Multi-Task Feature and Kernel Selection for SVMs

Tony Jebara JEBARA@CS.COLUMBIA.EDU
Computer Science Department, Columbia University, New York, NY 10027

Abstract

We compute a common feature selection or kernel selection configuration for multiple support vector machines (SVMs) trained on different yet inter-related datasets. The method is advantageous when multiple classification tasks and differently labeled datasets exist over a common input space. Different datasets can mutually reinforce a common choice of representation or relevant features for their various classifiers. We derive a multi-task representation learning approach using the maximum entropy discrimination formalism. The resulting convex algorithms maintain the global solution properties of support vector machines. However, in addition to multiple SVM classification/regression parameters they also jointly estimate an optimal subset of features or optimal combination of kernels. Experiments are shown on standardized datasets.

1. Introduction

In applied settings, many supervised datasets are available over a common input data-type (be it text, images, or sequence data) and represent different problem tasks (i.e. various classification or regression scenarios). Multi-task learning or meta-learning leverages these many datasets synergistically, aggregating them and augmenting the effective size of the total training data (Baxter, 1995; Thrun & Pratt, 1997; Caruana, 1997). This can lead to improvement in overall classification and regression performance compared to learning the tasks in isolation. We elaborate meta-learning in a support vector machine setting and focus on the case where many small datasets over different tasks select one common underlying representation. More specifically, we discuss optimizing the representation either as a feature selection configuration (Jebara & Jaakkola, 2000; Weston et al., 2000) or as a convex kernel combination (Lanckriet et al., 2002; Cristianini et al., 2001). This is done jointly while estimating a support vector classification or regression machine by using the maximum entropy discrimination (MED) framework. While previous efforts suggest finding these representations for a single task, recent theoretical results (Baxter, 2000; Ben-David & Schuller, 2003) suggest that improvements are possible with multi-task learning. This article combines the above motivations into a joint multi-task feature and kernel selection SVM framework.

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

This paper is organized as follows. Section 2 summarizes maximum entropy discrimination and how it generates the support vector machine. We then discuss augmenting the SVM with feature selection in Section 3 and kernel selection in Section 4 yet only in isolated learning settings. In Section 5 we upgrade feature selection to a meta-learning scenario where many classifiers share one common feature selection configuration. Section 6 similarly upgrades kernel selection. Experiments are interleaved within each section when appropriate. Section 7 ends with a discussion.

2. Support Vector Machines as a Maximum Entropy Problem

The maximum entropy discrimination MED formalism introduced in (Jaakkola et al., 1999) is a flexible generalization of support vector machines. MED produces a solution that is a distribution of parameter models $P(\Theta)$ rather than finding a single parameter setting Θ^*. It is this characteristic that makes it straightforward to augment MED solution distributions to be joint densities over not only parameters of a single classifier, but several classifiers, representations, feature selection and kernel selection configurations. In other words, one may consider M support vector machines sharing a common feature selection configuration s and readily solve for $P(\theta_1, \ldots, \theta_M, s)$ in the maximum entropy formalism. The implementation of these MED extensions still produces a large margin support vector while also estimating additional parameters all within a globally optimizable convex program.

We now summarize the maximum entropy discrimination formalism in its simplest form. Assume we are given training examples $X_t \in \mathbb{R}^d$ with their corresponding labels $y_t \in \{\pm 1\}$ for $t = 1 \ldots T$. We want to find a linear discriminant function $L(X;\Theta) = \theta^T X + b$ whose sign agrees with this labeled training set. The discriminant is specified by $\Theta = \{\theta, b\}$ containing both a linear parameter vector θ as well as a scalar bias value b. An SVM finds a specific Θ^* setting of these two deterministic quantities that agrees with the labeling, in other words $y_t L(X_t;\Theta^*) \geq \gamma_t \; \forall t$ where γ_t are scalars typically set to unity, i.e. $\gamma_t = 1$. This is done while ensuring that the magnitude $\frac{1}{2}\|\theta\|^2$ is kept small for regularization. In MED, however, we solve for a distribution over solutions $P(\Theta)$ such that the *expected* value of the discriminant under this distribution agrees with the labeling. In addition to finding a $P(\Theta)$ that satisfies classification constraints in the expectation, MED regularizes the solution distribution $P(\Theta)$ by either maximizing its entropy or minimizing its relative entropy towards some prior target distribution $P_0(\Theta)$. We use the relative Shannon entropy given by $D(P\|P_0) = \int_\Theta P(\Theta) \ln P(\Theta)/P_0(\Theta) d\Theta$. Note that minimizing relative entropy is more general since choosing $P_0(\Theta)$ uniform gives maximum entropy. Thus, MED solves the constrained optimization problem

$$\min_{P(\Theta)} D(P\|P_0) \text{ s.t. } \int_\Theta P(\Theta)[y_t L(X_t;\Theta) - \gamma_t] \geq 0 \;\; \forall t$$

