Two big challenges in machine learning

LÉON BOTTOU
FACEBOOK AI RESEARCH

ICML 2015 - LILLE
Machine Learning in 2015

- An established academic discipline
  - that attracts bright students.
- Pervasive applications
  - all over the Internet
  - and sometimes in the real world.
- Massive investments.
- Massive returns.
Machine Learning in 2015

- An established academic discipline that attracts bright students.
- Pervasive applications all over the Internet and sometimes in the real world.
- Massive investments.
- Massive returns.

Increased ambitions

more logos in your ICML booklet...
Machine Learning in 2015

- An established academic discipline that attracts bright students.
- Pervasive applications all over the Internet and sometimes in the real world.
- Massive investments.
- Massive returns.

Challenges of a new kind
Challenges of a new kind


2. Our experimental paradigm is reaching its limits. → Challenging the speed of our scientific progress.

3. Elements of a solution
1 Machine Learning disrupts software engineering
Engineering complex artifacts

Counting parts down to the smallest screw

- Automobile: ~ 30,000 parts
- Airplane: ~ 3,000,000 parts

Specifications and replication

- The engineer does not design the alloy and the threads of each screw. Standard screws come with known specifications (size, strength, weight, ...)
- Many parts and many assemblies are identical.
Abstraction in engineering

Thinking with the specifications instead of the details.
- part specifications instead of part details.
- assembly specifications instead of assembly details.

Abstractions leak!

Sometimes one needs to look more closely.
- Example: engine cooling assembly can have complicated heat inertia.
- Example: global resource allocation (cost, weight, etc.)

Abstraction leaks limits the complexity of what we can build.
Abstraction in mathematics and logic

Pure abstraction belongs to mathematics and logic
• When we use a theorem, we do not need to revisit its proof.
• Mathematical abstractions do not leak.

Computer programs are similar to mathematical proofs (in theory)
• Design with contracts – knowing the specification of a module is enough.
• In practice, there are abstraction leaks (but less.)
Computer programs

The closer to logic, the more complex the artifacts

- Automobile: ~30,000 parts
- Airplane: ~3,000,000 parts
- Processor: ~3,000,000,000 transistors
  (note: counts include many identical parts and assemblies.)
- MS Office: ~40,000,000 LOCs
- Facebook: ~60,000,000 LOCs
- Debian: ~400,000,000 LOCs
  (note: different metric: lines of codes are rarely identical.)
Programming versus learning

Two conceptions of computing

Digital computers (that one programs) are everywhere. Programming makes it easy to leverage the work of others.
Programming versus learning

Programming has real advantages


Remark

This book is not just about Rosenblatt’s perceptron! What Minsky and Papert call an “order-k perceptron” covers most machine learning models, including the convolutional neural networks that are changing computer vision.
Connectedness

Is a shape made of a single connected component?

M&P prove that a small-order perceptron cannot say, but a simple algorithm can.

Theorem 9.2: For any $\epsilon$ there is a 2-symbol Turing machine that can verify the connectedness of a figure $X$ on any rectangular array $R$, using less than $(2 + \epsilon) \log_2 |R|$ squares of tape.

Such an algorithm fulfils a clear contract: verifying connectedness.
Is connectedness easy for us?
What is easy for us?

This shape represents a mouse

This shape represents a piece of cheese
Why is learning gaining momentum?

- Since “connectedness” has a clear mathematical specification, we can envision provable algorithms.
- Since “mousiness” and “cheesiness” do not have such a specification, we cannot envision provable algorithms.

We must rely on
- heuristic specifications
- or learning techniques.
Why is learning gaining momentum?

- We must rely on
  - heuristic specifications (e.g. rules)
  - or learning techniques.

Big data and big computing power

When the volume of data increases
- defining heuristic specifications becomes harder.
- training learning systems becomes more effective.
Programming versus learning

Two conceptions of computing

Gaining momentum as we speak

Everywhere
Integrating machine learning in large software systems

Machine learning systems must* work with digital software because digital computers are everywhere.

- Use trained models as software modules.
- Use learning algorithms as software modules.

* at least until we solve AI...
Trained models as software modules

“DeepVisotron™,℠,® detects 1000 object categories with less than 1% errors.”

What is the nature of the contract?

