1 Training and Approximation of a Primal Multiclass Support Vector Machine

Alexander Zien[1,2] and Fabio De Bona[1] and Cheng Soon Ong[1,2]

[1] Friedrich Miescher Lab., Max Planck Soc., Spemannstr. 39, Tübingen, Germany
[2] MPI for Biological Cybernetics, Spemannstr. 38, Tübingen, Germany

Abstract. We revisit the multiclass support vector machine (SVM) and generalize the formulation to convex loss functions and joint feature maps. Motivated by recent work [Chapelle, 2006] we use logistic loss and softmax to enable gradient based primal optimization. Kernels are incorporated via kernel principal component analysis (KPCA), which naturally leads to approximation methods for large scale problems. We investigate similarities and differences to previous multiclass SVM approaches. Experimental comparisons to previous approaches and to the popular one-vs-rest SVM are presented on several different datasets.
Keywords: Multiclass SVM, Primal Optimization.

1.1 Introduction

SVMs [Schölkopf and Smola, 2002] have initially been developed for binary classification. Multiclass classification can be considered the simplest and most natural learning problem going beyond the binary setting. It is possible, and still common practice, to reduce multiclass learning to a set of binary classification problems [Duan and Keerthi, 2005]. [Crammer and Singer, 2001] provides evidence that, in general, genuine multiclass approaches can be superior, but other studies suggest that it is hard to beat the one-vs-rest heuristic, both in accuracy and computational complexity. Anyway the multiclass SVM is of theoretical interest as it is conceptually identical to large margin structured output learning [Taskar *et al.*, 2003,Tsochantaridis *et al.*, 2004].

Multiclass SVMs (in the following briefly called m-SVMs) have now been investigated for several years [Hsu and Lin, 2002,Weston and Watkins, 1999]. Most literature on m-SVMs (as on SVMs) focusses on dual optimization. Motivated by [Chapelle, 2006] we investigate the primal problem for m-SVMs. In the next section we set up optimization problem and gradients. In section 1.3, we present numerical experiments. We first compare primal m-SVM training to primal one-vs-rest. Then we investigate approximate optimization and relate the incurred accuracy drop to the speed-up.

1.2 Methods

1.2.1 Multiclass SVMs (m-SVMs)

We consider linear classifiers in a feature space $\mathcal{H}$ defined by a potentially non-linear map $\Phi : \mathcal{X} \to \mathcal{H}$. Binary SVMs learn a linear decision function

$f(\cdot; \mathbf{w}, b) : \mathcal{X} \to \mathbb{R}$ defined by $f(\mathbf{x}; \mathbf{w}, b) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b$. Let $\mathcal{Y}$ now be a set of $m > 2$ classes. We now consider m output functions, one for each class, that quantify the confidence in the corresponding prediction. To do so we follow the modelling used in [Tsochantaridis *et al.*, 2004]. The key idea of this approach is to use a joint feature map between data $\mathcal{X}$ and labels $\mathcal{Y}$ denoted by $\Phi(\mathbf{x}, y)$. The output function for a class $y \in \mathcal{Y}$ can then be defined as

$$f_y(\mathbf{x}; \mathbf{w}, \mathbf{b}) = \langle \mathbf{w}, \Phi(\mathbf{x}, y) \rangle + b_y \ ,$$

with $\mathbf{b} = (b_1, \ldots, b_m)$. Thus, the predicted class $\hat{y}$ for a point $\mathbf{x}$ is chosen to maximize the confidence in the prediction, $\hat{y} = \arg\max_{y \in \mathcal{Y}} f_y(\mathbf{x}; \mathbf{w}, \mathbf{b})$. Training amounts to finding parameters $\mathbf{w}$ and $\mathbf{b}$ that lead, at least to a large extent, to correct predictions. That is, they should satisfy

$$f_{y_i}(\mathbf{x}_i; \mathbf{w}, \mathbf{b}) > f_u(\mathbf{x}_i; \mathbf{w}, \mathbf{b}) \quad \forall u \in \mathcal{Y} - \{y_i\} \ ,$$

