Journal of Machine Learning Research 2 (2001) 265-292 Submitted 03/01; Published 12/01

On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines

Koby Crammer KOBICS@CS.HUJI.AC.IL
Yoram Singer SINGER@CS.HUJI.AC.IL
School of Computer Science & Engineering
Hebrew University, Jerusalem 91904, Israel

Editors: Nello Cristianini, John Shawe-Taylor and Bob Williamson

Abstract

In this paper we describe the algorithmic implementation of multiclass kernel-based vector machines. Our starting point is a generalized notion of the margin to multiclass problems. Using this notion we cast multiclass categorization problems as a constrained optimization problem with a quadratic objective function. Unlike most of previous approaches which typically decompose a multiclass problem into multiple independent binary classification tasks, our notion of margin yields a direct method for training multiclass predictors. By using the dual of the optimization problem we are able to incorporate kernels with a compact set of constraints and decompose the dual problem into multiple optimization problems of reduced size. We describe an efficient fixed-point algorithm for solving the reduced optimization problems and prove its convergence. We then discuss technical details that yield significant running time improvements for large datasets. Finally, we describe various experiments with our approach comparing it to previously studied kernel-based methods. Our experiments indicate that for multiclass problems we attain state-of-the-art accuracy.

Keywords: Multiclass problems, SVM, Kernel Machines

1. Introduction

Supervised machine learning tasks often boil down to the problem of assigning labels to instances where the labels are drawn from a finite set of elements. This task is referred to as multiclass learning. Numerous specialized algorithms have been devised for multiclass problems by building upon classification learning algorithms for binary problems, i.e., problems in which the set of possible labels is of size two. Notable examples for multiclass learning algorithms are the multiclass extensions for decision tree learning (Breiman et al., 1984, Quinlan, 1993) and various specialized versions of boosting such as AdaBoost.M2 and AdaBoost.MH (Freund and Schapire, 1997, Schapire and Singer, 1999). However, the dominating approach for solving multiclass problems using support vector machines has been based on reducing a single multiclass problems into multiple binary problems. For instance, a common method is to build a set of binary classifiers where each classifier distinguishes between one of the labels to the rest. This approach is a special case of using output codes for solving multiclass problems (Dietterich and Bakiri, 1995). However, while multiclass learning using output codes provides a simple and powerful framework it cannot capture correlations between the different classes since it breaks a multiclass problem into multiple *independent* binary problems.

In this paper we develop and discuss in detail a direct approach for learning multiclass support vector machines (SVM). SVMs have gained an enormous popularity in statistics, learning theory, and engineering (see for instance Vapnik, 1998, Schölkopf et al., 1998, Cristianini and Shawe-Taylor, 2000, and the many references therein). With a few exceptions most support vector learning algorithms have been designed for binary (two class) problems. A few attempts have been made to generalize SVM to multiclass problems (Weston and Watkins, 1999, Vapnik, 1998). These attempts to extend the binary case are acheived by adding constraints for every class and thus the size of the quadratic optimization is proportional to the number categories in the classification problems. The result is often a homogeneous quadratic problem which is hard to solve and difficult to store.

The starting point of our approach is a simple generalization of separating hyperplanes and, analogously, a generalized notion of margins for multiclass problems. This notion of a margin has been employed in previous research (Allwein et al., 2000) but not in the context of SVM. Using the definition of a margin for multiclass problems we describe in Section 3 a *compact* quadratic optimization problem. We then discuss its dual problem and the form of the resulting multiclass predictor. In Section 4 we give a decomposition of the dual problem into multiple small optimization problems. This decomposition yields a memory and time efficient representation of multiclass problems. We proceed and describe an iterative solution for the set of the reduced optimization problems. We first discuss in Section 5 the means of choosing which reduced problem to solve on each round of the algorithm. We then discuss in Section 6 an efficient fixed-point algorithm for finding an approximate solution for the reduced problem that was chosen. We analyze the algorithm and derive a bound on its rate of convergence to the optimal solution. The baseline algorithm is based on a main loop which is composed of an example selection for optimization followed by an invocation of the fixed-point algorithm with the example that was chosen. This baseline algorithm can be used with small datasets but to make it practical for large ones, several technical improvements had to be sought. We therefore devote Section 7 to a description of the different technical improvements we have taken in order to make our approach applicable to large datasets. We also discuss the running time and accuracy results achieved in experiments that underscore the technical improvements. In addition, we report in Section 8 the results achieved in evaluation experiments, comparing them to previous work. Finally, we give conclusions in Section 9.

Related work Naturally, our work builds on previous research and advances in learning using support vector machines. The space is clearly too limited to mention all the relevant work, and thus we refer the reader to the books and collections mentioned above. As we have already mentioned, the idea of casting multiclass problems as a single constrained optimization with a quadratic objective function was proposed by Vapnik (1998), Weston and Watkins (1999), Bredensteiner and Bennet (1999), and Guermeur et. al (2000). However, the size of the resulting optimization problems devised in the above papers is typically large and complex. The idea of breaking a large constrained optimization problem into small problems, where each of which employs a subset of the constraints was first explored in the context of support vector machines by Boser *et al.* (1992). These ideas were further developed by several researchers (see Joachims, 1998 for an overview). However, the roots of this line of research go back to the seminal work of Lev Bregman (1967) which was further developed by Yair Censor and colleagues (see Censor and Zenios, 1997 for an excellent overview). These ideas distilled in Platt's method, called SMO, for sequential minimal optimization. SMO works with reduced problems that are derived from a pair of examples while our approach employs a single example for each reduced optimization problem. The result is a simple optimization problem which can be solved analytically in binary classification problems (see Platt, 1998) and leads to an efficient numerical algorithm (that is guaranteed to converge) in multiclass settings. Furthermore, although not explored in this paper, it deems possible that the single-example reduction can be used in parallel applications. Many of the technical improvements we discuss in this paper have been proposed in previous work. In particular ideas such as using a working set and caching have been described by Burges (1998), Platt (1998), Joachims (1998), and others. Finally, we would like to note that this work is part of a general line of research on multiclass learning we have been involved with. Allwein et al. (2000) described and analyzed a general approach for multiclass problems using error correcting output codes (Dietterich and Bakiri, 1995). Building on that work, we investigated the problem of designing good output codes for multiclass problems (Crammer and Singer, 2000). Although the model of learning using output codes differs from the framework studied in this paper, some of the techniques presented in this paper build upon results from an earlier paper (Crammer and Singer, 2000). Finally, some of the ideas presented in this paper can also be used to build multiclass predictors in *online* settings using the mistake bound model as the means of analysis. Our current research on multiclass problems concentrates on analogous online approaches (Crammer and Singer, 2001).

2. Preliminaries

Let $S = \{(\bar{x}_1, y_1), \ldots, (\bar{x}_m, y_m)\}$ be a set of m training examples. We assume that each example $\bar{x}_i$ is drawn from a domain $\mathcal{X} \subseteq \Re^n$ and that each label y_i is an integer from the set $\mathcal{Y} = \{1, \ldots, k\}$. A (multiclass) classifier is a function $H : \mathcal{X} \rightarrow \mathcal{Y}$ that maps an instance $\bar{x}$ to an element y of $\mathcal{Y}$. In this paper we focus on a framework that uses classifiers of the form

$$H_{\mathbf{M}}(\bar{x}) = \arg \max_{r=1}^{k} \{\bar{M}_r \cdot \bar{x}\} ,$$

where $\mathbf{M}$ is a matrix of size $k \times n$ over $\Re$ and $\bar{M}_r$ is the rth row of $\mathbf{M}$. We interchangeably call the value of the inner-product of the rth row of $\mathbf{M}$ with the instance $\bar{x}$ the *confidence* and the *similarity score* for the r class. Therefore, according to our definition above, the predicted label is the index of the row attaining the highest similarity score with $\bar{x}$. This setting is a generalization of linear binary classifiers. Using the notation introduced above, linear binary classifiers predict that the label of an instance $\bar{x}$ is 1 if $\bar{w} \cdot \bar{x} > 0$ and 2 otherwise ($\bar{w} \cdot \bar{x} \leq 0$). Such a classifier can be implemented using a matrix of size $2 \times n$ where $\bar{M}_1 = \bar{w}$ and $\bar{M}_2 = -\bar{w}$. Note, however, that this representation is less efficient as it occupies twice the memory needed. Our model becomes parsimonious when $k \geq 3$ in which we maintain k prototypes $\bar{M}_1, \bar{M}_2, \ldots, \bar{M}_k$ and set the label of a new input instance by choosing the index of the most similar row of $\mathbf{M}$.

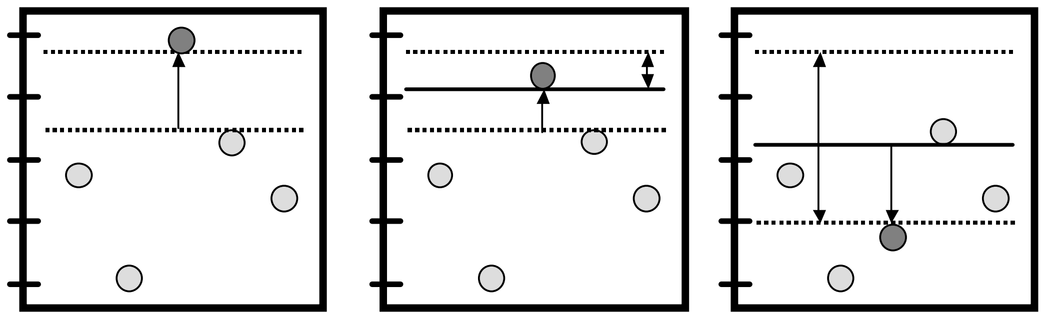

Figure 1: Illustration of the margin bound employed by the optimization problem.

