choose the best distribution $p(\mathbf{w})$ from $\mathcal{F}_1$, the *maximum entropy principle* suggests that one can consider the distribution that minimizes its relative entropy with respect to some chosen prior p_0, as measured by the Kullback-Leibler divergence, $KL(p||p_0) = \langle\log(p/p_0)\rangle_p$. To accommodate the discriminative prediction problem we concern, instead of minimizing the usual KL, we optimize the generalized entropy (Dudík et al., 2007; Lebanon & Lafferty, 2001), or a regularized KL-divergence, $KL(p(\mathbf{w})||p_0(\mathbf{w})) + U(\xi)$, where $U(\xi)$ is a closed proper convex function over the slack variables. This leads to the following Structured Maximum Entropy Discrimination Model:

Definition 1 (The Structured Maximum Entropy Discrimination Model) *Given training data $\mathcal{D} = \{\langle\mathbf{x}^i,\mathbf{y}^i\rangle\}_{i=1}^N$, a discriminant function $F(\mathbf{x},\mathbf{y};\mathbf{w})$, a loss function $\Delta\ell_{\mathbf{x}}(\mathbf{y})$, and an ensuing feasible subspace $\mathcal{F}_1$ (defined above) for parameter distribution $p(\mathbf{w})$, the SMED model that leads to a prediction function of the form of Eq. (2) is defined by the following generalized relative entropy minimization with respect to a parameter prior $p_0(\mathbf{w})$:*

$$\text{P1}: \quad \min_{p(\mathbf{w}),\xi} \ KL(p(\mathbf{w})||p_0(\mathbf{w})) + U(\xi)$$
$$\text{s.t. } \ p(\mathbf{w}) \in \mathcal{F}_1,\ \xi_i \geq 0, \forall i.$$

The P1 defined above is a variational optimization problem over $p(\mathbf{w})$ in a subspace of valid parameter distributions. Since both the KL and the function U in P1 are convex, and the constraints in $\mathcal{F}_1$ are linear, P1 is a convex program, which can be solved via applying the calculus of variations to the Lagrangian to obtain a variational extremum, followed by a dual transformation of P1. Due to space limit, a detailed derivation is given in an extended version of this paper, and below we state the main results as a theorem.

Theorem 2 (Solution to SMED) *The variational optimization problem P1 underlying the SMED model gives rise to the following optimum distribution of Markov network parameters $\mathbf{w}$:*

$$p(\mathbf{w}) = \frac{1}{Z(\alpha)}p_0(\mathbf{w})\exp\{\sum_{i,\mathbf{y}}\alpha_i(\mathbf{y})[\Delta F_i(\mathbf{y};\mathbf{w}) - \Delta\ell_i(\mathbf{y})]\}, \quad (3)$$

where the Lagrangian multipliers $\alpha_i(\mathbf{y})$ (corresponding to constraints in $\mathcal{F}_1$) can be obtained by solving the dual problem of P1:

$$\text{D1}: \quad \max_{\alpha} \ -\log Z(\alpha) - U^\star(\alpha)$$
$$\text{s.t. } \ \alpha_i(\mathbf{y}) \geq 0,\ \forall i,\ \forall\mathbf{y},$$

where $U^\star(\cdot)$ represents the conjugate of the slack function $U(\cdot)$, i.e., $U^\star(\alpha) = \sup_\xi\big(\sum_{i,\mathrm{y}}\alpha_i(\mathrm{y})\xi_i - U(\xi)\big)$.

For a closed proper convex function $\phi(\mu)$, its conjugate is defined as $\phi^\star(\nu) = \sup_\mu[\nu^\top\mu - \phi(\mu)]$. In problem D1, by convex duality, the log normalizer $\log Z(\alpha)$ can be shown to be the conjugate of the KL-divergence. If the slack function is $U(\xi) = C\|\xi\| = C\sum_i \xi_i$, it is easy to show that $U^\star(\alpha) = \mathbb{I}_\infty(\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) \leq C,\ \forall i)$, where $\mathbb{I}_\infty(\cdot)$ is a function that equals to zero when its argument holds true and infinity otherwise. Here, the inequality corresponds to the trivial solution $\xi = 0$, that is, the training data are perfectly separative. Ignoring this inequality does not affect the solution since the special case $\xi = 0$ is still included. Thus, the Lagrangian multipliers $\alpha_i(\mathbf{y})$ in the dual problem D1 comply with the set of constraints that $\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = C,\ \forall i$. Another example is $U(\xi) = KL(p(\xi)||p_0(\xi))$ by introducing uncertainty on the slack variables (Jaakkola et al., 1999). Some other U functions and their dual functions are studied in (Lebanon & Lafferty, 2001; Dudík et al., 2007).

