FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Fast Training of Multi-class Support Vector Machines

Ürün Dogan
Tobias Glasmachers
Christian Igel

Technical Report no. 03/2011

ISSN: 0107-8283

Fast Training of Multi-Class Support Vector Machines

Ürün Dogan
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
urun.dogan@ini.rub.de

Tobias Glasmachers
Dalle Molle Institute for Artificial Intelligence (IDSIA)
6928 Manno-Lugano, Switzerland
tobias@idsia.ch

Christian Igel
Department of Computer Science
University of Copenhagen
2000 Copenhagen Ø, Denmark
igel@diku.dk

March 22, 2011

Abstract

We present new decomposition algorithms for training multi-class support vector machines (SVMs), in particular the variants proposed by Lee, Lin, & Wahba (LLW) and Weston & Watkins (WW). Although these two types of machines have desirable theoretical properties, they have been rarely used in practice because efficient training algorithms have been missing. Training is accelerated by considering hypotheses without bias, by second order working set selection, and by using working sets of size two instead of applying sequential minimal optimization (SMO). We derive a new bound for the generalization performance of multi-class SVMs. The bound depends on the sum of target margin violations, which corresponds to the loss function employed in the WW machine. The improved training scheme allows us to perform a thorough empirical comparison of the Crammer & Singer (CS), the WW, and the LLW machine. In our experiments, all machines gave better generalization results than the baseline one-vs-all approach. The two-variable decomposition algorithm outperformed SMO. The LLW SVM performed best in terms of accuracy, at the cost of slower training. The WW SVM led to better generalizing hypotheses compared to the CS machine and did not require longer training times. Thus, we see no reason to prefer the CS variant over the WW SVM.

1 Introduction

Long training times limit the applicability of multi-class support vector machines (SVMs). In particular, the canonical extension of binary SVMs to multiple classes (referred to as WW, [32, 6, 31]) as well as the SVM proposed by Lee, Lin, & Wahba (LLW, [19]) are rarely used. These approaches are theoretically sound and experiments indicate that they lead to well-generalizing hypotheses, but efficient training algorithms are not available. Crammer & Singer (CS, [7]) proposed the arguably most popular modification of the WW formulation, mainly to speed-up the training process. Still, the one-vs-all method [31, 24] is most frequently used when SVMs are applied to multi-class problems.

Against this background, we consider batch training of multi-class SVMs with universal (i.e., non-linear) kernels and ask the questions: Can we increase the learning speed of multi-class SVMs by using a more efficient quadratic programming method? Do the convincing statistical properties of the LLW machine lead to better hypotheses in practice? Can we support our experience about the performance of these machines by an instructive generalization bound? This study gives positive answers to these questions. We provide efficient training algorithms for all machines. These make training of LLW machines practical and allow to train WW SVMs as fast as CS's variant. Extensive experiments demonstrate the superior performance of the LLW machine in terms of generalization performance. We derive a new generalization bound for multi-class SVMs that suggests that the loss function considered in the WW machine is a natural choice.

In the following, we first introduce multi-class SVMs and the corresponding optimization problems with an emphasis on the LLW machine. We then show how to solve these problems efficiently with decomposition algorithms. Then we prove our new generalization bound for multi-class SVMs. An extensive empirical comparison of the different training algorithms and SVM formulations closes this study.

2 Multi-Class SVMs

All multi-class SVMs considered in this study solve d-class classification problems by constructing decision functions of the form

$$x \mapsto \arg \max_{c \in \{1,\dots,d\}} \left[\langle \boldsymbol{w}_c, \boldsymbol{\phi}(x) \rangle + b_c \right] \tag{1}$$

given i.i.d. training data $\big((x_1, y_1), \dots, (x_\ell, y_\ell)\big) \in \big(X \times \{1, \dots, d\}\big)^\ell$. Here, $\boldsymbol{\phi} : X \to \mathcal{H}$, $\boldsymbol{\phi}(x) = k(x, \cdot)$, is a feature map into a reproducing kernel Hilbert space $\mathcal{H}$ with corresponding kernel k, and $\boldsymbol{w}_1, \dots, \boldsymbol{w}_d \in \mathcal{H}$ are class-wise weight vectors. The CS machine is usually only defined for hypotheses without bias term, that is, for $b_c = 0$ (see [14] for a formulation with bias parameters).

Two basic strategies to extend SVMs to multi-category classification can be distinguished. One approach is to combine separately trained binary SVM classifiers after training as done in the prominent one-versus-all method (OVA). In the second family of algorithms, a single optimization problem considering all classes is derived and solved

at once. These *all-in-one* methods are usually computationally more demanding [14]. Still, there are several reasons to prefer all-in-one SVMs. First, from our perspective the all-in-one approach is theoretically more sound, and the machines have better statistical properties (see below). Second, although no significant differences in classification accuracy between the two paradigms have been observed in some studies (perhaps due to insufficient model selection because of slow learning algorithms), in other studies (e.g., [8]) and our own experience all-in-one approaches perform significantly better than OVA. Third, it has been argued that approaches such as OVA are prohibitive and all-in-one methods are required for the general problem of learning structured responses [4, 29].

One-vs-all (OVA). In the popular OVA approach, d standard binary SVMs are built. The machine that learns the weight vector $\boldsymbol{w}_c$ is trained to separate the training patterns belonging to class c from all other training patterns. Even if all binary classifiers are consistent, the resulting OVA classifier in general is not.[1]

Weston & Watkins (WW) SVM. In contrast, all-in-one methods directly obtain all weight vectors $\boldsymbol{w}_c$ from a single optimization problem taking all class relations into account at once. The most prominent example is the WW multi-class SVM proposed in [32]. If we consider hypotheses without bias, this SVM is trained by solving the primal problem $\min_{\boldsymbol{w}_c} \frac{1}{2} \sum_{c=1}^{d} \langle \boldsymbol{w}_c, \boldsymbol{w}_c \rangle + C \sum_{n=1}^{\ell} \sum_{c=1}^{d} \xi_{n,c}$ subject to the constraints $\forall n \in \{1, \ldots, \ell\}, \forall c \in \{1, \ldots, d\} \setminus \{y_n\} : \langle \boldsymbol{w}_{y_n} - \boldsymbol{w}_c, \boldsymbol{\phi}(x_n) \rangle \geq 2 - \xi_{n,c}$ and $\forall n \in \{1, \ldots, \ell\}, \forall c \in \{1, \ldots, d\} : \xi_{n,c} \geq 0$. If the first set of inequality constraints is replaced by $\langle \boldsymbol{w}_{y_n} - \boldsymbol{w}_c, \boldsymbol{\phi}(x_n) \rangle \geq 1 - \xi_{n,c}$, this formulation equals the one suggested by Vapnik [31], which is equivalent after rescaling $\boldsymbol{w}_c$ and C. Bredensteiner & Bennett's multi-category SVM also coincides with the WW formulation [6]. This approach can be regarded as the canonical extension of binary SVMs to multiple classes, because the objective function is basically the sum of the objective functions of the binary SVMs. The method is more sophisticated than OVA, because the constraints are adapted to the decision scheme (1). The $\ell \times d$ slack variables $\xi_{n,c}$ correspond to the hinge loss when separating example x_n from the decision boundary between classes y_n and c.