for the training data points $(\mathbf{x}_i, y_i)$. However, for many real world learning problems these inequality systems are inconsistent. We use a loss function $\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$ to quantify the vexation about each inequality by applying it to the corresponding slack $s_{iu} = f_{y_i}(\mathbf{x}_i; \mathbf{w}, \mathbf{b}) - f_u(\mathbf{x}_i; \mathbf{w}, \mathbf{b})$. Given a convex monotonically decreasing loss ℓ, training can be implemented by the following optimization problem:

$$\min_{\mathbf{w}, \mathbf{b}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \max_{u \neq y_i} \{\ell\left(f_{y_i}(\mathbf{x}_i; \mathbf{w}, \mathbf{b}) - f_u(\mathbf{x}_i; \mathbf{w}, \mathbf{b})\right)\} \ , \tag{1.1}$$

where n is the number of examples in the training set, C is the regularization parameter, and we write $u \neq y_i$ short for $u \in \mathcal{Y} - \{y_i\}$. In addition to fitting the data by minimizing the loss, the objective function encourages learning "smooth" functions by minimizing some norm of $\mathbf{w}$ (regularization). By choosing the squared 2-norm, kernelization is enabled: optimization problem and solution can be expressed in terms of dot products which can be replaced by a kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$.

1.2.2 Structured Output Kernels and Multiclass

Although the above m-SVM (1.1) is more general, in the following we will restrict ourselves to the class of kernels that decompose into a kernel over $\mathcal{X}$ and a kernel over $\mathcal{Y}$. Let $k_{\mathcal{X}}$ be a kernel on $\mathcal{X}$, i.e. $k_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and let $k_{\mathcal{Y}}$ be a kernel on $\mathcal{Y}$, i.e. $k_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Then a joint kernel k can be defined by

$$k\left((\mathbf{x}, y), (\mathbf{x}', y')\right) = k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \cdot k_{\mathcal{Y}}(y, y') \ . \tag{1.2}$$

This structure is reflected in the feature space: the joint feature space will be the tensor product of the feature space on $\mathcal{X}$ with that on $\mathcal{Y}$. We can thus view both the feature representations $\Phi(\mathbf{x}, y)$ of data-label pairs and the hyperplane normal $\mathbf{w}$ as being $\dim(\mathcal{H}) \times |\mathcal{Y}|$ matrices.

When the kernel matrix on $\mathcal{Y}$ is diagonal, decomposable kernels offer significant computational advantages. In the matrix representation described above, $\Phi(\mathbf{x}, y)$ has a single non-zero column (indexed by y), which is equal to $\sqrt{k_{\mathcal{Y}}(y,y)}\Phi_{\mathcal{X}}(\mathbf{x})$ (where $\Phi_{\mathcal{X}}$ is the feature map on $\mathcal{X}$ implied by $k_{\mathcal{X}}$). In our experiments, we will use (1.2) with the matching kernel (or identity kernel) on $\mathcal{Y}$, i.e. $k_{\mathcal{Y}}(y, y') = \delta_{yy'}$. This kernel imposes the least structure on the classes.

Structured output learning [Taskar *et al.*, 2003,Tsochantaridis *et al.*, 2004] (over finite sets of strucures) is conceptually a multiclass problem: each possible output structure can be modelled as a different class. The number of classes is usually huge, e.g. exponential in the size of the size. This not only renders one-vs-rest computationally too expensive, but can also prevent generalization over classes. However, in the joint feature map framework efficient learning can still be possible for several examples of important output spaces [Tsochantaridis *et al.*, 2004]. Prior knowledge on correlations between related structures can be exploited for generalization by defining a corresponding $k_{\mathcal{Y}}$.