The optimum parameter distribution $p(\mathbf{w})$ defined by Eq. (3), along with the predictive function $h_1(x;\mathbf{w})$ given by Eq. (2), jointly form what we would like to call a **Bayesian M^{3}N** (BM^{3}N). The close connection of BM^{3}N and M^{3}N is suggested by the striking isomorphisms of the opt-problem P1, the feasible space $\mathcal{F}_1$, and the predictive function h_1 underlying an BM^{3}N, to their counterparts P0, $\mathcal{F}_0$, and h_0, respectively, underlying an M^{3}N. Indeed, by making a special choice of a parameter prior in Eq. (3), based on the above discussion of conjugate functions in D1, we arrive at a reduction of D1 to an M^{3}N optimization problem. The following theorem makes this explicit.

Theorem 3 (Reduction of BM^{3}N to M^{3}N) *Assuming $F(\mathbf{x},\mathbf{y};\mathbf{w}) = \mathbf{w}^\top\mathbf{f}(\mathbf{x},\mathbf{y})$, $U(\xi) = \sum_i \xi_i$, and $p_0(\mathbf{w}) = \mathcal{N}(\mathbf{w}|0,I)$, where I denotes an identity matrix, then the Lagrangian multipliers $\alpha_i(\mathbf{y})$ are obtained by solving the following dual problem:*

$$\max_{\alpha} \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y}) - \frac{1}{2}\|\sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})\|^2$$
$$\text{s.t.} \sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = C;\ \alpha_i(\mathbf{y}) \geq 0,\ \forall i,\ \forall \mathbf{y},$$

which, when applied to h_1, lead to a predictive function that is identical to $h_0(\mathbf{x};\mathbf{w})$ given by Eq. (1).

Proof: (sketch) Replacing $p_0(\mathbf{w})$ in Eq. (3) with $\mathcal{N}(\mathbf{w}|0,I)$, we can obtain the following closed-form expression of the $Z(\alpha)$ in $p(\mathbf{w})$:

$$\int \frac{1}{(2\pi)^{\frac{K}{2}}} \exp\{-\frac{\mathbf{w}^\top\mathbf{w}}{2} + \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})[\mathbf{w}^\top\Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y})]\}\,\mathrm{d}\mathbf{w}$$
$$= \exp\Big(-\sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y}) + \frac{1}{2}\|\sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})\|^2\Big).$$

As we have stated, the constraints $\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = C$ are due to the conjugate of $U(\xi) = \sum_i \xi_i$. □

Theorem 3 shows that in the supervised learning setting, M^{3}N is subsumed by the SMED model, and can be viewed as a special case of a Bayesian M^{3}N when the slack function is linear and the parameter prior is a standard normal. As described later, this connection renders many existing techniques for solving the M^{3}N directly applicable for solving the BM^{3}N. Note that although the distribution $p(\mathbf{w})$ in Eq. (3) has the same form as that of Bayesian CRFs (Qi et al., 2005), the underlying principles are fundamentally different.

Recent trend in pursuing "sparse" graphical models has led to the emergence of regularized version of CRFs (Andrew & Gao, 2007) and Markov networks (Lee et al., 2006; Wainwright et al., 2006). Interestingly, while such extensions have been successfully implemented by several authors in maximum likelihood learning of various sparse graphical models, they have not yet been explored in the context of maximum margin learning. Such a gap is not merely due to a negligence. Indeed, learning a sparse M^{3}N can be significantly harder as we discuss below.

As Theorem 3 reveals, an M^{3}N corresponds to a BM^{3}N with a standard normal prior for the weight vector $\mathbf{w}$. To encourage a sparse model, when using zero-mean normal prior, the weights of irrelevant features should peak around zero with very small variances. However, the isotropy of the variances in all dimensions in the standard normal prior makes M^{3}N infeasible to adjust the variances in different dimensions to fit sparse data. One way to learn a sparse model is to adopt the strategy of L_1-SVM to use L_1-norm instead of L_2-norm (a detailed description of this formulation and the duality derivation is available in the extended version of this paper). However, in both the primal and dual of an L_1-regularized M^{3}N, there is no obvious way to exploit the sparse dependencies among variables of the MN in order to efficiently deal with typically exponential number of constraints, which makes direct optimization or LP-formulation expensive. In this paper, we adopt the SMED framework that directly leads to a Bayesian M^{3}N, and employ a Laplace prior for $\mathbf{w}$ to learn a Laplace M^{3}N. When fitted to training data, the parameter posterior $p(\mathbf{w})$ under a Laplace M^{3}N has a shrinkage effect on small weights, which is similar to the L_1-regularizer in an M^{3}N. Although exact learning of a Laplace M^{3}N is still very hard, we show that it can be efficiently approximated by a variational inference procedure based on existing methods.