Crammer & Singer (CS) SVM. Crammer & Singer [7] proposed an alternative multi-class SVM. They also take all class relations into account at once and solve a single optimization problem, however, with fewer slack variables. The main reason for this modification of the WW primal problem was to speed-up the training, because the WW approach turned out to be too slow for many applications. The CS classifier is trained by solving the primal problem $\min_{\boldsymbol{w}_c} \frac{1}{2} \sum_{c=1}^{d} \langle \boldsymbol{w}_c, \boldsymbol{w}_c \rangle + C \sum_{n=1}^{\ell} \xi_n$ subject to

[1] As a minimal counterexample, consider a single-point input space $X = \{x_0\}$ and three classes. Let the input belong to the three classes with probabilities 0.4, 0.3, and 0.3, respectively. In the limit of large datasets, each class is less frequently sampled than it is not sampled. Then, according to Steinwart's analysis [26] we obtain $\langle w_c, \phi(x_0) \rangle + b_c = -1$ (with $w_c = 0$ and $b_c = -1$) uniformly for all classes $c \in \{1, \ldots, d\}$. Thus, the probability of error is 2/3, which is worse than the Bayes risk of 0.6.

$\forall n \in \{1,\dots,\ell\}, \forall c \in \{1,\dots,d\} \setminus \{y_n\} : \langle \boldsymbol{w}_{y_n} - \boldsymbol{w}_c, \boldsymbol{\phi}(x_n)\rangle \geq 1 - \xi_n$ and $\forall n \in \{1,\dots,\ell\} : \xi_n \geq 0$.

For learning structured data, CS's method is usually the SVM algorithm of choice.

Lee, Lin, & Wahba (LLW) SVM. Lee, Lin, & Wahba modified the standard multi-class SVM formulation for theoretical reasons. In contrast to the other machines, their SVM relies on a classification calibrated loss function, which implies Fisher consistency [28, 20]. Further, it was shown in [3] that, roughly speaking, for any surrogate loss function Φ used by a learning machine, no bound on the 0-1 loss excess risk (i.e., risk relative to the Bayes optimal value) in terms of the Φ excess risk is possible if Φ is not classification calibrated. However, up to now no efficient solver for the LLW SVM has been derived and implemented and thus empirical comparisons with other methods are rare.

The primal problem of the LLW SMV can be stated as

$$\begin{aligned}
\min_{\boldsymbol{w}_c} \quad & \frac{1}{2}\sum_{c=1}^{d}\langle \boldsymbol{w}_c, \boldsymbol{w}_c\rangle + C\sum_{n=1}^{\ell}\sum_{c=1}^{d}\xi_{n,c} \\
\text{s.t.} \quad & \forall\, n \in \{1,\dots,\ell\}, c \in \{1,\dots,d\} \setminus \{y_n\} : \langle \boldsymbol{w}_c, \boldsymbol{\phi}(x_n)\rangle + b_c \leq -\frac{1}{d-1} + \xi_{n,c} \\
& \forall\, n \in \{1,\dots,\ell\}, c \in \{1,\dots,d\} : \xi_{n,c} \geq 0 \\
& \forall \boldsymbol{h} \in \mathcal{H} : \sum_{c=1}^{d}(\langle \boldsymbol{w}_c, \boldsymbol{h}\rangle + b_c) = 0 \ . \qquad (2)
\end{aligned}$$

If the feature map is injective then the sum-to-zero constraint (2) can be expressed as $\sum_{c=1}^{d} \boldsymbol{w}_c = \mathbf{0}$ and $\sum_{c=1}^{d} b_c = 0$. The corresponding dual problem reads

$$\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \frac{1}{d-1}\cdot\sum_{n=1}^{\ell}\sum_{c=1}^{d}\alpha_{n,c} - \frac{1}{2}\sum_{n,m=1}^{\ell}\sum_{c,e=1}^{d}(\delta_{c,e} - 1/d)\alpha_{n,c}\alpha_{m,e}k(x_n, x_m) \\
\text{s.t.} \quad & \forall\, n \in \{1,\dots,\ell\}, c \in \{1,\dots,d\} \setminus \{y_n\} : 0 \leq \alpha_{n,c} \leq C \\
& \forall\, n \in \{1,\dots,\ell\} : \alpha_{n,y_n} = 0 \\
& \forall\, c \in \{1,\dots,d\} : \sum_{n=1}^{\ell}\alpha_{n,c} = D \ . \qquad (3)
\end{aligned}$$

The last constraint (3) ensures that all d sums $\sum_{n=1}^{\ell}\alpha_{n,c}$ take the same value $D \in \mathbb{R}$. The value of D itself does not matter.

3 Dropping the Bias Parameters

The constraint (3) makes the quadratic program above difficult to solve for decomposition techniques ([5], see next section), because a feasible step requires the modification

of at least d variables simultaneously. This problem concerns both the WW and the LLW approach. However, such a constraint is not present in the standard CS machine, because Crammer & Singer dropped the bias terms b_c, which are of minor importance when working with characteristic or universal kernels [23].

Instead of restricting this trick to the CS machine, we propose to apply it also to the WW and LLW SVMs. If we do so, the constraint (3) simply vanishes from the dual, while everything else remains the same.

This step of removing the bias terms is crucial, because it allows us for the first time to solve the WW and the LLW machine with elaborated decomposition techniques as discussed in the next section.

Dropping the bias terms in all machines is also a prerequisite for a fair empirical comparison of the approaches. First, it makes fast training and therefore appropriate model selection and evaluation on several benchmark problems feasible. Second, now all machines consider the same hypothesis space. Instead we could have introduced the bias term into the CS method, but then the resulting dual problem gets much more complicated, because we end up with two sets of interfering equality constraints, see also [14], which renders the solution technique presented in the next section intractable.

4 Solvers and Working Set Selection

In this section we present our specialized solvers for the quadratic programs arising from training of non-linear multi-class SVMs. Fast solvers are needed to make SVM training with large datasets and/or large collections of problems tractable. In addition, faster training of single machines allows for more thorough model selection as discussed in section 6.1.

First, we introduce standard decomposition techniques and the SMO algorithm as a basis for the much less common technique of sequential two-dimensional optimization (S2DO). This technique results in a considerable speed-up over conventional SMO for problems without equality constraints such as WW and LLW SVMs.

For deriving our training algorithms, we consider quadratic programs of the canonical form

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \forall n \in \{1,\dots,m\} : L_n \le \alpha_n \le U_n \end{aligned} \tag{4}$$

for $\boldsymbol{\alpha} \in \mathbb{R}^m$. Here $\boldsymbol{v} \in \mathbb{R}^m$ is some vector, $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is a (symmetric) positive definite matrix, and $L_n \le U_n$ are component-wise lower and upper bounds. The gradient $\boldsymbol{g} = \nabla f(\boldsymbol{\alpha})$ of (4) has components $g_n = \frac{\partial f(\boldsymbol{\alpha})}{\partial \alpha_n} = v_n - \sum_{i=1}^m \alpha_i Q_{in}$. The dual problems of the WW and LLW machines without bias can directly be written in this canonical form. The dual problem of the CS machine introduces a large number of additional equality constraints, which will be ignored for the moment and will be discussed in section 4.4.