1.2.3 Representation via Kernel PCA

While $\mathcal{Y}$ is finite by definition, $\mathcal{H}$ may be infinite; we now describe how this can be handled. Let the kernel matrix $\mathbf{K}_{\mathcal{X}} \in \mathbb{R}^{n\times n}$ be defined by $(K_{\mathcal{X}})_{ij} = k_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j)$. We can now use kernel PCA [Schölkopf and Smola, 2002] to find a finite- (n-) dimensional representation of the data. The representer theorem (e.g. [Schölkopf and Smola, 2002]) ensures that the optimal solution $\mathbf{w}$ lies in a finite dimensional subspace. Following [Chapelle and Zien, 2005], we first find a basis for this subspace, and then represent all points (and $\mathbf{w}$) in terms of that basis and work with that representation. For simplicity we make the assumption that the span of the data points has full dimensionality.

The basis $\{\mathbf{u}_1, \dots, \mathbf{u}_n\}$ we use should satisfy two criteria. First, each basis vector $\mathbf{u}_i$ has to be expressed in terms of the feature maps of the data points $\Phi(\mathbf{x}_k)$, since these are the only points we know that lie in the relevant subspace. Formally we have $\mathbf{u}_i = \sum_{k=1}^n A_{ki}\Phi(\mathbf{x}_k)$ with a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ of coefficients. Second, the basis should be orthonormal (the normality is just for convenience). This amounts to $\delta_{ij} = \mathbf{u}_i^\top \mathbf{u}_j = \sum_{k=1}^n \sum_{l=1}^n A_{ki}A_{lj}\Phi(\mathbf{x}_k)^\top\Phi(\mathbf{x}_l)$ or, in compact matrix notation, $\mathbf{A}^\top\mathbf{K}\mathbf{A} = \mathbf{I}$. From this it follows that $(\mathbf{A}\mathbf{A}^\top)^{-1} = \mathbf{K}$. One way of finding such a matrix $\mathbf{A}$ is to eigendecompose $\mathbf{K}$ as $\mathbf{K} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ and use $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}^{-1/2}$. Another choice of $\mathbf{A}$ would be the inverse of the Cholesky decomposition. Once the basis is constructed, we have to represent the data points in this basis. Consider the feature space image of a data point $\Phi(\mathbf{x}_j)$, which might be a training point ($j \le n$) or a test point ($n < j \le \tilde{n}$). In either case it is mapped to its projections onto the basis vectors, $\mathcal{H} \to \mathbb{R}^n$, $\Phi(\mathbf{x}_j) \mapsto \tilde{\mathbf{x}}$, as follows:

$$(\tilde{\mathbf{x}}_j)_i = \mathbf{u}_i^\top\Phi(\mathbf{x}_j) = \sum_{k=1}^m A_{ki}k_{\mathcal{X}}(\mathbf{x}_k, \mathbf{x}_j) \ , \tag{1.3}$$

or $\tilde{\mathbf{X}} = \mathbf{A}^\top \tilde{\mathbf{K}}$. Here the columns of $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{\tilde{n}}) \in \mathbb{R}^{n \times \tilde{n}}$ are the new representations of the points $\mathbf{x}_j, j = 1, \ldots, \tilde{n}$, and $\tilde{\mathbf{K}} \in \mathbb{R}^{n \times \tilde{n}}$ is the extended kernel matrix with $K_{ij} = k_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j)$ with $i = 1, \ldots, n$ and $j = 1, \ldots, \tilde{n}$.

To verify that the new representation makes sense, observe that $\tilde{\mathbf{x}}_i^\top \tilde{\mathbf{x}}_j = k_{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j)$ for training points (ie, $1 \leq i, j \leq n$). Proof: $(\tilde{\mathbf{X}}_{1,\ldots,n})^\top (\tilde{\mathbf{X}}_{1,\ldots,n}) = \mathbf{K}^\top \mathbf{A} \mathbf{A}^\top \mathbf{K} = \mathbf{K}$. Note that the corresponding equality for the test points does not hold in general, since the test points do not necessarily lie in the span of the training points. This will, however, not thwart the SVM's predictions, since by the representer theorem $\mathbf{w}$ is in the span of the training points, and thus the part orthogonal to it is irrelevant for the classification anyway. In the rest of the paper, we will refer to $\tilde{\mathbf{x}} \in \mathbb{R}^d$ by just writing $\mathbf{x}$; thus we can restrict ourselves to the linear case without losing generality. Further, it is possible to use less than the n training points for constructing the basis, e.g. the first d points. This results in an approximation (since the part of $\mathbf{w}$ orthogonal to the corresponding d-dimensional subspace is lost), but saves computation time. Another possibility for approximation is by using the incomplete Cholesky factorization. We discuss these approximations further in the experiments section.

