Journal of Machine Learning Research 6 (2005) 1889–1918 Submitted 04/05; Revised 10/05; Published 11/05

Working Set Selection Using Second Order Information for Training Support Vector Machines

Rong-En Fan B90098@CSIE.NTU.EDU.TW
Pai-Hsuen Chen R90008@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science, National Taiwan University
Taipei 106, Taiwan

Editor: Thorsten Joachims

Abstract

Working set selection is an important step in decomposition methods for training support vector machines (SVMs). This paper develops a new technique for working set selection in SMO-type decomposition methods. It uses second order information to achieve fast convergence. Theoretical properties such as linear convergence are established. Experiments demonstrate that the proposed method is faster than existing selection methods using first order information.

Keywords: support vector machines, decomposition methods, sequential minimal optimization, working set selection

1. Introduction

Support vector machines (SVMs) (Boser et al., 1992; Cortes and Vapnik, 1995) are a useful classification method. Given instances $\mathbf{x}_i, i = 1, \ldots, l$ with labels $y_i \in \{1, -1\}$, the main task in training SVMs is to solve the following quadratic optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{e}^T \boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \le \alpha_i \le C, i = 1, \ldots, l, \\ & \mathbf{y}^T \boldsymbol{\alpha} = 0, \end{aligned} \tag{1}$$

where $\mathbf{e}$ is the vector of all ones, C is the upper bound of all variables, Q is an l by l symmetric matrix with $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, and $K(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function.

The matrix Q is usually fully dense and may be too large to be stored. Decomposition methods are designed to handle such difficulties (e.g., Osuna et al., 1997; Joachims, 1998; Platt, 1998; Chang and Lin, 2001). Unlike most optimization methods which update the whole vector $\boldsymbol{\alpha}$ in each step of an iterative process, the decomposition method modifies only a subset of $\boldsymbol{\alpha}$ per iteration. This subset, denoted as the working set B, leads to a small sub-problem to be minimized in each iteration. An extreme case is the Sequential Minimal Optimization (SMO) (Platt, 1998), which restricts B to have only two elements. Then in each iteration one does not require any optimization software in order to solve a simple two-variable problem. This method is sketched in the following:

Algorithm 1 (SMO-type decomposition method)

1. Find $\boldsymbol{\alpha}^1$ as the initial feasible solution. Set $k = 1$.

2. If $\boldsymbol{\alpha}^k$ is an optimal solution of (1), stop. Otherwise, find a *two-element* working set $B = \{i, j\} \subset \{1, \ldots, l\}$. Define $N \equiv \{1, \ldots, l\} \backslash B$ and $\boldsymbol{\alpha}_B^k$ and $\boldsymbol{\alpha}_N^k$ to be sub-vectors of $\boldsymbol{\alpha}^k$ corresponding to B and N, respectively.

3. Solve the following sub-problem with the variable $\boldsymbol{\alpha}_B$:

$$\begin{aligned}
\min_{\boldsymbol{\alpha}_B} \quad & \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}_B^T & (\boldsymbol{\alpha}_N^k)^T\end{bmatrix}\begin{bmatrix}Q_{BB} & Q_{BN}\\ Q_{NB} & Q_{NN}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_B\\ \boldsymbol{\alpha}_N^k\end{bmatrix} - \begin{bmatrix}\mathbf{e}_B^T & \mathbf{e}_N^T\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_B\\ \boldsymbol{\alpha}_N^k\end{bmatrix}\\
& = \frac{1}{2}\boldsymbol{\alpha}_B^T Q_{BB}\boldsymbol{\alpha}_B + (-\mathbf{e}_B + Q_{BN}\boldsymbol{\alpha}_N^k)^T\boldsymbol{\alpha}_B + \text{ constant}\\
& = \frac{1}{2}\begin{bmatrix}\alpha_i & \alpha_j\end{bmatrix}\begin{bmatrix}Q_{ii} & Q_{ij}\\ Q_{ij} & Q_{jj}\end{bmatrix}\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} + (-\mathbf{e}_B + Q_{BN}\boldsymbol{\alpha}_N^k)^T\begin{bmatrix}\alpha_i\\ \alpha_j\end{bmatrix} + \text{ constant}\\
\text{subject to} \quad & 0 \le \alpha_i, \alpha_j \le C, \qquad (2)\\
& y_i\alpha_i + y_j\alpha_j = -\mathbf{y}_N^T\boldsymbol{\alpha}_N^k,
\end{aligned}$$

   where $\begin{bmatrix}Q_{BB} & Q_{BN}\\ Q_{NB} & Q_{NN}\end{bmatrix}$ is a permutation of the matrix Q.

4. Set $\boldsymbol{\alpha}_B^{k+1}$ to be the optimal solution of (2) and $\boldsymbol{\alpha}_N^{k+1} \equiv \boldsymbol{\alpha}_N^k$. Set $k \leftarrow k + 1$ and goto Step 2.