The most frequently used algorithms for solving quadratic programs of non-linear SVMs are decomposition methods [21, 22, 9, 11, 5]. These methods iteratively decompose

the quadratic program into subprograms, which are restricted to a subset B (working set) of variables, and then solve these subprograms. A desirable property of state-of-the-art decomposition algorithms is that iterations are fast in the sense that for any fixed upper bound $q \in \mathbb{N}$ on the working set size each iteration requires only $\mathcal{O}(m)$ operations. A general decomposition scheme for (4) is given in Algorithm 1.

Algorithm 1: Decomposition algorithm for problem (4).

Input: feasible initial point $\boldsymbol{\alpha}^{(0)}$, accuracy $\varepsilon \geq 0$
compute the initial gradient $\boldsymbol{g}^{(0)} \leftarrow \nabla f(\boldsymbol{\alpha}^{(0)}) = \boldsymbol{v} - \boldsymbol{Q}\boldsymbol{\alpha}^{(0)}$
$t \leftarrow 1$
while *stopping criterion not met* **do**
  select working indices $B^{(t)} \subset \{1, \ldots, m\}$
  solve subproblem restricted to $B^{(t)}$ and update $\boldsymbol{\alpha}^{(t)} \leftarrow \boldsymbol{\alpha}^{(t-1)} + \boldsymbol{\mu}^{\star(t)}$
  update the gradient $\boldsymbol{g}^{(t)} \leftarrow \boldsymbol{g}^{(t-1)} - \boldsymbol{Q}\boldsymbol{\mu}^{\star(t)}$
  set $t \leftarrow t + 1$

For a vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ and an index set $I \subset \{1, \ldots, m\}$ let $\boldsymbol{\alpha}_I = \sum_{i \in I} \alpha_i \boldsymbol{e}_i$ denote the projection to the components indexed by I, where $\boldsymbol{e}_i \in \mathbb{R}^m$ is the unit vector in which the i-th component is 1. If we assume that all variables except those in $B = \{b_1, \ldots, b_{|B|}\}$ are fixed, the subproblem can be written as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}_B} \quad & f(\boldsymbol{\alpha}_B) = f(\boldsymbol{\alpha}_F + \boldsymbol{\alpha}_B) = (\boldsymbol{v}_B - \boldsymbol{Q}\boldsymbol{\alpha}_F)^{\mathrm{T}} \boldsymbol{\alpha}_B - \frac{1}{2}\boldsymbol{\alpha}_B \boldsymbol{Q}\boldsymbol{\alpha}_B + \text{const} \\ \text{s.t.} \quad & \forall n \in B : L_n \leq \alpha_n \leq U_n \end{aligned} \tag{5}$$

Here, the complement $F = \{1, \ldots, m\} \setminus B$ of B contains the indices of the fixed variables.

The convergence properties of the decomposition method are determined by the heuristic for selecting the working indices. Given a feasible search point, the set of possible working indices that indicate a violation of the KKT optimality condition by the corresponding variable is called violating set [18]. We denote the set of violating indices in a search point $\boldsymbol{\alpha}$ by

$$\mathcal{B}(\boldsymbol{\alpha}) = \left\{ n \in \{1, \ldots, m\} \;\middle|\; \alpha_n > L_n \wedge g_n < 0 \text{ or } \alpha_n < U_n \wedge g_n > 0 \right\} .$$

If the working set has minimum size for generating feasible solutions, this approach is called sequential minimal optimization (SMO, [22]), which is the most frequently used technique for SVM training. The minimum working set size depends on the number of equality constraints. For problem (4) it is one. Next, we briefly discuss the trade-offs influencing the choice of the number of elements in the working set. Then we describe working set selection heuristics for solving (4).

4.1 Working Set Sizes for Decomposition Algorithms

The size of the working set B influences the overall performance of the decomposition algorithm in a number of different ways. First, the complexity of solving subproblem (5)

analytically grows combinatorially with $|B|$. This limits the working set size to few variables unless we are willing to use a numerical solver as done in [16]. Second, the larger $|B|$, the less well-founded is a heuristic for picking the actual working set from the $\mathcal{O}(m^{|B|})$ candidates, because we regard such a heuristic as acceptable only if its time complexity is $\mathcal{O}(m)$. At the same time large $|B|$ allows to find working sets on which large progress can be made. Third, the gradient update takes $\mathcal{O}(m \cdot |B|)$ operations. That is, small working sets result in fast iterations making few progress, while larger working sets result in slower iterations making larger progress. For example, we may roughly trade two iterations with $|B| = 1$ for one iteration with $|B| = 2$. The latter can take into account correlations between the variables. However, in two subsequent iteration the second iteration can profit from the gradient update of the first one, and can therefore make a better decision for picking its active variable.

Thus, working set sizes should be small in order to avoid unacceptable computation times for solving the subproblem. In addition, there is an inherent trade-off between many cheap iterations profiting from frequent gradient updates on the one side and fewer slow iterations, each with a larger choice of working sets, making larger progress per iteration on the other side. In our point of view, none of the above arguments enforces the working set size to be *minimal* (or *irreducible*) as in SMO.

Instead of always using the minimum working set, we propose to use working sets of size two whenever possible. We refer to this strategy as *sequential two-dimensional optimization* (S2DO). Considering two variables is a good compromise between (a) the complexity of solving the subproblems analytically, (b) the availability of well-motivated heuristics for working set selection, and (c) the computational cost per decomposition iteration.

4.2 Second Order Working Variable Selection for SMO

We adopt the second order working set selection introduced by [9].[2] Let us assume that the optimization problem is restricted to the single variable α_b, $b \in \mathcal{B}(\boldsymbol{\alpha})$. The update direction is thus $\boldsymbol{e}_b$. If we ignore the box constraints, the optimal step size is given by the Newton step $\hat{\mu} = g_b \,/\, Q_{bb}$ yielding a *gain* of $f(\boldsymbol{\alpha} + \hat{\mu} \cdot \boldsymbol{e}_b) - f(\boldsymbol{\alpha}) = \hat{\mu}^2 \cdot \frac{Q_{bb}}{2} = \frac{g_b^2}{2Q_{bb}}$. This definition of gain leads to the greedy heuristic $b^{(t)} = \arg\max \left\{ (g_n^{(t-1)})^2 / Q_{nn} \,\middle|\, n \in \mathcal{B}(\boldsymbol{\alpha}^{(t-1)}) \right\}$ for choosing the working index, where we define $g_n^{(t-1)} = \frac{\partial f}{\partial \alpha_n}(\boldsymbol{\alpha}^{(t-1)})$. To obtain a feasible new search point, the Newton step must be clipped to the feasible region by computing $\mu^\star = \max\left\{L_b - \alpha_b, \min\{U_b - \alpha_b, \hat{\mu}\}\right\}$. The update of the variables is simply given by $\boldsymbol{\alpha}^{(t)} = \boldsymbol{\alpha}^{(t-1)} + \mu^\star \boldsymbol{e}_b$.

