Journal of Machine Learning Research 7 (2006) 1437–1466 Submitted 9/05; Revised 10/05; Published 6/06

Maximum-Gain Working Set Selection for SVMs

Tobias Glasmachers TOBIAS.GLASMACHERS@NEUROINFORMATIK.RUB.DE
Christian Igel CHRISTIAN.IGEL@NEUROINFORMATIK.RUB.DE
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany

Editors: Kristin P. Bennett and Emilio Parrado-Hernández

Abstract

Support vector machines are trained by solving constrained quadratic optimization problems. This is usually done with an iterative decomposition algorithm operating on a small working set of variables in every iteration. The training time strongly depends on the selection of these variables. We propose the maximum-gain working set selection algorithm for large scale quadratic programming. It is based on the idea to greedily maximize the progress in each single iteration. The algorithm takes second order information from cached kernel matrix entries into account. We prove the convergence to an optimal solution of a variant termed hybrid maximum-gain working set selection. This method is empirically compared to the prominent most violating pair selection and the latest algorithm using second order information. For large training sets our new selection scheme is significantly faster.

Keywords: working set selection, sequential minimal optimization, quadratic programming, support vector machines, large scale optimization

1. Introduction

We consider 1-norm support vector machines (SVM) for classification. These classifiers are usually trained by solving convex quadratic problems with linear constraints. For large data sets, this is typically done with an iterative decomposition algorithm operating on a small working set of variables in every iteration. The selection of these variables is crucial for the training time.

Recently, a very efficient SMO-like (*sequential minimal optimization* using working sets of size 2, see Platt, 1999) decomposition algorithm was presented by Fan et al. (2005). The main idea is to consider second order information to improve the working set selection. Independent from this approach, we have developed a working set selection strategy sharing this basic idea but with a different focus, namely to minimize the number of kernel evaluations per iteration. This considerably reduces the training time of SVMs in case of large training data sets. In the following, we present our approach, analyze its convergence properties, and present experiments evaluating the performance of our algorithm. We close with a summarizing conclusion.

1.1 Support Vector Machine Learning

We consider 1-norm soft margin SVMs for classification (Vapnik, 1998; Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002). The learning problem at hand is defined by a set of ℓ training examples $\{(x_1,y_1),\ldots,(x_\ell,y_\ell)\}$, where the x_i are points in some input space $\mathcal{X}$ with corre-

sponding class labels $y_i = \pm 1$. A positive semi-definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ ensures the existence of a feature Hilbert space $\mathcal{F}$ with inner product $\langle \cdot, \cdot \rangle$ and a mapping $\Phi : \mathcal{X} \to \mathcal{F}$ such that $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$.

The SVM algorithm constructs a real-valued, affine linear function H on the feature space. The corresponding function $h := H \circ \Phi$ on the input space can be computed in terms of the kernel k without the need for explicit computations in $\mathcal{F}$. The zero set of H is called the separating hyperplane, because the SVM uses the sign of this function for class prediction. This affine linear function is defined through maximizing the margin, that is, the desired distance of correctly classified training patterns from the hyperplane, and reducing the sum of distances by which training examples violate this margin. The trade-off between these two objectives is controlled by a regularization parameter $C > 0$.

Training a 1-norm soft margin SVM is equivalent to solving the following ℓ-dimensional convex quadratic problem with linear constraints for $\alpha \in \mathbb{R}^\ell$:

$$\mathcal{P} \quad \left\{ \begin{array}{ll} \text{maximize} & f(\alpha) = v^T \alpha - \frac{1}{2} \alpha^T Q \alpha \\ \text{subject to} & y^T \alpha = z \\ \text{and} & 0 \leq \alpha_i \leq C \ , \ \forall i \in \{1, \ldots, \ell\} \ . \end{array} \right.$$

The requirements $y^T \alpha = z$ and $0 \leq \alpha_i \leq C$ are referred to as equality constraint and box constraints, respectively. In the SVM context the constants $v \in \mathbb{R}^\ell$ and $z \in \mathbb{R}$ are fixed to $v = (1, \ldots, 1)^T$ and $z = 0$. The matrix $Q \in \mathbb{R}^{\ell \times \ell}$ is defined as $Q_{ij} := y_i y_j k(x_i, x_j)$ and is positive semi-definite as the considered kernel function k is positive semi-definite. The vector $y := (y_1, \ldots, y_\ell)^T$, $y_i \in \{+1, -1\}$ for $1 \leq i \leq \ell$, is composed of the labels of the training patterns $x_1, \ldots, x_\ell$. The set of points α fulfilling the constraints is called the feasible region $\mathcal{R}(\mathcal{P})$ of problem $\mathcal{P}$.

An optimal solution α^* of this problem defines the function $h(x) = \sum_{i=1}^{\ell} \alpha_i^* y_i k(x_i, x) + b$, where the scalar b can be derived from α^* (e.g., see Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002).

1.2 Decomposition Algorithms

Making SVM classification applicable in case of large training data sets requires an algorithm for the solution of $\mathcal{P}$ that does not presuppose the $\ell(\ell+1)/2$ independent entries of the symmetric matrix Q to fit into working memory. The methods of choice in this situation are the so called decomposition algorithms (Osuna et al., 1997). These iterative algorithms start at an arbitrary feasible point $\alpha^{(0)}$ and improve this solution in every iteration t from $\alpha^{(t-1)}$ to $\alpha^{(t)}$ until some stopping condition is satisfied. In each iteration an active set or working set $B^{(t)} \subset \{1, \ldots, \ell\}$ is chosen. Its inactive complement is denoted by $N^{(t)} := \{1, \ldots, \ell\} \setminus B^{(t)}$. The improved solution $\alpha^{(t)}$ may differ from $\alpha^{(t-1)}$ only in the components in the working set, that is, $\alpha_i^{(t-1)} = \alpha_i^{(t)}$ for all $i \in N^{(t)}$. Usually the working set $B^{(t)}$ is limited to a fixed size $|B^{(t)}| \leq q \ll \ell$. The working set must always be larger than the number of equality constraints to allow for a useful, feasible step. In general a decomposition algorithm can be formulated as follows:

Decomposition Algorithm

1 $\alpha^{(0)} \leftarrow$ feasible starting point, $t \leftarrow 1$
2 **repeat**
3 | select working set $B^{(t)}$
4 | solve QP restricted to $B^{(t)}$ resulting in $\alpha^{(t)}$
5 | $t \leftarrow t+1$
6 **until** *stopping criterion is met*

The sub-problem defined by the working set in step 4 has the same structure as the full problem $\mathcal{P}$ but with only q variables.[1] Thus, the complete problem description fits into the available working memory and is small enough to be solved by standard tools.

For the SVM problem $\mathcal{P}$ the working set must have a size of at least two. Indeed, the sequential minimal optimization (SMO) algorithm selecting working sets of size $q = 2$ is a very efficient method (Platt, 1999). The great advantage of the SMO algorithm is the possibility to solve the sub-problem analytically (cf. Platt, 1999; Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2002).

1.3 Working Set Selection

Step 3 is crucial as the convergence of the decomposition algorithm depends strongly on the working set selection procedure. As the selection of the working set of a given size q that gives the largest improvement in a single iteration requires the knowledge of the full matrix Q, well working heuristics for choosing the variables using less information are needed.

There exist various algorithms for this task, an overview is given in the book by Schölkopf and Smola (2002). The most prominent ones share the strategy to select pairs of variables that mostly violate the Karush-Kuhn-Tucker (KKT) conditions for optimality and can be subsumed under the *most violating pair* (MVP) approach. Popular SVM packages such as SVM^{light} by Joachims (1999) and LIBSVM 2.71 by Chang and Lin (2001) implement this technique. The idea of MVP is to select one or more pairs of variables that allow for a feasible step and most strongly violate the KKT conditions.

Here we describe the approach implemented in the LIBSVM 2.71 package. Following Keerthi and Gilbert (2002) we define the sets

$$I := \{i \in \{1,\dots,\ell\} \,|\, y_i = +1 \wedge \alpha_i^{(t-1)} < C\} \cup \{i \in \{1,\dots,\ell\} \,|\, y_i = -1 \wedge \alpha_i^{(t-1)} > 0\}$$
$$J := \{i \in \{1,\dots,\ell\} \,|\, y_i = +1 \wedge \alpha_i^{(t-1)} > 0\} \cup \{i \in \{1,\dots,\ell\} \,|\, y_i = -1 \wedge \alpha_i^{(t-1)} < C\} \ .$$

Now the MVP algorithm selects the working set $B^{(t)} = \{b_1, b_2\}$ using the rule

$$b_1 := \operatorname*{argmax}_{i \in I} \left(y_i \frac{\partial f}{\partial \alpha_i}(\alpha) \right)$$
$$b_2 := \operatorname*{argmin}_{i \in J} \left(y_i \frac{\partial f}{\partial \alpha_i}(\alpha) \right) \ .$$

1. For a sub-problem the constants $v \in \mathbb{R}^q$ and $z \in \mathbb{R}$ in general differ from $(1,\dots,1)^T$ and 0, respectively, and depend on $\alpha_i^{(t-1)}$ for $i \in N^{(t)}$.

The condition $y_{b_1}\frac{\partial f}{\partial \alpha_{b_1}}(\alpha) - y_{b_2}\frac{\partial f}{\partial \alpha_{b_2}}(\alpha) < \varepsilon$ is used as the stopping criterion. In the limit $\varepsilon \to 0$ the algorithm checks the exact KKT conditions and only stops if the solution found is optimal. The MVP algorithm is known to converge to the optimum (Lin, 2001; Keerthi and Gilbert, 2002; Takahashi and Nishi, 2005).

SVM^{light} uses essentially the same working set selection method with the important difference that it is not restricted to working sets of size 2. The default algorithm selects 10 variables by picking the five most violating pairs. In each iteration an inner optimization loop determines the solution on the 10-dimensional sub-problem up to some accuracy.

Fan et al. (2005) propose a working set selection procedure which uses second order information. The first variable b_1 is selected as in the MVP algorithm. The second variable is chosen in a way that promises the maximal value of the target function f ignoring the box constraints. The selection rule is

$$b_2 := \operatorname*{argmax}_{i \in J} \left(f(\alpha^{\max}_{\{b_1,i\}}) \right) \quad .$$

Here, $\alpha^{\max}_{\{b_1,i\}}$ is the solution of the two-dimensional sub-problem defined by the working set $\{b_1, i\}$ at position $\alpha^{(t-1)}$ considering only the equality constraint. For this second order algorithm costly kernel function evaluations may become necessary which can slow down the entire algorithm. These kernel values are cached and can be reused in the gradient update step, see equation (2) in Section 2.1. Because this algorithm is implemented in version 2.8 of LIBSVM, we will refer to it as the LIBSVM-2.8 algorithm.

The simplest feasible point one can construct is $\alpha^{(0)} = (0,\ldots,0)^T$, which has the additional advantage that the gradient $\nabla f(\alpha^{(0)}) = v = (1,\ldots,1)^T$ can be computed without kernel evaluations. It is interesting to note that in the first iteration starting from this point all components of the gradient $\nabla f(\alpha^{(0)})$ of the objective function are equal. Thus, the selection schemes presented above have a freedom of choice for the selection of the first working set. In case of LIBSVM, for b_1 simply the variable with maximum index is chosen in the beginning. Therefore, the order in which the training examples are presented is important in the first iteration and can indeed significantly influence the number of iterations and the training time in practice.