The solution for the posterior $P(\Theta)$ is straightforward (Jaakkola et al., 1999) and is given by the usual maximum entropy method

$$P(\Theta) \;\; = \;\; \frac{P_0(\Theta)}{Z(\lambda)} e^{\sum_t \lambda_t [y_t L(X_t;\Theta) - \gamma_t]}.$$

Here $Z(\lambda)$ is obtained by normalizing $P(\Theta)$ and the optimal setting of the λ Lagrange multipliers (λ_t for $t = 1 \ldots T$) is found by maximizing the concave objective function $J(\lambda) = -\ln Z(\lambda)$. Given λ, the solution distribution $P(\Theta)$ is fully specified. It is then straightforward to use this distribution for predicting the label of a new exemplar X via $\hat{y} = \text{sign}(\int_\Theta P(\Theta) L(X;\Theta) d\Theta)$. Furthermore, often the effect of the distribution $P(\Theta)$ can be mimicked by a deterministic parameter setting on our discriminant function since we can solve the integrals analytically without computational effort.

Interestingly, the above framework *exactly reproduces* support vector machines. We simply assume the prior distribution factorizes into a prior over the vector parameters and a prior over scalar bias, $P_0(\Theta) = P_0(\theta)P_0(b)$. The first is a white zero-mean Gaussian of the form $P_0(\theta) = N(\theta|0, I)$ which encourages linear models of low magnitude (a traditional regularization principle in the literature). The second is a non-informative (i.e. flat) prior $P_0(b) = N(b|0, \infty)$ indicating that any scalar bias is equally probable a priori. The resulting objective function $J(\lambda)$ is identical to the SVM dual optimization problem

$$J(\lambda) \;\; = \;\; \sum_t \lambda_t - \frac{1}{2} \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} X_t^T X_{t'}$$

subject to non-negativity constraints on the λ_t values (since the expectation constraints in the maximum entropy problem used *greater-than* inequalities). Furthermore, the non-informative prior yields the equality constraint $\sum_t \lambda_t y_t = 0$. The above can be readily maximized using quadratic programming or axis-parallel methods that update only a limited number of Lagrange multipliers at a time while others are fixed. The final decision rule is given by $\int_\Theta P(\Theta) L(X;\Theta) d\Theta = \sum_t y_t \lambda_t X_t^T X + \hat{b}$ where the expected bias $\hat{b}$ is found via the Karush-Kuhn-Tucker conditions as usual. The above is readily kernelizable as well simply by replacing all inner products $X_t^T X_{t'}$ with kernel function evaluations $k(X_t, X_{t'})$.

It is now straightforward to obtain variations on support vector machines by considering other prior distributions instead of the Gaussians above. We can also *augment* the solution distribution $P(\Theta)$ such that it is a density over other variables in addition to the linear classifier's θ, b parameters on their own. For instance, we can consider a non-separable support vector machine if we make the γ_t scalars variable and consider solving for an MED distribution that is a joint function of both γ and the parameters Θ as $P(\Theta, \gamma)$. Assume the prior distribution factorizes over margins as follows $P_0(\Theta, \gamma) = P_0(\Theta) \prod_t P_0(\gamma_t)$. Here, we choose $P_0(\gamma_t) \propto \exp(-c + c\gamma_t), \gamma < 1$ as an exponential distribution which favors large margins yet also allows negative margin values (i.e. miss-classifications). The solution distribution $P(\Theta, \gamma)$ is found again as in the above general form (a product of the prior $P_0(\Theta, \gamma)$ and exponentiated constraints):

$$P(\Theta, \gamma) \;\; = \;\; \frac{P_0(\Theta, \gamma)}{Z(\lambda)} e^{\sum_t \lambda_t [y_t L(X_t;\Theta) - \gamma_t]}. \quad (1)$$

The resulting SVM-like concave maximization problem emerges from the partition function $Z(\lambda)$. This new objective function $J(\lambda) = -\ln Z(\lambda)$ equals

$$\sum_t [\lambda_t + \ln(1 - \lambda_t / c)] - \frac{1}{2} \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} X_t^T X_{t'}.$$

The above effectively adds barrier functions preventing λ_t's from growing beyond the regularization parameter c. The decision rule $\hat{y} = \int_{\Theta,\gamma} P(\Theta,\gamma)L(X;\Theta)d\Theta d\gamma$ remains identical. We now continue this line of reasoning and the notion of *augmented* solution distributions to endow support vector machines with other interesting properties beyond non-separability. For instance, we can now readily deal with feature and kernel selection.