In each iteration the algorithm needs the $b^{(t)}$-th column of the large kernel-based matrix $\boldsymbol{Q}$ for the gradient update. In addition, the diagonal entries Q_{nn} needed for

[2] Note that in the case of single-variable working sets first order and second order working set selection coincide for the important special case $\forall 1 \leq i, j \leq m : Q_{ii} = Q_{jj}$, e.g., for normalized kernels ($k(x, x) = 1$ for all $x \in X$). The same holds for the SMO working set selection strategy for the CS machine described in section 4.4.

second order working set selection should be precomputed and stored. This requires $\mathcal{O}(m)$ time and memory.

4.3 Second Order Working Pair Selection for S2DO

We now derive second order working set selection for (4) using $|B| = 2$. Our focus is on multi-class SVMs, but the selection scheme is also valid for binary SVMs without bias. We follow the common scheme proposed by [9]. This amounts to selecting the first index according to the maximum absolute value of the gradient component, $i = \arg\max_{k \in \mathcal{B}(\boldsymbol{\alpha})} |g_k|$, and the second component by maximization of the gain (which depends on the first variable).

In appendix A, we derive the gain and the update of the variables when using S2DO in the absence of equality constraints. This is a generalization of [27]. Note that the procedures differ considerably from the ones proposed in [9, 11, 5].

4.4 Solving the Crammer & Singer Multi-class SVM Using SMO

The dual problem of the CS machine has ℓ equality constraints. Still, the problem can be solved by decomposition algorithms using workings sets of size two. The trick is to ensure that in every iteration the two variables correspond to the same training pattern. By doing so, the solver naturally respects all ℓ equality constraints. This was also suggested in [4] for online learning using first-order working set selection. That is, the algorithm presented in section 4.2 is not applicable and SMO and S2DO coincide.

5 Generalization Analysis

In [30], the *empirical* risk of multi-class SVMs is upper bounded in terms of the mean of the slack variables. Based on this bound it is argued that the CS SVM has advantages compared to the WW formulation because it leads to lower values in the bounds. We think that this argument is not convincing for several reasons. First, one has to be careful when drawing conclusions based on upper bounds on performance in general. Second, the empirical error may only be a weak predictor of the generalization error (in particular for large values of C). Apart from these general arguments, the statement is not supported when looking at generalization bounds. These bounds are instructive, because they indicate why it may be beneficial to sum-up all margin violation in the multi-class SVM optimization problem. As an example, we extend a bound on the generalization error of binary SVMs by Shawe-Taylor and Cristianini [25] to the multi-class case in order to investigate the impact of the different loss functions on the generalization performance.

Let $h_c(x) = \langle \boldsymbol{w}_c, \boldsymbol{\phi}(x) \rangle$. After dropping the bias term in the WW machine the conceptual difference between the WW and the CS approach is the loss function used to measure margin violations. For a given training example (x_i, y_i) the WW machine penalizes the sum $\sum_{c \neq y_i} [1 - h_{y_i}(x_i) + h_c(x_i)]_+$ of margin violations,[3] while the CS

[3]For simplicity we use the target margin $\gamma = 1$ proposed in [31] for the analysis. This makes the

machine penalizes the maximum margin violation $\max_{c \neq y_i}[1 - h_{y_i}(x_i) + h_c(x_i)]_+$. Here we use the short notation $[t]_+ = \max\{0, t\}$.

The basic idea of our analysis is the following: There are $d-1$ mistakes one can make per example x_i, namely preferring class e over the true class y_i ($e \in \{1, \ldots, d\} \setminus y_i$). Each of these possible mistakes corresponds to one binary problem (having a decision function with normal $\boldsymbol{w}_{y_i} - \boldsymbol{w}_e$) indicating the specific mistake. One of these mistakes is sufficient for wrong classification and no "binary" mistake at all implies correct classification. A union bound over all mistakes gives the multi-class generalization result based on known bounds for binary classifiers.

Let us first restate a fundamental result from [25] for binary classification problems with labels $\{\pm 1\}$. It bounds the risk under the 0-1 loss depending on the fat shattering dimension (e.g., see [17, 1]) of the class of real-valed decision functions. We measure the margin violation of training pattern (x_i, y_i) by $z_i = [\gamma - y_i h(x_i)]_+$, collected in the vector $\boldsymbol{z} = (z_1, \ldots, z_\ell)^{\mathrm{T}} \in \mathbb{R}^\ell$. Then we have:

Theorem 1 (Corollary 6.14 from [25]). *Let $\mathcal{F}$ be a sturdy class of functions $h : \mathcal{F} \to [a, b] \in \mathbb{R}$ with fat shattering dimension $\mathrm{fat}_{\mathcal{F}}(\gamma)$. Fix a scaling of the output range $\eta \in \mathbb{R}$. Consider a fixed but unknown probability distribution on the input space X. Then with probability $1 - \delta$ over randomly drawn training sets T of size ℓ for all $0 < \gamma < b - a$ the risk of a function $h \in \mathcal{F}$ thresholded at zero is bounded by*

$$\epsilon_h = \frac{2}{\ell}\left(\left[\mathrm{fat}_{\mathcal{F}}(\gamma/16) + 64\tilde{D}^2\right] \log_2\left(65\ell(1+\tilde{D})^3\right) \right.$$
$$\left. \cdot \log_2(9e\ell(1+\tilde{D})) + \log_2\left(\frac{64\ell^{1.5}(b-a)}{\delta\eta}\right) \right)$$

with $\tilde{D} = 2(\sqrt{\|\boldsymbol{z}\|_1 \cdot (b-a)} + \eta)/\gamma$, provided $\ell \geq 2/\epsilon_h$ and there is no discrete probability on misclassified training points.

Ignoring logarithmic terms we have $\epsilon_h \in \tilde{\mathcal{O}}\left((\mathrm{fat}(\gamma/16) + \|\boldsymbol{z}\|_1/\gamma^2)/\ell\right)$.

Now we use a union bound over the $d(d-1)/2$ possible pairs of classes to transfer this result to the multi-class case. For a more elaborate treatment of fat shattering in the multi-class case we refer to the literature [17, 13].

We decompose the training set into subsets $T_c = \{(x_i, y_i) \in T \mid y_i = c\}$, $c \in \{1, \ldots, d\}$ according to the training labels and denote their sizes by $\ell_c = |T_c|$. The natural extension of the margin violations d to the loss functions used in the WW and CS machines is $z_{i,c} = [\gamma - h_{y_i}(x_i) + h_c(x_i)]_+$ for $c \neq y_i$. We collect these values in vectors $\boldsymbol{z}^{(c,e)} \in \mathbb{R}^{\ell_c + \ell_e}$ with entries $z_{i,e}$ and $z_{i,c}$ for $i \in T_c$ and $i \in T_e$, respectively, for each fixed pair (c, e) of different classes. We also define the vector $\boldsymbol{z} \in \mathbb{R}^{\ell \times (d-1)}$ collecting all margin violations.