Given a classifier $H_{\mathbf{M}}(\bar{x})$ (parametrized by a matrix $\mathbf{M}$) and an example $(\bar{x}, y)$, we say that $H_{\mathbf{M}}(\bar{x})$ misclassifies an example $\bar{x}$ if $H_{\mathbf{M}}(\bar{x}) \neq y$. Let $[\![\pi]\!]$ be 1 if the predicate π holds and 0 otherwise. Thus, the empirical error for a multiclass problem is given by

$$\epsilon_S(M) = \frac{1}{m} \sum_{i=1}^{m} [\![H_{\mathbf{M}}(x_i) \neq y_i]\!] \ . \tag{1}$$

Our goal is to find a matrix $\mathbf{M}$ that attains a small empirical error on the sample S and also generalizes well. Direct approaches that attempt to minimize the empirical error are computationally expensive (see for instance Höffgen and Simon, 1992, Crammer and Singer, 2000). Building on Vapnik's work on support vector machines (Vapnik, 1998), we describe in the next section our paradigm for finding a good matrix $\mathbf{M}$ by replacing the discrete empirical error minimization problem with a quadratic optimization problem. As we see later, recasting the problem as a minimization problem also enables us to replace inner-products of the form $\bar{a} \cdot \bar{b}$ with kernel-based inner-products of the form $K(\bar{a}, \bar{b}) = \bar{\phi}(\bar{a}) \cdot \bar{\phi}(\bar{b})$.

3. Constructing multiclass kernel-based predictors

To construct multiclass predictors we replace the misclassification error of an example, ($[\![H_{\mathbf{M}}(x) \neq y]\!]$), with the following piecewise linear bound,

$$\max_r \{\bar{M}_r \cdot \bar{x} + 1 - \delta_{y,r}\} - \bar{M}_y \cdot \bar{x} \ ,$$

where $\delta_{p,q}$ is equal 1 if $p = q$ and 0 otherwise. The above bound is zero if the confidence value for the correct label is larger by at least one than the confidences assigned to the rest of the labels. Otherwise, we suffer a loss which is linearly proportional to the difference between the confidence of the correct label and the maximum among the confidences of the other labels. A graphical illustration of the above is given in Figure 1. The circles in the figure denote different labels and the correct label is plotted in dark grey while the rest of the labels are plotted in light gray. The height of each label designates its confidence. Three settings are plotted in the figure. The left plot corresponds to the case when the margin is larger than one, and therefore the bound $\max_r \{\bar{M}_r \cdot \bar{x} + 1 - \delta_{y,r}\} - \bar{M}_y \cdot \bar{x}$ equals zero, and hence the example is correctly classified. The middle figure shows a case where the example is correctly classified but with a small margin and we suffer some loss. The right plot depicts the loss of a misclassified example.

Summing over all the examples in S we get an upper bound on the empirical loss,

$$\epsilon_S(M) \leq \frac{1}{m} \sum_{i=1}^{m} \left[\max_r \{ \bar{M}_r \cdot \bar{x}_i + 1 - \delta_{y_i,r} \} - \bar{M}_{y_i} \cdot \bar{x}_i \right] \ . \tag{2}$$

We say that a sample S is *linearly separable by a multiclass machine* if there exists a matrix $\mathbf{M}$ such that the above loss is equal to zero for all the examples in S, that is,

$$\forall i \quad \max_r \{ \bar{M}_r \cdot \bar{x}_i + 1 - \delta_{y_i,r} \} - \bar{M}_{y_i} \cdot \bar{x}_i = 0 \ . \tag{3}$$

Therefore, a matrix $\mathbf{M}$ that satisfies Eq. (3) would also satisfy the constraints,

$$\forall i, r \quad \bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,r} - \bar{M}_r \cdot \bar{x}_i \geq 1 \ . \tag{4}$$

Define the l_2-norm of a matrix $\mathbf{M}$ to be the l_2-norm of the vector represented by the concatenation of $\mathbf{M}$'s rows, $\|M\|_2^2 = \|(\bar{M}_1, \ldots, \bar{M}_k)\|_2^2 = \sum_{i,j} M_{i,j}^2$. Note that if the constraints given by Eq. (4) are satisfied, we can make the differences between $\bar{M}_{y_i} \cdot \bar{x}_i$ and $\bar{M}_r \cdot \bar{x}_i$ arbitrarily large. Furthermore, previous work on the generalization properties of large margin DAGs (Platt et al., 2000) for multiclass problems showed that the generalization properties depend on the l_2-norm of $\mathbf{M}$ (see also Crammer and Singer, 2000). We therefore would like to seek a matrix $\mathbf{M}$ of a small norm that satisfies Eq. (4). When the sample S is linearly separable by a multiclass machine, we seek a matrix $\mathbf{M}$ of the smallest norm that satisfies Eq. (4). The result is the following optimization problem,

$$\begin{aligned} &\min_{M} && \frac{1}{2}\|M\|_2^2 \\ &\text{subject to :} && \forall i, r \quad \bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,r} - \bar{M}_r \cdot \bar{x}_i \geq 1 \ . \end{aligned} \tag{5}$$

Note that m of the constraints for $r = y_i$ are automatically satisfied since,

$$\bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,y_i} - \bar{M}_{y_i} \cdot \bar{x}_i = 1 \ .$$

This property is an artifact of the separable case. In the general case the sample S might not be linearly separable by a multiclass machine. We therefore add slack variables $\xi_i \geq 0$ and modify Eq. (3) to be,

$$\forall i \quad \max_r \{ \bar{M}_r \cdot \bar{x}_i + 1 - \delta_{y_i,r} \} - \bar{M}_{y_i} \cdot \bar{x}_i = \xi_i \ . \tag{6}$$

We now replace the optimization problem defined by Eq. (5) with the following primal optimization problem,

$$\begin{aligned} &\min_{M,\xi} && \frac{1}{2}\beta\|M\|_2^2 + \sum_{i=1}^{m} \xi_i \\ &\text{subject to :} && \forall i, r \quad \bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,r} - \bar{M}_r \cdot \bar{x}_i \geq 1 - \xi_i \ . \end{aligned} \tag{7}$$

where $\beta > 0$ is a regularization constant and for $r = y_i$ the inequality constraints become $\xi_i \geq 0$. This is an optimization problem with "soft" constraints. We would like to note in passing that it is possible to cast an analogous optimization problem with "hard" constraints as in (Vapnik, 1998).

To solve the optimization problem we use the Karush-Kuhn-Tucker theorem (see for instance Vapnik, 1998, Cristianini and Shawe-Taylor, 2000). We add a dual set of variables, one for each constraint and get the Lagrangian of the optimization problem,

$$\begin{aligned} \mathcal{L}(M,\xi,\eta) &= \frac{1}{2}\beta\sum_r \|\bar{M}_r\|_2^2 + \sum_{i=1}^m \xi_i \\ &\quad + \sum_{i,r}\eta_{i,r}\left[\bar{M}_r\cdot\bar{x}_i - \bar{M}_{y_i}\cdot\bar{x}_i - \delta_{y_i,r} + 1 - \xi_i\right] \\ \text{subject to :} &\quad \forall i,r \quad \eta_{i,r}\ge 0 \ . \end{aligned} \tag{8}$$

We now seek a saddle point of the Lagrangian, which would be the minimum for the primal variables $\{M,\xi\}$ and the maximum for the dual variables η. To find the minimum over the primal variables we require,

$$\frac{\partial}{\partial \xi_i}\mathcal{L} = 1 - \sum_r \eta_{i,r} = 0 \qquad \Rightarrow \qquad \sum_r \eta_{i,r} = 1 \ . \tag{9}$$

Similarly, for $\bar{M}_r$ we require,

$$\begin{aligned} \frac{\partial}{\partial \bar{M}_r}\mathcal{L} &= \sum_i \eta_{i,r}\bar{x}_i - \sum_{i,y_i=r}\underbrace{\left(\sum_q \eta_{i,q}\right)}_{=1}\bar{x}_i + \beta\bar{M}_r \\ &= \sum_i \eta_{i,r}\bar{x}_i - \sum_i \delta_{y_i r}\bar{x}_i + \beta\bar{M}_r = 0 \ , \end{aligned}$$

which results in the following form

$$\bar{M}_r = \beta^{-1}\left[\sum_i (\delta_{y_i,r} - \eta_{i,r})\bar{x}_i\right] \ . \tag{10}$$

Eq. (10) implies that the solution of the optimization problem given by Eq. (5) is a matrix $\mathbf{M}$ whose rows are linear combinations of the instances $\bar{x}_1 \ldots \bar{x}_m$. Note that from Eq. (10) we get that the contribution of an instance $\bar{x}_i$ to $\bar{M}_r$ is $\delta_{y_i,r} - \eta_{i,r}$. We say that an example $\bar{x}_i$ is a *support pattern* if there is a row r for which this coefficient is not zero. For each row $\bar{M}_r$ of the matrix $\mathbf{M}$ we can partition the patterns with nonzero coefficients into two subsets by rewriting Eq. (10) as follows,

$$\bar{M}_r = \beta^{-1}\left[\sum_{i:y_i=r}(1-\eta_{i,r})\bar{x}_i + \sum_{i:y_i\neq r}(-\eta_{i,r})\bar{x}_i\right] \ .$$

The first sum is over all patterns that belong to the rth class. Hence, an example $\bar{x}_i$ labeled $y_i = r$ is a support pattern only if $\eta_{i,r} = \eta_{i,y_i} < 1$. The second sum is over the rest of the patterns whose labels are different from r. In this case, an example $\bar{x}_i$ is a support pattern only if $\eta_{i,r} > 0$. Put another way, since for each pattern $\bar{x}_i$ the set $\{\eta_{i,1}, \eta_{i,2}, \ldots, \eta_{i,k}\}$ satisfies the constraints $\eta_{i,1}, \ldots, \eta_{i,k} \geq 0$ and $\sum_r \eta_{i,r} = 1$, each set can be viewed as a probability distribution over the labels $\{1 \ldots k\}$. Under this probabilistic interpretation an example $\bar{x}_i$ is a support pattern if and only if its corresponding distribution is *not* concentrated on the correct label y_i. Therefore, the classifier is constructed using patterns whose labels are uncertain; the rest of the input patterns are ignored.