3. Feature Selection

Feature selection for support vector machines was discussed in earlier work in maximum entropy discrimination (Jebara & Jaakkola, 2000) as well as in deterministic SVM settings (Weston et al., 2000). For feature selection, consider modifying the discriminant with a binary feature selection switch vector $s = [s_1, \ldots, s_D]$ where $s_d \in \{0, 1\}$. This vector now becomes part of a more elaborate model $\Theta = \{\theta, b, s\}$. The parameters of the model interact via the discriminant function $L(X;\Theta) = \sum_d \theta_d s_d X_d + b$ which turns certain entries of the input vector X on and off depending on the setting of s. Here d indexes into the dimensionality of the D-dimensional vectors θ or X or s. The prior over models now also involves switches and we assume it factorizes as $P_0(\Theta,\gamma) = P_0(\theta)P_0(b)\prod_d P_0(s_d)\prod_t P_0(\gamma_t)$. We again assume a Gaussian prior over θ, a non-informative prior over b, and exponential priors over γ_t (although other choices are possible). The natural choice of a prior over switches s_d is a Bernoulli distribution given by $P_0(s_d) = \rho^{s_d}(1-\rho)^{1-s_d}$. Here the prior is controlled by ρ which affects the likelihood of including any given feature; $\rho = 1$ preserves all features and would simply give us an SVM. Meanwhile smaller ρ values in the prior encourage the SVM linear model solution to have many zeros and use fewer features. The solution distribution is found using Equation 1 yet computing the partition function now involves normalizing over the switch settings as well as over the parameters θ and b. This yields an alternative $J(\lambda)$ objective function

$$\sum_t \ln(e^{\lambda} - e^{\lambda}\lambda_t/c) - \sum_d \ln\left[1 - \rho + \rho e^{\frac{1}{2}(\sum_t \lambda_t y_t X_{t,d})^2}\right]$$

which is maximized subject to $\sum_t \lambda_t y_t = 0$. The $J(\lambda)$ above can be maximized iteratively via Newton-Raphson or axis-parallel methods (i.e. iteratively locking all Lagrange multipliers except for 1 or 2 and searching for their optimal setting). Clearly, if ρ is held at unity in the above, we once again obtain exactly the support vector machine optimization problem. However, smaller settings of ρ affect the optimization of Lagrange multipliers values and ultimately provides a sparsified linear model to use in the dot product with each input datum vector. Note that this $J(\lambda)$ remains a concave objective with convex constraints and hence can be solved without local minima problems. It yields the largest margin linear support vector machine while also sparsifying features. Given the final setting of the (non-negative) Lagrange multipliers and $\hat{b}$ (the expected value of b via the Karush-Kuhn Tucker conditions), we compute our discriminant function output $\hat{y}$ as the sign of $\int_{\theta,b,s} P(\theta,b,s)L(X;\theta,b,s)$ which simplifies to:

$$\sum_d \left(\frac{\rho\sum_t \lambda_t y_t X_{t,d}}{\rho + (1-\rho)\exp(-\frac{1}{2}[\sum_t \lambda_t y_t X_{t,d}]^2)}\right) X_d + \hat{b}.$$

In the above, $X_{t,d}$ refers to the d'th dimension of the vector representing the t'th datum in our training dataset. Note how we have terms that are attenuated nonlinearly with smaller ρ settings in their interaction with the query datum X. We use W_d to denote the terms multiplying each X_d in the parentheses. These behave as attenuated weights which are driven close to zero when ρ is small. While the model is still a linear classifier $\hat{y} = \text{sign}(\sum_d X_d W_d + \hat{b})$, most of the dimensions of X_d get multiplied with vanishingly small scalar W_d weights and will be ignored (selected out).

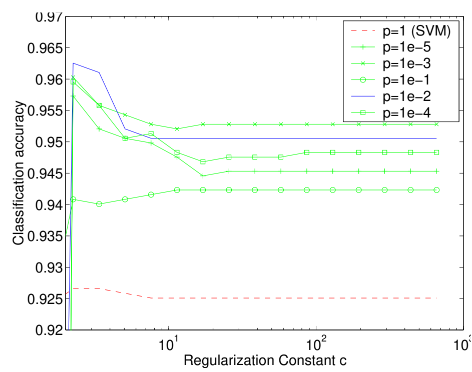

Figure 1. Feature selection for sequence classification. The dashed line is the SVM classification accuracy; solid lines show accuracy at various levels of feature selection.

To evaluate feature selection with SVMs in an isolated learning setting, the UCI splice-site dataset was used to classify gene sequences as intron-exon or exon-intron sites. Each datum in the dataset is a sequence of 60 nucleotides (A, G, C, T) which are represented as 4-tuple codes according to $A = (0001)$, $C = (0100)$, $G = (0010)$ and $T = (0001)$. Codes were mixed with weights to represent uncertain symbols such as "G or T". Therefore, our final input space's dimensionality is $D = 240$. Of a total of 1535 sequences, 200 were used for training an SVM and the remaining were used for testing. Figure 1 depicts the classification accuracy on test data of the feature selection SVM at various settings of the regularization parameter c as well as the sparsification parameter ρ. Note that at $\rho = 1$ we have a traditional support vector machine with no feature selection. Clearly, some feature selection is advantageous and improves generalization increasing overall accuracy from 92% up to approximately 96%. In Figure 2 we plot the 240 weights W_d of the resulting linear model for both a traditional support vector machine when $\rho = 1$ and a feature selection SVM with $\rho = 1e-4$. Feature selection can be clearly seen in the latter where many weights W_d are effectively 0.