The risk of each binary classifier $h_c - h_e$ separating the problem restricted to the

target margins of the two machines directly comparable.

classes c and e can be upper bounded by

$$\epsilon_h^{(c,e)} = \frac{2}{\ell^{(c,e)}} \Bigg(\left[\text{fat}_{\mathcal{F}}(\gamma/16) + 64\tilde{D}^2 \right] \log_2 \left(65\ell^{(c,e)}(1+\tilde{D})^3 \right) \\ \cdot \log_2(9e\ell^{(c,e)}(1+\tilde{D})) + \log_2 \left(\frac{128[\ell^{(c,e)}]^{1.5}(b-a)}{\delta\eta} \right) \Bigg)$$

with probability $1-\delta/2$, where $\ell^{(c,e)} = \ell_c + \ell_e$.

Now we estimate the probability that a pattern x belongs to either class c or e. This can be done by considering the observed class frequencies and applying the Hoeffding bound. We get

$$\frac{\ell^{(c,e)}}{\ell} + \frac{1}{\sqrt{\ell}} \sqrt{\frac{\log(d(d-1)) - \log\delta}{2}}$$

with a probability of $1-\delta'/2$ with $\delta' = 2\delta/(d(d-1))$. That is, this bound holds simultaneously for all $d(d-1)/2$ pairs of classes with a probability of $1-\delta/2$.

We again apply the union bound and get the risk bound for the multi-class case

$$\epsilon_h \leq \sum_{1 \leq c < e \leq d} \left(\frac{\ell^{(c,e)}}{\ell} + \frac{1}{\sqrt{\ell}} \sqrt{\frac{\log(d(d-1)) - \log\delta}{2}} \right) \cdot \epsilon_h^{(c,e)} \tag{6}$$

with a probability of $1-\delta$, where we measure the complexity of the class of $\mathbb{R}^d$-valued functions f used for multi-class classification by the maximal fat shattering dimension of the real-valued differences $h_e - h_c$. Thus, ignoring logarithmic terms we have

$$\epsilon_h \in \tilde{\mathcal{O}} \left(\frac{d(d-1)\text{fat}_{\mathcal{F}}(\gamma/16) + \|\boldsymbol{z}\|_1/\gamma^2}{\ell} \right) \ . \tag{7}$$

The bounds (6) and (7) indicate that regularized minimization of $\|\boldsymbol{z}\|_1$ is a natural learning strategy. The primal problem of the WW machine can be written as

$$\min \frac{1}{2} \sum_c \|\boldsymbol{w}_c\|^2 + C \cdot \sum_i \sum_c z_{i,c} = \frac{1}{2} \sum_c \|\boldsymbol{w}_c\|^2 + C \cdot \|\boldsymbol{z}\|_1$$

and the CS primal as

$$\min \frac{1}{2} \sum_c \|\boldsymbol{w}_c\|^2 + C \cdot \sum_i \max_c z_{i,c} \ .$$

Thus, the WW machine implements the strategy of minimizing $\|\boldsymbol{z}\|_1$ straightly, while the CS approach minimizes $\|\boldsymbol{z}\|_1$ only indirectly. Given that the bounds are not tight and the general warning above, this observation should not be over-interpreted.

6 Experiments and Results

The goal of our experiments was to rank the different machines in terms of generalization performance and training speed. Further, we wanted to assess whether we can speed-up training of multi-class SVMs by applying the S2DO algorithm instead of SMO.

6.1 Experimental setup

We trained all machines on a number of standard benchmark problems from the UCI repository [2]. We selected the SVM parameter C and the bandwidth parameter γ of the radial Gaussian kernel $k(x, x') = \exp(-\gamma\|x - x'\|^2)$ by nested grid search on logarithmic scale using 5-fold cross validation. We recorded the best parameter settings found (see Table 1) for the different machines and computed the error on the test sets.

Table 1: Best hyperparameters found in the model selection procedure. The values of C and γ are given in logarithmic units [$\log 2$].

	WW		**LLW**		**CS**		**OVA**	
	γ	C	γ	C	γ	C	γ	C
Abalone	0	-3	0	-6	0	-3	0	-9
Car	-2	5	-2	5	-2	5	-2	5
Glass	1	-4	-3	1	-3	2	-3	2
Iris	-9	9	-4	5	-6	6	-12	18
OptDigits	-5	1	-6	10	-6	5	-5	10
Page Blocks	-4	8	-7	16	-5	11	-9	20
Sat	-1	2	-1	2	-1	2	-1	4
Segment	-4	7	-4	15	-9	12	-5	10
SoyBean	-6	1	-7	4	-6	3	-7	3
Vehicle	-7	10	-7	11	-7	10	-8	12

It is well known that the generalization performance of SVMs depends crucially on the choice of the kernel and the regularization parameter. Thus, reliable model selection is important, in particular in a comparison study. The combination of grid search and K-fold cross validation gives solid performance across a wide variety of problems. However, it is a costly procedure that amounts to training K SVMs for each grid point, which sums up to hundreds of calls to the quadratic program solver per problem. This high computational cost is the price we pay for reliability – and a good reason for employing fast solvers as those presented in the previous section.

For the WW and LLW machines (with dual problems without equality constraints) we compared SMO and S2DO by measuring the number of decomposition iterations as well as training time.

For a fair comparison of training times of different types of SVMs it is important to choose comparable stopping criteria for the quadratic programming. Unfortunately, this is hardly possible when the quadratic programs differ. However, as a minimal requirement we made sure that all stopping conditions coincide for the case of binary classification, where all machines reduce to the standard SVM without bias. We use the standard technique to stop the solvers as soon as the KKT conditions for optimality are satisfied up to an accuracy of $\epsilon = 10^{-3}$. To rule out that our results are just an artifact of this choice, we repeated all CS experiments with $\epsilon = 10^{-5}$ (without observing increased classification accuracy).

All our solvers use the same highly efficient caching and shrinking techniques, which

have been specifically tailored to the structure of multi-class SVM dual problems. The kernel cache operates on the level of training examples, not single variables, while shrinking techniques work on both levels, that is, on single variables as well as on chunks of variables corresponding to training examples. All experiments were implemented using the Shark machine learning library [15].

All numbers reported in the results section are medians over 10 independent trials with different splits of the datasets into training and test sets. We applied the statistical testing procedure suggested in [10] with a significance level of $p < 0.05$ for all tests in this study.

6.2 Results

Table 2: Generalization performance of the different multi-class SVMs measured by accuracy on test sets.