Next, we develop the Lagrangian using only the dual variables by substituting Eqs. (9) and (10) into Eq. (8). Since the derivation is rather technical we defer the complete derivation to App. A. We obtain the following objective function of the dual program,

$$\mathcal{Q}(\eta) \quad = \quad -\frac{1}{2}\beta^{-1} \sum_{i,j} (\bar{x}_i \cdot \bar{x}_j) \left[\sum_r (\delta_{y_i,r} - \eta_{i,r})(\delta_{y_j,r} - \eta_{j,r}) \right] - \sum_{i,r} \eta_{i,r} \delta_{y_i,r} \ .$$

Let $\bar{1}_i$ be the vector whose components are all zero except for the ith component which is equal to one, and let $\bar{1}$ be the vector whose components are all one. Using this notation we can rewrite the dual program in the following vector form,

$$\begin{array}{ll} \max\limits_{\eta} & \mathcal{Q}(\eta) = -\frac{1}{2}\beta^{-1} \sum\limits_{i,j} (\bar{x}_i \cdot \bar{x}_j) \left[(\bar{1}_{y_i} - \bar{\eta}_i) \cdot (\bar{1}_{y_j} - \bar{\eta}_j) \right] - \sum\limits_i \bar{\eta}_i \cdot \bar{1}_{y_i} \\ \text{subject to :} & \forall i \ : \ \bar{\eta}_i \geq 0 \quad \text{and} \quad \bar{\eta}_i \cdot \bar{1} = 1 \ . \end{array} \tag{11}$$

It is easy to verify that $\mathcal{Q}(\eta)$ is concave in η. Since the set of constraints is convex, there is a unique maximum value of $\mathcal{Q}(\eta)$. To simplify the problem we now perform the following change of variables. Let $\bar{\tau}_i = \bar{1}_{y_i} - \bar{\eta}_i$ be the difference between the point distribution $\bar{1}_{y_i}$ concentrating on the correct label and the distribution $\bar{\eta}_i$ obtained by the optimization problem. Then Eq. (10) that describes the form of $\mathbf{M}$ becomes,

$$\bar{M}_r = \beta^{-1} \sum_i \tau_{i,r} \bar{x}_i \ . \tag{12}$$

Since we search for the value of the variables which maximize the objective function $\mathcal{Q}$ (and not the optimum value of $\mathcal{Q}$ itself), we can omit any additive and positive multiplicative constants and write the dual problem given by Eq. (11) as,

$$\begin{array}{ll} \max\limits_{\tau} & \mathcal{Q}(\tau) = -\frac{1}{2} \sum\limits_{i,j} (\bar{x}_i \cdot \bar{x}_j)(\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \sum\limits_i \bar{\tau}_i \cdot \bar{1}_{y_i} \\ \text{subject to :} & \forall i \quad \bar{\tau}_i \leq \bar{1}_{y_i} \quad \text{and} \quad \bar{\tau}_i \cdot \bar{1} = 0 \ . \end{array} \tag{13}$$

Finally, we rewrite the classifier $H(\bar{x})$ in terms of the variable τ,

$$H(\bar{x}) \quad = \quad \arg\max_{r=1}^{k} \left\{ \bar{M}_r \cdot \bar{x} \right\} = \arg\max_{r=1}^{k} \left\{ \sum_i \tau_{i,r} (\bar{x}_i \cdot \bar{x}) \right\} \ . \tag{14}$$

As in Support Vector Machines (Cortes and Vapnik, 1995), the dual program and the resulting classifier depend *only* on inner products of the form $(\bar{x}_i \cdot \bar{x})$. Therefore, we can perform inner-product calculations in some high dimensional inner-product space $\mathcal{Z}$ by replacing the inner-products in Eq. (13) and in Eq. (14) with a kernel function $K(\cdot,\cdot)$ that satisfies Mercer's conditions (Vapnik, 1998). The general dual program using kernel functions is therefore,

$$\max_{\tau} \quad \mathcal{Q}(\tau) = -\frac{1}{2}\sum_{i,j} K\left(\bar{x}_i, \bar{x}_j\right)\ \left(\bar{\tau}_i \cdot \bar{\tau}_j\right) + \beta \sum_i \bar{\tau}_i \cdot \bar{1}_{y_i} \tag{15}$$
$$\text{subject to}: \quad \forall i \quad \bar{\tau}_i \leq \bar{1}_{y_i} \quad \text{and} \quad \bar{\tau}_i \cdot \bar{1} = 0 \ \ ,$$

and the classification rule $H(\bar{x})$ becomes,

$$H(\bar{x}) = \arg\max_{r=1}^{k}\left\{\sum_i \tau_{i,r} K\left(\bar{x}, \bar{x}_i\right)\right\} \ . \tag{16}$$

Therefore, constructing a multiclass predictor using kernel-based inner-products is as simple as using standard inner-products.

Note the classifier of Eq. (16) does not contain a bias parameter b_r for $r = 1 \ldots k$. Augmenting these terms will add m more equality constraints to the dual optimization problem, increasing the complexity of the optimization problem. However, one can always use inner-products of the form $K(\bar{a}, \bar{b}) + 1$ which is equivalent of using bias parameters, and adding $\frac{1}{2}\beta \sum_r b_r^2$ to the objective function.

Also note that in the special case of $k = 2$ Eq. (7) reduces to the primal program of SVM by setting $\bar{w} = \bar{M}_1 - \bar{M}_2$ and $C = \beta^{-1}$. As mentioned above, Weston and Watkins (1999) also developed a multiclass version for SVM. Their approach compared the confidence $\bar{M}_y \cdot \bar{x}$ of the correct label to the confidences of *all* the other labels $\bar{M}_r \cdot \bar{x}$ and therefore used $m(k-1)$ slack variables in the primal problem. In contrast, in our framework the confidence of the correct label is compared to the highest similarity-score among the rest of the labels and uses only m slack variables in the primal program. As we describe in the sequel our compact formalization leads to a memory and time efficient algorithm for the above optimization problem.

4. Decomposing the optimization problem

The dual quadratic program given by Eq. (15) can be solved using standard quadratic programming (QP) techniques. However, since it employs mk variables, converting the dual program given by Eq. (15) into a standard QP form yields a representation that employs a matrix of size $mk \times mk$, which leads to a very large scale problem in general. Clearly, storing a matrix of that size is intractable for large problems. We now introduce a simple, memory efficient algorithm for solving the quadratic optimization problem given by Eq. (15) by decomposing it into small problems.

The core idea of our algorithm is based on separating the constraints of Eq. (15) into m disjoint sets, $\{\bar{\tau}_i | \bar{\tau}_i \leq \bar{1}_{y_i},\, \bar{\tau}_i \cdot \bar{1} = 0\}_{i=1}^{m}$. The algorithm we propose works in rounds. On each round the algorithm chooses a pattern p and improves the value of the objective function by updating the variables $\bar{\tau}_p$ under the set of constraints, $\bar{\tau}_p \leq \bar{1}_{y_p}$ and $\bar{\tau}_p \cdot \bar{1} = 0$.

Let us fix an example index p and write the objective function only in terms of the variables $\bar{\tau}_p$. For brevity we use $K_{i,j}$ to denote $K\left(\bar{x}_i, \bar{x}_j\right)$. We now isolate the contribution

Input $\{(\bar{x}_1, y_1), \ldots, (\bar{x}_m, y_m)\}$.
Initialize $\bar{\tau}_1 = \bar{0}, \ldots, \bar{\tau}_m = \bar{0}$.
Loop:

1. Choose an example p.
2. Calculate the constants for the reduced problem:
   - $A_p = K(\bar{x}_p, \bar{x}_p)$
   - $\bar{B}_p = \sum_{i \neq p} K(\bar{x}_i, \bar{x}_p)\bar{\tau}_i - \beta \bar{1}_{y_p}$
3. Set $\bar{\tau}_p$ to be the solution of the reduced problem :

$$\begin{array}{cc} \min\limits_{\tau_p} & \mathcal{Q}(\bar{\tau}_p) = \frac{1}{2} A_p (\bar{\tau}_p \cdot \bar{\tau}_p) + \bar{B}_p \cdot \bar{\tau}_p \\ \text{subject to :} & \bar{\tau}_p \leq \bar{1}_{y_p} \quad \text{and} \quad \bar{\tau}_p \cdot \bar{1} = 0 \end{array}$$

Output : $H(\bar{x}) = \arg \max\limits_{r=1}^{k} \left\{ \sum_i \tau_{i,r} K(\bar{x}, \bar{x}_i) \right\}$.

Figure 2: Skeleton of the algorithm for learning multiclass support vector machine.

of $\bar{\tau}_p$ in $\mathcal{Q}$.

$$\begin{aligned} \mathcal{Q}_p(\bar{\tau}_p) &\stackrel{\text{def}}{=} -\frac{1}{2} \sum_{i,j} K_{i,j} (\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \sum_i \bar{\tau}_i \cdot \bar{1}_{y_i} \\ &= -\frac{1}{2} K_{p,p} (\bar{\tau}_p \cdot \bar{\tau}_p) - \sum_{i \neq p} K_{i,p} (\bar{\tau}_p \cdot \bar{\tau}_i) \\ &\quad -\frac{1}{2} \sum_{i \neq p, j \neq p} K_{i,j} (\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \bar{\tau}_p \cdot \bar{1}_{y_p} + \beta \sum_{i \neq p} \bar{\tau}_i \cdot \bar{1}_{y_i} \\ &= -\frac{1}{2} K_{p,p} (\bar{\tau}_p \cdot \bar{\tau}_p) - \bar{\tau}_p \cdot \left[-\beta \bar{1}_{y_p} + \sum_{i \neq p} K_{i,p} \bar{\tau}_i \right] \\ &\quad + \left[-\frac{1}{2} \sum_{i \neq p, j \neq p} K_{i,j} (\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \sum_{i \neq p} \bar{\tau}_i \cdot \bar{1}_{y_i} \right] . \end{aligned} \tag{17}$$

Let us now define the following variables,

$$A_p = K_{p,p} > 0 \tag{18}$$

$$\bar{B}_p = -\beta \bar{1}_{y_p} + \sum_{i \neq p} K_{i,p} \bar{\tau}_i \tag{19}$$

$$C_p = -\frac{1}{2} \sum_{i,j \neq p} K_{i,j} (\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \sum_{i \neq p} \bar{\tau}_i \cdot \bar{1}_{y_i} \quad .$$

Using the variables defined above the objective function becomes,

$$\mathcal{Q}_p(\bar{\tau}_p) = -\frac{1}{2} A_p (\bar{\tau}_p \cdot \bar{\tau}_p) - \bar{B}_p \cdot \bar{\tau}_p + C_p \ .$$

For brevity, let us now omit all constants that do not affect the solution. Each reduced optimization problem has k variables and $k+1$ constraints,

$$\begin{array}{cl} \min\limits_{\tau} & \mathcal{Q}(\bar{\tau}) = \frac{1}{2} A_p (\bar{\tau}_p \cdot \bar{\tau}_p) + \bar{B}_p \cdot \bar{\tau}_p \\ \text{subject to :} & \bar{\tau}_p \leq \bar{1}_{y_p} \text{ and } \bar{\tau}_p \cdot \bar{1} = 0 \ . \end{array} \tag{20}$$

The skeleton of the algorithm is given in Figure 2. The algorithm is initialized with $\bar{\tau}_i = \bar{0}$ for $i = 1 \ldots m$ which, as we discuss later, leads to a simple initialization of internal variables the algorithm employs for efficient implementation. To complete the details of the algorithm we need to discuss the following issues. First, we need a stopping criterion for the loop. A simple method is to run the algorithm for a fixed number of rounds. A better approach which we discuss in the sequel is to continue iterating as long as the algorithm does not meet a predefined accuracy condition. Second, we need a scheme for choosing the pattern p on each round which then induces the reduced optimization problem given in Eq. (20). Two commonly used methods are to scan the patterns sequentially or to choose a pattern uniformly at random. In this paper we describe a scheme for choosing an example p in a greedy manner. This scheme appears to perform better empirically than other naive schemes. We address these two issues in Section 5.