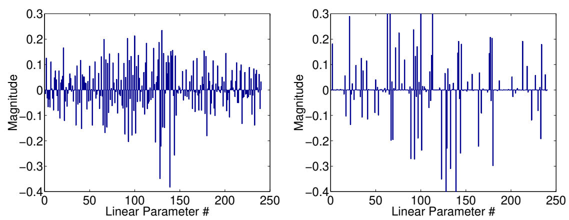

Figure 2. Sparsification of the linear model. On the left, values of the linear weights of the SVM are highly variable. On the right, the feature selection SVM (with $\rho = 1e - 4$) prunes the linear model which has many zeros.

The above feature selection technique is straightforward to derive for support vector machine regression and is not limited to classification (Jebara & Jaakkola, 2000). This is actually the case *for all subsequent derivations and SVM extensions* in this article. We next go beyond linear feature selection and consider the kernel selection problem.

4. Kernel Selection

Another crucial issue for support vector machines is choosing the kernel function. Kernels introduce different nonlinearities into the SVM problem by mapping input data X implicitly into Hilbert space via a function $\phi(X)$ where it may then be linearly separable. While the explicit computation of the features or the mapping to Hilbert space may be unwieldly, a support vector machine only requires inner products of these $\phi(X)$ features. A kernel is an efficient way to compute such an inner product and provides the same scalar output $k(X_t, X_{t'}) = \phi(X_t)^T\phi(X_{t'})$. It is straightforward to kernelize the SVM and related learning machines to accommodate nonlinear classification and to potentially handle non-vectorial inputs by swapping all inner products $X_t^T X_{t'}$ in the formulation with kernel function evaluations $k(X_t, X_{t'})$ (subject to some caveats, i.e. Mercer's condition). In fact, in implementing a support vector machine and optimizing for the Lagrange multipliers (for instance, via quadratic programming), it is natural to build a $T \times T$ Gram matrix K whose entries are given by the kernel function evaluated over all pairs of data-points $K_{t,t'} = k(X_t, X_{t'})$. Different kernels will accommodate different nonlinear mappings and the performance of the resulting SVM will often hinge on the appropriate choice of the kernel.

However, searching for different kernels either via trial-and-error or other exhaustive means can be a computationally daunting problem (Chapelle & Vapnik, 1999). This search is particularly difficult if we also want to consider combining kernels in convex combinations (a *continuous* optimization problem) to mix various nonlinear mappings in search of the optimal Hilbert space. More efficient kernel selection approaches involve either directly manipulating entries of the Gram matrix or, alternatively, searching over a convex combination of user-specified kernel functions (Lanckriet et al., 2002; Cristianini et al., 2001). In this article, we will approach the problem of kernel selection by estimating the optimal weighted combination of several prototype or base kernels provided by the user. It is well known that a legitimate kernel can be created by convex combination of kernels as long as they individually satisfy Mercer's condition. We will derive this additional estimation problem in a straightforward manner using the maximum entropy discrimination framework.

We cast the kernel selection problem by estimating a positive linear combination of a set of D kernel functions $k_d(.,.)$ where $d = 1 \ldots D$. Excuse the abuse of notation as we recycle variables and use d to index into the set of kernel or Hilbert space mappings under consideration (as opposed to indexing into the dimensionality of the input space). We wish to estimate the best combination of these base kernels while *jointly* finding the optimal support vector machine classifier. In other words, we will find a resulting final kernel of the form: $k(.,.) = \sum_d W_d k_d(.,.)$ where W_d are non-negative weights. In fact, we would also like to have many of these W_d weights drop close to zero to fully reject the contribution of some kernels that are not useful for the given classification task.

We approach the problem by noting that different combinations of kernels correspond to different weightings of their corresponding mappings $\phi_d(.)$ for $d = 1 \ldots D$. We are uncertain which mapping to use and consider using them all by concatenating all mappings into an augmented Hilbert space. Consider the resulting discriminant function:

$$L(X;\Theta) = \sum_d s_d \theta_d^T \phi_d(X) + b.$$

This model contains several parameter vectors θ_d for each mapping $\phi_d(.)$. The resulting full model $\Theta =$

$\{\theta_1, \ldots, \theta_D, b\}$ only interacts linearly with our aggregated mapping in this augmented Hilbert space, in other words, we could have written the following general form for the discriminant $L(X;\Theta) = \Theta^T \Phi(X)$. As with feature selection, we will assume that the switch vector s constitutes D binary switches s_d and consider a factorized Bernoulli prior distribution over them. The prior is thus $P_0(\Theta,\gamma) = P_0(b)\prod_d P(\theta_d)P(s_d)\prod_t P(\gamma_t)$. Here, each $P(\theta_d)$ is individually a zero-mean white Gaussian distribution over its vector θ_d (each θ_d vector can potentially be of a different dimensionality). For simplicity, we will also use an *informative* prior for the bias given by $P_0(b) = N(b|0,\sigma^2)$. This removes the equality constraint $\sum_t \lambda_t y_t = 0$ and replaces it with a quadratic penalty (to be derived shortly). The priors on margins γ_t are as before. Given the prior, we recover the posterior distribution $P(\Theta,\gamma)$ via Equation 1.