Other algorithms select a *rate certifying pair* (Hush and Scovel, 2003). The allurement of this approach results from the fact that analytical results have been derived not only about the guaranteed convergence of the algorithm, but even about the rate of convergence (Hush and Scovel, 2003; List and Simon, 2005). Unfortunately, in practice these algorithms seem to perform rather poorly.

1.4 Related Methods

Several new training algorithms for SVMs have been developed recently, which could be considered for large scale data sets. One possibility to reduce the computational cost for huge data sets is to determine only rough approximate solutions of the SVM problem. Algorithms emerged from this line of research include the Core Vector Machine by Tsang et al. (2005) and LASVM by Bordes et al. (2005), which have the additional advantage to produce even sparser solutions than the exact SVM formulation. In theory, both methods can solve the exact SVM problem with arbitrary accuracy, but their strength lies in the very fast computation of relatively rough approximate solutions. Both methods can profit from our working set selection algorithm presented below, as the Core Vector Machine uses an inner SMO loop and LASVM is basically an online version of SMO.

The SimpleSVM algorithm developed by Vishwanathan et al. (2003) provides an alternative to the decomposition technique. It can handle a wide variety of SVM formulations, but is limited in the large scale context by its extensive memory requirements. In contrast, Keerthi et al. (2000) present a geometrically inspired algorithm with modest memory requirements for the exact solution of the SVM problem. A drawback of this approach is that it is not applicable to the standard one-norm slack penalty soft margin SVM formulation, which we consider here, because it requires the classes to be linearly separable in the feature space $\mathcal{F}$.

2. Maximum-Gain Working Set Selection

Before we describe our new working set selection method, we recall how the quadratic problem restricted to a working set can be solved (cf. Platt, 1999; Cristianini and Shawe-Taylor, 2000; Chang and Lin, 2001). Then we compute the progress, the functional gain, that is achieved by solving a single sub-problem. Picking the variable pair maximizing the functional gain while minimizing kernel evaluations—by reducing cache misses when looking up rows of Q—leads to the new working set selection strategy.

2.1 Solving the Problem Restricted to the Working Set

In every iteration of the decomposition algorithm all variables indexed by the inactive set N are fixed and the problem $\mathcal{P}$ is restricted to the variables indexed by the working set $B = \{b_1, \ldots, b_q\}$. We define

$$\alpha_B = (\alpha_{b_1}, \ldots, \alpha_{b_q})^T \quad , \qquad Q_B = \begin{pmatrix} Q_{b_1 b_1} & \cdots & Q_{b_q b_1} \\ \vdots & \ddots & \vdots \\ Q_{b_1 b_q} & \cdots & Q_{b_q b_q} \end{pmatrix} \quad , \qquad y_B = (y_{b_1}, \ldots, y_{b_q})^T$$

and fix the values

$$v_B = \left(1 - \sum_{i \in N} Q_{i b_1} \alpha_i, \quad \ldots \quad , 1 - \sum_{i \in N} Q_{i b_q} \alpha_i\right)^T \in \mathbb{R}^q \quad \text{and} \quad z_B = -\sum_{i \in N} y_i \alpha_i \in \mathbb{R}$$

not depending on α_B. This results in the convex quadratic problem (see Joachims, 1999)

$$\mathcal{P}_{B,\alpha} \qquad \begin{cases} \text{maximize} & f_B(\alpha_B) = v_B^T \alpha_B - \frac{1}{2} \alpha_B^T Q_B \alpha_B \\ \text{subject to} & y_B^T \alpha_B = z_B \\ \text{and} & 0 \le \alpha_i \le C \quad \forall i \in B \ . \end{cases}$$

The value z_B can easily be determined in time linear in q, but the computation of v_B takes time linear in q and ℓ. Then $\mathcal{P}_{B,\alpha}$ can be solved using a ready-made quadratic program solver in time independent of ℓ.

We will deal with this problem under the assumption that we know the gradient vector

$$G := \nabla f_B(\alpha_B) = \left(\frac{\partial}{\partial \alpha_{b_1}} f_B(\alpha_B), \ldots, \frac{\partial}{\partial \alpha_{b_q}} f_B(\alpha_B)\right)^T \tag{1}$$

of partial derivatives of f_B with respect to all q variables $\alpha_{b_1}, \ldots, \alpha_{b_q}$ indexed by the working set. In the following, we consider SMO-like algorithms using working sets of size $q = 2$. In this case

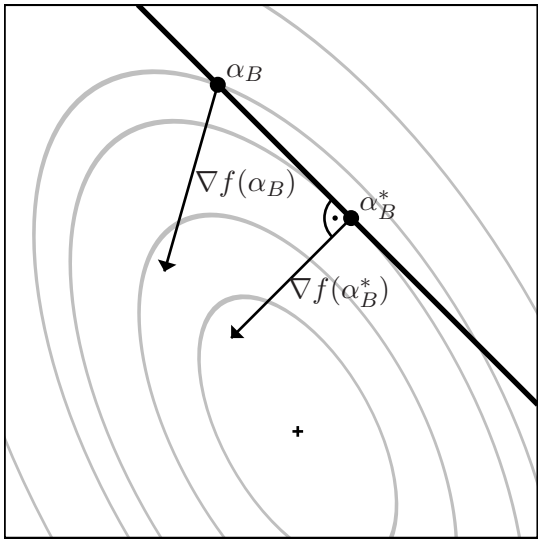

Figure 1: The 2-dimensional SMO sub-problem restricted to the equality constraint (solid 'feasible' line) and the box constraints (boundary). The point fulfilling the equality constraint with gradient orthogonal to the feasible line is a candidate for the solution of the sub-problem. If it is not feasible w.r.t. the box constraints it has to be moved along the line onto the box boundary.

the equality constraint restricts us to a line. Due to the box constraints only a bounded segment of this line is feasible (see Figure 1). To solve the restricted problem we define the vector $w_B := (1, -y_{b_1} y_{b_2})^T$ pointing along the 1-dimensional feasible hyperplane. To find the maximum on this line we look at the gradient $\nabla f_B(\alpha_B) = v_B - Q_B \alpha_B$ and compute the step $\mu_B \cdot w_B$ ($\mu_B \in \mathbb{R}$) such that the gradient $\nabla f_B(\alpha_B + \mu_B \cdot w_B)$ is orthogonal to w_B,

$$\begin{aligned} 0 &= \langle \nabla f_B(\alpha_B + \mu_B w_B), w_B \rangle \\ &= \langle v_B - Q_B \alpha_B - \mu_B Q_B w_B, w_B \rangle \\ &= \langle \nabla f_B(\alpha_B) - \mu_B Q_B w_B, w_B \rangle \ . \end{aligned}$$

Using $\nabla f_B(\alpha_B) = (G_{b_1}, G_{b_2})^T$ we get the solution

$$\mu_B^{\max} = (G_{b_1} - y_{b_1} y_{b_2} G_{b_2}) / (Q_{b_1 b_1} + Q_{b_2 b_2} - 2 y_{b_1} y_{b_2} Q_{b_1 b_2}) \ .$$

The corresponding point on the feasible line is denoted by $\alpha_B^{\max} = \alpha_B + \mu_B^{\max} w_B$. Of course, $\alpha_B^{\max}$ is not necessarily feasible. We can easily apply the box constraints to $\mu_B^{\max}$. The new solution clipped to the feasible line segment is denoted μ_B^*. The maximum of $\mathcal{P}_{B,\alpha}$ can now simply be expressed as $\alpha_B^* = \alpha_B + \mu_B^* w_B$.

After the solution of the restricted problem the new gradient

$$\nabla f(\alpha^{(t)}) = \nabla f(\alpha^{(t-1)}) - Q(\alpha^{(t)} - \alpha^{(t-1)}) \tag{2}$$

has to be computed. As the formula indicates this is done by an update of the former gradient. Because $\Delta\alpha = \alpha^{(t)} - \alpha^{(t-1)}$ differs from zero in only the b_1th and b_2th component only the corresponding two matrix rows of Q have to be known to determine the update.

2.2 Computing the Functional Gain

Expressing the target function on the feasible line by its Taylor expansion in the maximum $\alpha_B^{\max}$ we get

$$\begin{aligned}\tilde{f}_B(\xi) &:= f_B(\alpha_B^{\max} + \xi w_B) \\ &= f_B(\alpha_B^{\max}) - \frac{1}{2}(\xi w_B)^T Q_B(\xi w_B) \\ &= f_B(\alpha_B^{\max}) - \left(\frac{1}{2} w_B^T Q_B w_B\right)\xi^2 \quad .\end{aligned}$$

Now it is possible to calculate the gain as

$$\begin{aligned}f_B(\alpha_B^*) - f_B(\alpha_B) &= \tilde{f}_B(\mu_B^* - \mu_B^{\max}) - \tilde{f}_B(0 - \mu_B^{\max}) \\ &= \left(\frac{1}{2} w_B^T Q_B w_B\right)((\mu_B^{\max})^2 - (\mu_B^* - \mu_B^{\max})^2) \\ &= \left(\frac{1}{2} w_B^T Q_B w_B\right)(\mu_B^*(2\mu_B^{\max} - \mu_B^*)) \\ &= \frac{1}{2}(Q_{b_1b_1} + Q_{b_2b_2} - 2y_{b_1}y_{b_2}Q_{b_1b_2})(\mu_B^*(2\mu_B^{\max} - \mu_B^*)) \quad . \qquad (3)\end{aligned}$$

The diagonal matrix entries Q_{ii} needed for the calculation can be precomputed before the decomposition loop starts using time and memory linear in ℓ. Thus, knowing only the derivatives (1), C, y_B, and $Q_{b_1b_2}$ (and the precomputed diagonal entries) makes it possible to compute the gain in f. Usually in an SVM implementation the derivatives are already at hand because they are required for the optimality test in the stopping criterion. Of course we have access to the labels and the regularization parameter C. The only remaining quantity needed is $Q_{b_1b_2}$, which unfortunately requires evaluating the kernel function.

2.3 Maximum-Gain Working Set Selection

Now, a straightforward working set selection strategy is to look at all $\ell(\ell-1)/2$ possible variable pairs, to evaluate the gain (3) for every one of them, and to select the best, that is, the one with maximum gain. It can be expected that this greedy selection policy leads to very fast convergence measured in number of iterations needed. However, it has two major drawbacks making it advisable only for very small problems: looking at all possible pairs requires the knowledge of the complete matrix Q. As Q is in general too big to fit into the working memory, expensive kernel function evaluations become necessary. Further, the evaluation of all possible pairs scales quadratically with the number of training examples.

Fortunately, modern SVM implementations use a considerable amount of working memory as a cache for the rows of Q computed in recent iterations. In all cases, this cache contains the two rows corresponding to the working set chosen in the most recent iteration, because they were needed for

the gradient update (2). This fact leads to the following maximum-gain working pair selection (MG) algorithm:

Maximum-Gain Working Set Selection in step t

if $t = 1$ **then**
  select arbitrary working set $B^{(1)} = \{b_1, b_2\}, y_{b_1} \neq y_{b_2}$
else
  select pair $B^{(t)} \leftarrow \operatorname{argmax}_{B=\{b_1,b_2\} \mid b_1 \in B^{(t-1)}, b_2 \in \{1,\ldots,\ell\}} g_B(\alpha)$

In the first iteration, usually no cached matrix rows are available. Thus, an arbitrary working set $B^{(1)} = \{b_1, b_2\}$ fulfilling $y_{b_1} \neq y_{b_2}$ is chosen. In all following iterations, given the previous working set $B^{(t-1)} = \{b_1, b_2\}$, the gain of all combinations $\{b_1, b\}$ and $\{b_2, b\}$ ($b \in \{1, \ldots, \ell\}$) is evaluated and the best one is selected.