Note that the set B changes from one iteration to another, but to simplify the notation, we just use B instead of B^k.

Since only few components are updated per iteration, for difficult problems, the decomposition method suffers from slow convergences. Better methods of working set selection could reduce the number of iterations and hence are an important research issue. Existing methods mainly rely on the violation of the optimality condition, which also corresponds to first order (i.e., gradient) information of the objective function. Past optimization research indicates that using second order information generally leads to faster convergence. Now (1) is a quadratic programming problem, so second order information directly relates to the decrease of the objective function. There are several attempts (e.g., Lai et al., 2003a,b) to find working sets based on the reduction of the objective value, but these selection methods are only heuristics without convergence proofs. Moreover, as such techniques cost more than existing ones, fewer iterations may not lead to shorter training time. This paper develops a simple working set selection using second order information. It can be extended for indefinite kernel matrices. Experiments demonstrate that the training time is shorter than existing implementations.

This paper is organized as follows. In Section 2, we discuss existing methods of working set selection and propose a new strategy. Theoretical properties of using the new selection technique are in Section 3. In Section 4 we extend the proposed selection method to other SVM formulas such as ν-SVM. A detailed experiment is in Section 5. We then in Section 6 discuss and compare some variants of the proposed selection method. Finally, Section 7 concludes this research work. A pseudo code of the proposed method is in the Appendix.

2. Existing and New Working Set Selections

In this section, we discuss existing methods of working set selection and then propose a new approach.

2.1 Existing Selections

Currently a popular way to select the working set B is via the "maximal violating pair:"

WSS 1 (Working set selection via the "maximal violating pair")

1. Select

$$\begin{aligned} i &\in \arg\max_t \{-y_t \nabla f(\boldsymbol{\alpha}^k)_t \mid t \in I_{\text{up}}(\boldsymbol{\alpha}^k)\}, \\ j &\in \arg\min_t \{-y_t \nabla f(\boldsymbol{\alpha}^k)_t \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k)\}, \end{aligned}$$

where

$$\begin{aligned} I_{\text{up}}(\boldsymbol{\alpha}) &\equiv \{t \mid \alpha_t < C, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\}, \text{ and} \\ I_{\text{low}}(\boldsymbol{\alpha}) &\equiv \{t \mid \alpha_t < C, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\}. \end{aligned} \tag{3}$$

2. Return $B = \{i, j\}$.

This working set was first proposed in Keerthi et al. (2001) and is used in, for example, the software LIBSVM (Chang and Lin, 2001). WSS 1 can be derived through the Karush-Kuhn-Tucker (KKT) optimality condition of (1): A vector $\boldsymbol{\alpha}$ is a stationary point of (1) if and only if there is a number b and two nonnegative vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ such that

$$\begin{gathered} \nabla f(\boldsymbol{\alpha}) + b\mathbf{y} = \boldsymbol{\lambda} - \boldsymbol{\mu}, \\ \lambda_i \alpha_i = 0, \mu_i (C - \alpha_i) = 0, \lambda_i \geq 0, \mu_i \geq 0, i = 1, \ldots, l, \end{gathered}$$

where $\nabla f(\boldsymbol{\alpha}) \equiv Q\boldsymbol{\alpha} - \mathbf{e}$ is the gradient of $f(\boldsymbol{\alpha})$. This condition can be rewritten as

$$\nabla f(\boldsymbol{\alpha})_i + b y_i \geq 0 \qquad \text{if } \alpha_i < C, \tag{4}$$

$$\nabla f(\boldsymbol{\alpha})_i + b y_i \leq 0 \qquad \text{if } \alpha_i > 0. \tag{5}$$

Since $y_i = \pm 1$, by defining $I_{\text{up}}(\boldsymbol{\alpha})$ and $I_{\text{low}}(\boldsymbol{\alpha})$ as in (3), and rewriting (4)-(5) to

$$\begin{aligned} -y_i \nabla f(\boldsymbol{\alpha})_i \leq b, \forall i \in I_{\text{up}}(\boldsymbol{\alpha}), \text{ and} \\ -y_i \nabla f(\boldsymbol{\alpha})_i \geq b, \forall i \in I_{\text{low}}(\boldsymbol{\alpha}), \end{aligned}$$

a feasible $\boldsymbol{\alpha}$ is a stationary point of (1) if and only if

$$m(\boldsymbol{\alpha}) \leq M(\boldsymbol{\alpha}), \tag{6}$$

where

$$m(\boldsymbol{\alpha}) \equiv \max_{i \in I_{\text{up}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i, \text{ and } M(\boldsymbol{\alpha}) \equiv \min_{i \in I_{\text{low}}(\boldsymbol{\alpha})} -y_i \nabla f(\boldsymbol{\alpha})_i.$$

Note that $m(\boldsymbol{\alpha})$ and $M(\boldsymbol{\alpha})$ are well defined except a rare situation where all $y_i = 1$ (or -1). In this case the zero vector is the only feasible solution of (1), so the decomposition method stops at the first iteration.