4. Laplace M^{3}N

The Laplace prior is $p_0(\mathbf{w}) = \prod_{k=1}^K \frac{\sqrt{\lambda}}{2} e^{-\sqrt{\lambda}|w_k|} = \big(\frac{\sqrt{\lambda}}{2}\big)^K e^{-\sqrt{\lambda}\|\mathbf{w}\|}$. The Laplace density is heavy tailed

and peaked at zero. Thus, it encodes the prior belief that the distribution of $\mathbf{w}$ is strongly peaked around zero. Another nice property is that the Laplace density is log-convex, which can be exploited to get convex estimation problems like LASSO (Tibshirani, 1996).

4.1. Variational Learning with Laplace Prior

Although in principle we have a closed-form solution of $p(\mathbf{w})$ in Theorem 2, the parameters $\alpha_i(\mathbf{y})$ are hard to estimate when using the Laplace prior. As we shall see in Section 4.2, exact integration will lead to a dual function that is difficult to maximize. Thus, we present a variational approximate learning approach.

Our approach is based on the hierarchical interpretation (Figueiredo, 2003) of the Laplace prior, that is, each w_k has a zero-mean Gaussian distribution $p(w_k|\tau_k) = \mathcal{N}(w_k|0, \tau_k)$ and the variance τ_k has an exponential hyper-prior density,

$$p(\tau_k|\lambda) = \frac{\lambda}{2}\exp\big\{-\frac{\lambda}{2}\tau_k\big\}, \quad \text{for } \tau_k \geq 0.$$

Let $p(\mathbf{w}|\tau) = \prod_{k=1}^K p(w_k|\tau_k)$, $p(\tau|\lambda) = \prod_{k=1}^K p(\tau_k|\lambda)$, then, $p_0(\mathbf{w}) = \int p(\mathbf{w}|\tau)p(\tau|\lambda)\,\mathrm{d}\tau$. Using the hierarchical representation and applying the Jensen's inequality, we get the following upper bound:

$$\begin{aligned} KL(p||p_0) &= -H(p) - \langle \log \int p(\mathbf{w}|\tau)p(\tau|\lambda)\,\mathrm{d}\tau \rangle_p \\ &\leq -H(p) - \langle \int q(\tau)\log\frac{p(\mathbf{w}|\tau)p(\tau|\lambda)}{q(\tau)}\,\mathrm{d}\tau \rangle_p \\ &\triangleq \mathcal{L}(p(\mathbf{w}), q(\tau)), \end{aligned}$$

where $q(\tau)$ is a variational distribution which is used to approximate $p(\tau|\lambda)$.

Substituting this upper bound for the KL in P1, we now solve the following problem,

$$\min_{p(\mathbf{w})\in\mathcal{F}_1; q(\tau); \xi} \mathcal{L}(p(\mathbf{w}), q(\tau)) + U(\xi). \tag{4}$$

This problem can be solved with an iterative minimization algorithm alternating between $p(\mathbf{w})$ and $q(\tau)$, as outlined in Algorithm 1, and detailed below.

Algorithm 1 Variational Bayesian Learning

Input: data $\mathcal{D} = \{\langle \mathbf{x}^i, \mathbf{y}^i\rangle\}_{i=1}^N$, constants C and λ, iteration number T
Output: posterior mean $\langle \mathbf{w}\rangle_p^T$
Initialize $\langle \mathbf{w}\rangle_p^1 \leftarrow 0$, $\Sigma_{\mathbf{w}}^1 \leftarrow I$
for $t = 1$ **to** $T-1$ **do**
  Step 1: solve (5) or (6) for $\langle \mathbf{w}\rangle_p^{t+1} = \Sigma_{\mathbf{w}}^t\eta$; update $\langle \mathbf{w}\mathbf{w}^\top\rangle_p^{t+1} \leftarrow \Sigma_{\mathbf{w}}^t + \langle \mathbf{w}\rangle_p^{t+1}(\langle \mathbf{w}\rangle_p^{t+1})^\top$.
  Step 2: use (7) to update $\Sigma_{\mathbf{w}}^{t+1} \leftarrow \mathrm{diag}(\sqrt{\frac{\langle w_k^2\rangle_p^{t+1}}{\lambda}})$.
end for