The third issue we need to address is how to solve efficiently the reduced problem given by Eq. (20). Since this problem constitutes the core and the inner-loop of the algorithm we develop an efficient method for solving the reduced quadratic optimization problem. This method is more efficient than using the standard QP techniques, especially when it suffices to find an approximation to the optimal solution. Our specialized solution enables us to solve problems with a large number of classes k when a straightforward approach could not be applicable. This method is described in Section 6.

5. Example selection for optimization

To remind the reader, we need to solve Eq. (15),

$$\begin{array}{cl} \min\limits_{\tau} & \mathcal{Q}(\tau) = \frac{1}{2} \sum\limits_{i,j} K_{i,j} \ (\bar{\tau}_i \cdot \bar{\tau}_j) - \beta \sum\limits_i \bar{\tau}_i \cdot \bar{1}_{y_i} \\ \text{subject to :} & \forall i \quad \bar{\tau}_i \leq \bar{1}_{y_i} \quad \text{and} \quad \bar{\tau}_i \cdot \bar{1} = 0 \ , \end{array}$$

where as before $K_{i,j} = K(\bar{x}_i, \bar{x}_j)$. We use the Karush-Kuhn-Tucker theorem (see Cristianini and Shawe-Taylor, 2000) to find the necessary conditions for a point τ to be an optimum of Eq. (15). The Lagrangian of the problem is,

$$\begin{aligned} \mathcal{L}(\tau, u, v) \ &= \ \frac{1}{2} \sum_{i,j} K_{i,j} \sum_r \tau_{i,r} \tau_{j,r} - \beta \sum_{i,r} \tau_{i,r} \delta_{y_i,r} \qquad (21) \\ &\quad + \sum_{i,r} u_{i,r} (\tau_{i,r} - \delta_{y_i,r}) - \sum_i v_i \sum_r \tau_{i,r} \\ \text{subject to :} & \qquad \forall i, r \quad u_{i,r} \geq 0 \ . \end{aligned}$$

The first condition is,

$$\frac{\partial}{\partial \tau_{i,r}} \mathcal{L} \ = \ \sum_j K_{i,j} \tau_{j,r} - \beta \delta_{y_i,r} + u_{i,r} - v_i = 0 \ . \tag{22}$$

Let us now define the following auxiliary set of variables,

$$F_{i,r} \;\; = \;\; \sum_j K_{i,j}\tau_{j,r} - \beta\delta_{y_i,r} \ . \tag{23}$$

For each instance $\bar{x}_i$, the value of $F_{i,r}$ designates the confidence in assigning the label r to $\bar{x}_i$. A value of β is subtracted from the correct label confidence in order to obtain a margin of at least β. Note that from Eq. (19) we get,

$$F_{p,r} = B_{p,r} + k_{p,p}\tau_{p,r} \ . \tag{24}$$

We will make use of this relation between the variables F and B in the next section in which we discuss an efficient solution to the quadratic problem.

Taking the derivative with respect to the dual variables of the Lagrangian given by Eq. (21) and using the definition of $F_{i,r}$ from Eq. (23) and KKT conditions we get the following set of equality constraints on a feasible solution for the quadratic optimization problem,

$$\forall i, r \qquad F_{i,r} + u_{i,r} = v_i \ \ , \tag{25}$$

$$\forall i, r \qquad u_{i,r}(\tau_{i,r} - \delta_{y_i,r}) = 0 \ \ , \tag{26}$$

$$\forall i, r \qquad u_{i,r} \geq 0 \ \ . \tag{27}$$

We now further simplify the equations above. We do so by considering two cases. The first case is when $\tau_{i,r} = \delta_{y_i,r}$. In this case Eq. (26) holds automatically. By combining Eq. (27) and Eq. (25) we get that,

$$F_{i,r} \leq v_i \ \ . \tag{28}$$

In the second case $\tau_{i,r} < \delta_{y_i,r}$. In order for Eq. (26) to hold we must have $u_{i,r} = 0$. Thus, using Eq. (25) we get that,

$$F_{i,r} = v_i \ .$$

We now replace the single equality constraint with the following two inequalities,

$$F_{i,r} \geq v_i \ \text{ and } \ F_{i,r} \leq v_i \ \ . \tag{29}$$

To remind the reader, the constraints on $\bar{\tau}$ from the optimization problem given by Eq. (15) imply that for all i, $\bar{\tau}_i \leq \bar{1}_{y_i}$ and $\bar{\tau}_i \cdot \bar{1} = 0$. Therefore, if these constraints are satisfied there must exist at least one label r for which $\tau_{i,r} < \delta_{y_i,r}$. We thus get that $v_i = \max_r F_{i,r}$. Note also that if $\bar{\tau}_i = 0$ then $F_{i,y_i} = v_i = \max_r F_{i,r}$ and F_{i,y_i} is the unique maximum. We now combine the set of constraints from Eqs. (28) and (29) into a single inequality,

$$\max_r F_{i,r} \leq v_i \leq \min_{r \,:\, \tau_{i,r} < \delta_{y_i,r}} F_{i,r} \ \ . \tag{30}$$

Finally, dropping v_i we obtain,

$$\max_r F_{i,r} \leq \min_{r \,:\, \tau_{i,r} < \delta_{y_i,r}} F_{i,r} \ . \tag{31}$$

We now define,

$$\psi_i = \max_r F_{i,r} - \min_{r \,:\, \tau_{i,r} < \delta_{y_i,r}} F_{i,r} \ \ . \tag{32}$$

Since $\max_r F_{i,r} \geq \min_{r\,:\,\tau_{i,r}<\delta_{y_i,r}} F_{i,r}$ then the necessary and sufficient condition for a feasible vector $\bar{\tau}_i$ to be an optimum for Eq. (15) is that, $\psi_i = 0$. In the actual numerical implementation it is sufficient to find $\bar{\tau}_i$ such that $\psi_i \leq \epsilon$ where ϵ is a predefined accuracy parameter. We therefore keep performing the main loop of Figure 2 so long as there are examples $(\bar{x}_i, y_i)$ whose values ψ_i are greater than ϵ.

The variables ψ_i also serve as our means for choosing an example for an update. In our implementation we try to keep the memory requirements as small as possible and thus manipulate a single example on each loop. We choose the example index p for which ψ_p is maximal. We then find the vector $\bar{\tau}_p$ which is the (approximate) solution of the reduced optimization problem given by Eq. (15). Due to the change in $\bar{\tau}_p$ we need to update $F_{i,r}$ and ψ_i for all i and r. The pseudo-code describing this process is deferred to the next section in which we describe a simple and efficient algorithm for finding an approximate solution for the optimization problem of Eq. (15). Lin (2001) showed that this scheme does converge to the solution in a finite number of steps. Finally, we would like to note that some of the underlying ideas described in this section have been also explored by Keerthi and Gilbert (2000).

6. Solving the reduced optimization problem

The core of our algorithm relies on an efficient method for solving the reduced optimization given by Eq. (15) or the equivalent problem as defined by Eq. (20). In this section we describe an efficient fixed-point algorithm that finds an approximate solution to Eq. (20). We would like to note that an exact solution can also be derived. In (Crammer and Singer, 2000) we described a closely related algorithm for solving a similar quadratic optimization problem in the context of output coding. A simple modification of the algorithm can be used here. However, the algorithm needs to sort k values on each iteration and thus might be slow when k is large. Furthermore, as we discuss in the next section, we found empirically that the quality of the solution is quite insensitive to how well we fulfill the Karush-Kuhn-Tucker condition by bounding ψ_i. Therefore, it is enough to find a vector $\bar{\tau}_p$ that decreases significantly the value of $\mathcal{Q}(\bar{\tau})$ but is not necessarily the optimal solution.

We start by rewriting $\mathcal{Q}(\bar{\tau})$ from Eq. (20) using a completion to quadratic form and dropping the pattern index p,

$$\begin{aligned} \mathcal{Q}(\bar{\tau}) &= -\frac{1}{2}A(\bar{\tau}\cdot\bar{\tau}) - \bar{B}\cdot\bar{\tau} \\ &= -\frac{1}{2}A[(\bar{\tau}+\frac{\bar{B}}{A})\cdot(\bar{\tau}+\frac{\bar{B}}{A})] + \frac{\bar{B}\cdot\bar{B}}{2A}\ . \end{aligned}$$

We now perform the following change of variables,

$$\bar{\nu} = \bar{\tau} + \frac{\bar{B}}{A} \qquad \bar{D} = \frac{\bar{B}}{A} + \bar{1}_y\ . \tag{33}$$

At this point, we omit additive constants and the multiplicative factor A since they do not affect the value of the optimal solution. Using the above variable the optimization problem from Eq. (20) now becomes,

$$\begin{aligned} \min_{\nu} \qquad & \mathcal{Q}(\bar{\nu}) = \|\bar{\nu}\|^2 \\ \text{subject to :} \quad & \bar{\nu} \leq \bar{D} \text{ and } \bar{\nu}\cdot\bar{1} = \bar{D}\cdot\bar{1} - 1\ . \end{aligned} \tag{34}$$

We would like to note that since

$$F_{i,r} = B_{i,r} + A_i \tau_{i,r} \quad ,$$

we can compute ψ_i from $\bar{B}_i$ and thus need to store either $B_{i,r}$ or $F_{i,r}$.