Following Keerthi et al. (2001), we define a "violating pair" of the condition (6).

Definition 1 (Violating pair) *If $i \in I_{\text{up}}(\boldsymbol{\alpha}), j \in I_{\text{low}}(\boldsymbol{\alpha})$, and $-y_i \nabla f(\boldsymbol{\alpha})_i > -y_j \nabla f(\boldsymbol{\alpha})_j$, then $\{i, j\}$ is a "violating pair."*

From (6), indices $\{i, j\}$ which most violate the optimality condition are a natural choice of the working set. They are called a "maximal violating pair" in WSS 1. It is known that violating pairs are important in the working set selection:

Theorem 2 (Hush and Scovel, 2003) *Assume Q is positive semi-definite. SMO-type methods have the strict decrease of the function value (i.e., $f(\boldsymbol{\alpha}^{k+1}) < f(\boldsymbol{\alpha}^k), \forall k$) if and only if B is a violating pair.*

Interestingly, the maximal violating pair is related to first order approximation of $f(\boldsymbol{\alpha})$. As explained below, $\{i, j\}$ selected via WSS 1 satisfies

$$\{i, j\} = \arg \min_{B:|B|=2} \text{Sub}(B), \tag{7}$$

where

$$\begin{aligned}
\text{Sub}(B) \equiv \min_{\mathbf{d}_B} \quad & \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B && \text{(8a)}\\
\text{subject to} \quad & \mathbf{y}_B^T \mathbf{d}_B = 0, \\
& d_t \geq 0, \text{ if } \alpha_t^k = 0, t \in B, && \text{(8b)}\\
& d_t \leq 0, \text{ if } \alpha_t^k = C, t \in B, && \text{(8c)}\\
& -1 \leq d_t \leq 1, t \in B. && \text{(8d)}
\end{aligned}$$

Problem (7) was first considered in Joachims (1998). By defining $\mathbf{d}^T \equiv [\mathbf{d}_B^T, \mathbf{0}_N^T]$, the objective function (8a) comes from minimizing *first order approximation* of $f(\boldsymbol{\alpha}^k + \mathbf{d})$:

$$\begin{aligned}
f(\boldsymbol{\alpha}^k + \mathbf{d}) &\approx f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)^T \mathbf{d} \\
&= f(\boldsymbol{\alpha}^k) + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B.
\end{aligned}$$

The constraint $\mathbf{y}_B^T \mathbf{d}_B = 0$ is from $\mathbf{y}^T(\boldsymbol{\alpha}^k + \mathbf{d}) = 0$ and $\mathbf{y}^T \boldsymbol{\alpha}^k = 0$. The condition $0 \leq \alpha_t \leq C$ leads to inequalities (8b) and (8c). As (8a) is a linear function, the inequalities $-1 \leq d_t \leq 1, t \in B$ avoid that the objective value goes to $-\infty$.

A first look at (7) indicates that we may have to check all $\binom{l}{2}$ B's in order to find an optimal set. Instead, WSS 1 efficiently solves (7) in $O(l)$ steps. This result is discussed in Lin (2001a, Section II), where more general settings ($|B|$ is any even integer) are considered. The proof for $|B| = 2$ is easy, so we give it in Appendix A for completeness.

The convergence of the decomposition method using WSS 1 is proved in Lin (2001a, 2002).

2.2 A New Working Set Selection

Instead of using first order approximation, we may consider more accurate second order information. As f is a quadratic,

$$\begin{aligned}
f(\boldsymbol{\alpha}^k + \mathbf{d}) - f(\boldsymbol{\alpha}^k) &= \nabla f(\boldsymbol{\alpha}^k)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 f(\boldsymbol{\alpha}^k) \mathbf{d} \\
&= \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B + \frac{1}{2} \mathbf{d}_B^T \nabla^2 f(\boldsymbol{\alpha}^k)_{BB} \mathbf{d}_B
\end{aligned} \tag{9}$$

is exactly the reduction of the objective value. Thus, by replacing the objective function of (8) with (9), a selection method using *second order information* is

$$\min_{B:|B|=2} \text{Sub}(B), \tag{10}$$

where

$$\begin{aligned} \text{Sub}(B) \equiv \min_{\mathbf{d}_B} \quad & \frac{1}{2}\mathbf{d}_B^T \nabla^2 f(\boldsymbol{\alpha}^k)_{BB}\mathbf{d}_B + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B && \text{(11a)} \\ \text{subject to} \quad & \mathbf{y}_B^T \mathbf{d}_B = 0, && \text{(11b)} \\ & d_t \geq 0, \text{ if } \alpha_t^k = 0, t \in B, && \text{(11c)} \\ & d_t \leq 0, \text{ if } \alpha_t^k = C, t \in B. && \text{(11d)} \end{aligned}$$