Step 1: Keep $q(\tau)$ fixed, we optimize (4) with respect to $p(\mathbf{w})$. Taking the same procedure as in solving P1, we get the posterior distribution $p(\mathbf{w})$ as follows,

$$\begin{aligned} p(\mathbf{w}) &\propto \exp\{\int q(\tau)\log p(\mathbf{w}|\tau)\,\mathrm{d}\tau - b\}\cdot\exp\{\mathbf{w}^\top\eta - L\} \\ &\propto \exp\{-\frac{1}{2}\mathbf{w}^\top\langle A^{-1}\rangle_q\mathbf{w} - b + \mathbf{w}^\top\eta - L\} \\ &= \mathcal{N}(\mathbf{w}|\mu_{\mathbf{w}}, \Sigma_{\mathbf{w}}), \end{aligned}$$

where $\eta = \sum_{i,\mathbf{y}}\alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})$, $L = \sum_{i,\mathbf{y}}\alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y})$, $A = \mathrm{diag}(\tau_k)$, and $b = KL(q(\tau)||p(\tau|\lambda))$ is a constant. The posterior mean and variance are $\langle \mathbf{w}\rangle_p = \mu_{\mathbf{w}} = \Sigma_{\mathbf{w}}\eta$ and $\Sigma_{\mathbf{w}} = (\langle A^{-1}\rangle_q)^{-1} = \langle \mathbf{w}\mathbf{w}^\top\rangle_p - \langle \mathbf{w}\rangle_p\langle \mathbf{w}\rangle_p^\top$, respectively. The dual parameters α are estimated by solving the following dual problem:

$$\begin{aligned} \max_{\alpha}\ & \sum_{i,\mathbf{y}}\alpha_i(\mathbf{y})\Delta\ell_i(\mathbf{y}) - \frac{1}{2}\eta^\top\Sigma_{\mathbf{w}}\eta \\ \text{s.t.}\ & \sum_{\mathbf{y}}\alpha_i(\mathbf{y}) = C;\ \alpha_i(\mathbf{y}) \geq 0,\ \forall i,\ \forall\mathbf{y}. \end{aligned} \tag{5}$$

This dual problem can be directly solved using existing algorithms developed for M^{3}N, such as (Taskar et al., 2003; Bartlett et al., 2004). Alternatively, we can solve the following primal problem:

$$\begin{aligned} \min_{\mathbf{w},\xi}\ & \frac{1}{2}\mathbf{w}^\top\Sigma_{\mathbf{w}}^{-1}\mathbf{w} + C\sum_{i=1}^N\xi_i \\ \text{s.t.}\ & \mathbf{w}^\top\Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i;\ \xi_i \geq 0,\ \ \forall i,\ \forall\mathbf{y} \neq \mathbf{y}^i. \end{aligned} \tag{6}$$

It is easy to show that the solution of problem (6) leads to the posterior mean of $\mathbf{w}$ under $p(\mathbf{w})$. The primal problem can be solved with subgradient (Ratliff et al., 2007) or extragradient (Taskar et al., 2006) methods.

Step 2: Keep $p(\mathbf{w})$ fixed, we optimize (4) with respect to $q(\tau)$. Take the derivative of $\mathcal{L}$ with respect to $q(\tau)$ and set it to zero, then we get $q(\tau) = \prod_{k=1}^K q(\tau_k)$. Each $q(\tau_k)$ is computed as follows:

$$\begin{aligned} \forall k: \quad q(\tau_k) &\propto p(\tau_k|\lambda)\exp\big\{\langle \log p(w_k|\tau_k)\rangle_p\big\} \\ &\propto \mathcal{N}(\sqrt{\langle w_k^2\rangle_p}|0,\tau_k)\exp(-\frac{1}{2}\lambda\tau_k). \end{aligned}$$

The normalization factor is $\int\mathcal{N}(\sqrt{\langle w_k^2\rangle_p}|0,\tau_k)\cdot\frac{\lambda}{2}\exp(-\frac{1}{2}\lambda\tau_k)\,\mathrm{d}\tau_k = \frac{\sqrt{\lambda}}{2}\exp(-\sqrt{\lambda\langle w_k^2\rangle_p})$. The expectations $\langle\tau_k^{-1}\rangle_q$ required in calculating $\langle A^{-1}\rangle_q$ are calculated as follows,

$$\langle\frac{1}{\tau_k}\rangle_q = \int\frac{1}{\tau_k}q(\tau_k)\,\mathrm{d}\tau_k = \sqrt{\frac{\lambda}{\langle w_k^2\rangle_p}}. \tag{7}$$

We iterate between the above two steps until convergence. Then, we use the posterior distribution $p(\mathbf{w})$ to make prediction. For irrelevant features, the variances should converge to zeros and thus lead to a sparse estimation. The intuition behind this iterative minimization algorithm is as follows. First, we use a Gaussian distribution to approximate the Laplace distribution and thus get a QP problem that is analogous to that of M^{3}N; then, the second step updates the covariance matrix in the QP problem with an exponential hyper-prior on the variance.