We next turn to the computation of the partition function which normalizes this $P(\Theta,\gamma)$. We will simplify it by considering the contribution of each of the variables θ, s, b, γ to the partition individually as follows

$$\begin{aligned} Z(\lambda) &= \int_\Theta P_0(\Theta) e^{\sum_t \lambda_t y_t L(X_t;\Theta)} \int_\gamma P_0(\gamma) e^{-\sum_t \lambda_t \gamma_t} \\ &= Z_\Theta(\lambda) Z_\gamma(\lambda) \end{aligned}$$

We can then further simplify the $Z_\Theta(\lambda)$ term as

$$\int_{\theta,s} P_0(\theta,s) e^{\sum_t \lambda_t y_t \sum_d s_d \theta_d^T \phi_d(X_t)} \int_b P_0(b) e^{\sum_t \lambda_t y_t b}.$$

The left integral evaluates $Z_{\theta,s}(\lambda)$ while the right integral evaluates $Z_b(\lambda)$. The computation of the bias contribution is straightforward producing $Z_b(\lambda) = e^{\frac{1}{2}\sigma^2(\sum_t y_t \lambda_t)^2}$. Note that if we let $\sigma \to \infty$, the partition function diverges unless we have the constraint $\sum_t y_t \lambda_t = 0$ as in the non-informative prior case.

The remaining term involves the contribution of all the $d = 1 \ldots D$ models and the binary switches, and is given by

$$\begin{aligned} Z_{\theta,s}(\lambda) &= \int_{\theta,s} \prod_d P_0(s_d) P_0(\theta_d) e^{\sum_t \lambda_t y_t \sum_d s_d \theta_d^T \phi_d(X_t)} \\ &= \prod_d \sum_{s_d=0}^{1} P_0(s_d) e^{s_d \frac{1}{2} \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} k_d(X_t, X_{t'})}. \end{aligned}$$

The resulting overall MED objective function is then the negated logarithm of the product of all the partition function terms:

$$\begin{aligned} J(\lambda) = \sum_t \lambda_t + \ln(1 - \lambda_t/c) \; - \frac{\sigma^2}{2} \left(\sum_t y_t \lambda_t \right)^2 \\ - \sum_d \ln \left(1 - \rho + \rho e^{\frac{1}{2} \sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} k_d(X_t, X_{t'})} \right) \end{aligned}$$

which is maximized *only* over non-negativity constraints on the Lagrange multipliers. This is straightforward to do via Newton-Raphson or axis-parallel updates on individual values of λ_t.

Once again, if we select $\rho = 1$, we note that the above kernel selection problem degenerates since all logarithms cancel with the exponential functions and the optimization is equivalent to the case where all kernels are summed equally. This means our final kernel is just the total of the base kernels $k(.,.) = \sum_d k_d(.,.)$. However, in the cases where ρ is less than unity, we note that the objective function involves a nonlinear combination of the quadratic terms from the different kernels, i.e. a nonlinear mixing of multiple support vector machine quadratic cost problems.

To compute the actual discriminant function, the MED recipe suggests taking the expectation under $P(\Theta,\gamma)$ of the discriminant function $L(X;\Theta)$. This gives the following decision rule:

$$\hat{y} \;=\; \mathrm{sign}\left(\sum_t y_t \lambda_t \sum_d \hat{s}_d k_d(X_t, X) + \hat{b} \right)$$

The formula involves the expected value of the bias $\hat{b}$ and the expected value of the binary switches $\hat{s}_d$. It also implicitly involves the expected value of the models $\hat{\theta}_d$ but these are only written in terms of the Lagrange multipliers. In the non-informative case, the *expected* value of the bias is given by the mean of the Gaussian posterior $P(b)$ and is merely $\hat{b} = \sigma^2 \sum_t y_t \lambda_t$. The expected value of each switch (or the weight of each kernel in our SVM problem) is

$$\hat{s}_d \;=\; \frac{\rho}{\rho + (1-\rho)\exp(-\frac{1}{2}\sum_{t,t'} \lambda_t \lambda_{t'} y_t y_{t'} k_d(X_t, X_{t'}))}.$$

Since these are actually positive scalars, the above final formula for the decision rule involves a positive combination of the base kernels. We can immediately rewrite the decision rule as a standard SVM classifier yet using the aggregated kernel $k(.,.) = \sum_d \hat{s}_d k_d(.,.)$. Another benefit is that, as in the feature selection case, many $\hat{s}_d$ values will be vanishingly small (particularly as we use smaller values of ρ). For computationally efficiency in practice, we round off small values of $\hat{s}_d$ to zero and avoid computing their corresponding kernel evaluations altogether in the final classifier's formula. This indicates that some base kernels will be pruned away completely from our final combination while the remaining ones will be mixed with various weights. In practice, we assume all base kernels are normalized and $k_d(X_t, X_t) = 1$ since the switches combining them are on the same scale $\hat{s}_d \in [0,1]$. One standard way to normalize the base kernels is to replace them as follows:

$$k_d(X_t, X_t') \quad \leftarrow \quad \frac{k_d(X_t, X_t')}{\sqrt{k_d(X_t, X_t) k_d(X_{t'}, X_{t'})}}$$

Finally, the aggregated kernel is legitimate since positive combinations of Mercer base-kernels are Mercer.