	WW	**LLW**	**CS**	**OVA**
Abalone	0.2605	**0.2682**	0.2165	0.2672
Car	0.9807	**0.9846**	0.9807	0.9807
Glass	**0.7231**	**0.7231**	0.7077	0.6923
Iris	**0.9556**	**0.9556**	**0.9556**	0.9333
OptDigits	0.9761	**0.9789**	0.9750	0.9761
Page Blocks	**0.9342**	0.9330	0.9294	0.9312
Sat	0.9235	0.9230	**0.9240**	0.9135
Segment	0.9639	**0.9683**	0.9495	0.9625
SoyBean	0.9032	**0.9247**	**0.9247**	**0.9247**
Vehicle	**0.8425**	**0.8425**	0.8150	0.8346

The main results are summarized in Tables 2 and 3. In our experiments, S2DO was statistically significantly faster than SMO with respect to training time and number of iterations. The time taken by one S2DO iteration was roughly equivalent to two SMO iterations.[4]

The LLW SVM was significantly the best machine in terms of accuracy. Albeit its training times were acceptable when using our decomposition algorithms—in particular when using S2DO—, it was the slowest.

Training the CS machines was slower than training WW SVMs in seven of the benchmarks. For six of the datasets they needed more iterations. A statistical comparison did not support a significant difference between CS and WW SVMs in terms of training complexity. However, the accuracy of WW's algorithm was statistically significantly superior to the CS SVMs for both values of ϵ.

[4]The **Iris** data set is an exception from this rule of thumb. With 105 training examples and three classes it is the smallest data set in our benchmark suite. The SMO algorithm performed several fast shrinking and unshrinking operations, the S2DO none because it solved the problem so quickly. Thus, each S2DO iteration considered the complete set of variables, most SMO iterations only subsets. Therefore, a single SMO iteration took less time on average. However, SMO needed much more iterations.

The computational complexity of the OVA approach scales linearly with the number of classes d while the all-in-one methods are in $\omega(d)$. Accordingly, OVA is considerably faster than the other algorithms, but yielded hypotheses with a statistically significantly worse accuracy.

7 Discussion and Conclusions

We presented a fast training algorithm for WW and LLW SVMs. By dropping the bias term—as done in the CS approach—we get rid of the equality constraints in the dual problems for both machines. This makes decomposition methods easily applicable. We proposed a second order working set selection algorithm using working sets of size two for these problems. Instead of choosing the smallest, irreducible working set size, we in general propose to use a working set size of two whenever possible. This allows for a still tractable analytic solution of the sub-problem and in our experience corresponds to a significantly better trade-off between iteration complexity (as, e.g., determined by the working set selection heuristic and the gradient update) and progress. That is, we favor *sequential two-dimensional optimization* (S2DO) over the strict SMO heuristic. This is also supported by the findings in [27] for binary SVMs. The S2DO heuristic is not restricted to the SVMs considered in this study, but can be applied to any machine involving quadratic programs without equality constraints.

In our experiments, the WW approach generated hypotheses with higher classification accuracy compared to the CS machine. Both approaches outperformed the one-versus-all method in this respect. Using S2DO, the original WW multi-class SVM now becomes at least as fast as the CS method trained with tailored, state-of-the-art second order working set selection. This indicates that the faster training times observed for the CS SVM compared to the WW formulation were not achieved by reducing the number of slack variables, but rather by dropping the bias term from the hypotheses (this is in accordance with the findings in [14], where training times increased drastically when adding bias parameters to the CS machine).

We derived a new generalization bound for multi-class SVMs based on performance guarantees for binary machines. It builds on a bound in terms of the fat shattering dimension. However, the proof technique can be transferred to other types of bounds relying on different complexity measures. The bound suggests that minimizing the sum of target margin violations is a proper learning strategy. As opposed to the CS algorithm, the WW machine directly performs regularized minimization of this quantity.

Given our empirical and theoretical results, we see no reason anymore for *a priori* preferring the CS SVM to the original method. We hope that our work makes the WW method more popular among practitioners, because it offers improved accuracy without additional costs in training time compared to CS.

From a theoretical point of view, the decisive property of the LLW multi-class SVM is the classification calibration of its loss function [28]. Our efficient solver makes LLW training practical and thereby allowed for the first extensive empirical comparison of

LLW with alternative multi-class SVMs.[5] The LLW method is the only classification calibrated machine in our comparison [28] and showed the best generalization performance. This improved accuracy required considerably more training time. However, if training time does not matter, the LLW machine is the multi-class SVM of choice. This experimental result corroborates the theoretical advantages of the LLW machine.

In this study, we considered batch learning of multi-class SVMs. For binary classification, it has been shown that improved second-order working set selection derived for batch learning is even more advantageous when applied to online learning in LASVM [12]. Therefore, we are confident that the results in this study also carry over to the popular LaRank online multi-class SVM [4].

References

[1] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical F oundations.* Cambridge University Press, 1999.

[2] A. Asuncion and D. J. Newman. UCI machine learning repository, 2007.

[3] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[4] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In Z. Ghahramani, editor, *Proceedings of the 24th International Machine Learning Conference (ICML)*, pages 89–96. OmniPress, 2007.

[5] L. Bottou and C. J. Lin. Support vector machine solvers. In L. Bottou et al., editors, *Large Scale Kernel Machines*, pages 1–28. MIT Press, 2007.

[6] E. J. Bredensteiner and K. P. Bennett. Multicategory classification by support vector machines. *Computational Optimization and Applications*, 12(1):53–79, 1999.

[7] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002.

[8] C. Demirkesen and H. Cherifi. A comparison of multiclass SVM methods for real world natural scenes. In J. Blanc-Talon and other, editors, *Advanced Concepts for Intelligent Vision Systems (ACIVS 2008)*, volume 5259 of *LNCS*, pages 763–763. Springer, 2008.

[9] R. E. Fan, P. Chen, and C. J. Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[5] Source code implementing all machines and solvers will be made available if the manuscript is accepted.

[10] S. García and F. Herrera. An extension on statistical "comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[11] T. Glasmachers and C. Igel. Maximum-gain working set selection for SVMs. *Journal of Machine Learning Research*, 7:1437–1466, 2006.

[12] T. Glasmachers and C. Igel. Second order SMO improves SVM online and active learning. *Neural Computation*, 20(2):374–382, 2008.

[13] Y. Guermeur. VC theory for large margin multi-category classifiers. *Journal of Machine Learning Research*, 8:2551–2594, 2007.

[14] C. W. Hsu and C. J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[15] C. Igel, T. Glasmachers, and V. Heidrich-Meisner. Shark. *Journal of Machine Learning Research*, 9:993–996, 2008.

[16] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf et al., editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 169–184. MIT Press, 1998.

[17] M. J. Kearns and R. E. Shapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48(3):464–497, 1994.

[18] S. Keerthi and E. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46(1):351–360, 2002.

[19] Y. Lee, Y. Lin, and G. Wahba. Multicategory Support Vector Machines: Theory and Application to the Classification of Microarray Data and Satellite Radiance Data. *Journal of the American Statistical Association*, 99(465):67–82, 2004.

[20] Y. Liu. Fisher consistency of multicategory support vector machines. In M. Meila and X. Shen, editors, *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 289–296, 2007.