Let us denote by θ and α_i the variables of the dual problem of Eq. (34). Then, the Karush-Kuhn-Tucker conditions imply that,

$$\forall r \quad \nu_r \le D_r \;\; ; \;\; \alpha_r(\nu_r - D_r) = 0 \;\; ; \;\; \nu_r + \alpha_r - \theta = 0 \quad . \tag{35}$$

Note that since $\alpha_r \ge 0$ the above conditions imply that $\nu_r \le \theta$ for all r. Combining this inequality with the constraint that $\nu_r \le D_r$ we get that the solution satisfies

$$\nu_r \le \min\{\theta, D_r\} \quad . \tag{36}$$

If $\alpha_r = 0$ we get that $\nu_r = \theta$ and if $\alpha_r > 0$ we must have that $\nu_r = \mathcal{D}_r$. Thus, Eq. (36) holds with equality, namely, the solution is of the form, $\nu_r = \min\{\theta, D_r\}$. Now, since $\bar{\nu} \cdot \bar{1} = \bar{D} \cdot \bar{1} - 1$ we get that θ satisfies the following constraint,

$$\sum_{r=1}^{k} \min_r\{\theta, D_r\} = \sum_{r=1}^{k} D_r - 1 \quad . \tag{37}$$

The above equation uniquely defines θ since the sum $\sum_{r=1}^{k} \min_r\{\theta, D_r\}$ is a strictly monotone and continuous function in θ. For $\theta = \max_r D_r$ we have that $\sum_{r=1}^{k} \min_r\{\theta, D_r\} > \sum_{r=1}^{k} D_r - 1$ while $\sum_{r=1}^{k} \min_r\{\theta, D_r\} \to -\infty$ as $\theta \to -\infty$. Therefore, there always exists a unique value $\theta^\star$ that satisfies Eq. (37). The following theorem shows that $\theta^\star$ is indeed the optimal solution of the quadratic optimization problem.

Theorem 1 *Let $\nu_r^* = \min\{\theta^\star, D_r\}$ where $\theta^\star$ is the solution of $\sum_{r=1}^{k} \min_r\{\theta, D_r\} = \sum_{r=1}^{k} D_r - 1$. Then, for every point $\bar{\nu}$ we have that $\|\bar{\nu}\|^2 > \|\bar{\nu}^*\|^2$.*

Proof Assume by contradiction that there is another feasible point $\bar{\nu} = \bar{\nu}^* + \bar{\Delta}$ which minimizes the objective function. Since $\bar{\nu} \ne \bar{\nu}^*$ we know that $\bar{\Delta} \ne 0$. Both $\bar{\nu}$ and $\bar{\nu}^*$ satisfy the equality constraint of Eq. (34), thus $\sum_r \Delta_r = 0$. Also, both points satisfy the inequality constraint of Eq. (34) thus $\Delta_r \le 0$ when $\nu_r^* = D_r$. Combining the last two equations with the assumption that $\bar{\Delta} \ne 0$ we get that $\Delta_s > 0$ for some s with $\nu_s^* = \theta$. Using again the equality $\sum_r \Delta_r = 0$ we have that there exists an index u with $\Delta_u < 0$. Let us denote by $\epsilon = \min\{|\Delta_s|, |\Delta_u|\}$. We now define a new feasible point $\bar{\nu}'$ as follows. Let $\nu_s' = \nu_s - \epsilon$, $\nu_u' = \nu_u + \epsilon$, and $\nu_r' = \nu_r$ otherwise. We now show that the norm of $\bar{\nu}'$ is smaller than the norm of $\bar{\nu}$. Since $\bar{\nu}$ and $\bar{\nu}'$ differ only in their s and u coordinates, we have that,

$$\|\bar{\nu}'\|^2 - \|\bar{\nu}\|^2 = (\nu_s')^2 + (\nu_u')^2 - (\nu_s)^2 - (\nu_u)^2 \quad .$$

Writing the values of $\bar{\nu}'$ in terms of $\bar{\nu}$ and ϵ we get,

$$\|\bar{\nu}'\|^2 - \|\bar{\nu}\|^2 = 2\epsilon\,(\epsilon - \nu_s + \nu_u) \quad .$$

From our construction of $\bar{\nu}'$ we have that either $\nu_u - \epsilon = \theta > \nu_s$ or $\nu_u - \epsilon > \theta \geq \nu_s$ and therefore we get $\nu_u - \epsilon > \nu_s$. This implies that

$$\|\bar{\nu}'\|^2 - \|\bar{\nu}\|^2 < 0 \ \ ,$$

which is clearly a contradiction. ■

We now use the above characterization of the solution to derive a simple fixed-point algorithm that finds $\theta^\star$. We use the simple identity $\min\{\theta, D_r\} + \max\{\theta, D_r\} = \theta + D_r$ and replace the minimum function with the above sum in Eq. (37) and get,

$$\sum_{r=1}^{k} [\theta + D_r - \max\{\theta, D_r\}] = \sum_{r=1}^{k} D_r - 1 \ ,$$

which amounts to,

$$\theta^* = \frac{1}{k}\left[\sum_{r=1}^{k} \max\{\theta^*, D_r\}\right] - \frac{1}{k} \ . \tag{38}$$

Let us define,

$$F(\theta) = \frac{1}{k}\left[\sum_{r=1}^{k} \max\{\theta, D_r\}\right] - \frac{1}{k} \ . \tag{39}$$

Then, the optimal value θ^* satisfies

$$\theta^* = F(\theta^*) \ . \tag{40}$$

Eq. (40) can be used for the following iterative algorithm. The algorithm starts with an initial value of θ and then computes the next value using Eq. (39). It continues iterating by substituting each new value of θ in $F(\cdot)$ and producing a series of values for θ. The algorithm halts when a required accuracy is met, that is, when two successive values of θ are close enough. A pseudo-code of the algorithm is given in Figure 3. The input to the algorithm is the vector $\bar{D}$, an initial suggestion for θ, and a required accuracy ϵ. We next show that if $\theta_1 \leq \max_r D_r$ then the algorithm does converge to the correct value of θ^*.

Theorem 2 *Let θ^* be the fixed point of Eq. (40) ($\theta^* = F(\theta^*)$). Assume that $\theta_1 \leq \max_r D_r$ and let $\theta_{l+1} = F(\theta_l)$. Then for $l \geq 1$*

$$\frac{|\theta_{l+1} - \theta^*|}{|\theta_l - \theta^*|} \leq 1 - \frac{1}{k} \ ,$$

where k is the number of classes.

Proof Assume without loss of generality that $\max_r D_r = D_1 \geq D_2 \geq \ldots \geq D_k \geq D_{k+1} \stackrel{\text{def}}{=} -\infty$. Also assume that $\theta^* \in (D_{s+1}, D_s)$ and $\theta_l \in (D_{u+1}, D_u)$ where $u, s \in \{1, 2, \ldots, k\}$.

`FixedPointAlgorithm(`$\bar{D}, \theta, \epsilon$`)`

Input $\bar{D}$, θ_1, ϵ.
Initialize $l = 0$.
Repeat

- $l \leftarrow l + 1$.
- $\theta_{l+1} \leftarrow \frac{1}{k}\left[\sum_{r=1}^{k} \max\{\theta_l, D_r\}\right] - \frac{1}{k}$.

Until $\left|\dfrac{\theta_l - \theta_{l+1}}{\theta_l}\right| \le \epsilon$.
Assign for $r = 1, \ldots, k$: $\nu_r = \min\{\theta_{l+1}, D_r\}$
Return: $\bar{\tau} = \bar{\nu} - \frac{\bar{B}}{A}$.

Figure 3: The fixed-point algorithm for solving the reduced quadratic program.

Thus,

$$\begin{aligned}
\theta_{l+1} &= F(\theta_l) \\
&= \frac{1}{k}\left[\sum_{r=1}^{k} \max\{\theta_l, D_r\}\right] - \frac{1}{k} \\
&= \frac{1}{k}\left(\sum_{r=u+1}^{k} \theta_l\right) + \frac{1}{k}\left(\sum_{r=1}^{u} D_r\right) - \frac{1}{k} \\
&= \left(1 - \frac{u}{k}\right)\theta_l + \frac{1}{k}\left(\sum_{r=1}^{u} D_r - 1\right) \ . \qquad (41)
\end{aligned}$$

Note that if $\theta_l \le \max_r D_r$ then $\theta_{l+1} \le \max_r D_r$. Similarly,

$$\begin{aligned}
\theta^* = F(\theta^*) = \left(1 - \frac{s}{k}\right)\theta^* + \frac{1}{k}\left(\sum_{r=1}^{s} D_r - 1\right) \\
\Rightarrow \theta^* = \frac{1}{s}\left(\sum_{r=1}^{s} D_r - 1\right) \ .
\end{aligned} \qquad (42)$$

We now need to consider three cases depending on the relative order of s and u. The first case is when $u = s$. In this case we get that,

$$\begin{aligned}
\frac{|\theta_{l+1} - \theta^*|}{|\theta_l - \theta^*|} &= \frac{\left|\left(1 - \frac{s}{k}\right)\theta_l + \frac{1}{k}\left(\sum_{r=1}^{s} D_r - 1\right) - \theta^*\right|}{|\theta_l - \theta^*|} \\
&= \frac{\left|\left(1 - \frac{s}{k}\right)\theta_l + \frac{s}{k}\theta^* - \theta^*\right|}{|\theta_l - \theta^*|} \\
&= 1 - \frac{s}{k} \le 1 - \frac{1}{k} \ .
\end{aligned}$$

where the second equality follows from Eq. (42). The second case is where $u > s$. In this case we get that for all $r = s+1, \ldots, u$:

$$\theta_l \le D_r \le \theta^* \ . \qquad (43)$$

Using Eq. (41) and Eq. (42) we get,

$$\begin{aligned}
\theta_{l+1} &= \left(1-\frac{u}{k}\right)\theta_l + \frac{1}{k}\left(\sum_{r=1}^{u} D_r - 1\right) \\
&= \left(1-\frac{u}{k}\right)\theta_l + \frac{s}{k}\frac{1}{s}\left(\sum_{r=1}^{s} D_r - 1\right) + \frac{1}{k}\left(\sum_{r=s+1}^{u} D_r\right) \\
&= \left(1-\frac{u}{k}\right)\theta_l + \frac{s}{k}\theta^* + \frac{1}{k}\left(\sum_{r=s+1}^{u} D_r\right) \ .
\end{aligned}$$