- This does not mean that one rolls a dice for each picture.
- This statement refers to a specific testing set.
  The error guarantee is lost if the image distribution changes.
Weak contracts

- A smart programmer makes an inventive use of a **sorting routine**.
- The sorting routine **fulfils** the contract by **sorting** the data (however strange.)
- The code of the smart programmer **does what was intended**.

- A smart programmer makes an inventive use of a **trained object recognizer**.
- The object recognizer receives data that **does not resemble the testing data** and outputs nonsense.
- The code of the smart programmer **does not work**.
Weak contracts

- A smart programmer makes an inventive use of a **sorting routine**.
- The sorting routine *fulfils* the contract by *sorting* the data (however strange.)
- The code of the smart programmer *does what was intended*.

- A smart programmer makes an inventive use of a **trained object recognizer**.
- The object recognizer receives data that *does not resemble the testing data* and outputs nonsense.
- The code of the smart programmer *does not work*.

Collecting new data and retraining may help ... or may not!
Learning algorithms as software modules

---

2.1 Entanglement

From a high level perspective, a machine learning package is a tool for mixing data sources together. That is, machine learning models are machines for creating entanglement and making the isolation of improvements effectively impossible.
Many web sites offer lists of items (search results, recommendations, ads ...) and select them by modeling click probabilities.
A common click model

\[ P(\text{click}|\text{context, item, pos}) = F(\text{pos}) \times G(\text{context, item}) \]

Why separating position effect and item effect?

- Click probabilities of this form allow a greedy placement algorithm:
  
  1. Sort items by clickability \( G(\text{context, item}) \).
  2. Allocate them greedily to the best positions.
A common click model

Why separating position effect and item effect?
- Click probabilities of this form work well in auction theory results.
- There are easy ways to estimate $F(position)$ and $G(context, item)$.

How incorrect is this model?
- All ML models are incorrect.
- Model misspecification should only degrade the performance gracefully.
- How bad can things go?
How bad things can go?

- Assume there are only two contexts C1 and C2 and two positions P1,P2
- Assume the true click probability is
  \[ P^*(\text{click}|\text{context, item, pos}) = F^*(\text{context, pos}) \times G^*(\text{context, item}) \]

<table>
<thead>
<tr>
<th></th>
<th>F*</th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0.8</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>0.6</td>
<td>0.4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>G*</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0.12</td>
<td>0.10</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>0</td>
<td>0</td>
<td>0.12</td>
<td>0.10</td>
<td></td>
</tr>
</tbody>
</table>

- Our model estimates some intermediate click curve \( F(pos) \)
How bad can things go in context C1?

True position effect steeper than estimated.

- $F(P1) < F^*(C1, P1)$ $\implies$ overestimate clickability of items shown in P1
- $F(P2) > F^*(C1, P2)$ $\implies$ underestimate clickability of items shown in P2

Combined with the greedy placement algorithm.

- After retraining, whatever we placed in position P1 looks better than it really is. Therefore we keep placing it in position P1.

- After retraining, whatever we placed in position P2 looks worse than it really is. Therefore we keep placing it in position P2 (or stop showing it.)
How bad can things go in context C1?

True position effect steeper than estimated.

- $F(P1) < F^*(C1, P1) \Rightarrow$ overestimate clickability of items shown in P1
- $F(P2) > F^*(C1, P2) \Rightarrow$ underestimate clickability of items shown in P2

Combined with the greedy placement algorithm:

- After retraining, whatever we placed in position P1 looks better than it really is. Therefore we keep placing it in position P1.
- After retraining, whatever we placed in position P2 looks worse than it really is. Therefore we keep placing it in position P2 (or stop showing it.)

Exploration issue.
- New items eligible for context C1 may have a hard time competing with the current winner.
How bad can things go in context C2?

True position effect less steep than estimated.

- $F(P1) > F^*(C1, P1) \implies$ underestimate clickability of items shown in P1
- $F(P2) < F^*(C1, P2) \implies$ overestimate clickability of items shown in P2

Combined with the greedy placement algorithm.

- After retraining, whatever we placed in position P1 looks worse than it really is. Therefore we might move it down to position P2.
- After retraining, whatever we placed in position P2 looks better than it really is. Therefore we might move it up position P1.
How bad can things go in context C2?

True position effect less steep than estimated.

- $F(P1) > F^*(C1, P1) \Rightarrow$ underestimate clickability of items shown in P1
- $F(P2) < F^*(C1, P2) \Rightarrow$ overestimate clickability of items shown in P2

Combined with greedy placement algorithm.