The complexity of the working set selection is linear in the number of training examples. It is important to note that the algorithm uses second order information from the matrix cache. These information are ignored by all existing working set selection strategies, albeit they are available for free, that is, without spending any additional computational effort. This situation is comparable to the improvement of using the gradient for the analytical solution of the sub-problem in the SMO algorithm. Although the algorithm by Fan et al. (2005) considers second order information, these are in general not available from the matrix cache.

The maximum gain working pair selection can immediately be generalized to the class of *maximum-gain working set selection* algorithms (see Section 2.5). Under this term we want to subsume all working set selection strategies choosing variables according to a greedy policy with respect to the functional gain computed using cached matrix rows. In the following, we restrict ourselves to the selection of pairs of variables as working sets.

In some SVM implementations, such as LIBSVM, the computation of the stopping condition is done using information provided during the working set selection. LIBSVM's MVP algorithm stops if the sum of the violations of the pair is less than a predefined constant ε. The simplest way to implement a roughly comparable stopping condition in MG is to stop if the value μ_B^* defining the length of the constrained step is smaller than ε.

It is worth noting that the MG algorithm does not depend on the caching strategy. The only requirement for the algorithm to efficiently profit from the kernel cache is that the cache always contains the two rows of the matrix Q that correspond to the previous working set. This should be fulfilled by every efficient caching algorithm, because recently active variables have a high probability to be in the working set again in future iterations. That is, the MG algorithm does not require a change of the caching strategy. Instead, it improves the suitability of all caching strategies that at least store the information most recently used.

2.4 Hybrid Maximum-Gain Working Set Selection

The MG algorithm can be used in combination with other methods. In order to inherit the convergence properties from MVP we introduce the *hybrid maximum gain* (HMG) working set selection algorithm. The algorithm is defined as follows:

Hybrid Maximum-Gain Working Set Selection in step t, $0 < \eta \ll 1$

```
1 if t = 1 then
2     select arbitrary working set B^(1) = {b_1, b_2}, y_{b_1} ≠ y_{b_2}
3 else
4     if ∀i ∈ B^(t-1) : α_i ≤ η·C ∨ α_i ≥ (1-η)·C then
5         select B^(t) according to MVP
6     else
7         select pair B^(t) ← argmax_{B={b_1,b_2} | b_1∈B^(t-1), b_2∈{1,...,ℓ}} g_B(α)
```

In the first iteration, usually no cached matrix rows are available. Thus, an arbitrary working set $\{b_1, b_2\}$ fulfilling $y_{b_1} \neq y_{b_2}$ is selected. If in iteration $t > 1$ both variables indexed by the previous working set $B^{(t-1)} = \{b_1, b_2\}$ are no more than $\eta \cdot C$, $0 < \eta \ll 1$, from the bounds, then the MVP algorithm is used. Otherwise the working set is selected according to MG. Figure 2 illustrates the HMG decision rule. The stopping condition tested by the decomposition algorithm is the one

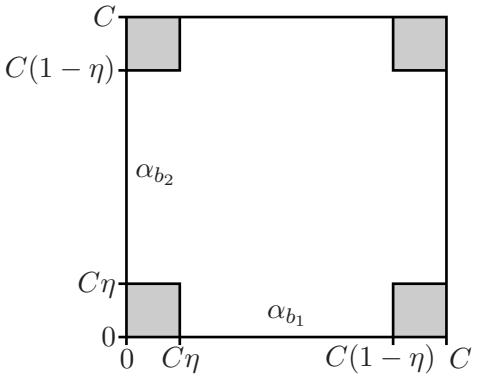

Figure 2: Illustration of the HMG algorithm. The plain defined by the previous working set $B^{(t-1)} = \{b_1, b_2\}$ is drawn. If the algorithm ended up in one of the gray corners then the MVP algorithm is used in iteration t.

from the working set selection algorithm used in the current iteration. That is, the decomposition algorithm stops if $y_{b_1} \frac{\partial f}{\partial \alpha_{b_1}}(\alpha) - y_{b_2} \frac{\partial f}{\partial \alpha_{b_2}}(\alpha)$ or μ_B^* falls below the threshold ε depending on whether MVP or MG has been selected.

The HMG algorithm is a combination of MG and MVP using MVP only in special situations. In our experiments, we set $\eta = 10^{-8}$. This choice is arbitrary and makes no difference to $\eta = 0$ in nearly all cases. In practice, in almost all iterations MG will be active. Thus, this algorithm inherits the speed of the MG algorithm. It is important to note that η is not a parameter influencing the convergence speed (as long as the parameter is small) and is therefore not subject to tuning.

The technical modification ensures the convergence of the algorithm. This is formally expressed by the following theorem.

Theorem 1 *We consider problem $\mathcal{P}$. Let $(\alpha^{(t)})_{t\in\mathbb{N}}$ be a sequence of feasible points produced by the decomposition algorithm using the HMG policy. Then, the limit point of every convergent subsequence is optimal for $\mathcal{P}$.*

The proof can be found in Section 3.

2.5 Generalization of the Algorithm

In the following, we discuss some of the potential variants of the basic MG or HMG algorithm.

It is possible to use a larger set of precomputed rows, say, 10, for the working set selection. In the extreme case we can run through all cached rows of Q. Then the working set selection algorithm becomes quite time consuming in comparison to the gradient update. As the number of iterations does not decrease accordingly, as we observed in real world applications, we recommend to use only the two rows of the matrix from the previous working set. We refer to Section 4.6 for a comparison.

A small change speeding up the working set selection is fixing one element of the working set in every iteration. When alternating the fixed position, every element is used two times successively. Only $\ell - 2$ pairs have to be evaluated in every iteration. Though leading to more iterations this policy can speed up the MG algorithm for small problems (see Section 4.6).

The algorithm can be extended to compute the gain for tuples of size $q > 2$. It is a severe disadvantage that such sub-problems can not be solved analytically and an iterative solver has to be used for the solution of the sub-problem. Note that this becomes necessary also for every gain computation during the working set selection. To keep the complexity of the working set selection linear in ℓ only one element new to the working set can be evaluated. Due to this limitation this method becomes even more cache friendly. The enlarged working set size may decrease the number of iterations required, but at the cost of the usage of an iterative solver. This should increase the speed of the SVM algorithm only on large problems with extremely complicated kernels, where the kernel matrix does not fit into the cache and the kernel evaluations in every iteration take much longer than the working set selection.

3. Convergence of the Algorithms

In this section we discuss the convergence properties of the decomposition algorithm using MG and HMG working set selection. First, we give basic definitions and prove a geometrical criterion for optimality. Then, as a motivation and a merely theoretical result, we show some properties of the gain function and prove that the greedy strategy w.r.t. the gain converges to an optimum. Returning to our algorithm, we give a counter example proving that there exist scenarios where pure MG looking at pairs of variables may stop after finitely many iterations without reaching an optimum. Finally, we prove that HMG converges to an optimum.

3.1 Prerequisites

In our convergence analysis, we consider the limit $\varepsilon \to 0$, that is, the algorithms only stop if the quantities checked in the stopping conditions vanish. We will discuss the convergence of the infinite sequence $(\alpha^{(t)})_{t\in\mathbb{N}}$ produced by the decomposition algorithm. If the decomposition algorithm stops in some iteration t_0 at $\alpha^{(t_0-1)}$, then by convention we set $\alpha^{(t)} \leftarrow \alpha^{(t_0-1)}$ for all $t \geq t_0$. The definition of convergence used here directly implies the convergence in finite time to a solution arbitrarily

close to the optimum considered in other proofs (Keerthi and Gilbert, 2002; Takahashi and Nishi, 2005).

The Bolzano-Weierstraß property states that every sequence on a compact set contains a convergent sub-sequence. Because of the compactness of $\mathcal{R}(\mathcal{P})$ the sequence $(\alpha^{(t)})_{t\in\mathbb{N}}$ always contains a convergent sub-sequence denoted $(\alpha^{(t)})_{t\in S}$ with limit point $\alpha^{(\infty)}$. From the construction of the decomposition algorithm it follows that the sequence $(f(\alpha^{(t)}))_{t\in\mathbb{N}}$ increases monotonically. The compactness of $\mathcal{R}(\mathcal{P})$ implies that it is bounded. It therefore converges and its limit $f(\alpha^{(\infty)})$ does not depend on the choice of the convergent sub-sequence. The gain sequence $g_{B^{(t)}}(\alpha^{(t-1)}) = f(\alpha^{(t)}) - f(\alpha^{(t-1)})$ is non-negative and converges to zero. It will be the aim of this section to prove that $\alpha^{(\infty)}$ is the maximizer of f within the feasible region $\mathcal{R}(\mathcal{P})$ if the decomposition algorithm is used with HMG working set selection.

List and Simon (2004) introduced the (technical) restriction that all principal 2×2 minors of Q have to be positive definite. For Theorem 4, which was shown by List and Simon (2004), and the proof of Lemma 9 (and thus of Theorem 1) we adopt this requirement. The assumption is not very restrictive because it does not require the whole matrix Q to be positive definite. If in contrast Q is indeed positive definite (for example for Gaussian kernels with distinct examples) then this property is inherited by the principal minors.[2]

If we fix any subset of variables of $\mathcal{P}$ at any feasible point $\alpha\in\mathcal{R}(\mathcal{P})$ then the resulting restricted problem is again of the form $\mathcal{P}$. By analytically solving the problem restricted to a working set B we can compute the gain $g_B(\alpha)$. The set $V_{\mathcal{P}} := \{\alpha\in\mathbb{R}^\ell \,|\, \langle y,\alpha\rangle = z\}$ is the hyperplane defined by the equality constraint. It contains the compact convex feasible region $\mathcal{R}(\mathcal{P})$. The set of possible working sets is denoted by $\mathcal{B}(\mathcal{P}) := \big\{B \,\big|\, B\subset\{1,\ldots,\ell\}, |B|=2\big\}$. We call two working sets $B_1, B_2\in\mathcal{B}(\mathcal{P})$ related if $B_1\cap B_2\neq\emptyset$. With a working set $B=\{b_1,b_2\}$, $b_1<b_2$, we associate the vector w_B with components $(w_B)_{b_1}=1$, $(w_B)_{b_2}=-y_{b_1}y_{b_2}$ and $(w_B)_i=0$ otherwise. It points into the direction in which α can be modified using the working set.

If a feasible point α is not optimal then there exists a working set B on which it can be improved. This simply follows from the fact that there are working set selection policies based on which the decomposition algorithm is known to converge (Lin, 2001; Keerthi and Gilbert, 2002; Fan et al., 2005; Takahashi and Nishi, 2005). In this case the gain $g_B(\alpha)$ is positive.

Next, we give a simple geometrically inspired criterion for the optimality of a solution.