Note that inequality constraints $-1 \leq d_t \leq 1, t \in B$ in (8) are removed, as later we will see that the optimal value of (11) does not go to $-\infty$. Though one expects (11) is better than (8), $\min_{B:|B|=2} \text{Sub}(B)$ in (10) becomes a challenging task. Unlike (7)-(8), which can be efficiently solved by WSS 1, for (10) and (11) there is no available way to avoid checking all $\binom{l}{2}$ B's. Note that except the working set selection, the main task per decomposition iteration is on calculating the two kernel columns Q_{ti} and Q_{tj}, $t = 1, \ldots, l$. This requires $O(l)$ operations and is needed only if Q is not stored. Therefore, each iteration can become l times more expensive if an $O(l^2)$ working set selection is used. Moreover, from simple experiments we know that the number of iterations is, however, not decreased l times. Therefore, an $O(l^2)$ working set selection is impractical.

A viable implementation of using second order information is thus to heuristically check several B's only. We propose the following new selection:

WSS 2 (Working set selection using second order information)

1. Select
$$i \in \arg\max_t \{-y_t \nabla f(\boldsymbol{\alpha}^k)_t \mid t \in I_{\text{up}}(\boldsymbol{\alpha}^k)\}.$$

2. Consider $\text{Sub}(B)$ defined in (11) and select
$$j \in \arg\min_t \{\text{Sub}(\{i, t\}) \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i\}. \tag{12}$$

3. Return $B = \{i, j\}$.

By using the same i as in WSS 1, we check only $O(l)$ possible B's to decide j. Alternatively, one may choose $j \in \arg M(\boldsymbol{\alpha}^k)$ and search for i by a way similar to (12)[1]. In fact, such a selection is the same as swapping labels $\mathbf{y}$ first and then applying WSS 2, so the performance should not differ much. It is certainly possible to consider other heuristics, and the main concern is how good they are if compared to the one by fully checking all $\binom{l}{2}$ sets. In Section 7 we will address this issue. Experiments indicate that a full check does not reduce iterations of using WSS 2 much. Thus WSS 2 is already a very good way of using second order information.

1. To simplify the notations, we denote $\arg M(\boldsymbol{\alpha})$ as $\arg\min_{t \in I_{\text{low}}(\boldsymbol{\alpha})} -y_t \nabla f(\boldsymbol{\alpha})_t$ and $\arg m(\boldsymbol{\alpha})$ as $\arg\max_{t \in I_{\text{up}}(\boldsymbol{\alpha})} -y_t \nabla f(\boldsymbol{\alpha})_t$, respectively.

Despite the above issue of how to effectively use second order information, the real challenge is whether the new WSS 2 can cause shorter training time than WSS 1. Now the two selection methods differ only in selecting j, so we can also consider WSS 2 as a direct attempt to improve WSS 1. The following theorem shows that one could efficiently solve (11), so the working set selection WSS 2 does not cost a lot more than WSS 1.

Theorem 3 *If $B = \{i, j\}$ is a violating pair and $K_{ii} + K_{jj} - 2K_{ij} > 0$, then (11) has the optimal objective value*

$$-\frac{(-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j)^2}{2(K_{ii} + K_{jj} - 2K_{ij})}.$$

Proof Define $\hat{d}_i \equiv y_i d_i$ and $\hat{d}_j \equiv y_j d_j$. From $\mathbf{y}_B^T \mathbf{d}_B = 0$, we have $\hat{d}_i = -\hat{d}_j$ and

$$\begin{aligned} & \frac{1}{2}\begin{bmatrix} d_i & d_j \end{bmatrix}\begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix}\begin{bmatrix} d_i \\ d_j \end{bmatrix} + \begin{bmatrix} \nabla f(\boldsymbol{\alpha}^k)_i & \nabla f(\boldsymbol{\alpha}^k)_j \end{bmatrix}\begin{bmatrix} d_i \\ d_j \end{bmatrix} \\ = \; & \frac{1}{2}(K_{ii} + K_{jj} - 2K_{ij})\hat{d}_j^2 + (-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j)\hat{d}_j. \end{aligned} \tag{13}$$

Since $K_{ii} + K_{jj} - 2K_{ij} > 0$ and B is a violating pair, we can define

$$a_{ij} \equiv K_{ii} + K_{jj} - 2K_{ij} > 0 \quad \text{and} \quad b_{ij} \equiv -y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j > 0. \tag{14}$$

Then (13) has the minimum at

$$\hat{d}_j = -\hat{d}_i = -\frac{b_{ij}}{a_{ij}} < 0, \tag{15}$$

and

$$\text{the objective function (11a)} = -\frac{b_{ij}^2}{2a_{ij}}.$$

Moreover, we can show that $\hat{d}_i$ and $\hat{d}_j$ (d_i and d_j) indeed satisfy (11c)-(11d). If $j \in I_{\text{low}}(\boldsymbol{\alpha}^k)$, $\alpha_j^k = 0$ implies $y_j = -1$ and hence $d_j = y_j \hat{d}_j > 0$, a condition required by (11c). Other cases are similar. Thus $\hat{d}_i$ and $\hat{d}_i$ defined in (15) are optimal for (11). ■