4.2. Insights

To see how the Laplace prior affects the posterior distribution, we do the following calculations. Substitute the hierarchical representation of the Laplace prior into $p(\mathbf{w})$ in Theorem 2, and we get:

$$\begin{aligned} Z(\alpha) &= \int\int p(\mathbf{w}|\tau)p(\tau|\lambda)\,\mathrm{d}\tau \cdot \exp\{\mathbf{w}^\top\eta - L\}\,\mathrm{d}\mathbf{w} \\ &= \int p(\tau|\lambda)\int p(\mathbf{w}|\tau)\cdot\exp\{\mathbf{w}^\top\eta - L\}\,\mathrm{d}\mathbf{w}\;\mathrm{d}\tau \\ &= \exp\{-L\}\prod_{k=1}^{K}\frac{\lambda}{\lambda-\eta_k^2}, \end{aligned} \tag{8}$$

where $\eta_k = \sum_{i,\mathbf{y}}\alpha_i(\mathbf{y})(f_k(\mathbf{x}^i,\mathbf{y}^i) - f_k(\mathbf{x}^i,\mathbf{y}))$ and an additional constraint is $\forall k,\ \eta_k^2 < \lambda$. Otherwise, the integration is infinity. Using the result (8), we can get:

$$\frac{\partial \log Z}{\partial \alpha_i(\mathbf{y})} = \mu^\top\Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y}), \tag{9}$$

where μ is a column vector and $\mu_k = \frac{2\eta_k}{\lambda-\eta_k^2}$, $\forall 1 \le k \le K$. An alternative way is using the definition of Z : $Z = \int p_0(\mathbf{w})\cdot\exp\{\mathbf{w}^\top\eta - L\}\,\mathrm{d}\mathbf{w}$. We can get:

$$\frac{\partial \log Z}{\partial \alpha_i(\mathbf{y})} = \langle\mathbf{w}\rangle_p^\top\Delta\mathbf{f}_i(\mathbf{y}) - \Delta\ell_i(\mathbf{y}). \tag{10}$$

Comparing Eqs. (9) and (10), we get $\langle\mathbf{w}\rangle_p = \mu$, that is, $\langle w_k\rangle_p = \frac{2\eta_k}{\lambda-\eta_k^2}$, $\forall 1 \le k \le K$. Similar calculation can lead to the result that in M^{3}N (standard normal prior) $\langle\mathbf{w}\rangle_p = \eta$. Figure 1 shows the posterior means (any dimension) when the priors are standard normal, Laplace with $\lambda = 4$, and Laplace with $\lambda = 6$. We can see that with a Laplace prior, the parameters are shrunk around zero. The larger the λ value is, the greater the shrinkage effect. For a fixed λ, the shape of the posterior mean is smoothly nonlinear but no component is explicitly discarded, that is, no weight is set to zero. This is different from the shape of a L_1-regularized maximum likelihood estimation (Kaban, 2007) where an interval exists around the origin and parameters falling into this interval are set to zeros.

Note that if we use the exact integration as in Eq. (8), the dual problem D1 will maximize $L - \sum_{k=1}^{K}\log\frac{\lambda}{\lambda-\eta_k^2}$. Since η_k^2 appears within a logarithm, the optimization problem would be very hard to solve. Thus, we turn to a variational approximation method.

5. Generalization bound

The PAC-Bayes bound (Langford et al., 2001) provides a theoretical motivation to learn an averaging model as in P1 which minimizes the KL-divergence and simultaneously satisfies the discriminative classification constraints. To apply it to our structured learning setting, we assume that the discriminant functions are bounded, that is, $F \in \mathcal{H} : \mathcal{X}\times\mathcal{Y} \to [-c, c]$ for all $\mathbf{w}$,

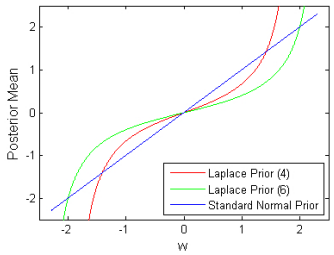


Figure 1. Posterior mean with different priors against the estimation of M^{3}N (i.e. with the standard normal prior).