Lemma 2 *We consider the problem $\mathcal{P}$. For a feasible point α_0 the following conditions are equivalent:*

1. *α_0 is optimal.*
2. *$\langle(\alpha-\alpha_0),\nabla f(\alpha_0)\rangle\le 0$ for all $\alpha\in\mathcal{R}(\mathcal{P})$.*
3. *$\langle\mu\cdot w_B,\nabla f(\alpha_0)\rangle\le 0$ for all $\mu\in\mathbb{R}$, $B\in\mathcal{B}(\mathcal{P})$ fulfilling $\alpha_0+\mu\cdot w_B\in\mathcal{R}(\mathcal{P})$.*

Proof The proof is organized as $(1)\Rightarrow(2)\Rightarrow(3)\Rightarrow(1)$. We consider the Taylor expansion

$$f(\alpha) = f(\alpha_0) + \langle(\alpha-\alpha_0),\nabla f(\alpha_0)\rangle - \frac{1}{2}(\alpha-\alpha_0)^T Q(\alpha-\alpha_0)$$

2. Usually there is only a zero set of training data sets that violate this condition. This is obviously true if the input space is an open subset of some $\mathbb{R}^n$ and the distribution generating the training data has a density w.r.t. the Lebesgue measure.

of f in α_0. Let us assume that (2) does not hold, that is, there exists $\alpha \in \mathcal{R}(\mathcal{P})$ such that $q := \langle(\alpha - \alpha_0), \nabla f(\alpha_0)\rangle > 0$. From the convexity of $\mathcal{R}(\mathcal{P})$ it follows that $\alpha_\mu := \mu\alpha + (1-\mu)\alpha_0 \in \mathcal{R}(\mathcal{P})$ for $\mu \in [0,1]$. We further set $r := \frac{1}{2}(\alpha - \alpha_0)^T Q(\alpha - \alpha_0) \geq 0$ and have $\langle(\alpha_\mu - \alpha_0), \nabla f(\alpha_0)\rangle = \mu q$ and $\frac{1}{2}(\alpha_\mu - \alpha_0)^T Q(\alpha_\mu - \alpha_0) = \mu^2 r$. We can chose $\mu_0 \in (0,1]$ fulfilling $\mu_0 q > \mu_0^2 r$. Then it follows

$$\begin{aligned} f(\alpha_{\mu_0}) &= f(\alpha_0) + \langle(\alpha_{\mu_0} - \alpha_0), \nabla f(\alpha_0)\rangle - \frac{1}{2}(\alpha_{\mu_0} - \alpha_0)^T Q(\alpha_{\mu_0} - \alpha_0) \\ &= f(\alpha_0) + \mu_0 q - \mu_0^2 r \\ &> f(\alpha_0) \ , \end{aligned}$$

which proves that α_0 is not optimal. Thus (1) implies (2). Of course (3) follows from (2). Now we assume α_0 is not optimal. From the fact that there are working set selection policies for which the decomposition algorithm converges to an optimum it follows that there exists a working set B on which α_0 can be improved, which means $g_B(\alpha_0) > 0$. Let α_1 denote the optimum on the feasible line segment within $\mathcal{R}(\mathcal{P})$ written in the form $\alpha_1 = \alpha_0 + \mu \cdot w_B$. Using the Taylor expansion above at α_1 and the positive semi-definiteness of Q we get

$$\begin{aligned} f(\alpha_1) = f(\alpha_0) + \langle(\alpha_1 - \alpha_0), \nabla f(\alpha_0)\rangle - \frac{1}{2}(\alpha_1 - \alpha_0)^T Q(\alpha_1 - \alpha_0) &> f(\alpha_0) \\ \Leftrightarrow \ \langle(\alpha_1 - \alpha_0), \nabla f(\alpha_0)\rangle &> \frac{1}{2}(\alpha_1 - \alpha_0)^T Q(\alpha_1 - \alpha_0) \geq 0 \\ \Rightarrow \ \langle \mu \cdot w_B, \nabla f(\alpha_0)\rangle &> 0 \end{aligned}$$

showing that (3) implies (1). ■

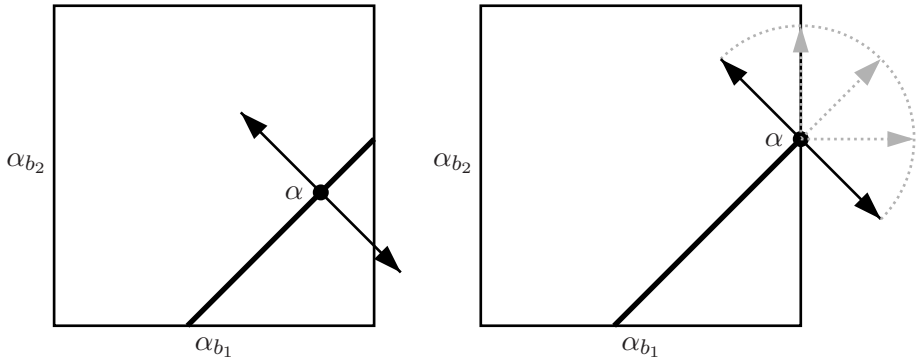

Figure 3: This figure illustrates the optimality condition given in Lemma 2 for one working set. On the left the case of two free variables α_{b_1} and α_{b_2} is shown, while on the right the variable α_{b_1} is at the bound C. The fat lines represent the feasible region for the 2-dimensional problem induced by the working set. The arrows show the possible gradient directions not violating the optimality conditions.

3.2 Convergence of the Greedy Policy

Before we look at the convergence of the MG algorithm, we use Theorem 4 by List and Simon (2004) to prove the convergence of the decomposition algorithm using the greedy policy with respect to the gain for the working set selection. For this purpose we will need the concept of a 2-sparse witness of suboptimality.

Definition 3 (2-sparse witness of suboptimality) *A family of functions* $(C_B)_{B\in\mathcal{B}(\mathcal{P})}$

$$C_B : \mathcal{R}(\mathcal{P}) \to \mathbb{R}^{\geq 0}$$

fulfilling the conditions
(C1) *C_B is continuous,*
(C2) *if α is optimal for $\mathcal{P}_{B,\alpha}$, then $C_B(\alpha) = 0$, and*
(C3) *if a feasible point α is not optimal for $\mathcal{P}$, then there exists B such that $C_B(\alpha) > 0$*
is called a 2-sparse witness of suboptimality *(List and Simon, 2004).*

Every 2-sparse witness of suboptimality induces a working set selection algorithm by

$$B^{(t)} := \operatorname*{argmax}_{B\in\mathcal{B}(\mathcal{P})} \left(C_B(\alpha^{(t-1)}) \right) \quad .$$

List and Simon (2004) call this the induced decomposition algorithm. Now we can quote a general convergence theorem for decomposition methods induced by a 2-sparse witness of suboptimality:[3]

Theorem 4 (List and Simon, 2004) *We consider the problem $\mathcal{P}$ and a 2-sparse witness of suboptimality $(C_B)_{B\in\mathcal{B}(\mathcal{P})}$. Let $(\alpha^{(t)})_{t\in\mathbb{N}}$ denote a sequence of feasible points generated by the decomposition method induced by (C_B) and $(\alpha^{(t)})_{t\in S}$ a convergent sub-sequence with limit point $\alpha^{(\infty)}$. Then, the limit point is optimal for $\mathcal{P}$.*

The following lemma allows for the application of this theorem.

Lemma 5 *The family of functions $(g_B)_{B\in\mathcal{B}(\mathcal{P})}$ is a 2-sparse witness of suboptimality.*

Proof Property **(C2)** is fulfilled directly per construction. Property **(C3)** follows from the fact that there exist working set selection strategies such that the decomposition method converges (Lin, 2001; Takahashi and Nishi, 2005; List and Simon, 2005). It is left to prove property **(C1)**. We fix a working set $B = \{b_1, b_2\}$ and the corresponding direction vector w_B. Choosing the working set B is equivalent to restricting the problem to this direction. We define the affine linear function

$$\varphi : V_{\mathcal{P}} \to \mathbb{R} \ , \qquad \alpha \mapsto \left.\frac{\partial}{\partial\mu}\right|_{\mu=0} f(\alpha + \mu w_B)$$

and the ($(\ell-2)$-dimensional) hyperplane $H := \{\alpha \in V_{\mathcal{P}} \,|\, \varphi(\alpha) = 0\}$ within the ($(\ell-1)$-dimensional) vector space $V_{\mathcal{P}}$. This set always forms a hyperplane because $Qw_B \neq 0$ is guaranteed by the assumption that all 2×2 minors or Q are positive definite. This hyperplane contains the optima of f restricted to the lines $\alpha + \mathbb{R}\cdot w_B$ considering the equality constraint but not the box constraints. We

3. Theorem 1 in List and Simon (2004) is more general as it is not restricted to working sets of size two.

introduce the map π_H projecting $V_\mathcal{P}$ onto H along w_B, that is the projection mapping the whole line $\alpha + \mathbb{R} \cdot w_B$ onto its unique intersection with H. The hyperplane H contains the compact subset

$$\tilde{H} := \left\{ \alpha \in H \,\middle|\, \alpha + \mathbb{R} \cdot w_B \cap \mathcal{R}(\mathcal{P}) \neq \emptyset \right\} = \pi_H(\mathcal{R}(\mathcal{P}))$$

on which we define the function

$$\delta : \tilde{H} \to \mathbb{R} \ , \qquad \alpha \mapsto \operatorname*{argmin}_{\{\mu \in \mathbb{R} \,|\, \alpha + \mu w_B \in \mathcal{R}(\mathcal{P})\}} |\mu| \ .$$

The term $\delta(\pi_H(\alpha))w_B$ describes the shortest vector moving $\alpha \in \tilde{H}$ to the feasible region on the line along w_B. On $\tilde{H} \setminus \mathcal{R}(\mathcal{P})$ it parameterizes the boundary of the feasible region (see Figure 4). These properties enable us to describe the optimal solution of the sub-problem induced by the working set. Starting from $\alpha \in \mathcal{R}(\mathcal{P})$ the optimum $\pi_H(\alpha)$ is found neglecting the box constraints. In case this point is not feasible it is clipped to the feasible region by moving it by $\delta(\pi_H(\alpha))w_B$. Thus, per construction it holds $g_B(\alpha) = f(\pi_H(\alpha) + \delta(\pi_H(\alpha))w_B) - f(\alpha)$. The important point here is that convexity and compactness of $\mathcal{R}(\mathcal{P})$ guarantee that δ is well-defined and continuous. We conclude that g_B is continuous as it is a concatenation of continuous functions. ■

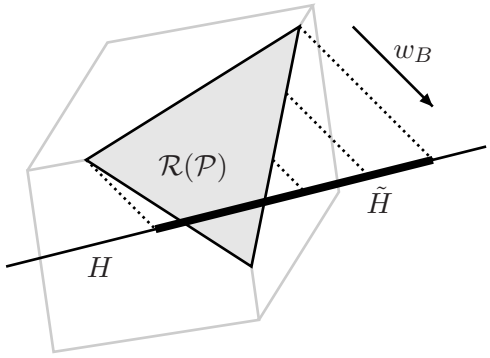

Figure 4: The feasible region $\mathcal{R}(\mathcal{P})$ within the $(\ell-1)$-dimensional vector space $V_\mathcal{P}$ is illustrated. The ℓ-dimensional box constraints are indicated in light gray. The thin line represents the hyperplane H containing the compact subset $\tilde{H}$ drawn as a fat line segment. The lengths of the dotted lines indicate the absolute values of the function δ on $\tilde{H}$. The function δ vanishes within the intersection of $\tilde{H}$ and $\mathcal{R}(\mathcal{P})$.