Applying Eq. (43) we obtain,

$$\begin{aligned}
\theta_{l+1} &\leq \left(1-\frac{u}{k}\right)\theta_l + \frac{s}{k}\theta^* + \frac{1}{k}(u-s)\theta^* \\
&= \left(1-\frac{u}{k}\right)\theta_l + \frac{u}{k}\theta^* \ .
\end{aligned}$$

Since θ_{l+1} is bounded by a convex combination of θ_l and θ^*, and θ^* is larger than θ_l, then $\theta^* \geq \theta_{l+1}$. We therefore finally get that,

$$\begin{aligned}
\frac{|\theta_{l+1}-\theta^*|}{|\theta_l-\theta^*|} &= \frac{\theta^*-\theta_{l+1}}{\theta^*-\theta_l} \\
&\leq \frac{\theta^* - \left(1-\frac{u}{k}\right)\theta_l - \frac{u}{k}\theta^*}{\theta^*-\theta_l} \\
&= 1-\frac{u}{k} \leq 1-\frac{1}{k} \ .
\end{aligned}$$

The last case, where $u < s$, is derived analogously to the second case, interchanging the roles of u and s. ■

From the proof we see that the best convergence rate is obtained for large values of u. Thus, a good feasible initialization for θ_1 can be $\min_r D_r$. In this case

$$\theta_2 = F(\theta_1) = \frac{1}{k}\left(\sum_{r=1}^{k} D_r\right) - \frac{1}{k} \ .$$

This gives a simple initialization of the algorithm which ensures that the initial rate of convergence will be fast.

We are now ready to describe the complete implementation of the algorithm for learning multiclass kernel machine. The algorithm gets a required accuracy parameter and the value of β. It is initialized with $\bar{\tau}_i = 0$ for all indices $1 \leq i \leq m$. This value yields a simple initialization of the variables $F_{i,r}$. On each iteration we compute from $F_{i,r}$ the value ψ_i for each example and choose the example index p for which ψ_p is the largest. We then call the fixed-point algorithm which in turn finds an approximate solution to the reduced quadratic optimization problem for the example indexed p. The fixed-point algorithm returns a set of new values for $\bar{\tau}_p$ which triggers the update of $F_{i,r}$. This process is repeated until the value ψ_i is smaller than ϵ for all $1 \leq i \leq m$. The pseudo-code of the algorithm is given in Figure 4.

Input $\{(\bar{x}_1, y_1), \ldots, (\bar{x}_m, y_m)\}$.
Initialize for $i = 1, \ldots, m$:

- $\bar{\tau}_i = \bar{0}$
- $F_{i,r} = -\beta \delta_{r,y_i}$ $(r = 1 \ldots k)$
- $A_i = K(\bar{x}_i, \bar{x}_i)$

Repeat:

- Calculate for $i = 1 \ldots m$: $\psi_i = \max_r F_{i,r} - \min_{r \,:\, \tau_{i,r} < \delta_{y_i,r}} F_{i,r}$
- Set: $p = \arg\max\{\psi_i\}$
- Set for $r = 1 \ldots k$: $D_r = \frac{F_{p,r}}{A_p} - \tau_{p,r} + \delta_{r,y_p}$ and $\theta = \frac{1}{k}\left(\sum_{r=1}^{k} D_r\right) - \frac{1}{k}$
- Call: $\bar{\tau}_p^\star$ =`FixedPointAlgorithm(`$\bar{D}, \theta, \epsilon/2$`)`. (See Figure 3)
- Set: $\Delta\bar{\tau}_p = \bar{\tau}_p^\star - \bar{\tau}_p$
- Update for $i = 1 \ldots m$ and $r = 1 \ldots k$: $F_{i,r} \leftarrow F_{i,r} + \Delta\tau_{p,r} K(\bar{x}_p, \bar{x}_i)$
- Update: $\bar{\tau}_p \leftarrow \bar{\tau}_p^\star$

Until $\psi_p < \epsilon\beta$

Output : $H(\bar{x}) = \arg\max_r \left\{ \sum_i \tau_{i,r} K(\bar{x}, \bar{x}_i) \right\}$.

Figure 4: Basic algorithm for learning a multiclass, kernel-based, support vector machine using KKT conditions for example selection.

7. Implementation details

We have discussed so far the underlying principal and algorithmic issues that arise in the design of multiclass kernel-based vector machines. However, to make the learning algorithm practical for large datasets we had to make several technical improvements to the baseline implementation. While these improvements do not change the underlying design principals they lead to a significant improvement in running time. We therefore devote this section to a description of the implementation details. To compare the performance of the different versions presented in this section we used the MNIST OCR dataset[1]. The MNIST dataset contains $60,000$ training examples and $10,000$ test examples and thus can underscore significant implementation improvements. Before diving into the technical details we would like to note that many of the techniques are by no means new and have been used in prior implementation of two-class support vector machines (see for instance Platt, 1998, Joachims, 1998, Collobert and Bengio, 2001). However, a few of our implementation improvements build on the specific algorithmic design of multiclass kernel machines.

1. Available at `http://www.research.att.com/~yann/exdb/mnist/index.html`

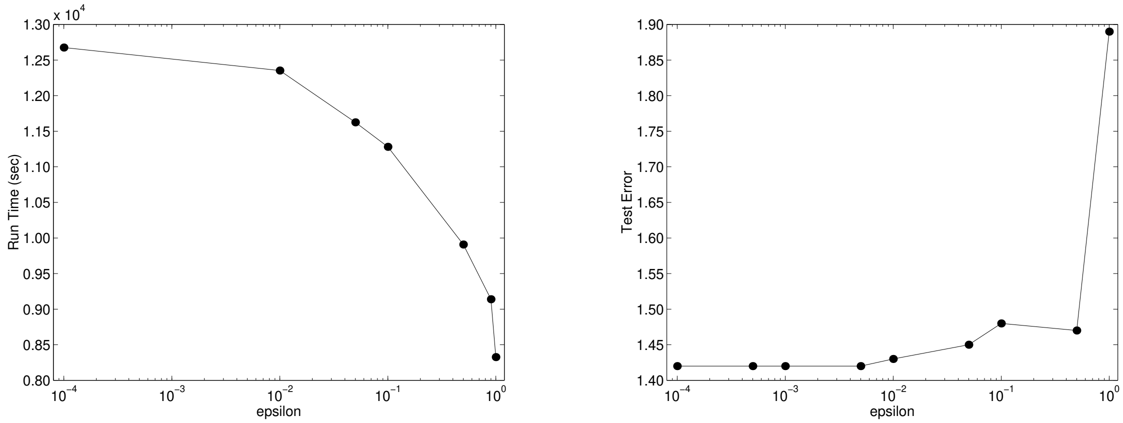

Figure 5: The run time (left) and test error (right) as a function of required accuracy ϵ.

Our starting point and base-line implementation is the algorithm described in Figure 2 combined with the fixed-point algorithm for solving the reduced quadratic optimization problem. In the base-line implementation we simply cycle through the examples for a fixed number of iterations, solving the reduced optimization problem, ignoring the KKT conditions for the examples. This scheme is very simple to implement and use. However, it spends unnecessary time in the optimization of patterns which are correctly classified with a large confidence. We use this scheme to illustrate the importance of the efficient example selection described in the previous section. We now describe the different steps we took, starting from the version described in Section 5.

Using KKT for example selection This is the algorithm described in Figure 4. For each example i and label r we compute $F_{i,r}$. These variables are used to compute ψ_i as described in Section 5. On each round we choose the example p for which ψ_p is the largest and iterate the process until the value of ψ_i is smaller for a predefined accuracy denoted by ϵ. It turns out that the choice of ϵ is not crucial and a large range of values yield good results. The larger ϵ is the sooner we terminate the main loop of the algorithm. Therefore, we would like to set ϵ to a large value as long as the generalization performance is not effected. In Figure 5 we show the running time and the test error as a function of ϵ. The results show that a moderate value of ϵ of 0.1 already yields good generalization. The increase in running time when using smaller values for ϵ is between %20 to %30. Thus, the algorithm is rather robust to the actual choice of the accuracy parameter ϵ so long as it is not set to a value which is evidently too large.

Maintaining an active set The standard implementation described above scans the entire training set and computes ψ_i for each example $\bar{x}_i$ in the set. However, if only a few support patterns constitute the multiclass machine then the vector $\bar{\tau}$ is the zero vector for many example. We thus partition the set of examples into two sets. The first, denoted by A and called the active set, is composed of the set of examples that contribute to the solution, that is, $A = \{i|\bar{\tau}_i \neq \bar{0}\}$. The second set is simply its complement, $A^c = \{i|\bar{\tau}_i = \bar{0}\}$. During the course of the main loop we first search for an example to update from the set A. Only if such an example does not exist, which can happen iff $\forall i \in A, \psi_i < \epsilon$, we scan the

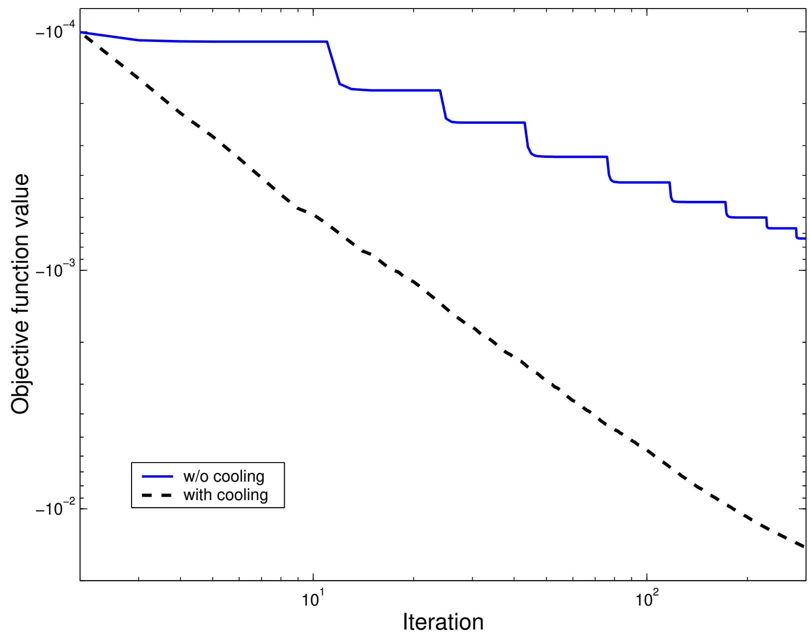

Figure 6: The value of the objective function $\mathcal{Q}$ as a function of the number of iteration for a fixed and variable scheduling of the accuracy parameter ϵ.

set A^c for an example p with $\psi_p > \epsilon$. If such an example exists we remove it from A^c, add it to A, and call the fixed-point algorithm with that example. This procedure spends most of it time adjusting the weights of examples that constitute the active set and adds a new example only when the active set is exhausted. A natural implication of this procedure is that the support patterns can come only from the active set.