- After retraining, whatever we placed in position P1 looks worse than it really is. Therefore we might move it down to position P2.
- After retraining, whatever we placed in position P2 looks better than it really is. Therefore we might move it up position P1.

This is an expensive oscillator!
- Items alternate positions whenever we retrain the click probability model.
- Damping the oscillation leads to the same problem as in context C1.
Feedback loops in machine learning

“Information (signal) feedback loops are everywhere.”
“They are central to adaptation and learning...”

See (Bottou et al., JMLR 2013) for a possible treatment of causal loops.

Norbert Wiener, Cybernetics, 1948
Example 2
Team work

**Red Team**
Owns the code that selects content to recommend.

**Blue Team**
Owns the code that selects fonts and colors for all page components.
Red team

The red team knows the bandit literature

- There are lots of potential recommendations to select from.
- One can only model the click probabilities of the recommendations that have been shown often enough in each context.
- Therefore the red team code performs $\epsilon$-greedy exploration, showing a small proportion of random recommendations.
- Although exploration has a cost, such the system cannot discover the most relevant recommendations without it.
Blue team

The blue team also knows machine learning.

- The job of the blue team is to emphasize things that the user will like.
- The blue team code runs a click probability model to spot what users like.
- The click probability model easily discovers that some of the recommendations generated by the red team are not very good.
- Therefore these recommendations will be shown with a smaller font.
Blue team

The blue team also knows machine learning.

- The job of the blue team is to emphasize the things the user will like.
- The blue team code runs a click probability model to spot what users like.
- The click probability model easily discovers that some of the recommendations generated by the red team are not very good.
- Therefore these recommendations will be shown with a smaller font.

All the exploration items are displayed with a very small font. → They receive less clicks.
Red team

The red team is very content!

- Analysis of the exploration traffic shows that there isn’t much to improve.
  → they can safely reduce the exploration level!

- Whenever they try new features in the click model, randomized testing shows a reduction in performance
  → it is amazing, but it seems they got their model right the first time!
Red team

The red team is very content!

- Analysis of the exploration traffic shows that there isn’t much to improve. → they can safely reduce the exploration level.

- Whenever they try new features, randomized testing shows a reduction in performance. → it is amazing, but they got their model right the first time!
Paying attention

“A manager of the two teams should have paid more attention.”

“The software architect should have paid more attention.”

Micro-management does not work because it does not scale...
beyond two teams
Conclusion of part 1
Machine learning in large software systems

- Use trained models as software modules.
  → problematic because trained models offer weak contracts.

- Use learning algorithms as software modules.
  → problematic because the output of the learning algorithm depends on the training data which itself depends on every other module.

Working around these difficulties could save billions...
Our experimental paradigm is reaching its limits.
Minsky and Papert 1968, F.A.Q.

13.5 Why Prove Theorems?
Why did you prove all these complicated theorems? Couldn't you just take a perceptron and see if it can recognize $\psi_{\text{connected}}$?

No.

Are we dealing with an exact science (like mathematics) or an empirical science (like physics)?
# Essential epistemology

<table>
<thead>
<tr>
<th>Deals with</th>
<th>Exact sciences</th>
<th>Experimental sciences</th>
<th>Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axioms &amp; Theorems</td>
<td>Facts &amp; Theories</td>
<td>Artifacts</td>
<td></td>
</tr>
<tr>
<td>Truth is</td>
<td>Forever</td>
<td>Temporary</td>
<td>It works.</td>
</tr>
<tr>
<td>Examples</td>
<td>Mathematics C.S. theory</td>
<td>Physics Biology</td>
<td>Lots...</td>
</tr>
</tbody>
</table>

Truth is forever temporary. It works.

Examples: Mathematics, C.S. theory, Physics, Biology, Lots...
# Essential epistemology – Exact sciences

## Exact sciences

- **Deals with**
  - Axioms & Theorems

- **Truth is**
  - Forever

- **Examples**
  - Mathematics
  - C.S. theory
  - Physics
  - Biology
  - C.S.