We evaluated the kernel selection technique on the UCI Isolet dataset. We used 200 training examples and 600 testing examples. The task is to compute a class label representing which letter of the alphabet was spoken from a vector of auditory features. This is a 26-class multi-class classification problem. We convert this problem into 26 binary classification problems by considering the one versus rest scenario. At this point, the standard approach is to train individual support vector machines on these 26 binary classification problems. To help improve accuracy, we investigate finding the best combination of different kernels on the auditory features. Each binary classifier optimizes the kernel selection individually by choosing a combination of polynomial kernels and radial basis function kernels. The polynomials used were of order 1,2,3,4 and the RBF kernels used were of standard deviation 10,1,0.1, and 0.01. Thus, a total of 8 base kernels were combined to form a final aggregate kernel for each classifier in isolation. We show the test error of the resulting classifiers in terms of the total binary error rate, in other words, the sum of all the errors the individual binary classifiers made. This is a pessimistic error estimate since other methods may be used to reduce errors by fusing the output of the 26 binary classifiers (for example via error-correcting codes). Figure 3 summarizes the total error rate for the kernel selection method as we explore various settings of the ρ and c parameters. At $\rho = 1$, we have the SVM case which assumes that our aggregated kernel is the sum of all 8 base kernels with equal (unity) weight. Lower values of ρ prune away poor kernels and reduce error by selecting the subset of kernels which are more appropriate for the task. The fact that lowering ρ keeps improves accuracy suggests that pruning kernels improves performance and that one kernel is consistently outperforming the others.

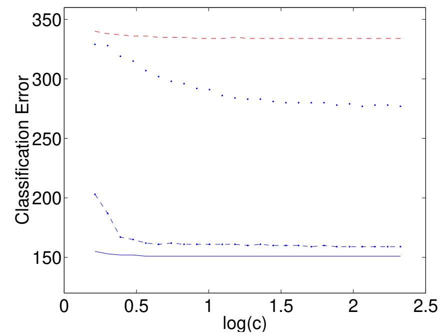

Figure 3. Kernel selection test error rate over 8 different base kernels (polynomials and RBF) for the Isolet dataset under varying c and ρ levels. The dashed line is a regular SVM, the dotted line is kernel selection with $\rho = 1e-1$, the dotted-dashed line $\rho = 1e-2$ and the solid line $\rho = 1e-3$.

5. Multi-Task Feature Selection

So far, we have seen a representational aspect of the learning process, be it a feature selection on the input space or a kernel selection, yet it was only driven by a single task and model. However, such types of representations may be learned for multiple tasks more powerfully than in isolation. We will now explicate the case where many tasks are present requiring different classification models yet dealing with the same input space which is consistently corrupted by nuisance or irrelevant features. For instance, we may have a dataset of face images that classifies each face as male or female and in another dataset, face images are classified as child or adult. These datasets can synergistically discover that only the pixels which are occupied by facial imagery are useful for either task and that pixels corresponding to background imagery should be ignored. This intuition can be formalized via the theoretical work of (Baxter, 2000) and (Ben-David & Schuller, 2003). The latter discusses how generalization can benefit from a multi-task or meta-learning situation when several classifiers are $\mathcal{F}$-related and involve controlled changes from a base classifier. The authors propose various settings where improvements can be made depending on the VC-dimension of the base classifier and the $\mathcal{F}$-relation tying the individual learners. We can view a feature selection configuration $s = [s_1, \ldots, s_D]$ as a way of estimating parameters for this base classifier while many subsequent θ_m linear classifier models are spanning various $\mathcal{F}$-related joint distributions over (X, y) or over hypotheses.

For the multi-task feature selection SVM, we will assume we are dealing with inputs $X \in \mathbb{R}^D$. We will have M different training datasets containing $X_{t,m} \in \mathbb{R}^D$ with corresponding binary labels $y_{t,m} \in \{0, 1\}$ for $t = 1 \ldots T_m$ data-points in $m = 1 \ldots M$ tasks. These tasks are to be classified with M different discriminant functions all sharing a single feature selection configuration. Each of the corresponding M models $\theta_1, \ldots, \theta_M$ is a D-dimensional vector and has its own scalar bias $b_1, \ldots, b_M$. However, all share a common D-dimensional binary feature selection vector s. The aggregated parameters of the model are denoted by