where c is a positive constant. Recall that our averaging model is $h(\mathbf{x},\mathbf{y}) = \langle F(\mathbf{x},\mathbf{y};\mathbf{w})\rangle_{p(\mathbf{w})}$. We define the margin of an example $(\mathbf{x},\mathbf{y})$ for such a function h as $M(h,\mathbf{x},\mathbf{y}) = h(\mathbf{x},\mathbf{y}) - \max_{\mathbf{y}'\neq\mathbf{y}} h(\mathbf{x},\mathbf{y}')$. Clearly, the model h makes a wrong prediction on $(\mathbf{x},\mathbf{y})$ only if $M(h,\mathbf{x},\mathbf{y}) \le 0$. Let Q be a distribution over $\mathcal{X}\times\mathcal{Y}$, and let $\mathcal{D}$ be a sample of N examples randomly drawn from Q. We have the following PAC-Bayes theorem.

Theorem 4 (PAC-Bayes Bound of BM^{3}N) *Let p_0 be any continuous probability distribution over $\mathcal{H}$ and let $\delta \in (0,1)$. If $F \in \mathcal{H} : \mathcal{X}\times\mathcal{Y} \to [-c,c]$ for all $\mathbf{w}$, then with probability at least $1-\delta$ over random samples $\mathcal{D}$ of Q, for very distribution p over $\mathcal{H}$ and for all margin thresholds $\gamma > 0$:*

$$\begin{aligned} &\Pr_Q(M(h,\mathbf{x},\mathbf{y}) \le 0) \le \Pr_{\mathcal{D}}(M(h,\mathbf{x},\mathbf{y}) \le \gamma) \\ &\quad + O\left(\sqrt{\frac{\gamma^{-2}KL(p||p_0)\ln(N|\mathcal{Y}|) + \ln N + \ln\delta^{-1}}{N}}\right). \end{aligned}$$

Here, $\Pr_Q(.)$ stands for $\langle.\rangle_Q$ and $\Pr_{\mathcal{D}}(.)$ stands for the empirical average on $\mathcal{D}$. The proof follows the same structure as the original PAC-Bayes bound proof, with consideration of the margins. Due to space limit, details of the proof are given in the extended paper.

6. Experiments

In this section, we present some empirical results of LapM^{3}N on both synthetic and real data sets. We compare LapM^{3}N with M^{3}N, CRFs, L_1-regularized CRFs (L_1-CRFs), and L_2-regularized CRFs (L_2-CRFs). We use the quasi-Newton method (Andrew & Gao, 2007) to learn L_1-CRFs.

6.1. Synthetic Data Sets

6.1.1. I.I.D Features

The first experiment is conducted on synthetic sequence data with 100 i.i.d features. We generate three types of data sets with 10, 30, and 50 relevant features. For each setting, we randomly generate 10 linear-chain CRFs with 8 binary labeling states. The feature functions include: a real valued state-feature function over a one dimensional input feature and a class label; and

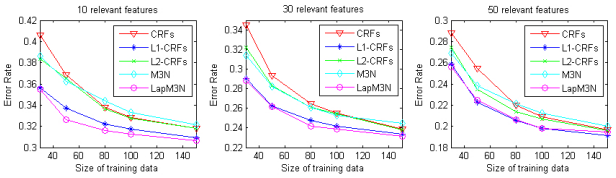


Figure 2. Evaluation results on data sets with i.i.d features.

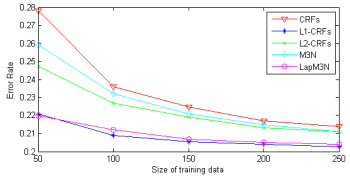


Figure 3. Results on data sets with 30 relevant features.

4 (2×2) binary transition-feature functions capturing pairwise label dependencies. For each model we generate a data set of 1000 samples. For each sample, we first *independently* draw the 100 features from a standard normal distribution, and then apply a Gibbs sampler to assign a label sequence with 5000 iterations.

For each data set, we randomly draw a part as training data and use the rest for testing. The numbers of training data are 30, 50, 80, 100, and 150. The QP problem is solved with the exponentiated gradient method (Bartlett et al., 2004). In all the following experiments, the regularization constant of L_1-CRFs and L_2-CRFs is chosen from $\{0.01, 0.1, 1, 4, 9, 16\}$ by a 5-fold cross-validation in training. For LapM^{3}N, we use the same method to choose λ from 20 roughly evenly spaced values between 1 and 268. For each setting, the average over 10 data sets is the final performance.