Corollary 6 *We consider problem $\mathcal{P}$. Let $(\alpha^{(t)})_{t\in\mathbb{N}}$ be a sequence of feasible points produced by the decomposition algorithm using the greedy working set selection policy. Then, every limit point $\alpha^{(\infty)}$ of a converging sub-sequence $(\alpha^{(t)})_{t\in S}$ is optimal for $\mathcal{P}$.*

Proof For a feasible point α and a working set B the achievable gain is computed as $g_B(\alpha)$. Thus, the decomposition method induced by the family (g_B) selects the working set resulting in the maximum gain, which is exactly the greedy policy. Lemma 5 and Theorem 4 complete the proof. ■

It is straightforward to use the more general version of the theorem from List and Simon (2004) to extend the construction for working sets limited in size to some $q > 2$.

3.3 Convergence of the MG Algorithm

Theorem 7 *Given a feasible point $\alpha^{(t)}$ for $\mathcal{P}$ and a previous working set $B^{(t-1)}$ of size 2 as a starting point. Then, in general, the MG algorithm may get stuck, that means, it may stop after finitely many iterations without reaching an optimum.*

Proof As a proof we give a counter example. The MG algorithm may get stuck before reaching the optimum because it is restricted to reselect one element of the previous working set. For $\ell < 4$ this poses no restriction. Thus, to find a counter example, we have to use some $\ell \geq 4$. Indeed, using four training examples is already sufficient. We consider $\mathcal{P}$ for $\ell = 4$ with the values

$$Q = \begin{pmatrix} 2 & \sqrt{3} & 1 & \sqrt{3} \\ \sqrt{3} & 4 & \sqrt{3} & 3 \\ 1 & \sqrt{3} & 2 & \sqrt{3} \\ \sqrt{3} & 3 & \sqrt{3} & 4 \end{pmatrix} , \qquad C = \frac{1}{10} , \qquad y = \begin{pmatrix} -1 \\ -1 \\ +1 \\ +1 \end{pmatrix} .$$

The matrix Q is positive definite with eigenvalues $(9,1,1,1)$. We assume the previous working set to be $B^{(1)} = \{1,3\}$ resulting in the point $\alpha^{(1)} = (C,0,C,0)^T$. Note that this is the result of the first iteration starting from $\alpha^{(0)} = (0,0,0,0)$ greedily choosing the working set $B^{(1)} = \{1,3\}$. It is thus possible that the decomposition algorithm reaches this state in the SVM context. We compute the gradient

$$\nabla f(\alpha^{(1)}) = \mathbf{1} - Q\alpha^{(1)} = (\frac{7}{10}, 1 - \frac{\sqrt{3}}{5}, \frac{7}{10}, 1 - \frac{\sqrt{3}}{5})^T \approx (0.7, 0.65, 0.7, 0.65)^T$$

(which is orthogonal to y). Using Lemma 2 we compute that the sub-problems defined by all working sets with exception $B = \{2,4\}$ are already optimal. The working sets $B^{(1)}$ and B have no element in common. Thus, the maximum gain working pair algorithm cannot select $B^{(2)} = B$ and gets stuck although the point $\alpha^{(1)}$ is not optimal. From Figure 5 we can see that the same example works for all points on the edge $\alpha^{(1)} = (C,\nu,C,\nu)$ for $\nu \in [0,C)$. Lemma 8 states that indeed the edges of the octahedron are the only candidates for the MG algorithm to get stuck. ■

3.4 Convergence of the HMG Algorithm

The above result makes it advisable to use a different algorithm whenever MG is endangered to get stuck. For this purpose the HMG algorithm was designed. In this section we prove that this modification indeed guarantees convergence to an optimum.

The following lemma deals with the specific property of the MG algorithm to reselect one element of the working set, that is, to select related working sets in consecutive iterations. It is a major building block in the proof of the main result stated in Theorem 1.

Lemma 8 *We consider $\mathcal{P}$, a current non-optimal feasible point α and a previous working set $B_1 = \{b_1, b_2\}$. If at least one of the variables α_{b_1} and α_{b_2} is free (not at the bounds 0 or C) then there exists a working set B_2 related to B_1 such that positive gain $g_{B_2}(\alpha) > 0$ can be achieved.*

Proof We show that no counter example exists. The idea is to reduce the number of possible scenarios to a finite number and to inspect each case individually.

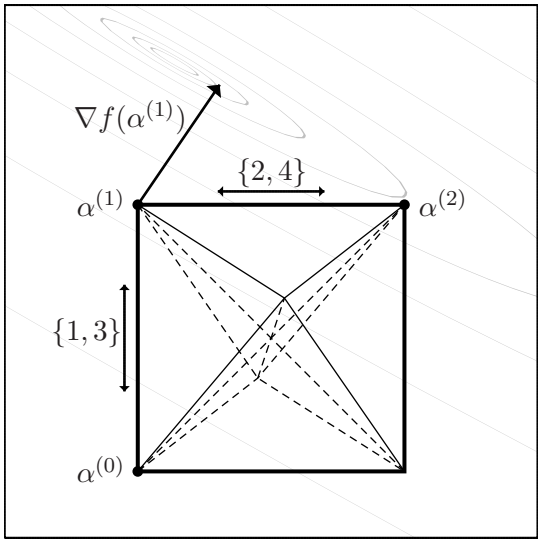

Figure 5: Illustration of the counter example. The feasible region $\mathcal{R}(\mathcal{P})$ forms an octahedron within the 3-dimensional space $V_{\mathcal{P}}$. The possible directions of movement using working sets of size 2 are parallel to the edges of the octahedron. The SVM algorithm starts in $\alpha^{(0)} = (0,0,0,0)^T$. During the first two iterations the greedy policy reaches the points $\alpha^{(1)} = (C,0,C,0)^T$ and $\alpha^{(2)} = (C,C,C,C)^T$. The MG algorithm gets stuck after the first iteration at $\alpha^{(1)}$. The plane spanned by the directions $(1,0,1,0)^T$ and $(0,1,0,1)^T$ defined by the working sets $\{1,3\}$ and $\{2,4\}$ respectively, is drawn. The gray lines are level sets of the target function f within this plane. In the point $\alpha^{(1)}$ the gradient $\nabla f(\alpha^{(1)})$ (which lies within the plane) has an angle of less than $\pi/2$ only with the horizontally drawn edge corresponding to the working set $\{2,4\}$.

For $\ell \leq 3$ the condition that B_1 and B_2 are related is no restriction and we are done. In the main part of the proof, we consider the 4-dimensional case and set $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)^T$ and $B_1 = \{1,2\}$ with free variable $\alpha_1 \in (0,C)$. In the end, we will reduce the general case to $\ell \leq 4$.

Let us have a look at potential counter examples. A feasible point α is a counter example if it is not optimal and does not allow for positive gain on any working set related to B_1. These conditions are equivalent to

$$g_B(\alpha) \begin{cases} = 0 & \text{for } B \neq \{3,4\} \\ > 0 & \text{for } B = \{3,4\} \end{cases} . \tag{4}$$

Looking at the six possible working sets B we observe from Lemma 2 that we have to distinguish three cases for sub-problems induced by the working sets:

- The current point α is at the bounds for a variable indexed by B and the points $\alpha + \mu \cdot w_B$ lie within $\mathcal{R}(\mathcal{P})$ only for $\mu \leq 0$. Then Lemma 2 states that α can only be optimal if $\langle w_B, \nabla f(\alpha) \rangle \geq 0$.

- The current point α is at the bounds for a variable indexed by B and the points $\alpha + \mu \cdot w_B$ lie within $\mathcal{R}(\mathcal{P})$ only for $\mu \geq 0$. Then Lemma 2 states that α can only be optimal if $\langle w_B, \nabla f(\alpha) \rangle \leq 0$.
- The current point α is not at the bounds for both variables indexed by B. Thus, there are positive and negative values for μ such that $\alpha + \mu \cdot w_B$ lies within $\mathcal{R}(\mathcal{P})$. From Lemma 2 it follows that α can only be optimal if $\langle w_B, \nabla f(\alpha) \rangle = 0$.

We conclude that the signs (< 0, $= 0$, or > 0) of the expressions

$$\langle w_B, \nabla f(\alpha) \rangle = \frac{\partial f}{\partial \alpha_{b'_0}}(\alpha) - y_{b'_0} y_{b'_1} \frac{\partial f}{\partial \alpha_{b'_1}}(\alpha) \quad \text{for} \quad B = \{b'_0, b'_1\} \subset \{1,2,3,4\} \tag{5}$$

and the knowledge about which variables are at which bound are sufficient for the optimality check. Further, it is not important which exact value a free variable takes. The possible combinations of vectors w_B occurring in equation (5) are generated by the label vector y. Combining these insights, we define the maps

$$\text{sign} : \mathbb{R} \to \{-1, 0, +1\}, \qquad x \mapsto \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ +1 & \text{if } x > 0 \end{cases}$$

$$\text{bound} : [0, C] \to \{0, \frac{C}{2}, C\}, \qquad x \mapsto \begin{cases} 0 & \text{if } x = 0 \\ \frac{C}{2} & \text{if } 0 < x < C \\ C & \text{if } x = C \end{cases}$$

and a mapping of all possible counter examples onto a finite number of cases

$$\Psi : \mathcal{R}(\mathcal{P}) \times \mathbb{R}^4 \times \{-1,+1\}^4 \to \{-1,0,+1\}^6 \times \{0, C/2, C\}^3 \times \{-1,+1\}^4,$$
$$\begin{pmatrix} \alpha \\ \nabla f(\alpha) \\ y \end{pmatrix} \mapsto \begin{pmatrix} \text{bound}(\alpha_i),\ i \in \{2,3,4\} \\ \text{sign}(\langle w_B, \nabla f(\alpha) \rangle),\ B \in \mathcal{B}(\mathcal{P}) \\ y \end{pmatrix} .$$

A possible counter example is fully determined by a candidate point $\alpha \in \mathcal{R}(\mathcal{P})$, the gradient $G = \nabla f(\alpha)$, and the label vector y. As the parameters of problem $\mathcal{P}$ are not fixed here, the equality constraint can be ignored, because every point fulfilling the box constraints can be made feasible by shifting the equality constraint hyperplane. The relation $\Psi(\alpha, G, y) = \Psi(\tilde{\alpha}, \tilde{G}, \tilde{y})$ divides the pre-image of Ψ into equivalence classes. For each element of one equivalence class the check of condition (4) using Lemma 2 is the same. Formally, we have

$$\begin{aligned} &\Psi(\alpha, G, y) = \Psi(\tilde{\alpha}, \tilde{G}, \tilde{y}) \\ \Rightarrow &\text{condition (4) holds for } (\alpha, G, y) \text{ if and only if condition (4) holds for } (\tilde{\alpha}, \tilde{G}, \tilde{y}) \ . \end{aligned}$$

It follows that any finite set containing representatives of all the non-empty equivalence classes is sufficient to check for the existence of a counter example in the infinite pre-image of Ψ.[4] The checking can be automated using a computer program. A suitable program can be downloaded from the online appendix

4. The function Ψ itself helps designing such a set of representatives of the non-empty classes. For every fixed α and y the set $\{-4, -3, \ldots, 3, 4\}^4 \subset \mathbb{R}^4$ is sufficient to generate all possible combinations of $\text{sign}(\langle w_B, \nabla f(\alpha) \rangle)$, $B \in \mathcal{B}(\mathcal{P})$.

`http://www.neuroinformatik.ruhr-uni-bochum.de/PEOPLE/igel/wss/ .`

The outcome of the program is that there exists no counter example.