1.2.4 Primal Optimization via Conjugate Gradient

Since we are interested in gradient optimization, we replace the max over the classes by a softmax. For a vector $\mathbf{t} = (t_u)_{u=1\ldots m}$ we define

$$\mathrm{smax}_\gamma(\mathbf{t}) = \frac{1}{\gamma} \log \left((1-m) + \sum_{u=1}^{m} e^{\gamma t_u} \right) . \tag{1.4}$$

This version of the softmax, already proposed in [Chapelle and Zien, 2005], has the nice property that it interpolates between maximizing and summing (which might also be considered a useful way of aggregating losses over wrong classes): $\lim_{\gamma \to \infty} \mathrm{smax}_\gamma(\mathbf{t}) = \max_u t_u$, $\lim_{\gamma \to 0} \mathrm{smax}_\gamma(\mathbf{t}) = \sum_u t_u$.

Following [Chapelle, 2006], we train the m-SVM in the primal by a gradient descent technique. Thus we prefer the logistic loss, $\ell(s) = \log(1 + e^{-s})$, to the popular hinge loss, which is not differentiable.

Both the logistic loss and the softmax are continuously differentiable and thus amendable to gradient based optimization. The gradients are given by $\frac{\partial \ell(s)}{\partial s} = \frac{-e^{-s}}{1+e^{-s}}$ and $\frac{\partial \mathrm{smax}_\gamma(\mathbf{t})}{\partial t_v} = \frac{e^{\gamma t_v}}{(1-m)+\sum_{u=1}^m e^{\gamma t_u}}$. For numerical stability we must avoid exponentiating values $\gamma t_u \gg 1$. Thus we compute the softmax by using the equality $\mathrm{smax}_\gamma(\mathbf{t}) = a + \frac{1}{\gamma} \log\left((1-m)e^{-\gamma a} + \sum_{u=1}^m e^{\gamma(t_u - a)}\right)$ with $a = \max_u t_u$; thus all exponents are negative. Similarly, we compute the gradient by $e^{\gamma(t_v - a)} / \left((1-k)e^{-\gamma a} + \sum_{u=1}^m e^{\gamma(t_u - a)}\right)$ with the same a.

We calculate the gradient associated separately for each data point i. For convenience, we define vectors $\mathbf{s}$ and $\mathbf{t}$ (ommitting the subscript i to lighten notation) by $s_u = f_{y_i}(\mathbf{x}_i; \mathbf{w}, \mathbf{b}) - f_u(\mathbf{x}_i; \mathbf{w}, \mathbf{b})$ and $\mathbf{t} = \ell(\mathbf{s})$ where ℓ is extended to vectors by applying it componentwise; thus $\mathbf{s}$ is a function of $\mathbf{w}$ and $\mathbf{b}$, and $\mathbf{t}$ is a function of $\mathbf{s}$. By using the chain rule, the gradients with respect $\mathbf{w}$ and $\mathbf{b}$ can be written as

$$\frac{\partial}{\partial \mathbf{w}} \mathrm{smax}_\gamma \ell \left(\mathbf{w}^\top (\Phi(\mathbf{x}_i, y_i) - \Phi(\mathbf{x}_i, u)) + b_{y_i} - b_u \right) = \mathbf{c} \cdot \mathbf{A} \ , \tag{1.5}$$