[21] E. Osuna, R. Freund, and F. Girosi. Improved Training Algorithm for Support Vector Machines. In J. Principe et al., editors, *Neural Networks for Signal Processing VII*, pages 276–285. IEEE Press, 1997.

[22] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf et al., editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 185–208. MIT Press, 1998.

[23] T. Poggio, S. Mukherjee, R. Rifkin, A. Rakhlin, and A. Verri. b. In J. Winkler and M. Niranjan, editors, *Uncertainty in Geometric Computations*, chapter 11, pages 131–141. Kluwer Academic Publishers, 2002.

[24] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[25] J. Shawe-Taylor and N. Cristianini. Robust bounds on generalization from the margin distribution. NeuroCOLT2 Technical Report NC2-TR-1998-029, 1998.

[26] I. Steinwart. Support Vector Machines are Universally Consistent. *Journal of Complexity*, 18(3):768–791, 2002.

[27] I. Steinwart, D. Hush, and C. Scovel. Training SVMs Without Offset. *Journal of Machine Learning Research*, 12:141–202, 2011.

[28] A. Tewari and P. L. Bartlett. On the Consistency of Multiclass Classification Methods. *Journal of Machine Learning Research*, 8:1007–1025, 2007.

[29] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[30] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[31] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[32] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In M. Verleysen, editor, *Seventh European Symposium on Artificial Neural Networks (ESANN)*, pages 219–224, 1999.

A Analytic Solution of the S2DO Problem

Sections A.1 and A.2 show how to compute the gain and the update of the variables in S2DO in the absence of equality constraints. Note that the procedures differ considerably from the ones proposed in [9, 11, 5].

A.1 Gain Computation

Let the optimization problem be restricted to $\boldsymbol{\alpha}_B = (\alpha_a, \alpha_b)^{\mathrm{T}}$. Let the update of the current point $\boldsymbol{\alpha}_B$ be $\boldsymbol{\mu}_B^\star = (\mu_a^\star, \mu_b^\star)^T$.[6] We ask for the *unconstrained* optimum $\hat{\boldsymbol{\alpha}}_B = (\hat{\alpha_a}, \hat{\alpha_b})^{\mathrm{T}}$ of (5) and the corresponding gain $f(\hat{\boldsymbol{\alpha}}_B) - f(\boldsymbol{\alpha}_B)$. We write the second order Taylor expansion of (5) around $\hat{\boldsymbol{\alpha}}_B$ using $\boldsymbol{\mu}_B = \hat{\boldsymbol{\alpha}}_B - \boldsymbol{\alpha}_B$ as $f(\boldsymbol{\alpha}_B) = f(\hat{\boldsymbol{\alpha}}_B) + \boldsymbol{\mu}_B^{\mathrm{T}} \nabla f(\hat{\boldsymbol{\alpha}}_B) - \frac{1}{2}\boldsymbol{\mu}_B^{\mathrm{T}} \boldsymbol{Q}_B \boldsymbol{\mu}_B$, where the matrix $\boldsymbol{Q}_B \in \mathbb{R}^{2\times 2}$ is the restriction of $\boldsymbol{Q}$ to entries corresponding to the working set indices. At the optimal point the first order term vanishes and the gain is $f(\hat{\boldsymbol{\alpha}}_B) - f(\boldsymbol{\alpha}_B) = \frac{1}{2}\boldsymbol{\mu}_B^{\mathrm{T}} \boldsymbol{Q}_B \boldsymbol{\mu}_B$. The gradient $\hat{\boldsymbol{g}}_B = \boldsymbol{g}_B - \boldsymbol{Q}_B \boldsymbol{\mu}_B$

[6] In the binary case with bias, $\mu_a^\star$ must be equal to $\pm\mu_b^\star$ due to the corresponding equality constraint [5]. Without equality constraint this restriction can be dropped.

at $\hat{\boldsymbol{\alpha}}_B$ vanishes and a Newton step gives $\boldsymbol{\mu}_B = {\boldsymbol{Q}_B}^{-1}\boldsymbol{g}_B$. However, this computation assumes that the matrix $\boldsymbol{Q}_B$ can be inverted. If this is indeed true ($\det(\boldsymbol{Q}_B) > 0$), combining our results directly gives the gain

$$\frac{g_a^2 Q_{bb} - 2g_a g_b Q_{ab} + g_b^2 Q_{aa}}{2(Q_{aa}Q_{bb} - Q_{ab}^2)} \ .$$

In the case of $\det(\boldsymbol{Q}_B) = 0$ the calculation of the gain is more complicated. For $\boldsymbol{Q}_B = \boldsymbol{0}$ we have two cases. For $\boldsymbol{g}_B = \boldsymbol{0}$ we have a constant objective function and the gain is zero, and for $\boldsymbol{g}_B \neq 0$, we face a linear function with infinite gain. The case that $\boldsymbol{Q}_B$ is a rank one matrix remains. Let $\boldsymbol{q}_B$ be an eigenvector spanning the null-eigenspace. For ${\boldsymbol{g}_B}^{\mathrm{T}}\boldsymbol{q}_B \neq \boldsymbol{0}$ the gain is infinite and only if $\boldsymbol{g}_B$ and $\boldsymbol{q}_B$ are orthogonal the problem is reduced to a one dimensional quadratic equation. In this case the (non-unique) optimum can be computed as follows: Let $\boldsymbol{w}_B$ be a nonzero vector orthogonal to $\boldsymbol{q}_B$ or in other words an eigenvector corresponding to the non-null eigenspace of $\boldsymbol{Q}_B$. Then the point $\hat{\boldsymbol{\alpha}}_B = \boldsymbol{\alpha}_B + (\boldsymbol{w}_B^{\mathrm{T}}\boldsymbol{g}_B)/(\boldsymbol{w}_B^{\mathrm{T}}\boldsymbol{Q}_B\boldsymbol{w}_B)$ is optimal and the corresponding gain is $((\boldsymbol{w}_B^{\mathrm{T}}\boldsymbol{g}_B)^2)/(2\boldsymbol{w}_B^{\mathrm{T}}\boldsymbol{Q}_B\boldsymbol{w}_B)$. The vectors $\boldsymbol{g}_B$ and $\boldsymbol{w}_B$ are aligned in this case ($\boldsymbol{g}_B = \lambda\boldsymbol{w}_B$ for some $\lambda \in \mathbb{R}$), such that $\boldsymbol{g}_B$ can directly take the role of $\boldsymbol{w}_B$, resulting in $((g_a^2 + g_b^2)^2)/(2(g_a^2 Q_{aa} + 2g_a g_b Q_{ab} + g_b^2 Q_{bb}))$.