It is left to prove the lemma for $\ell > 4$. This case can be reduced to the situations already considered. Because α is not optimal there exists a working set B_* with $g_{B_*}(\alpha) > 0$. The set $W := B_1 \cup B_*$ defines an at most 4-dimensional problem. The proof above shows that there exists a working set $B \subset W$ with the required properties. ■

Following List and Simon (2004), one can bound $\|\alpha^{(t)} - \alpha^{(t-1)}\|$ in terms of the gain:

Lemma 9 *We consider problem $\mathcal{P}$ and a sequence $(\alpha^{(t)})_{t\in\mathbb{N}}$ produced by the decomposition algorithm. Then, the sequence $\left(\|\alpha^{(t)} - \alpha^{(t-1)}\|\right)_{t\in\mathbb{N}}$ converges to* 0.

Proof The gain sequence $\left(g_{B^{(t)}}(\alpha^{(t-1)})\right)_{t\in\mathbb{N}}$ converges to zero as it is non-negative and its sum is bounded from above. The inequality

$$g_{B^{(t)}}(\alpha^{(t-1)}) \geq \frac{\sigma}{2}\|\alpha^{(t)} - \alpha^{(t-1)}\|^2 \quad \Leftrightarrow \quad \|\alpha^{(t)} - \alpha^{(t-1)}\| \leq \sqrt{\frac{2}{\sigma} g_{B^{(t)}}(\alpha^{(t-1)})}$$

holds, where σ denotes the minimal eigenvalue of the 2×2 minors of Q. By the technical assumption that all principal 2×2 minors of Q are positive definite we have $\sigma > 0$. ■

Before we can prove our main result we need the following lemma.

Lemma 10 *We consider problem $\mathcal{P}$, a sequence $(\alpha^{(t)})_{t\in\mathbb{N}}$ produced by the decomposition algorithm and the corresponding sequence of working sets $(B^{(t)})_{t\in\mathbb{N}}$. Let the index set $S \subset \mathbb{N}$ correspond to a convergent sub-sequence $(\alpha^{(t)})_{t\in S}$ with limit point $\alpha^{(\infty)}$.*

(*i*) *Let*

$$I := \left\{ B \in \mathcal{B}(\mathcal{P}) \;\middle|\; |\{t \in S \,|\, B^{(t)} = B\}| = \infty \right\}$$

denote the set of working sets selected infinitely often. Then, no gain can be achieved in the limit point $\alpha^{(\infty)}$ using working sets $B \in I$.

(*ii*) *Let*

$$R := \left\{ B \in \mathcal{B}(\mathcal{P}) \setminus I \;\middle|\; B \text{ is related to some } \tilde{B} \in I \right\} .$$

denote the set of working sets related to working sets in I. If the decomposition algorithm chooses MG working sets, no gain can be achieved in the limit point $\alpha^{(\infty)}$ using working sets $B \in R$.

This is obvious from equation (5) and the fact that these cases cover different as well as equal absolute values for all components together with all sign combinations. Hence, it is sufficient to look at these 9^4 gradient vectors, or in other words, the map from $\{-4,-3,\ldots,3,4\}^4$ to $\text{sign}(\langle w_B, \nabla f(\alpha)\rangle)$, $B \in \mathcal{B}(\mathcal{P})$ is surjective. The mapping of the 3^3 points $\alpha_1 = C/2$, $\alpha_i \in \{0, C/2, C\}$ for $i \in \{2,3,4\}$ onto $\text{bound}(\alpha_i)$, $i \in \{2,3,4\}$ is bijective. Of course the identity map $y \mapsto y$ of the 2^4 possible labels is bijective, too. Now we have constructed a set of $9^4 \cdot 3^3 \cdot 2^4 = 2,834,352$ cases. If there is no counter example among them, we know that no counter example exists in the whole infinite set.

Proof First, we prove (i). Let us assume these exists $B \in I$ on which positive gain can be achieved in the limit point $\alpha^{(\infty)}$. Then we have $\varepsilon := g_B(\alpha^{(\infty)}) > 0$. Because g_B is continuous there exists t_0 such that $g_B(\alpha^{(t)}) > \varepsilon/2$ for all $t \in S$, $t > t_0$. Because B is selected infinitely often it follows $f(\alpha^{(\infty)}) = \infty$. This is a contradiction to the fact that f is bounded on $\mathcal{R}(\mathcal{P})$.

To prove (ii) we define the index set $S_{(+1)} := \{t+1 \,|\, t \in S\}$. Using Lemma 9 we conclude that the sequence $(\alpha^{(t)})_{t \in S_{(+1)}}$ converges to $\alpha^{(\infty)}$. Let us assume that the limit point can be improved using a working set $B \in R$ resulting in $\varepsilon := g_B(\alpha^{(\infty)}) > 0$. Because g_B is continuous, there exists t_0 such that it holds $g_B(\alpha^{(t)}) > \varepsilon/2$ for all $t \in S_{(+1)}$, $t > t_0$. By the convergence of the sequence $(f(\alpha^{(t)}))_{t \in \mathbb{N}}$ we find t_1 such that for all $t > t_1$ it holds $g_{B^{(t)}}(\alpha^{(t-1)}) < \varepsilon/2$. The definition of I yields that there is a working set $\tilde{B} \in I$ related to B which is chosen in an iteration $t > \max\{t_0, t_1\}$, $t \in S$. Then in iteration $t+1 \in S_{(+1)}$ due to the MG policy the working set B (or another working set resulting in larger gain) is selected. We conclude that the gain achieved in iteration $t+1$ is greater and smaller than $\varepsilon/2$ at the same time which is a contradiction. Thus, $\alpha^{(\infty)}$ can not be improved using a working set $B \in R$. ■

Proof of Theorem 1 First we consider the case that the algorithm stops after finitely many iterations, that it, the sequence $(\alpha^{(t)})_{t \in \mathbb{N}}$ becomes stationary. We again distinguish two cases depending on the working set selection algorithm used just before the stopping condition is met. In case the MVP algorithm is used the stopping condition checks the exact KKT conditions. Thus, the point reached is optimal. Otherwise Lemma 8 asserts the optimality of the current feasible point.

For the analysis of the infinite case we distinguish two cases again. If the MG algorithm is used only finitely often then we can simply apply the convergence proof of SMO (Keerthi and Gilbert, 2002; Takahashi and Nishi, 2005). Otherwise we consider the set

$$T := \{t \in \mathbb{N} \,|\, \text{MG is used in iteration } t\}$$

of iterations in which the MG selection is used. The compactness of $\mathcal{R}(\mathcal{P})$ ensures the existence of a subset $S \subset T$ such that the sub-sequence $(\alpha^{(t)})_{t \in S}$ converges to some limit point $\alpha^{(\infty)}$. We define the sets

$$I := \left\{ B \in \mathcal{B}(\mathcal{P}) \;\middle|\; |\{t \in S \,|\, B^{(t)} = B\}| = \infty \right\}$$
$$R := \left\{ B \in \mathcal{B}(\mathcal{P}) \setminus I \;\middle|\; B \text{ is related to some } \tilde{B} \in I \right\}$$

and conclude from Lemma 10 that $\alpha^{(\infty)}$ can not be improved using working sets $B \in I \cup R$. Now let us assume that the limit point can be improved using any other working set. Then Lemma 8 states that all coordinates $\alpha_i^{(\infty)}$ for all $i \in B \in I$ are at the bounds. By the definition of the HMG algorithm this contradicts the assumption that the MG policy is used on the whole sequence $(\alpha^{(t)})_{t \in S}$. Thus, the limit point $\alpha^{(\infty)}$ is optimal for $\mathcal{P}$. From the strict increase and the convergence of the sequence $(f(\alpha^{(t)}))_{t \in \mathbb{N}}$ it follows that the limit point of every convergent sub-sequence $(\alpha^{(t)})_{t \in \tilde{S}}$ is optimal. ■

4. Experiments

The main purpose of our experiments is the comparison of different working set selection policies for large scale problems. This comparison focuses on SMO-like algorithms. The experiments were

carried out using LIBSVM (Chang and Lin, 2001). We implemented our HMG selection algorithm within LIBSVM to allow for a direct comparison. The modified source code of LIBSVM is given in the online appendix

`http://www.neuroinformatik.ruhr-uni-bochum.de/PEOPLE/igel/wss/ .`

Three SMO-like working set selection policies were compared, namely the LIBSVM-2.71 MVP algorithm, the second order LIBSVM-2.8 algorithm, and HMG working set selection.

To provide a baseline, we additionally compared these three algorithms to SVM^{light} (Joachims 1999) with a working set of size ten. In these experiments we used the same configuration and cache size as for the SMO-like algorithms. It is worth noting that neither the iteration count nor the influence of shrinking are comparable between LIBSVM and SVM^{light}. As we do not want to go into details on conceptual and implementation differences between the SVM packages, we only compared the plain runtime for the most basic case as it most likely occurs in applications. Still, as the implementations of the SMO-like algorithms and SVM^{light} differ, the results have to be interpreted with care.

We consider 1-norm soft margin SVM with radial Gaussian kernel functions

$$k_\sigma(x_i, x_j) := \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{6}$$

with kernel parameter σ and regularization parameter C. If not stated otherwise, the SVM was given 40 MB of working memory to store matrix rows. The accuracy of the stopping criterion was set to $\varepsilon = 0.001$. This value is small compared to the components of the gradient of the target function in the starting position $\alpha^{(0)}$. The shrinking heuristics for speeding up the SVM algorithm is turned on, see Section 4.4. Shrinking may cause the decomposition algorithm to require more iterations, but in most cases it considerably saves time. All of these settings correspond to the LIBSVM default configuration. If not stated otherwise, the hyperparameters C and σ were fixed to values giving well generalizing classifiers. These were determined by grid search optimizing the error on independent test data.

For the determination of the runtime of the algorithms we used a 1533 MHz AMD Athlon-XP system running Fedora Linux.

In most experiments we measured both the number of iterations needed and the runtime of the algorithm.[5] Although the runtime depends highly on implementation issues and programming skills, this quantity is in the end the most relevant in applications.

The comparison of the working set selection algorithms involves one major difficulty: The stopping criteria are different. It is in general not possible to compute the stopping criterion of one

5. We did not investigate the classification performance of the different approaches. As we are comparing algorithms converging to the exact solution of the SVM problem and the stopping criteria are chosen appropriately, we can expect that the machines trained using the different methods are equally well suited for classification. Due to the finite accuracy, the direction from which the optimum is approached, the exact path taken by the optimization, and the stopping criterion influence the value of the final solution. Thus, small differences in classification performance may occur between the algorithms. In contrast to the comparison of rough approximation methods or completely distinct types of classifiers, these effects are highly random, as they can depend on the presence of single input patterns or even on the order in which the training examples are presented. Another well known effect is that the classification accuracy measured on a test set does not necessarily increase with the solution accuracy. Thus, besides the prior knowledge that the differences are negligible, a comparisons of the classification accuracy is not meaningful in this context.

algorithm in another without additional computational effort. As the comparability of the runtime depends on an efficient implementation, each algorithm in the comparisons uses its own stopping criterion. The computation of the final value of the objective function reveals that the two stopping conditions are roughly comparable (see Section 4.2 and Table 2).

As discussed in Section 1.3, the order in which the training examples are presented influences the initial working set and thereby considerably the speed of optimization. Whether a certain ordering leads to fast or slow convergence is dependent on the working set selection method used. Therefore, we always consider the median over 10 independent trials with different initial working sets if not stated otherwise. In each trial the different algorithms started from the same working set. Whenever we claim that one algorithm requires less iterations or time these results are highly significant (two-tailed Wilcoxon rank sum test, $p < 0.001$).