$$\frac{\partial}{\partial \mathbf{b}} \mathrm{smax}_\gamma \ell \left(\mathbf{w}^\top (\Phi(\mathbf{x}_i, y_i) - \Phi(\mathbf{x}_i, u)) + b_{y_i} - b_u \right) = \mathbf{c} \cdot \mathbf{B} \ , \tag{1.6}$$

where $\mathbf{A} \in \mathbb{R}^{m \times (d \cdot m)}$, $\mathbf{B} \in \mathbb{R}^{m \times m}$, and $\mathbf{c} \in \mathbb{R}^{1 \times m}$ are defined by

$$A_{u\cdot} = (\Phi(\mathbf{x}_i, y_i) - \Phi(\mathbf{x}_i, u))^\top \qquad B_{uv} = \delta_{y_i v} - \delta_{uv}$$

$$c_v = \frac{\partial \mathrm{smax}_\gamma(\mathbf{t})}{\partial t_v} \frac{\partial \ell(s_v)}{\partial s_v} \ .$$

Using this gradient, training was performed by running the conjugate gradient code `minimize.m` by Carl Rasmussen.[1] The matlab implementation is available at `http://www.kyb.tuebingen.mpg.de/bs/people/zien/mSVM/`.

1.2.5 Comparison with previous multiclass approaches

We briefly compare and constrast our formulation, eq. (1.1), to other multiclass SVMs in the literature. Using the matching kernel on $\mathcal{Y}$, i.e. $k_\mathcal{Y}(y, y') = \delta_{y,y'}$, the hyperplane normal $\mathbf{w}$ in the joint feature map decomposes into a separate $\mathbf{w}_u \in \mathcal{X}$ for each class u. To enable comparison we plug in the commonly used hinge loss. Expressing it by inequalities yields:

$$\begin{array}{ll} \min\limits_{\mathbf{w}, \mathbf{b}, 0 \le \xi} & \frac{1}{2} \sum\limits_u \|\mathbf{w}_u\|^2 + C \sum\limits_{i=1}^n \mathrm{smax}_\gamma \left(\xi_{iu} \right)_{u \neq y_i} \\ \text{s.t.} & \forall i : \forall u \neq y_i : \ \langle \mathbf{w}_{y_i} - \mathbf{w}_u, \Phi(\mathbf{x}_i) \rangle + b_{y_i} - b_u \geqslant 1 - \xi_{iu} \end{array} \tag{1.7}$$

The first genuine multiclass SVM was proposed by [Vapnik, 1998] and [Weston and Watkins, 1999]. It is essentially the special case of (1.7) obtained by taking γ to 0, i.e. using the sum instead of the softmax.

$$\begin{array}{ll} \min\limits_{\mathbf{w}, \mathbf{b}, 0 \le \xi} & \frac{1}{2} \sum\limits_u \|\mathbf{w}_u\|^2 + C \sum\limits_{i=1}^n \sum\limits_{u \neq y_i} \xi_{iu} \\ \text{s.t.} & \forall i : \forall u \neq y_i : \ \langle \mathbf{w}_{y_i} - \mathbf{w}_u, \Phi(\mathbf{x}_i) \rangle + b_{y_i} - b_u \geqslant 2 - \xi_{iu} \end{array}$$

In the dual formulation, it turns out that the bias terms make it difficult to decouple the constraints, and hence make active set methods ineffective. However, as was proposed in [Hsu and Lin, 2002], this can be alleviated by adding a regularization of the bias, $\|\mathbf{b}\|^2$, to the objective.

Alternatively to penalizing all margin violations, one may penalize only the wrong class with maximal loss, thus replacing the sum $\sum_{u \neq y_i} \xi_{iu}$ by

[1] `http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/`

$\max_{u \neq y_i}\{\xi_{iu}\} =: \xi_i$. Thereby we obtain (up to the bias terms) the model of [Crammer and Singer, 2001]:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{b},0\leq\xi} \quad & \frac{1}{2}\sum_u \|\mathbf{w}_u\|^2 + C\sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \forall i : \forall u \neq y_i : \ \langle \mathbf{w}_{y_i} - \mathbf{w}_u, \varPhi(\mathbf{x}_i)\rangle + b_{y_i} - b_u \geqslant 1 - \xi_i . \end{aligned}$$

Our softmax can interpolate between these two modes of aggregation.