Note that if K is positive definite, then for any $i \neq j$, $K_{ii} + K_{jj} - 2K_{ij} > 0$. Using Theorem 3, (12) in WSS 2 is reduced to a very simple form:

$$j \in \arg\min_t \left\{ -\frac{b_{it}^2}{a_{it}} \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i \right\},$$

where a_{it} and b_{it} are defined in (14). If K is not positive definite, the leading coefficient a_{ij} of (13) may be non-positive. This situation will be addressed in the next sub-section.

Note that (8) and (11) are used only for selecting the working set, so they do not have to maintain the feasibility $0 \leq \alpha_i^k + d_i \leq C, \forall i \in B$. On the contrary, feasibility must hold for the sub-problem (2) used to obtain $\boldsymbol{\alpha}^{k+1}$ after B is determined. There are some earlier attempts to use second order information for selecting working sets (e.g., Lai et al., 2003a,b), but they always check the feasibility. Then solving sub-problems during the

selection procedure is more complicated than solving the sub-problem (11). Besides, these earlier approaches do not provide any convergence analysis. In Section 6 we will investigate the issue of maintaining feasibility in the selection procedure, and explain why using (11) is better.

2.3 Non-Positive Definite Kernel Matrices

Theorem 3 does not hold if $K_{ii} + K_{jj} - 2K_{ij} \leq 0$. For the linear kernel, sometimes K is only positive semi-definite, so it is possible that $K_{ii} + K_{jj} - 2K_{ij} = 0$. Moreover, some existing kernel functions (e.g., sigmoid kernel) are not the inner product of two vectors, so K is even not positive semi-definite. Then $K_{ii} + K_{jj} - 2K_{ij} < 0$ may occur and (13) is a concave objective function.

Once B is decided, the same difficulty occurs for the sub-problem (2) to obtain $\boldsymbol{\alpha}^{k+1}$. Note that (2) differs from (11) only in constraints; (2) strictly requires the feasibility $0 \leq \alpha_i + d_i \leq C, \forall t \in B$. Therefore, (2) also has a concave objective function if $K_{ii} + K_{jj} - 2K_{ij} < 0$. In this situation, (2) may possess multiple local minima. Moreover, there are difficulties in proving the convergence of the decomposition methods (Palagi and Sciandrone, 2005; Chen et al., 2006). Thus, Chen et al. (2006) proposed adding an additional term to (2)'s objective function if $a_{ij} \equiv K_{ii} + K_{jj} - 2K_{ij} \leq 0$:

$$\begin{aligned} \min_{\alpha_i, \alpha_j} \quad & \frac{1}{2} \begin{bmatrix} \alpha_i & \alpha_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + (-\mathbf{e}_B + Q_{BN}\boldsymbol{\alpha}_N^k)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + \\ & \frac{\tau - a_{ij}}{4}((\alpha_i - \alpha_i^k)^2 + (\alpha_j - \alpha_j^k)^2) \\ \text{subject to} \quad & 0 \leq \alpha_i, \alpha_j \leq C, \\ & y_i\alpha_i + y_j\alpha_j = -\mathbf{y}_N^T \boldsymbol{\alpha}_N^k, \end{aligned} \tag{16}$$

where τ is a small positive number. By defining $\hat{d}_i \equiv y_i(\alpha_i - \alpha_i^k)$ and $\hat{d}_j \equiv y_j(\alpha_j - \alpha_j^k)$, (16)'s objective function, in a form similar to (13), is

$$\frac{1}{2}\tau\hat{d}_j^2 + b_{ij}\hat{d}_j, \tag{17}$$

where b_{ij} is defined as in (14). The new objective function is thus strictly convex. If $\{i, j\}$ is a violating pair, then a careful check shows that there is $\hat{d}_j < 0$ which leads to a negative value in (17) and maintains the feasibility of (16). Therefore, we can find $\boldsymbol{\alpha}^{k+1} \neq \boldsymbol{\alpha}^k$ satisfying $f(\boldsymbol{\alpha}^{k+1}) < f(\boldsymbol{\alpha}^k)$. More details are in Chen et al. (2006).

For selecting the working set, we consider a similar modification: If $B = \{i, j\}$ and a_{ij} is defined as in (14), then (11) is modified to:

$$\begin{aligned} \text{Sub}(B) \equiv \min_{d_B} \quad & \frac{1}{2}\mathbf{d}_B^T \nabla^2 f(\boldsymbol{\alpha}^k)_{BB}\mathbf{d}_B + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B + \frac{\tau - a_{ij}}{4}(d_i^2 + d_j^2) \\ \text{subject to} \quad & \text{constraints of (11).} \end{aligned} \tag{18}$$

Note that (18) differs from (16) only in constraints. In (18) we do not maintain the feasibility of $\alpha_t^k + d_t, t \in B$. We are allowed to do so because (18) is used only for identifying the working set B.