The results are shown in Figure 2. All the results of LapM^{3}N are achieved with 3 iterations of the variational learning. Under different settings LapM^{3}N consistently outperforms M^{3}N and performs comparably with L_1-CRFs. But note that the synthetic data come from simulated CRFs. Both L_1-CRFs and L_2-CRFs outperform the un-regularized CRFs. One interesting result is that M^{3}N and L_2-CRFs perform comparably. This is reasonable because as derived by Lebanon and Lafferty (2001) and noted by Globerson et al. (2007) the L_2-regularized MLE of CRFs has a similar convex dual as that of M^{3}N. The only difference is the loss they try to optimize. CRFs optimize the log-loss while M^{3}N optimizes the hinge-loss. As the number of training data increase, all the algorithms consistently get higher performance. The advantage of LapM^{3}N is more obvious when there are fewer relevant features.

6.1.2. Correlated Features

In reality, most data sets contain redundancy and the features are usually correlated. So, we evaluate our models on synthetic data sets with correlated features. We take the similar procedure as in generating the data sets with i.i.d features to first generate one linear-chain CRF model. Then, we use the CRF model to generate 10 data sets of which each sample has 30 relevant features. The 30 relevant features are partitioned into 10 groups. For the features in each group, we first draw a real-value from a standard normal distribution and then 'spoil' the feature with a random Gaussian noise to get 3 correlated features. The noise Gaussian has a zero mean and standard variance 0.05. Here and in all the remaining experiments, we use the sub-gradient method (Ratliff et al., 2007) to solve the QP problem in both M^{3}N and LapM^{3}N. We use the learning rate and complexity constant that are suggested by the authors, that is, $\alpha_t = \frac{1}{2\beta\sqrt{t}}$ and $C = 200\beta$, where β is a parameter we introduced to adjust α_t and C. We do K-fold CV on each data set and take the average over the 10 data sets as the final results. Like (Taskar et al., 2003), in each run we choose one part to do training and test on the rest K-1 parts. We vary K from 20, 10, 7, 5, to 4. In other words, we use 50, 100, about 150, 200, and 250 samples during the training. We use the same grid search to choose λ and β from $\{9, 16, 25, 36, 49, 64\}$ and $\{1, 10, 20, 30, 40, 50, 60\}$ respectively. Results are shown in Figure 3. We can get the same conclusions as in the previous results.

6.2. Real-World OCR Data Set

The OCR data set is partitioned into 10 subsets for 10-fold CV (Taskar et al., 2003; Ratliff et al., 2007). We randomly select N samples from each fold for our experiments. We vary N from 100, 150, 200, to 250, and denote the selected data sets by OCR100, OCR150, OCR200, and OCR250 respectively. When $\beta = 4$ on OCR100 and OCR150, $\beta = 2$ on OCR200 and OCR250, and $\lambda = 36$, results are shown in Figure 4.

Overall, as the number of training data increases, all algorithms achieve lower error rates and smaller variances. Generally, LapM^{3}N consistently outperforms all the other models. M^{3}N outperforms the standard, non-regularized, CRFs and the L_1-CRFs. Again, L_2-CRFs perform comparably to M^{3}N. This is a bit surprising but still reasonable due to the understanding of their only difference on loss functions (Globerson et al., 2007). By examining the prediction accuracy, we can see an obvious over-fitting in CRFs and L_1-CRFs. In contrast, L_2-CRFs are very robust. This is because unlike the synthetic data sets, features in real-world data are usually not completely irrelevant. In this case, putting small weights to zero as in L_1-CRFs will hurt generalization ability and also lead to instability to regularization constants as shown later. Instead, L_2-CRFs do not put small weights to zero but

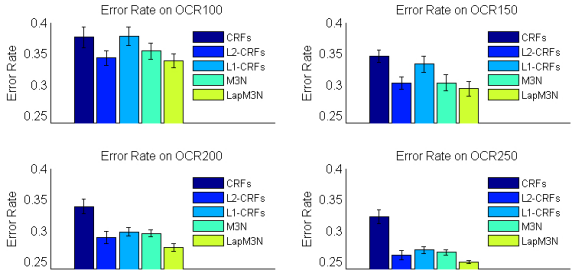


Figure 4. Evaluation results on OCR data set with different numbers of selected data.