The historically first approaches to multiclass problems were to split them up into several binary classification problems. The two main methods are one-vs-rest and one-vs-one. Our m-SVM corresponds to a one-vs-rest setting, as there is a single hyperplane for each class, while there is one for each pair of classes in one-vs-one. However, the one-vs-rest setup in principle allows points to be labeled with any subset of classes, while in m-SVMs each point belongs to exactly one class.

1.3 Experiments

In the following, we evaluate our method on the five UCI datasets described in [Duan and Keerthi, 2005], using the same splits into training data and test data. For each dataset, three different sizes of training sets are fixed, called `small`, `medium`, and `large`. For each combination of dataset and size, 20 splits into training set and test set are provided.[2] We use the Gaussian radial basis function (RBF) kernel on the inputs and, as discussed above, the matching kernel on the classes.

1.3.1 Comparison of m-SVM to one-vs-rest

For each dataset and size, we perform three-fold cross-validation for model selection over σ and C (γ is set to 1.0). Here, σ is the width parameter of the Gaussian RBF kernel. Both parameters are chosen from a small grid, namely $\sigma \in \{1/2, 1, 2, 4, 8\}$ and $C \in \{1, 10, 10^2, 10^3, 10^4\}$ for C. These values are taken to be relative to default values, which are calculated from the matrix of pairwise distances (for σ) and the kernel function values (for C) as described in [Chapelle and Zien, 2005]. The results are shown in Table 1.1.

For the small training sets, we provide results by other approaches for comparison. First, we show the results achieved in [Duan and Keerthi, 2005] which is the best out of four investigated methods, which are consistently better than ours. We conjecture that the best method in [Duan and Keerthi, 2005] performs better due to probabilistic post-processing, which we do not consider here. Our results are still comparable to (even slightly better than) those achieved in [Duan and Keerthi, 2005] with a one-vs-rest method. To verify that this is not due to modifications to the optimization problem (e.g. the

[2] available at `http://guppy.mpe.nus.edu.sg/~mpessk/multiclass.shtml`

Set	m-SVM small	m-SVM medium	m-SVM large	one vs. rest small	PWC_PSVM small
ABE	1.94 ± 0.63	1.06 ± 0.38	0.61 ± 0.23	1.75 ± 0.65	1.16 ± 0.63
DNA	9.70 ± 0.57	7.85 ± 0.98	5.59 ± 0.47	9.71 ± 0.75	9.23 ± 1.73
SAT	11.12 ± 0.56	10.14 ± 0.37	9.77 ± 0.53	10.89 ± 0.94	10.27 ± 0.92
SEG	7.85 ± 1.44	5.25 ± 0.73	4.27 ± 0.57	8.88 ± 0.90	6.66 ± 2.24
WAV	15.80 ± 1.31	14.91 ± 1.08	14.09 ± 0.82	16.96 ± 0.86	13.20 ± 3.70

Table 1.1. Average test error ($\pm$ standard error) on five datasets. The second to fourth columns show the results for m-SVM. For the fifth column m-SVM was run in a one-vs-rest mode. The rightmost column shows the best method from Duan and Keerthi (2005), which uses a sigmoid to estimate posterior probabilities of binary SVMs and combines them via a pairwise coupling strategy.