<table>
<thead>
<tr>
<th>Exact sciences</th>
<th>Experimental sciences</th>
<th>Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axioms &amp; Theorems</td>
<td>Facts &amp; Theories</td>
<td>Artifacts</td>
</tr>
<tr>
<td>Forever</td>
<td>Temporary</td>
<td>It works.</td>
</tr>
</tbody>
</table>

- **Deal with axioms and theorems.**

- Axioms are neither true or false. Theorems are true once **proven**. They remain true **forever**.

- **Examples**
  - Mathematics
  - Theoretical C.S. (e.g., algorithms, type theory, …)
  - Theoretical Physics (e.g., string theory)
Essential epistemology – Experimental sciences

Experimental sciences

- Deal with facts and theories.
- Facts are true when they can be replicated. Theories are true when they predict the facts. New facts can invalidate theories that were previously considered true (K. Popper).

Examples
- Physics, Biology,…
- Social sciences, Experimental psychology,…

The word “true” has a different meaning.
Essential epistemology – Engineering

### Engineering

- **Deals with artifacts.**
- **A claim is true when the artifact works,** (the artifact fulfils a contract.)
- **Examples**
  - Mechanical engineering, ...
  - Nuclear engineering, ...
  - Computer science.
Machine learning?

What is machine learning?

- An exact science? Yes: statistical/algorithmic learning theory.
- An experimental science? Yes: papers often report on experiments.

This is all good
Machine learning?

What is machine learning?

- An exact science? Yes: statistical/algorithmic learning theory.
- An experimental science? Yes: most papers include experiments.

This is all good ... as long as we don’t mix the genres.

- “My software works really well, therefore my theory is true.”
- “This is NP-complete, therefore this experiment is wrong.”
ML as an experimental science

- Some problems of interest do not have a clear specification. They are not amenable to provable algorithms.

- Machine learning replaces the missing specification by lots of data.

- Important aspects of the data cannot be described by a compact mathematical statement (otherwise we have a specification!)

- Therefore experimentation is necessary!
ML as an experimental science

Progress during the last few decades has been driven by a single experimental paradigm!
ML as an experimental science

Progress during the last few decades has been driven by a single experimental paradigm!

This is unusual for an experimental science!

Physicists, biologists, experimental psychologists, ..., must work a lot harder to design experiments and interpret their results.
Where does the data come from?

How to improve the performance of a ML application?
- Get better algorithm? +
- Get better features? +++
- Get better data? ++++++

Data collection is hard work
- The data distribution must match the operational conditions.
- Rely on manual data curation to avoid dataset bias.
The impact of dataset bias

Training/testing on biased datasets gives unrealistic results.


<table>
<thead>
<tr>
<th>task</th>
<th>Train on:</th>
<th>Test on:</th>
<th>SUN09</th>
<th>LabelMe</th>
<th>PASCAL</th>
<th>ImageNet</th>
<th>Caltech101</th>
<th>MSRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;car&quot; classification</td>
<td>SUN09</td>
<td>28.2</td>
<td>29.5</td>
<td>16.3</td>
<td>14.6</td>
<td>16.9</td>
<td>21.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LabelMe</td>
<td>14.7</td>
<td>34.0</td>
<td>16.7</td>
<td>22.9</td>
<td>43.6</td>
<td>24.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PASCAL</td>
<td>10.1</td>
<td>25.5</td>
<td>35.2</td>
<td>43.9</td>
<td>44.2</td>
<td>39.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ImageNet</td>
<td>11.4</td>
<td>29.6</td>
<td>36.0</td>
<td>57.4</td>
<td>52.3</td>
<td>42.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Caltech101</td>
<td>7.5</td>
<td>31.1</td>
<td>19.5</td>
<td>33.1</td>
<td>96.9</td>
<td>42.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MSRC</td>
<td>9.3</td>
<td>27.0</td>
<td>24.9</td>
<td>32.6</td>
<td>40.3</td>
<td>68.4</td>
<td></td>
</tr>
<tr>
<td>Mean others</td>
<td></td>
<td>10.6</td>
<td>28.5</td>
<td>22.7</td>
<td>29.4</td>
<td>39.4</td>
<td>34.1</td>
<td></td>
</tr>
</tbody>
</table>
Increased ambitions: Big Data

Too much data for manual curation

- Big data exists because data collection is automated.
- Nobody checks that the data distribution matches the operational conditions of the system.

→ All large scale datasets are biased.

Training/testing on a big dataset can give unrealistic results.
Increased ambitions: A.I.