By reformulating (18) to (17) and following the same argument in Theorem 3, the optimal objective value of (18) is

$$-\frac{b_{ij}^2}{2\tau}.$$

Therefore, a generalized working set selection is as the following:

WSS 3
(Working set selection using second order information: any symmetric K)

1. Define a_{ts} and b_{ts} as in (14), and

$$\bar{a}_{ts} \equiv \begin{cases} a_{ts} & \text{if } a_{ts} > 0, \\ \tau & \text{otherwise.} \end{cases} \tag{19}$$

Select

$$\begin{aligned} i &\in \arg\max_t \{-y_t \nabla f(\boldsymbol{\alpha}^k)_t \mid t \in I_{\text{up}}(\boldsymbol{\alpha}^k)\}, \\ j &\in \arg\min_t \left\{ -\frac{b_{it}^2}{\bar{a}_{it}} \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i \right\}. \end{aligned} \tag{20}$$

2. Return $B = \{i, j\}$.

In summary, an SMO-type decomposition method using WSS 3 for the working set selection is:

Algorithm 2 (An SMO-type decomposition method using WSS 3)

1. Find $\boldsymbol{\alpha}^1$ as the initial feasible solution. Set $k = 1$.
2. If $\boldsymbol{\alpha}^k$ is a stationary point of (1), stop. Otherwise, find a working set $B = \{i, j\}$ by WSS 3.
3. Let a_{ij} be defined as in (14). If $a_{ij} > 0$, solve the sub-problem (2). Otherwise, solve (16). Set $\boldsymbol{\alpha}_B^{k+1}$ to be the optimal point of the sub-problem.
4. Set $\boldsymbol{\alpha}_N^{k+1} \equiv \boldsymbol{\alpha}_N^k$. Set $k \leftarrow k + 1$ and goto Step 2.

In the next section we study theoretical properties of using WSS 3.

3. Theoretical Properties

To obtain theoretical properties of using WSS 3, we consider the work (Chen et al., 2006), which gives a general study of SMO-type decomposition methods. It considers Algorithm 2 but replaces WSS 3 with a general working set selection[2]:

WSS 4 (A general working set selection discussed in Chen et al., 2006)

1. Consider a fixed $0 < \sigma \leq 1$ for all iterations.

2. In fact, Chen et al. (2006) consider an even more general framework for selecting working sets, but for easy description, we discuss WSS 4 here.

2. Select any $i \in I_{\text{up}}(\boldsymbol{\alpha}^k), j \in I_{\text{low}}(\boldsymbol{\alpha}^k)$ satisfying

$$-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j \geq \sigma(m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k)) > 0. \tag{21}$$

3. Return $B = \{i, j\}$.

Clearly (21) ensures the quality of the selected pair by linking it to the maximal violating pair. It is easy to see that WSS 3 is a special case of WSS 4: Assume $B = \{i, j\}$ is the set returned from WSS 3 and $\bar{j} \in \arg M(\boldsymbol{\alpha}^k)$. Since WSS 3 selects $i \in \arg m(\boldsymbol{\alpha}^k)$, with $\bar{a}_{ij} > 0$ and $\bar{a}_{i\bar{j}} > 0$, (20) in WSS 3 implies

$$\frac{-(-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j)^2}{\bar{a}_{ij}} \leq \frac{-(m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k))^2}{\bar{a}_{i\bar{j}}}.$$

Thus,

$$-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j \geq \sqrt{\frac{\min_{t,s} \bar{a}_{t,s}}{\max_{t,s} \bar{a}_{t,s}}}(m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k)),$$

an inequality satisfying (21) for $\sigma = \sqrt{\min_{t,s} \bar{a}_{t,s} / \max_{t,s} \bar{a}_{t,s}}$.

Therefore, all theoretical properties proved in Chen et al. (2006) hold here. They are listed below.

It is known that the decomposition method may not converge to an optimal solution if using improper methods of working set selection. We thus must prove that the proposed selection leads to the convergence.

Theorem 4 (Asymptotic convergence (Chen et al., 2006, Theorem 3 and Corollary 1))

Let $\{\boldsymbol{\alpha}^k\}$ be the infinite sequence generated by the SMO-type method Algorithm 2. Then any limit point of $\{\boldsymbol{\alpha}^k\}$ is a stationary point of (1)*. Moreover, if Q is positive definite, $\{\boldsymbol{\alpha}^k\}$ globally converges to the unique minimum of* (1)*.*

As the decomposition method only asymptotically approaches an optimum, in practice, it is terminated after satisfying a stopping condition. For example, we can pre-specify a small tolerance $\epsilon > 0$ and check if the maximal violation is small enough:

$$m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k) \leq \epsilon. \tag{22}$$

Alternatively, one may check if the selected working set $\{i, j\}$ satisfies

$$-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j \leq \epsilon, \tag{23}$$

because (21) implies $m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k) \leq \epsilon/\sigma$. These are reasonable stopping criteria due to their closeness to the optimality condition (6). To avoid an infinite loop, we must have that under any $\epsilon > 0$, Algorithm 2 stops in a finite number of iterations. The finite termination of using (22) or (23) as the stopping condition is implied by (26) of Theorem 5 stated below.