Cooling of the accuracy parameter The employment of an active set yields significant reduction in running time. However, the scheme also forces the algorithm to keep updating the vectors $\bar{\tau}_i$ for $i \in A$ as long there is even a single example i for which $\psi_i > \epsilon$. This may result in minuscule changes and a slow decrease in $\mathcal{Q}$ once most examples in A have been updated. In Figure 6 we plot in bold line the value of $\mathcal{Q}$ as a function of the number of iterations when ϵ is kept fixed. The line has a staircase-like shape. Careful examination of the iterations in which there was a significant drop in $\mathcal{Q}$ revealed that these are the iterations on which new examples were added to the active set. After each addition of a new example numerous iterations are spent in adjust the weights $\bar{\tau}_i$. To accelerate the process, especially on early iterations during which we mostly add new examples to the active set, we use a variable accuracy parameter, rather than a fixed accuracy. On early iterations the accuracy value is set to a high value so that the algorithm will mostly add new examples to the active set and spend only a small time on adjusting the weights of the support patterns. As the number of iterations increases we decrease ϵ and spend more time on adjusting the weights of support patterns. The result is a smoother and more rapid decrease in $\mathcal{Q}$ which leads to faster convergence of the algorithm. We refer to this process of gradually decreasing

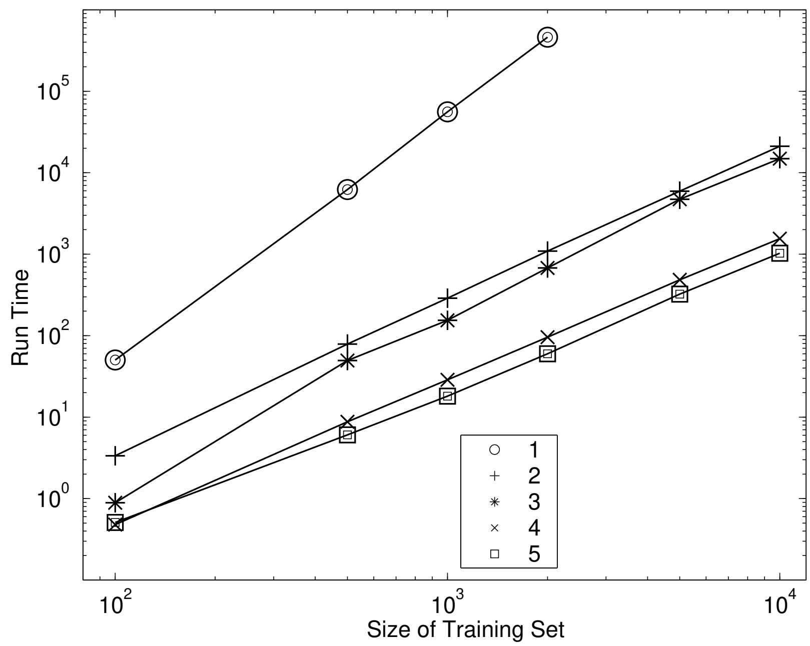

Figure 7: Comparison of the run-time on the MNIST dataset of the different versions as a function of the training-set size. Version 1 is the baseline implementation. Version 2 uses KKT conditions for selecting an example to update. Version 3 adds the usage of an active set and cooling of ϵ. Version 4 adds caching of inner-products. Finally, version 5 uses data structures for representing and using sparse inputs.

ϵ as *cooling*. We tested the following cooling schemes (for $t = 0, 1, \ldots$): (a) exponential: $\epsilon(t) = \epsilon_0 \exp(-t)$; (b) linear: $\epsilon(t) = \epsilon_0/(t+1)$ (c) logarithmic: $\epsilon(t) = \epsilon_0/\log_{10}(t+10)$. The initial accuracy ϵ_0 was set to 0.999. We found that all of these cooling schemes improve the rate of decrease in $\mathcal{Q}$, especially the logarithmic scheme for which $\epsilon(t)$ is relatively large for a long period and than decreases moderately. The dashed line in Figure 6 designate the value of $\mathcal{Q}$ as a function of the number of iterations using a logarithmic cooling scheme for ϵ. In the particular setting of the figure, cooling reduces the number of iterations, and thus the running time, by an order of magnitude.

Caching Previous implementations of algorithms for support vector machines employ a cache for saving expensive kernel-based inner-products (see for instance Platt, 1998, Joachims, 1998, Collobert and Bengio, 2001). Indeed, one of the most expensive steps in the algorithm is the evaluation of the kernel. Our scheme for maintaining a cache is as follows. For small datasets we store in the cache all the kernel evaluations between each example in the active set and all the examples in the training set. For large problems with many support patterns (and thus a large active set) we use a least-recently-used (LRU) scheme as a caching strategy. In this scheme, when the cache is full we replace least used inner-products of an example with the inner-products of a new example. LRU caching is also used in SVM^{light} (Joachims, 1998).

Name	No. of Training Examples	No. of Test Examples	No. of Classes	No. of Attributes
satimage	4435	2000	6	36
shuttle	5000	9000	7	9
mnist	5000	10000	10	784
isolet	6238	1559	26	617
letter	5000	4000	26	16
vowel	528	462	11	10
glass	214	5-fold cval	7	9

Table 1: Description of the small databases used in the experiments.

Data-structures for sparse input instances Platt (1998) and others have observed that when many components of the input vectors are zero, a significant saving of space and time can be achieved using data-structures for sparse vectors and computing only the products of the non-zero components in kernel evaluations. Our implementation uses linked list for sparse vectors. Experimental evaluation we performed indicate that it is enough to have 20% sparseness of the input instances to achieve a speedup in time and reduction in memory over a non-sparse implementation.

To conclude this section we give in Figure 7a comparison of the run-time of the various technical improvements we outlined above. Each version that we plot include all of the previous improvements. The running-time of the version that includes all the algorithmic and implementation improvements is two orders of magnitude faster than the baseline implementation. It took the fastest version 3 hours and 25 minutes to train a *multiclass* kernel-machine on the MNIST dataset using a Pentium III computer running at 600MHz with 2Gb of physical memory. (The fastest version included two more technical improvements which are not discussed here but will be documented in the code that we will shortly make available.) In comparison, Platt (1998) reports a training time of 8 hours and 10 minutes for learning a single classifier that discriminates one digit from the rest. Therefore, it takes over 80 hours to train a set of 10 classifiers that constitute a multiclass predictor. Platt's results were obtained using a Pentium II computer running at 266MHz. While there are many different factors that influence the running time, the running time results above give an indication of the power of our approach for multiclass problems. We believe that the advantage of our direct approach to multiclass problems will become even more evident in problems with a large number of classes such as Kanji character recognition. We would also like to note that the improved running time is not achieved at the expense of deteriorated classification accuracy. In the next section we describe experiments that show that our approach is comparable to, and sometimes better than, the standard support vector technology.

8. Experiments

In this section we report the results of experiments we conducted in order to evaluate our implementation of the multiclass support vector machine described in this paper. We

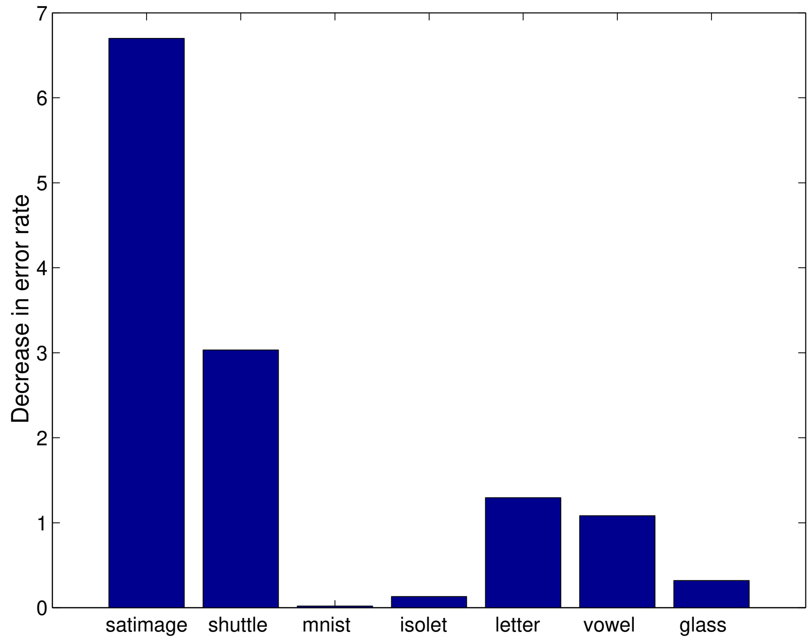

Figure 8: Comparison of a multiclass SVM build using the one-against-rest approach with the multiclass support vector machines studied in this paper.

selected seven datasets, six from UCI repository [2] and the `MNIST` dataset. All but one of the datasets (`glass`), contain a separate test set. For the remaining dataset we used five fold cross validation to compute the error rate. A description of the datasets we used is given in Table 1. Note that on `MNIST`, `Letter` and `Shuttle` we used only a subset of the training set. On each data set we ran two algorithms. The first algorithm uses the multiclass SVM of this paper in a binary mode by training one classifier for each class. Each such classifier is trained to distinguish between the class to the rest of the classes. To classify a new instance we compute the output of each of the binary classifiers and predict the label which attains the highest confidence value. The second algorithm we compared is the multiclass SVM described in this paper. We used our multiclass SVM as the basis for the two algorithms in order to have a common framework for comparison. In both algorithms we used Gaussian kernels. To determine the value of β and σ we used cross validation on the training set. We used 5-fold cross validation for the large datasets and 10-fold cross validation for the small datasets. In all the experiments we set the value of ϵ to 0.001.