For normalized kernels we have $Q_{aa} = Q_{bb} = 1$ and so the case $\boldsymbol{Q}_B = \boldsymbol{0}$ is impossible and $\det(\boldsymbol{Q}_B) = 0$ amounts to the two cases $Q_{ab} \in \{\pm 1\}$, resulting in $\boldsymbol{q}_B = (-Q_{ab}, 1)^T$ and $\boldsymbol{w}_B = (1, Q_{ab})^T$. For $\boldsymbol{g}_B^T\boldsymbol{q}_B = g_b - Q_{ab}g_a = 0$ the gain is given by $(g_a^2 + Q_{ab}g_b)^2/8$.

A.2 Parameter Update

In the case of S2DO, the update of the $\boldsymbol{\alpha}$ vector is non-trivial and differs considerably from the standard updates described in [9, 11, 5]. In the following, we present the solution of the quadratic subproblem in the case of working sets of size $|B| = 2$. This derivation was first provided by [27] for the special case of normalized kernels (i.e., $Q_{ii} = 1$ for all $i \in \{1, \ldots, m\}$), and $\boldsymbol{v} = \boldsymbol{1}$. Here, we outline the solution for the general case.

Let $B = \{i, j\}$. We consider the subproblem

$$\begin{aligned} \max \quad & f(\boldsymbol{\alpha_B} + \boldsymbol{\mu}_B) = \boldsymbol{g}_B^T\boldsymbol{\mu}_B - \frac{1}{2}\boldsymbol{\mu}_B^T\boldsymbol{Q}\boldsymbol{\mu}_B + \text{const} \\ \text{s.t.} \quad & \boldsymbol{\alpha}_B + \boldsymbol{\mu}_B \in \mathcal{F} = [L_i, U_i] \times [L_j, U_j] \ . \end{aligned}$$

Setting the partial derivatives w.r.t. μ_i and μ_j to zero gives $g_i = Q_{ii}\mu_i + Q_{ij}\mu_j$ and $g_j = Q_{ij}\mu_i + Q_{jj}\mu_j$.

In the sequel we will solve a number of one-dimensional sub-problems where one of the variables μ_i or μ_j is fixed to one of its bounds. W.l.o.g. assume that $\alpha_i + \mu_i^\star = L_i$. Then the optimal solution is given by

$$\mu_j^\star = \min\left\{\max\left\{\frac{g_j - Q_{ij}\mu_i^\star}{Q_{jj}}, L_j - \alpha_j\right\}, U_j - \alpha_j\right\} \ .$$

We distinguish three different cases according to the rank of $\boldsymbol{Q}_B$: For $\boldsymbol{Q}_B = \mathbf{0}$ the solution is found by following the gradient, i.e., $\mu_i^\star = U_i - \alpha_i$ for $g_i > 0$, $\mu_i^\star = L_i - \alpha_i$ for $g_i < 0$, and $\mu_i^\star = 0$ for $g_i = 0$; with analogous rules for $\mu_j^\star$.

Now assume that $\boldsymbol{Q}_B$ has rank one. Then the objective function is linear on each line segment $S_{\boldsymbol{p}} = \{\boldsymbol{p} + \lambda \cdot (-Q_{ij}, Q_{ii})^T \mid \lambda \in \mathbb{R}\} \cap \mathcal{F}$, $\boldsymbol{p} \in \mathcal{F}$, with derivative $\gamma = \partial f / \partial \lambda = Q_{ii} g_j - Q_{ij} g_i$ in parameter direction. For $\gamma \geq 0$ the optimum is obtained on one of the line segments at the maximal parameter value. These points cover either one or two adjacent edges of the parameter rectangle $[L_i, U_i] \times [L_j, U_j]$, depending on the signs of Q_{ii} and Q_{ij}. We solve the one-dimensional sub-problem for each of the edges involved. The best solution obtained from the one-dimensional sub-problems is the optimum $\boldsymbol{\mu}_B^\star$. The case $\gamma < 0$ is handled analogously with the opposite edge(s).

If $\boldsymbol{Q}_B$ has full rank then we compute the unconstrained optimum

$$\hat{\boldsymbol{\mu}}_B = \boldsymbol{Q}_B^{-1} \boldsymbol{g}_B = \frac{1}{\det(\boldsymbol{Q}_B)} \cdot \begin{pmatrix} Q_{jj} g_i - Q_{ij} g_j \\ Q_{ii} g_j - Q_{ij} g_i \end{pmatrix} .$$

If this solution is feasible we have $\boldsymbol{\mu}_B^\star = \hat{\boldsymbol{\mu}}_B$. Otherwise first assume that only one of the variables $\hat{\mu}_i$ and $\hat{\mu}_j$ is outside the bounds; w.l.o.g. assume $\alpha_i + \hat{\mu}_i > U_i$. Then by convexity we conclude that the optimum is found on the edge $\{U_i\} \times [L_j, U_j]$, which amounts to a one-dimensional problem. In case that both variables violate the constraints, w.l.o.g. $\alpha_i + \hat{\mu}_i < L_i$ and $\alpha_j + \hat{\mu}_j > U_j$, the same convexity argument ensures that the optimum is located on one of the adjacent edges $\{L_i\} \times [L_j, U_j]$ and $[L_i, U_i] \times \{U_j\}$. As above, the better solution of the two one-dimensional problems constitutes the optimum.

Table 3: Training times and numbers of iterations for all multi-class SVMs using decomposition algorithms with minimal working set size (SMO) and working set size two (S2DO), as well as the baseline OVA approach. The training times in brackets are given in seconds along with the number of iterations of the decomposition algorithms needed by the all-in-one SVMs.

	LLW		**WW**		**CS**	**OVA**
	SMO	**S2DO**	**SMO**	**S2DO**	**S2DO/SMO**	**SMO**
Abalone	122,853 (671)	52,501 (493)	92,705 (361)	40,514 (319)	118,368 (44)	(60)
Car	130,199 (9.86)	31,360 (6.37)	15,309 (0.85)	2,973 (0.73)	17,408 (2.21)	(0.19)
Glass	38,030 (1.08)	5475 (0.48)	742 (0.048)	372 (0.037)	2,473 (0.19)	(0.18)
Iris	21,145 (0.065)	1697 (0.049)	146,387 (0.153)	554 (0.022)	21,362 (0.107)	(0.035)
OptDigits	529,247 (532)	195,362 (520)	24,102 (59)	10,419 (77)	16,264 (46)	(6.4)
Page Blocks	$693 \cdot 10^6$ (34,078)	$381 \cdot 10^6$ (25,259)	1,037,684 (30)	93,251 (11)	$16 \cdot 10^6$ (4,438)	(252)
Sat	219,895 (191)	95,643 (276)	59,495 (105)	22,001 (96)	29,753 (109)	(15)
Segment	$55 \cdot 10^6$ (6,507)	$19 \cdot 10^6$ (5,161)	206,149 (9.4)	17,782 (4.7)	$13 \cdot 10^6$ (448)	(1.0)
SoyBean	728,480 (66.3)	214,096 (51.8)	7,073 (0.56)	1,627 (0.49)	9,657 (1.71)	(0.02)
Vehicle	$17 \cdot 10^6$ (566)	1,743,176 (163)	1,391,588 (18.1)	203,840 (6.3)	1,637,912 (26.4)	(0.9)