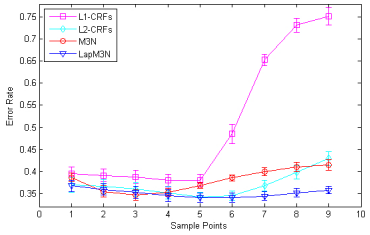


Figure 5. Error rates of different models on OCR100 with different regularization constants. From left to right, the regularization constants are 0.0001, 0.001, 0.01, 0.1, 1, 4, 9, 16, and 25 for L_1-CRFs and L_2-CRFs, and for M^{3}N and LapM^{3}N they are 1, 4, 9, 16, 25, 36, 49, 64, and 81.

shrink them towards zero as in LapM^{3}N. The non-regularized MLE can also easily lead to over-fitting.

6.3. Sensitivity to Regularization Constants

Figure 5 shows the error rates of different models on OCR100. From the results, we can see that the L_1-CRFs are much sensitive to the regularization constants. However, L_2-CRFs, M^{3}N, and LapM^{3}N are much less sensitive. Among all the models, LapM^{3}N is the most stable one. The stability of LapM^{3}N is due to the posterior weighting instead of hard-thresholding to set small weights to zero as in L_1-CRFs.

7. Conclusions

We proposed a *Structured Maximum Entropy Discrimination* formalism for Bayesian max-margin learning in structured prediction. This formalism gives rise to a general class of Bayesian M^{3}Ns and subsumes the standard M^{3}N as a spacial case where the predictive model is assumed to be linear and the parameter prior is a standard normal. We show that the adoption of a Laplace prior of the parameter leads to a Laplace M^{3}N that enjoys properties expected from a sparsified Bayesian M^{3}N. Unlike the L_1-regularized MLE which sets small weights to zeros to achieve sparsity, LapM^{3}N weights the parameters *a posteriori*. Features with smaller weights are shrunk more. This posterior weighting effect makes LapM^{3}N more stable with respect to the magnitudes of the regularization coefficients and more generalizable.

Acknowledgements

We thank Ivor Tsang for inspiring discussions. This work was conceived and completed while J.Z. was a visiting researcher at CMU under a State Scholarship from China, and supports from NSF DBI-0546594 and DBI-0640543 awarded to Eric Xing. J.Z. and B.Z. are also supported by Chinese NSF Grant 60321002; and the National Key Foundation R&D Projects 2004CB318108 and 2007CB311003.

References

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden markov support vector machines. *ICML*.

Andrew, G., & Gao, J. (2007). Scalable training of l_1-regularized log-linear models. *ICML*.

Bartlett, P., Collins, M., Taskar, B., & McAllester, D. (2004). Exponentiated gradient algorithms for large-margin structured classification. *NIPS*.

Bennett, K. P., & Mangasarian, O. L. (1992). Robust linear programming discrimination of two linearly inseparable sets. *Optim. Methods Softw*, 23–34.

Chan, A. B., Vasconcelos, N., & Lanckriet, G. R. G. (2007). Direct convex relaxations of sparse svm. *ICML*.

Dudík, M., Phillips, S. J., & Schapire, R. E. (2007). Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *JMLR*, 1217–1260.

Figueiredo, M. (2003). Adaptive sparseness for supervised learning. *IEEE Trans. on PAMI*, *25*, 1150–1159.

Globerson, A., Koo, T. Y., Carreras, X., & Collins, M. (2007). Exponentiated gradient algorithms for log-linear structured prediction. *ICML*.

Jaakkola, T., Meila, M., & Jebara, T. (1999). Maximum entropy discrimination. *NIPS*.

Kaban, A. (2007). On bayesian classification with laplace priors. *Pattern Recognition Letters*, *28*, 1271–1282.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Langford, J., Seeger, M., & Megiddo, N. (2001). An improved predictive accuracy bound for averaging classifiers. *ICML*.

Lebanon, G., & Lafferty, J. (2001). Boosting and maximum likelihood for exponential models. *NIPS*.

Lee, S.-I., Ganapathi, V., & Koller, D. (2006). Efficient structure learning of markov networks using l1-regularization. *NIPS*.

Qi, Y. A., Szummer, M., & Minka, T. P. (2005). Bayesian conditional random fields. *AISTATS*.

Ratliff, N. D., Bagnell, J. A., & Zinkevich, M. A. (2007). (online) subgradient methods for structured prediction. *AISTATS*.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. *NIPS*.

Taskar, B., Lacoste-Julien, S., & Jordan, M. I. (2006). Structured prediction via the extragradient method. *NIPS*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc., B*, 267–288.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *ICML*.

Wainwright, M. J., Ravikumar, P., & Lafferty, J. (2006). High-dimensional graphical model selection using l1-regularized logistic regression. *NIPS*.

Author Index

Author Index

H

I

J

K

L

M

Author Index

Author Index

T

U

V

W

X

Y

Author Index

Z