Shrinking and caching (Joachims, 1998) are two effective techniques to make the decomposition method faster. The former removes some bounded components during iterations, so smaller reduced problems are considered. The latter allocates some memory space (called cache) to store recently used Q_{ij}, and may significantly reduce the number of kernel evaluations. The following theorem explains why these two techniques are useful in practice:

Theorem 5 (Finite termination and explanation of caching and shrinking techniques (Chen et al., 2006, Theorems 4 and 6))

Assume Q is positive semi-definite.

1. *The following set is independent of any optimal solution $\bar{\boldsymbol{\alpha}}$:*

$$I \equiv \{i \mid -y_i \nabla f(\bar{\boldsymbol{\alpha}})_i > M(\bar{\boldsymbol{\alpha}}) \text{ or } -y_i \nabla f(\bar{\boldsymbol{\alpha}})_i < m(\bar{\boldsymbol{\alpha}})\}. \tag{24}$$

   Problem (1) *has unique and bounded optimal solutions at $\alpha_i, i \in I$.*

2. *Assume Algorithm 2 generates an infinite sequence $\{\boldsymbol{\alpha}^k\}$. There is $\bar{k}$ such that after $k \geq \bar{k}$, every $\alpha_i^k, i \in I$ has reached the unique and bounded optimal solution. It remains the same in all subsequent iterations and $\forall k \geq \bar{k}$:*

$$i \notin \{t \mid M(\boldsymbol{\alpha}^k) \leq -y_t \nabla f(\boldsymbol{\alpha}^k)_t \leq m(\boldsymbol{\alpha}^k)\}. \tag{25}$$

3. *If* (1) *has an optimal solution $\bar{\boldsymbol{\alpha}}$ satisfying $m(\bar{\boldsymbol{\alpha}}) < M(\bar{\boldsymbol{\alpha}})$, then $\bar{\boldsymbol{\alpha}}$ is the unique solution and Algorithm 2 reaches it in a finite number of iterations.*

4. *If $\{\boldsymbol{\alpha}^k\}$ is an infinite sequence, then the following two limits exist and are equal:*

$$\lim_{k\to\infty} m(\boldsymbol{\alpha}^k) = \lim_{k\to\infty} M(\boldsymbol{\alpha}^k) = m(\bar{\boldsymbol{\alpha}}) = M(\bar{\boldsymbol{\alpha}}), \tag{26}$$

   where $\bar{\boldsymbol{\alpha}}$ is any optimal solution.

Finally, the following theorem shows that Algorithm 2 is linearly convergent under some assumptions:

Theorem 6 (Linear convergence (Chen et al., 2006, Theorem 8))

Assume problem (1) *satisfies*

1. *Q is positive definite. Therefore, (1) has a unique optimal solution $\bar{\boldsymbol{\alpha}}$.*

2. *The nondegenency condition. That is, the optimal solution $\bar{\boldsymbol{\alpha}}$ satisfies that*

$$\nabla f(\bar{\boldsymbol{\alpha}})_i + \bar{b} y_i = 0 \text{ if and only if } 0 < \bar{\alpha}_i < C, \tag{27}$$

   where $\bar{b} = m(\bar{\boldsymbol{\alpha}}) = M(\bar{\boldsymbol{\alpha}})$ according to Theorem 5.

For the sequence $\{\boldsymbol{\alpha}^k\}$ generated by Algorithm 2, there are $c < 1$ and $\bar{k}$ such that for all $k \geq \bar{k}$,

$$f(\boldsymbol{\alpha}^{k+1}) - f(\bar{\boldsymbol{\alpha}}) \leq c(f(\boldsymbol{\alpha}^k) - f(\bar{\boldsymbol{\alpha}})).$$

This theorem indicates how fast the SMO-type method Algorithm 2 converges. For any fixed problem (1) and a given tolerance ϵ, there is $\bar{k}$ such that within

$$\bar{k} + O(\log(1/\epsilon))$$

iterations,

$$|f(\boldsymbol{\alpha}^k) - f(\bar{\boldsymbol{\alpha}})| \leq \epsilon.$$

Note that $O(\log(1/\epsilon))$ iterations are necessary for decomposition methods according to the analysis in Lin (2001b)[3]. Hence the result of linear convergence here is already the best worst case analysis.