$\Theta = \{\theta_1, \ldots, \theta_M, b_1, \ldots, b_M, s\}$. Each of the M discriminant functions can then be written as:

$$L(X; \Theta_m) = L(X; \theta_m, b_m, s) = \sum_d s_d \theta_{m,d} X_d + b_m.$$

Unlike before, the MED classification constraints are now over many datasets as follows:

$$\int_{\Theta,\gamma} P(\Theta, \gamma)[y_{m,t} L(X_{m,t}; \theta_m, b_m, s) - \gamma_{m,t}] \geq 0 \quad \forall m, t$$

There are $\sum_m T_m$ total such constraints from our M classification problems and we therefore anticipate having $\sum_m T_m$ non-negative Lagrange multipliers $\lambda_{m,t}$. Minimizing the relative entropy of $P(\Theta, \gamma)$ to a prior $P_0(\Theta, \gamma)$ subject to the above constraints again produces the classical maximum entropy solution, a product of the prior and the exponentiated constraints:

$$P(\Theta, \gamma) = \frac{P_0(\Theta, \gamma)}{Z(\lambda)} e^{\sum_{m,t} \lambda_{m,t} [y_{m,t} L(X_{m,t}; \Theta_m) - \gamma_{m,t}]}.$$

We select a prior that factorizes as follows $P_0(\Theta, \gamma) = \prod_d P_0(s_d) \prod_m P_0(\theta_m) P_0(b_m) \prod_t P_0(\gamma_{m,t})$. Due to the lack of a priori knowledge about how tasks interact, we start with a factorized prior over tasks yet will eventually find a posterior that need not remain factorized afterwards (starting with non-factorized priors may be possible as in related work by (Evegniou & Pontil, 2004)). We again utilize zero-mean white Gaussian priors for models, Bernoulli priors for switches, zero-mean Gaussian priors for biases and exponential priors for margins. We can now readily evaluate the partition function. Focusing on the contribution from the biases, we obtain $Z_{b_1,\ldots,b_M} = \prod_m e^{\frac{1}{2}\sigma^2 (\sum_t y_{m,t} \lambda_{m,t})^2}$. We also have the contribution from the vector models and the single switch; this term $Z_{\theta_1,\ldots,\theta_M,s}(\lambda)$ is

$$\int_{\theta_1,\ldots,\theta_M,s} P_0(s) \prod_m P_0(\theta_m) e^{\sum_{m,t} \lambda_{m,t} y_{m,t} \sum_d s_d \theta_{m,d} X_{m,t,d}}$$

$$= \prod_d \sum_{s_d=0}^{1} P_0(s_d) e^{s_d \frac{1}{2} \sum_{m=1}^{M} \left[\sum_{t=1}^{T_m} \lambda_{m,t} y_{m,t} X_{m,t,d}\right]^2}.$$

Multiplying all contributions of the partition function and taking the negative logarithm yields $J(\lambda)$ as

$$\sum_{m,t} \lambda_{m,t} + \ln(1 - \lambda_{m,t}/c) - \sum_m \frac{\sigma^2}{2} \left(\sum_t y_{m,t} \lambda_{m,t} \right)^2$$
$$- \sum_d \ln \left(1 - \rho + \rho e^{\frac{1}{2} \sum_m \left[\sum_t \lambda_{m,t} y_{m,t} X_{m,t,d}\right]^2} \right).$$

The objective is maximized subject to non-negativity constraints and produces the optimal Lagrange multipliers. To recover the final decision rule, each of the $m = 1 \ldots M$ classifiers is then computed from the sign of the expected m'th discriminant:

$$\int_\Theta P(\Theta) L(X; \vec{\theta}_m, b_m, s) = \sum_d \hat{s}_d \hat{\theta}_{m,d} X_d + \hat{b}_m$$

where we have the following expected quantities from the solution distribution $P(\Theta)$:

$$\hat{\theta}_m = \sum_t y_{m,t} \lambda_{m,t} X_{m,t}$$
$$\hat{s}_d = \frac{\rho}{\rho + (1-\rho)\exp(-\frac{1}{2}\sum_m [\sum_t \lambda_{m,t} y_{m,t} X_{m,t,d}]^2)}$$
$$\hat{b}_m = \sigma^2 \sum_t y_{m,t} \lambda_{m,t}.$$

Note that the meta-learning case performs no feature selection when $\rho = 1$ and there is no notion of multi-task learning. All models become optimized individually as separate SVMs and the switches are all unity. Conversely, for smaller settings of ρ we will find a single feature selection configuration that achieves good classification performance for all models.

The UCI Dermatology dataset was used to evaluate meta-learning with linear feature selection. This is a 6-class dataset which was converted to 6 binary one-versus-rest classification problems. Typically, individual SVMs would be trained on each binary classification. If, however, we are to estimate a feature selection configuration for this dataset, we would like all 6 SVMs to share the same one. This makes the problem a meta-learning one [1]. The inputs here were 33-dimensional and we used 200 training and 166 testing data points. Figure 4 reports the total binary classification error for the 6 tasks as we vary ρ and cross-validate over c. The dashed line depicts performance when each SVM for each binary classification task has its own feature selection configuration (i.e., the independent learning case). Meanwhile, the solid line depicts the performance when each SVM has to share a common feature selection configuration (i.e., the meta-learning case). The upper left corner of the plot is the performance of an SVM when no feature selection whatsoever is used (there, meta-learning equals isolated learning). Feature selection improves the SVM error while meta-learned feature selection improves results even further.