Besides the overall performance of the working set selection strategies we investigated the influence of a variety of conditions. The experiments compared different values of the kernel parameter σ, the regularization parameter C, and the cache size. Further, we evaluated the performance with LIBSVM's shrinking algorithm turned on or off. Finally, we compared variants of the HMG strategy using different numbers of cached matrix rows for the gain computation.

4.1 Data Set Description

Four benchmark problems were considered. The 60,000 training examples of the MNIST handwritten digit database (LeCun et al., 1998) were split into two classes containing the digits $\{0,1,2,3,4\}$ and $\{5,6,7,8,9\}$, respectively. Every digit is represented as a 28×28 pixel array making up a 784 dimensional input space.

The next two data sets are available from the UCI repository (Blake and Merz, 1998). The spam-database contains 4,601 examples with 57 features extracted from e-mails. There are 1,813 positive examples (spam) and 2,788 negative ones. We transformed every feature to zero mean and unit variance. Because of the small training set, HMG is not likely to excel at this benchmark.

The connect-4 opening database contains 67,557 game states of the connect-4 game after 8 moves together with the labels 'win', 'loose', or 'draw'. For binary classification the 'draw' examples were removed resulting in 61,108 data points. Every situation was transformed into a 42-dimensional vector containing the entries 1, 0, or -1 for the first player, no player, or the second player occupying the corresponding field, respectively. The representation is sparse as in every vector only 8 components are non-zero. The data were split roughly into two halves making up training and test data. For the experiments only the training data were used.

The face data set contains 20,000 positive and 200,000 negative training examples. Every example originates from the comparison of two face images. Two pictures of the same person were compared to generate positive examples, while comparisons of pictures of different persons make up negative examples. The face comparison is based on 89 similarity features. These real-world data were provided by the Viisage Technology AG and are not available to the public.

Training an SVM using the large data sets face and MNIST takes very long. Therefore these two problems were not considered in all experiments.

We determined appropriate values for σ and C for each benchmark problem, see Table 1. We did coarse grid searches. The parameter combinations resulting in the smallest errors on corresponding test sets were chosen.

We want to pay special attention to the size of the kernel matrices in comparison to the cache size (see Table 1). The data sets cover a wide range of kernel matrix sizes which fit into the cache by nearly 50 % to only 0.02 %. It is a hopeless approach to adapt the cache size in order to fit larger parts of the kernel matrix into working memory. Because the space requirement for the kernel matrix grows quadratically with ℓ, large scale real world problems exceed any physically available cache.

data set	dim.	ℓ	cache	σ	C	SV	BSV
spam-database	57	4,601	47.2 %	10	50	18.5 %	11.7 %
connect-4	42	30,555	1.07 %	1.5	4.5	27.0 %	7.7 %
MNIST	784	60,000	0.28 %	3,500	50	10.5 %	4.6 %
face	89	220,000	0.02 %	3	5	2.6 %	1.2 %

Table 1: SVM parameters used in the comparison together with solution statistics. The column "dim." gives the input space dimension while ℓ is the number of training examples. The "cache"-column shows how much of the kernel matrix fits into the kernel cache. The fractions of support vectors and bounded support vectors are denoted by "SV" and "BSV". These percentage values might slightly differ between the algorithms because of the finite accuracy of the solutions.

4.2 Comparison of Working Set Selection Strategies

We trained SVMs on all data sets presented in the previous section. We monitored the number of iterations and the time until the stopping criterion was met. The results are shown in Table 2. The final target function values $f(\alpha^*)$ are also presented to prove the comparability of the stopping criteria (for the starting state it holds $f(\alpha^{(0)}) = 0$). Indeed, the final values are very close and which algorithm is most accurate depends on the problem.

It becomes clear from the experiments that the LIBSVM-2.71 algorithm performs worst. This is no surprise because it does not take second order information into account. In the following we will concentrate on the comparison of the second order algorithms LIBSVM-2.8 and HMG.

As the smallest problem considered the spam-database consists of 4,601 training examples. The matrix Q requires about 81 MB of working memory. The cache size of 40 MB should be sufficient when using the shrinking technique. The LIBSVM-2.8 algorithm profits from the fact that the kernel matrix fits into the cache after the first shrinking event. It takes less iterations and (in the mean) the same time per iteration as the HMG algorithm and is thus the fastest in the end.

In the connect-4 problem the kernel matrix does not fit into the cache even if shrinking is used to reduce the problem size. Thus, even in late iterations kernel evaluations can occur. Here, HMG outperforms the old and the new LIBSVM algorithm. This situation is even more pronounced for the MNIST data set and the face problem. Only a small fraction of Q fit into the cache making expensive kernel evaluations necessary. Note that for all of these large problems the LIBSVM-2.8 algorithm minimizes the number of iterations while HMG minimizes the training time. The HMG algorithm is the fastest on the three large scale problems, because it makes use of the kernel cache more efficiently.

data set (ℓ)	algorithm	iterations	runtime	$f(\alpha^*)$
spam-database (4,601)	LIBSVM-2.71	36,610	11.21 s	27,019.138
	LIBSVM-2.8	**9,228**	**8.44** s	27,019.140
	HMG	10,563	9.17 s	27,019.140
connect-4 (30,555)	LIBSVM-2.71	65,167	916 s	13,557.542
	LIBSVM-2.8	**45,504**	734 s	13,557.542
	HMG	50,281	**633 s**	13,557.536
MNIST (60,000)	LIBSVM-2.71	185,162	13,657 s	143,199.142
	LIBSVM-2.8	**110,441**	9,957 s	143,199.146
	HMG	152,873	**7,485 s**	143,199.160
face (220,000)	LIBSVM-2.71	37,137	14,239 s	15,812.666
	LIBSVM-2.8	**32,783**	14,025 s	15,812.666
	HMG	42,303	**11,278 s**	15,812.664

Table 2: Comparison of the number of iterations of the decomposition algorithm and training times for the different working set selection approaches. In each case the best value is highlighted. The differences are highly significant (Wilcoxon rank sum test, $p < 0.001$). Additionally, the final value of the objective function showing the comparability of the results is given.

We performed the same experiments with the SVM^{light} support vector machine implementation. The results are summarized in Table 3. We relaxed the stopping condition such that the SVM^{light} solutions are less accurate than the LIBSVM solutions. Nevertheless, the SVM^{light} algorithm is slower than the LIBSVM implementation using the SMO algorithm (see Table 2). Please note that according to the numerous implementation differences these experiments do not provide a fair comparison between SMO-like methods and decomposition algorithms using larger working sets.

data set (ℓ)	iterations	runtime	$f(\alpha^*)$
spam-database (4,601)	9,450	23.97 s	27,019,125
connect-4 (30,555)	17,315	5,589 s	13,557.520
MNIST (60,000)	42,347	282,262 s	143,175.447
face (220,000)	9,806	51,011 s	15,812.643

Table 3: Iterations, runtime and objective function value of the SVM^{light} experiments with working set size $q = 10$. Because of the enormous runtime, only one trial was conducted for the MNIST task.

4.3 Analysis of Different Parameter Regimes

The choice of the regularization parameter C and the parameter σ of the Gaussian kernel (6) influence the quadratic problem induced by the data. We analyzed this dependency using grid search on the connect-4 problem. The results are plotted in Figure 6.

All parameter configurations where LIBSVM-2.71 or LIBSVM-2.8 outperformed HMG have one important property in common, namely, that it is a bad idea to reselect an element of the previous working set. This is true when after most iterations both coordinates indexed by the working set are already optimal. This can happen for different reasons:

- For $\sigma \to 0$ the feature vectors corresponding to the training examples become more and more orthogonal and the quadratic problem $\mathcal{P}$ becomes (almost) separable.
- For increasing values of σ the example points become more and more similar in the feature space until they are hard to distinguish. This increases the quotient of the largest and smallest eigenvalue of Q. Thus, the solution of $\mathcal{P}$ is very likely to lie in a corner or on a very low dimensional edge of the box constraining the problem, that is, many of the α_i^* end up at the constraints 0 or C. This is even more likely for small values of C.

We argue that in practice parameter settings leading to those situations are not really relevant because they tend to produce degenerate solutions. Either almost all examples are selected as support vectors (and the SVM converges to nearest neighbor classification) or the information available are used inefficiently, setting most support vector coefficients to C. Both extremes are usually not intended in SVM learning. In our experiments, HMG performs best in the parameter regime giving well generalizing solutions.

4.4 Influence of the Shrinking Algorithm

A shrinking heuristics in a decomposition algorithm tries to predict whether a variable α_i will end up at the box constraint, that is, whether it will take one of the values 0 or C. In this case the variable is fixed at the boundary and the optimization problem is reduced accordingly. Of course, every heuristics can fail and thus when the stopping criterion is met these variables must be reconsidered. The temporary reduction of the problem restricts working set selection algorithms to a subset of possible choices. This may cause more iterations but has the potential to save a lot of runtime.

We repeated our experiments with the LIBSVM shrinking heuristics turned off to reveal the relation between the working set selection algorithms and shrinking, see Table 4. The experiments show that the influence of the shrinking algorithm on the different working set selection policies is highly task dependent. The time saved and even the algorithm for which more time was saved differs from problem to problem. For some problems the differences between the methods increase, for others they decrease. Compared to the experiments with shrinking turned on the results qualitatively remain the same.

4.5 Influence of the Cache Size

The speed (in terms of runtime, not iterations) of the SVM algorithm depends on the fraction of matrix rows fitting into the cache. We used the connect-4 data set to test the dependency of speed on the cache size. The full representation of the matrix Q requires nearly 3.5 GB of working memory for this problem. We trained SVMs with 20 MB (0.56% of the matrix), 40 MB (1.12%), 100 MB

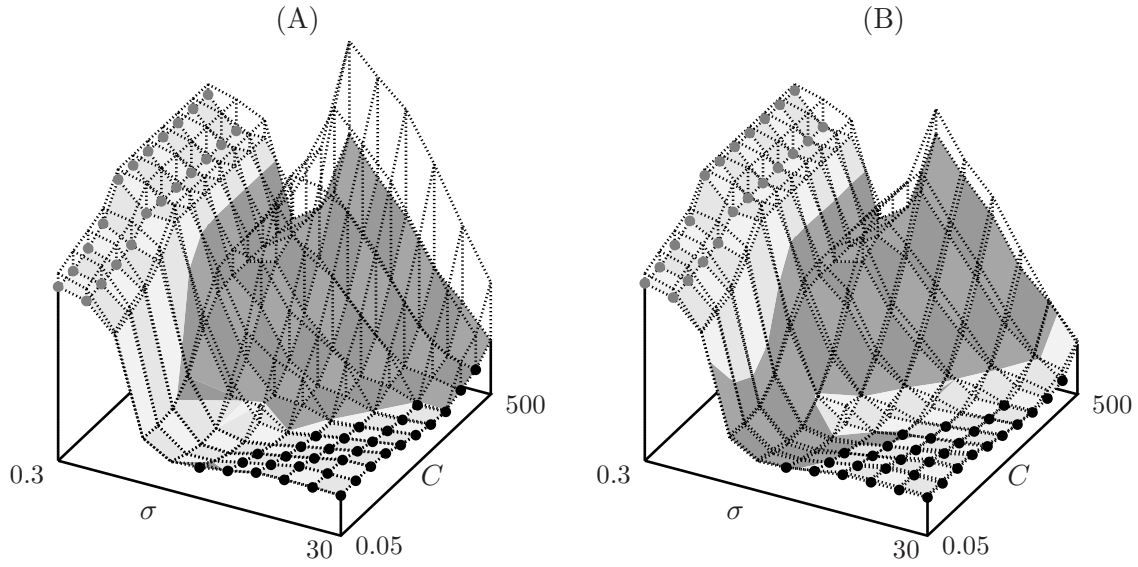

Figure 6: Influence of C and σ on the runtime for the connect-4 data set. The plots show on logarithmic scales the runtime of the SVM depending on C and σ. The comparison of HMG to LIBSVM-2.71 is plotted in (A), while plot (B) shows the comparison of HMG to LIBSVM-2.8. The colored shapes indicate the method needing less runtime, in light and dark gray for LIBSVM and HMG, respectively. Only the lower surface corresponding to the faster algorithm is drawn solid while the higher surface is indicated by the dotted grid lines. Both $\gamma = 1/(2\sigma^2)$ and C are considered in a range of factor $10,000$ containing the well generalizing parameter regime, see Table 2. The dots mark degenerate (and thus not desirable) solutions. The gray dots indicate that the solution uses at least 99 % of the training data as support vectors. If at least 99 % of the support vectors are at the upper bound C the solution is marked with a black dot.