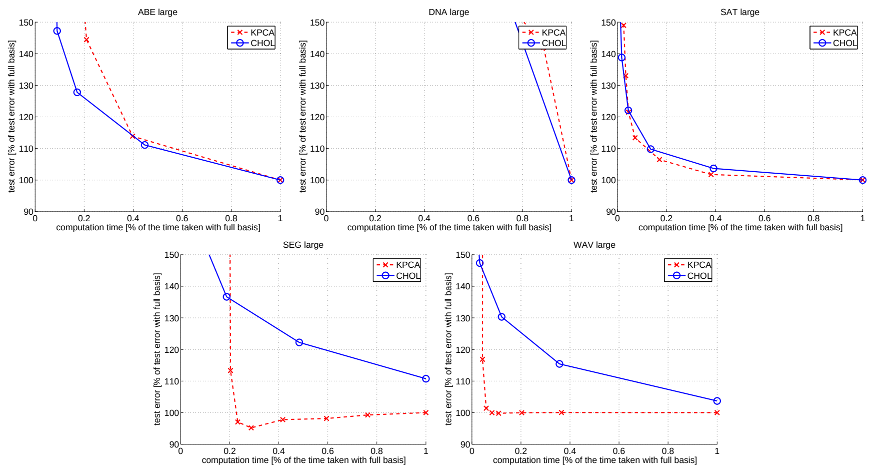

Fig. 1.1. Trade-off of training time versus prediction accuracy for the KPCA and Cholesky decomposition performed on all, half, 1/4, 1/8, ... (right to left) of the training points.

different loss function) we run one-vs-rest with exactly the same algorithm, just using it for a set of m binary problems.

The results shown above suggest that m-SVMs do not consistenly outperform the simple one-vs-rest heuristic. However, the real strength and purpose of true multiclass methods lies in problems with a priori known structure on a large output space. The computational complexity of such structured output learning still limits its applicability.

1.3.2 Speedup by approximation

In this section, we train SVMs using the low-rank versions of the decompositions described in Section 1.2.3. In Figure 1.1 we show how approximate optimization can be used to speed up training, and how this affects the predictive accuracy. Note that for some datasets (e.g. SAT), the accuracy of the approximate methods quickly decreases to the accuracy of the exact m-SVM. However, for other datasets (e.g. DNA), the accuracy deteriorates terribly upon approximation of the kernel, suggesting that class membership does not correlate with the predominant strucure in the data.

Interestingly, for WAV and SEG, KPCA and Cholesky decomposition perform very differently. The reason for this remains unclear. Since a low rank approximation can be considered to be some sort of regularization, it is interesting to observe that for SEG with KPCA approximation, the faster version actually is also more accurate.

1.4 Conclusion

We present a primal optimization method for m-SVMs. Based on a careful comparison to a one-vs-rest approach, keeping fixed everything else (e.g. the loss), we confirm that it achieves roughly equal accuracy. This seems to contradict the findings reported in [Crammer and Singer, 2001]; the reasons for this remain to be understood.

For structured output learning problems one-vs-rest cannot be used, thus research on m-SVM remains important. We show how the benefits of training in the primal [Chapelle, 2006], in particular high quality approximate solutions, can be earned for m-SVM training.

References

[Chapelle and Zien, 2005]O. Chapelle and A. Zien. Semi-Supervised Classification by Low Density Separation. In *AISTATS*, pages 57–64, 2005.

[Chapelle, 2006]O. Chapelle. Training a Support Vector Machine in the Primal. *Neural Computation*, 2006.

[Crammer and Singer, 2001]K. Crammer and Y. Singer. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[Duan and Keerthi, 2005]Kai-Bo Duan and S. Sathiya Keerthi. Which Is the Best Multiclass SVM Method? An Empirical Study. In *Multiple Classifier Systems*, pages 278–285, 2005.

[Hsu and Lin, 2002]C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

[Schölkopf and Smola, 2002]B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.

[Taskar *et al.*, 2003]B. Taskar, C. Guestrin, and D. Koller. Max-Margin Markov Networks. In *NIPS*, December 2003.

[Tsochantaridis *et al.*, 2004]I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *ICML*, 2004.

[Vapnik, 1998]V. N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 1998.

[Weston and Watkins, 1999]J. Weston and C. Watkins. Support Vector Machines for Multi-Class Pattern Recognition. In *ESANN*, 1999.