[1]Note that this form of meta-learning is adverserial since the input data for each binary task is redundant. Meta-learning can only improve error via the interdependencies between the binary tasks and not because of the availability of additional input data samples from different supervised problems.

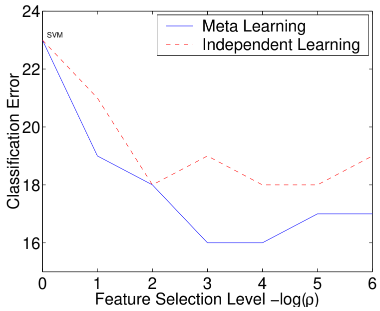

Figure 4. Meta learning and feature selection classification test errors. A multi-class classification problem is converted to multiple binary classification problems and handled with meta-learning to obtain a common feature selection configuration. Varying levels of feature selection $-\ln(\rho)$ are shown after optimizing over the regularization parameter c. The dashed line is the independent learning case where feature selection is done individually for each task. The solid line is the meta-learning case which ties a common feature selection across all tasks.

6. Multi-Task SVM Kernel Selection

It is now immediate to derive the multi-task kernel selection case since it is merely a combination of the extensions from Section 3 to Section 4 and from Section 3 to Section 5. We only write out the resulting objective function and the decision rule. For brevity, we rewrite indexes without any commas, i.e. $\lambda_{mt} = \lambda_{m,t}$. The concave objective $J(\lambda)$ is then

$$\sum_{mt} \lambda_{mt} + \ln(1 - \lambda_{mt}/c) - \sum_{m} \frac{\sigma^2}{2} \left(\sum_t y_{mt}\lambda_{mt} \right)^2 - \sum_d \ln \left(1 - \rho + \rho e^{\frac{1}{2} \sum_{mtt'} \lambda_{mt}\lambda_{mt'} y_{mt} y_{mt'} k_d(X_{mt}, X_{mt'})} \right)$$

which is maximized over non-negativity constraints. The final decision rule for the m'th classifier is

$$\hat{y} \quad = \quad \text{sign} \left(\sum_t y_{mt}\lambda_{mt} \sum_d \hat{s}_d k_d(X_{mt}, X) + \hat{b}_m \right)$$

where $\hat{b}_m$ are given as in Section 5. The expected switch values $\hat{s}_d$ are

$$\frac{1}{1 + \frac{(1-\rho)}{\rho} \exp(-\frac{1}{2} \sum_{mtt'} \lambda_{mt}\lambda_{mt'} y_{mt} y_{mt'} k_d(X_{mt}, X_{mt'}))}$$

which give us the weights on the combination of base kernels to use for multi-tasks.

7. Discussion

A framework for feature and kernel selection in an SVM classification (and regression) setting was put forward. However, here, features or kernel combinations are found from *multiple* synergistic problems. If such problems have commonalities in terms of the features they should be sensitive to or in terms of the nonlinearities they require for good classification, they can mutually benefit from multi-task or meta-learning. Maximum entropy discrimination generated these novel extensions of support vector machines which were implemented as convex programs. These avoid local minima as they simultaneously estimate classifier models and perform feature/kernel selection. Preliminary empirical results were promising and encourage further empirical and theoretical work.

References

Baxter, J. (1995). Learning internal representations. *Proceedings of the 8th International ACM Workshop on Computational Learning Theory*.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, 149–198.

Ben-David, S., & Schuller, R. (2003). Exploiting task relatedness for multiple task learning. *Conference on Learning Theory and Kernel Machines*.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Chapelle, O., & Vapnik, V. (1999). Model selection for support vector machines. *NIPS*.

Cristianini, N., Kandola, J., Elisseeff, A., & Shawe-Taylor, J. (2001). On kernel target alignment. *NIPS*.

Evegniou, T., & Pontil, M. (2004). *Regularized multi–task learning* (Technical Report RN/04/04). Dept of Computer Science, UCL.

Jaakkola, T., Meila, M., & Jebara, T. (1999). Maximum entropy discrimination. *NIPS*.

Jebara, T., & Jaakkola, T. (2000). Feature selection and dualities in maximum entropy discrimination. *Uncertainty in Artifical Intelligence 16*.

Lanckriet, G., Cristianini, N., Bartlett, P., El Ghaoui, L., & Jordan, M. (2002). Learning the kernel matrix with semi-definite programming. *ICML*.

Thrun, S., & Pratt, L. (1997). *Learning to learn*. Kluwer Academic.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2000). Feature selection for SVMs. *NIPS 13*.