**Statistical machine learning**
- Building a classifier that works well under a certain distribution.

**Machine learning for artificial intelligence**
- Building a classifier that recognizes a “concept”.
- Concepts exist independently of the pattern distribution.
- Training data will never cover the whole range of possible inputs.

*In fact, a system that recognizes a “concept” fulfils a stronger contract than a classifier that works well under a certain distribution.*
Concepts ≠ Statistics

Example: detection of action “giving a phone call”

(Oquab et al., CVPR 2014)
Concepts ≠ Statistics

Computer vision is not a statistical problem

Car examples in ImageNet

Is this less of a car because the context is wrong?
Concepts ≠ Statistics

Convolutional networks can be fooled.

(Nguyen et al, CVPR 2015)
Concepts ≠ Statistics

A herd of giraffes walk down the street in the middle of some trees.

From a talk of (Zitnick 14)

Caption generation

- Eight independent papers in 2014. According to the BLEU score, some of these systems perform better than humans...
- They met in Berkeley in January 2015.
- Evaluation is very difficult.
- Example of difficulty: caption generation vs retrieval
Conclusion of part 2
Training/testing only goes so far…

Training/testing is an unusually convenient experimental paradigm…
→ it makes experimentation easier than in other sciences
→ it has played a role in the fast progress of ML.

…but may not fulfil our increased ambition (big data, AI.)
→ the end of an anomaly?

Working around this could save decades.
3 Elements of a solution
1. Process challenges

Challenges of a new kind

- We are used to face technical challenges.
- These are “process challenges.”
1. Process challenges

Engineering process

- There is a rich literature about the software engineering process.
- What should be the machine learning engineering process?
1. Process challenges

Scientific process

- We cannot *solely* rely on training/testing experiments.
- We can understand more things with ad hoc experiments.
  
  Example in question answering: the “BABI” tasks (Bordes et al., 2015)
  - questions that require combining one/two/more factoids.
  - questions that require spatial/temporal reasoning.
  - etc.

- What are the best practices in other experimental sciences?
1. Process challenges

A constructive proposition

- Machine learning papers rarely show the limits of their approach.
  - For fear that reviewers will use them against their work.
- Reviewers should instead demand that authors discuss these limits.
  - Additional credits if they illustrate these limits with experiments.
  - This would make the papers more informative...
  - and would therefore facilitate scientific progress.
2. ML with stronger contracts

How to better resist distribution shifts?

- Optimize measure of coverage instead of average accuracy.
  - Selective classification (El Yaniv’s, …)
  - KWIK (Li & Littman)
  - Conformal learning (Vovk, …)

- Very interesting properties when the data is Zipf distributed…
2. ML with stronger contracts

Zipf distributed patterns

Queries sorted in frequency order
2. ML with stronger contracts

Doubling the size of a Zipf distributed dataset.

Diminishing returns for average accuracy improvements.

No diminishing returns on number of queries for which we can learn correct answers.
3 How to package the work of others

- **Digital computers** : “software”

- **Learning machines** :
  - Trained module as a software component?
    - ❌ Trained components only offer “weak contracts”.
  - Training software?
    - ✓ If you can get the training data and replicate the rig. Recent example: AlexNet.
  - Task specific trained features
    - ✓ Nearly as good.
Example: face recognition

Interesting task: “Recognizing the faces of $10^6$ persons.”

- How many labeled images per person can we obtain?

Auxiliary task: “Do these faces belong to the same person?”

- Two faces in the same picture usually are different persons.
- Two faces in successive frames are often the same person.

(Matt Miller, NEC, 2006)
Example: NLP tagging

(Binary encoded sentence words) —> (Words embedded in 50-100 dim space) —> (Five Time-Delay Multi-layer networks):

- Part Of Speech Tagging (treebank, split 02-21 / 23)
- Named Entity Recognition (treebank, Stanford NER)
- Chunking (treebank)
- Semantic Role Labeling (propbank)
- Language Model (wikipedia, 620M examples)

(Collobert, Weston, et al., 2008-2011)
Example: object recognition

Dogs in ImageNet
(~$10^6$ dogs)

Dogs in Pascal VOC
(only ~$10^4$ images)
Example: object recognition

Several CVPR 2014 papers – figure from (Oquab et al., 2014)
Conclusion
Two process challenges

- Our experimental paradigm is reaching its limits. → Challenging the speed of our scientific progress.