(2.8%) and 200 MB (5.6%) cache. Because the shrinking heuristics reduces the amount of memory required for the storage of the relevant part of Q, the percentage values should be viewed with care. If all variables ending up at the box constraints are removed, the storage size of the matrix Q is about 134 MB. This matrix already fits into the 200 MB cache.

The results listed in Table 5 and plotted in Figure 7 clearly show that for small cache sizes the HMG algorithm is advantageous while for a large cache the LIBSVM-2.8 algorithm catches up.

These results can easily be explained. As long as there is a considerable chance to find a matrix row in the cache it is not necessary to use the cache friendly HMG strategy. In this case it is reasonable to minimize the number of iterations. This is best achieved by the LIBSVM-2.8 algorithm. If the cache is too small to store a relevant part of the kernel matrix it becomes advantageous to use HMG, because HMG produces at most one cache miss per iteration. We conclude that the HMG algorithm should be used for large scale problems.

data set	algorithm	iterations		runtime	
spam-database	LIBSVM-2.71	33,340	91.1 %	12.77 s	114 %
	LIBSVM-2.8	9,123	98.9 %	8.98 s	106 %
	HMG	9,342	88.4 %	11.41 s	124 %
connect-4	LIBSVM-2.71	65,735	100.9 %	2,223 s	243 %
	LIBSVM-2.8	45,466	99.9 %	1,567 s	213 %
	HMG	49,512	98.5 %	1,005 s	159 %
MNIST	LIBSVM-2.71	187,653	101.3 %	94,981 s	695 %
	LIBSVM-2.8	110,470	100.0 %	58,213 s	585 %
	HMG	155,182	101.5 %	41,097 s	549 %
face	LIBSVM-2.71	37,060	99.8 %	55,057 s	387 %
	LIBSVM-2.8	32,796	100.0 %	48,922 s	349 %
	HMG	43,066	101.8 %	33,001 s	293 %

Table 4: Iterations and time needed for solving the quadratic problem without shrinking. The percentage values refer to the corresponding results with shrinking turned on, that is, iterations and runtime of the experiments with shrinking turned on define the 100% mark. Due to the enormous runtime, for the data sets MNIST and face only one trial was conducted.

cache size	LIBSVM-2.71	LIBSVM-2.8	HMG
20 MB	958 s	766 s	656 s
40 MB	916 s	734 s	633 s
100 MB	758 s	649 s	583 s
200 MB	603 s	547 s	555 s

Table 5: The training time for the connect-4 task for the different working set selection algorithms depending on the cache size.

4.6 Number of Matrix Rows Considered

In the definition of the HMG algorithm we restrict ourselves to computing the gain using the two cached matrix rows corresponding to the previous working set. This seems to be an arbitrary restriction. To determine the influence of the number of rows considered we compared the HMG algorithm to two modified versions.

We computed the gain using only one matrix row corresponding to one element of the working set. The element chosen was alternated in every iteration such that a selected variable was used in exactly two successive iterations. This policy reduces the time required for the working set selection by about 50%. It can be considered as the minimal strategy avoiding asymmetries between the variables. The results comparing the usage of one and two rows are shown in Table 6.

Although the stopping criteria used are the same we get different final target function values. This happens due to the reduced number of pairs over which the maximum is taken in the one-row strategy. The values listed indicate that the experiments are roughly comparable, but the one-row strategy produces less accurate solutions in general.

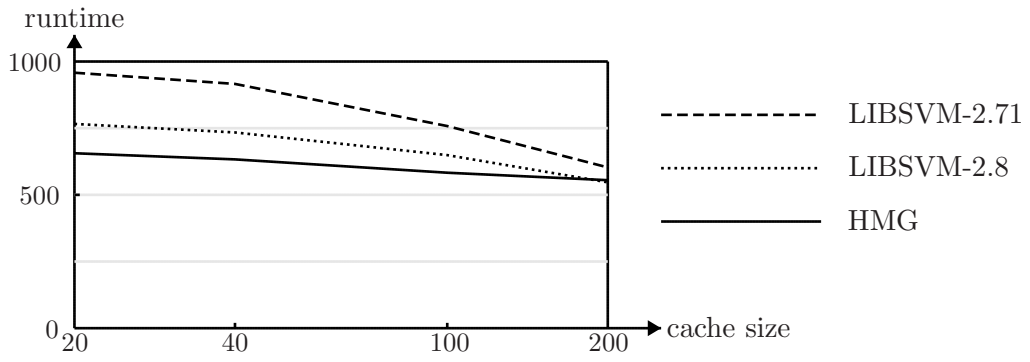

Figure 7: The training time in seconds for the connect-4 data set is plotted over the cache size in MB.

	two rows			one row		
data set	iterations	runtime	$f(\alpha^*)$	iterations	runtime	$f(\alpha^*)$
spam-database	**10,563**	**9.17 s**	27,019.140	14,023	10.99 s	27,019.134
connect-4	**50,281**	**633 s**	13,557.536	83,305	4,967 s	13,557.529

Table 6: Comparison of the one-row and the two-row strategy. The better values are printed in bold face. The differences are highly significant (Wilcoxon rank sum test, $p < 0.001$). The final target function values are lower using the one-row strategy.

The two-row strategy performed clearly better. The reasons for the poor performance of the one-row strategy are the higher number of iterations and, what is worse, the far higher number of unshrinking events. Because of the reduced amount of pairs considered this strategy is endangered to wrongly detect optimality on the shrunk problem causing a costly unshrinking process. Simple experiments indicated that the one-row strategy can compete if the problem is small. However, in this case both HMG strategies were outperformed by the LIBSVM algorithm.

The other strategy tested is to compute the gain for every cached matrix element available. Of course this algorithm is extremely time consuming and thus not practical for applications. This test gives us the minimum number of iterations the HMG working pair algorithm can achieve, as it is the strategy using all information available. It thus provides a bound on the performance of possible extensions of the algorithm using more than two cached matrix rows to determine the working pair. In the case where the whole matrix Q fits into the cache and all rows have already been computed, the strategy coincides with the exact greedy policy w.r.t. the gain. We compared the results to the the two-row strategy, see Table 7.

Again it is difficult to compare the experiments because the algorithm using all cached rows available generally stops later than the two-row strategy. We will nevertheless interpret the results, although this difficulty indicates that the bound on the possible performance of HMG algorithms using more than two rows may not be tight.

The behavior is clearly problem specific. On the connect-4 task both strategies nearly showed the same performance. This reveals that on some real world problems the two-row strategy cannot

	two rows		whole cache		iterations
data set	iterations	$f(\alpha^*)$	iterations	$f(\alpha^*)$	saved
spam-database	10,563	27,019.140	7,280	27,019.143	31 %
connect-4	50,281	13,557.536	51,285	13,557.538	0 %

Table 7: Comparison of the strategies using two matrix rows and the whole matrix cache available.

be outperformed by HMG strategies using more than two matrix rows. In contrast for the spam-database, the whole-cache strategy saved 31 percent of the iterations. This is a considerable amount which was bought dearly using all cached matrix rows for the working set selection. In practice one would like to use a faster method, which, for example, looks at a small fixed number of matrix rows. The reduction of the number of iterations will presumably be less for such strategies. Additionally, the non-trivial question for the row selection policy arises. Thus, for simplicity as well as performance we recommend to stick to the two-row HMG algorithm.

5. Conclusion

The time needed by a decomposition algorithm to solve the support vector machine (SVM) optimization problem up to a given accuracy depends highly on the working set selection. In our experiments with large data sets, that is, when training time really matters, our new hybrid maximum-gain working set selection (HMG) saved a lot of time compared to the latest second order selection algorithm. This speed-up is achieved by the avoidance of cache misses in the decomposition algorithm. In contrast, for small problems the LIBSVM-2.8 algorithm is faster. This result suggest a mixed strategy which switches between the algorithms depending on cache and problem size.

The main advantage of the HMG algorithm is its efficient usage of the matrix cache. It reselects almost always one element of the previous working set. Therefore, at most one matrix row needs to be computed in every iteration. The new algorithm obtains strong theoretical support as it is known to converge to an optimum under weak prerequisites, see Section 3.

The HMG algorithm is especially efficient for appropriate kernel and regularization parameter settings leading to well-generalizing solutions. Thus, it is the method of choice when parameters suiting the problem at hand are *roughly* known. It is for example a good idea to find out well working parameters using a small subset of the data and then train the SVM with the HMG algorithm using the whole data set.

Although LIBSVM-2.8 and HMG both select working sets using second order information, different target functions and variable sets are considered. It is an issue of future work to investigate the performance and the convergence properties of possible combinations of methods. In particular, an elaborate cooperation between the kernel cache strategy and the working set selection algorithm is promising to increase the efficiency of future algorithms.

Acknowledgment

We would like to thank Nikolas List for fruitful discussions and hints. We acknowledge support from Viisage Technology AG under contract “SecureFaceCheck”.

References

C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

A. Bordes, S. Ertekin, J. Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 5:1579–1619, 2005.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

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

D. Hush and C. Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51:51–71, 2003.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 169–184. MIT Press, 1999.

S. S. Keerthi and E. G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46:351–360, 2002.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Transactions on Neural Networks*, 11(1):124–136, 2000.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

C.-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12:1288–1298, 2001.

N. List and H. U. Simon. A general convergence theorem for the decomposition method. In John Shawe-Taylor and Yoram Singer, editors, *Proceedings of the 17th Annual Conference on Learning Theory, COLT 2004*, volume 3120 of *LNCS*, pages 363–377. Springer-Verlag, 2004.

N. List and H. U. Simon. General polynomial time decomposition algorithms. In Peter Auer and Ron Meir, editors, *Proceedings of the 18th Annual Conference on Learning Theory, COLT 2005*, volume 3559 of *LNCS*, pages 308–322. Springer-Verlag, 2005.

E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. In J. Principe, L. Giles, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII*, pages 276–285. IEEE Press, 1997.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 12, pages 185–208. MIT Press, 1999.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

N. Takahashi and T. Nishi. Rigorous proof of termination of SMO algorithm for support vector machines. *IEEE Transaction on Neural Networks*, 16(3):774–776, 2005.

I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, New-York, 1998.

S. V. N. Vishwanathan, A. J. Smola, and M. N. Murty. SimpleSVM. In T. Fawcett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pages 760–767. AAAI Press, 2003.