A summary of the results is depicted in Figure 8. Each bar in the figure is proportional to the difference in the test error between the two algorithms. Positive value means that the algorithm proposed in this paper achieved a lower error rate than the strawman algorithm based on the 'one-vs-rest' approach. In general the multiclass support vector machine achieved lower error rate than the 'one-vs-rest' method where for the datasets with a large example per class ratio this improvement is significant.

2. Available at http://www.ics.uci.edu/~mlearn/MLRepository.html

Figure 9: Examples of images which are support patterns with a large norm of $\bar{\tau}$ (MNIST dataset).

Name	Test error(%)
mnist	1.42
USPS	4.38
shuttle	0.12
letter	1.95

Table 2: Summary of experiments with large datasets.

We also ran the multiclass SVM on the complete training set of `MNIST`, `Letter` and `Shuttle` and on the `USPS` [3] dataset. The results are summarized in Table 2. Schölkopf *et al.* (1997) achieved a test error of 1.4% on the `MNIST` dataset and a test error of 4.2% on the `USPS` (Schölkopf et al., 1996) using a set of binary SVMs training by the 'one-vs-rest' method. In another work, Dietterich (2000) achieved a test error of 2.71% on the `Letter` dataset and a test error of 0.01% on the `Shuttle` dataset by boosting $C4.5$. Note that in the first three cases our algorithm is comparable to the other algorithms, while it is 10 times worse on the `Shuttle` dataset. This gap can be explained by the fact that most of the instances in the dataset belong to the same class, which is a phenomena that is easier for boosting algorithms to cope with. Finally, we would like to note that using the polynomial kernel of degree 9 and the normalization described by DeCoste and Schölkopf (2001) we achieve an error rate of 1.24% on the test set of `MNIST`.

In Figure 9 we give training images taken from the `MNIST` dataset with high norm of $\bar{\tau}$. All the images we plot attain a margin value of less than β. For each image we give its correct label and in parentheses the probability vector $\bar{\eta}$ using a sparse representation: each entry of the vector is of the form *label:probability*. If there is a training error on an image we added a trailing E in the text. As discussed in the previous sections, the vector $\bar{\eta}$ is a probability vector that is concentrated on the corrected label if it is *not* a support pattern. Therefore, all the above images are support pattern and the vectors $\bar{\eta}$ designate the set of confusable labels – the labels for which the corresponding entries in $\bar{\eta}$ are non-zero. Examination of the different images reveal that the confusion is indeed correlated with the actual digit each training example can be confused with. For instance, the top image on the left hand-side from Figure 9 can be mistakenly classified as 3, 7, or 9. In many cases, the examples with a large norm of $\bar{\tau}$ correspond to mislabeled examples or corrupted images. Therefore, the norm of $\bar{\tau}$ can be used as an aid for data cleaning.

9. Summary

In this paper we presented a novel approach for building multiclass support vector machines. We described in detail an efficient learning algorithm and discussed various implementation issues required for making it practical. Our methods achieve state of the art results with a running time that is competitive with previous methods, and is one order of magnitude faster in some problems. An interesting questions that stem from our research is whether the approach taken in this paper can also be used for other machine learning tasks with kernels, such as ranking problems and regression. We leave this for future research. Another

3. Available at ftp.kyb.tuebingen.mpg.de

interesting direction that we are currently pursuing is analogous online algorithms that use multiple prototypes for multiclass problems.

Acknowledgments

We would like to thank Elisheva Bonchek for useful comments and carefully reading the manuscript. Thanks also to Chih-Jen Lin for fruitful email exchange.

Appendix A. Derivation of the dual optimization problem

We develop the Lagrangian using only the dual variables. For simplicity we use the identity Kernel $K(x_i, x_j) = x_i \cdot x_j$. Substituting Eq. (9) to Eq. (8) we obtain,

$$\begin{aligned}\mathcal{Q}(\eta) &= \sum_{i=1}^{m} \xi_i + \sum_{i,r} \eta_{i,r} x_i \cdot \bar{M}_r - \sum_{i,r} \eta_{i,r} x_i \cdot \bar{M}_{y_i} - \sum_i \xi_i \underbrace{\sum_r \eta_{i,r}}_{=1} + \sum_{i,r} \eta_{i,r} b_{i,r} + \frac{1}{2}\beta \sum_r \|\bar{M}_r\|_2^2 \\ &= \overbrace{\sum_{i,r} \eta_{i,r} x_i \cdot \bar{M}_r}^{\stackrel{\text{def}}{=} S_1} - \overbrace{\sum_{i,r} \eta_{i,r} x_i \cdot \bar{M}_{y_i}}^{\stackrel{\text{def}}{=} S2} + \overbrace{\frac{1}{2}\beta \sum_r \|\bar{M}_r\|_2^2}^{\stackrel{\text{def}}{=} S_3} + \sum_{i,r} \eta_{i,r} b_{i,r} \ . \end{aligned} \tag{44}$$

We substitute $\bar{M}_r$ in the above equation using Eq. (10) and get,

$$\begin{aligned} S_1 &= \sum_{i,r} \eta_{i,r} x_i \cdot \bar{M}_r \\ &= \sum_{i,r} \eta_{i,r} x_i \cdot \beta^{-1} \sum_j x_j (\delta_{y_j,r} - \eta_{j,r}) \\ &= \beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r \eta_{i,r} (\delta_{y_j,r} - \eta_{j,r}) \ , \end{aligned} \tag{45}$$

$$\begin{aligned} S_2 &= \sum_{i,r} \eta_{i,r} x_i \cdot \bar{M}_{y_i} \\ &= \sum_{i,r} \eta_{i,r} x_i \cdot \beta^{-1} \sum_j x_j (\delta_{y_j,y_i} - \eta_{j,y_i}) \\ &= \beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r \eta_{i,r} (\delta_{y_j,y_i} - \eta_{j,y_i}) \\ &= \beta^{-1} \sum_{i,j} x_i \cdot x_j (\delta_{y_j,y_i} - \eta_{j,y_i}) \underbrace{\sum_r \eta_{i,r}}_{=1} \\ &= \beta^{-1} \sum_{i,j} x_i \cdot x_j (\delta_{y_j,y_i} - \eta_{j,y_i}) \\ &= \beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r \delta_{y_i,r} (\delta_{y_j,r} - \eta_{j,r}) \ , \end{aligned} \tag{46}$$

$$S_3 = \frac{1}{2}\beta \sum_r \bar{M}_r \cdot \bar{M}_r$$

$$
\begin{aligned}
&= \frac{1}{2}\beta \sum_r [\beta^{-1} \sum_i x_i(\delta_{y_i,r} - \eta_{i,r})][\beta^{-1} \sum_j x_j(\delta_{y_j,r} - \eta_{j,r})] \\
&= \frac{1}{2}\beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r (\delta_{y_i,r} - \eta_{i,r})(\delta_{y_j,r} - \eta_{j,r}) \ . \qquad (47)
\end{aligned}
$$

Taking the difference $S_1 - S_2$ while using Eqs. (45) and (46) we get,

$$
\begin{aligned}
S_1 - S_2 &= \beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r \eta_{i,r}(\delta_{y_j,r} - \eta_{j,r}) - \beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r \delta_{y_i,r}(\delta_{y_j,r} - \eta_{j,r}) \\
&= -\beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r (\delta_{y_i,r} - \eta_{i,r})(\delta_{y_j,r} - \eta_{j,r}) \ . \qquad (48)
\end{aligned}
$$

Finally, plugging the values for S_1,S_2 and S_3 from Eqs. (47) and (48) into Eq. (44) we get,

$$
\begin{aligned}
\mathcal{Q}(\eta) &= -\beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r (\delta_{y_i,r} - \eta_{i,r})(\delta_{y_j,r} - \eta_{j,r}) \\
&\quad + \frac{1}{2}\beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r (\delta_{y_i,r} - \eta_{i,r})(\delta_{y_j,r} - \eta_{j,r}) + \sum_{i,r} \eta_{i,r} b_{i,r} \\
&= -\frac{1}{2}\beta^{-1} \sum_{i,j} x_i \cdot x_j \sum_r (\delta_{y_i,r} - \eta_{i,r})(\delta_{y_j,r} - \eta_{j,r}) + \sum_{i,r} \eta_{i,r} b_{i,r} \ .
\end{aligned}
$$

References

E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Machine Learning: Proceedings of the Seventeenth International Conference*, 2000.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

E. J. Bredensteiner and K. P. Bennet. Multicategory classification by support vector machines. *Computational Optimizations and Applications*, 12:53–79, 1999.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth & Brooks, 1984.

C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):1–47, 1998.

Yair Censor and Stavros A. Zenios. *Parallel optimization: Theory, Algorithms and Applications.* Oxford University Press, 1997.

Ronan Collobert and Samy Bengio. SVMtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, September 1995.

Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 2000.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, 2001.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

Dennis DeCoste and Bernhard Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1–3):133–168, 2001.

Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–158, 2000.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, January 1995.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

Y. Guermeur, A. Elisseeff, and H. Paugam-Moisy. A new multi-class svm based on a uniform convergence result. In V. Piuri et al., editor, *Proceedings of IJCNN-2000*, 2000.

Klaus-U. Höffgen and Hans-U. Simon. Robust trainability of single neurons. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 428–439, Pittsburgh, Pennsylvania, July 1992.

Thorsten Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

S.S. Keerthi and E.G. Gilbert. Convergence of a generalized smo algorithm for svm classifier design. Technical Report CD-00-01, Control Division Dept. of Mechanical and Production Engineering National University of Singapore, 2000.

C.-J. Lin. Stopping criteria of decomposition methods for support vector machines: a theoretical justification. Technical report, Depratment of Computer Science and Information Engineering, National Taiwan University, May 2001.

J.C. Platt. Fast training of Support Vector Machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

J.C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553. MIT Press, 2000.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):1–40, 1999.

B. Schölkopf. *Support Vector Learning*. PhD thesis, GMD First, 1997.

B. Schölkopf, C. Burges, and A. Smola, editors. *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. Technical Report A.I. Memo No. 1599, Massachusetts Institute of Techology, 1996.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium on Artificial Neural Networks*, April 1999.