3. Lin (2001b) gave a three-variable example and explained that the SMO-type method using WSS 1 is linearly convergent. A careful check shows that the same result holds for any method of working set selection.

4. Extensions

The proposed WSS 3 can be directly used for training support vector regression (SVR) and one-class SVM because they solve problems similar to (1). More detailed discussion about applying WSS 4 (and hence WSS 3) to SVR and one-class SVM is in Chen et al. (2006, Section IV).

Another formula which needs special attention is ν-SVM (Schölkopf et al., 2000), which solves a problem with one more linear constraint:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} \\ \text{subject to} \quad & \mathbf{y}^T \boldsymbol{\alpha} = 0, \\ & \mathbf{e}^T \boldsymbol{\alpha} = \nu, \\ & 0 \le \alpha_i \le 1/l, i = 1, \ldots, l, \end{aligned} \tag{28}$$

where $\mathbf{e}$ is the vector of all ones and $0 \le \nu \le 1$.

Similar to (6), $\boldsymbol{\alpha}$ is a stationary point of (28) if and only if it satisfies

$$m_p(\boldsymbol{\alpha}) \le M_p(\boldsymbol{\alpha}) \text{ and } m_n(\boldsymbol{\alpha}) \le M_n(\boldsymbol{\alpha}), \tag{29}$$

where

$$\begin{aligned} m_p(\boldsymbol{\alpha}) &\equiv \max_{i \in I_{\text{up}}(\boldsymbol{\alpha}), y_i = 1} -y_i \nabla f(\boldsymbol{\alpha})_i, \qquad & M_p(\boldsymbol{\alpha}) &\equiv \min_{i \in I_{\text{low}}(\boldsymbol{\alpha}), y_i = 1} -y_i \nabla f(\boldsymbol{\alpha})_i, \text{ and} \\ m_n(\boldsymbol{\alpha}) &\equiv \max_{i \in I_{\text{up}}(\boldsymbol{\alpha}), y_i = -1} -y_i \nabla f(\boldsymbol{\alpha})_i, \qquad & M_n(\boldsymbol{\alpha}) &\equiv \min_{i \in I_{\text{low}}(\boldsymbol{\alpha}), y_i = -1} -y_i \nabla f(\boldsymbol{\alpha})_i. \end{aligned}$$

A detailed derivation is in, for example, Chen et al. (2006, Section VI).

In an SMO-type method for ν-SVM the selected working set $B = \{i, j\}$ must satisfy $y_i = y_j$. Otherwise, if $y_i \ne y_j$, then the two linear equalities make the sub-problem have only one feasible point $\boldsymbol{\alpha}_B^k$. Therefore, to select the working set, one considers positive (i.e., $y_i = 1$) and negative (i.e., $y_i = -1$) instances separately. Existing implementations such as LIBSVM (Chang and Lin, 2001) check violating pairs in each part and select the one with the largest violation. This strategy is an extension of WSS 1. By a derivation similar to that in Section 2, the selection can also be from first or second order approximation of the objective function. Using $\text{Sub}(\{i, j\})$ defined in (11), WSS 2 in Section 2 is modified to

WSS 5 (Extending WSS 2 for ν-SVM)

1. Find

$$\begin{aligned} & i_p \in \arg m_p(\boldsymbol{\alpha}^k), \\ & j_p \in \arg \min_t \{\text{Sub}(\{i_p, t\}) \mid y_t = 1, \boldsymbol{\alpha}_t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_{i_p} \nabla f(\boldsymbol{\alpha}^k)_{i_p}\}. \end{aligned}$$

2. Find

$$\begin{aligned} & i_n \in \arg m_n(\boldsymbol{\alpha}^k), \\ & j_n \in \arg \min_t \{\text{Sub}(\{i_n, t\}) \mid y_t = -1, \boldsymbol{\alpha}_t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_{i_n} \nabla f(\boldsymbol{\alpha}^k)_{i_n}\}. \end{aligned}$$

Problem	#data	#feat.	Problem	#data	#feat.	Problem	#data	#feat.
image	1,300	18	breast-cancer	690	10	abalone*	1,000	8
splice	1,000	60	diabetes	768	8	cadata*	1,000	8
tree	700	18	fourclass	862	2	cpusmall*	1,000	12
a1a	1,605	119	german.numer	1,000	24	mg	1,385	6
australian	683	14	w1a	2,477	300	space_ga*	1,000	6

Table 1: Data statistics for small problems (left two columns: classification, right column: regression). *: subset of the original problem.

3. Check $\text{Sub}(\{i_p, j_p\})$ and $\text{Sub}(\{i_n, j_n\})$. Return the set with a smaller value.

By Theorem 3 in Section 2, it is easy to solve $\text{Sub}(\{i_p, t\})$ and $\text{Sub}(\{i_n, t\})$ in the above procedure.

5. Experiments

In this section we aim at comparing the proposed WSS 3 with WSS 1, which selects the maximal violating pair. As indicated in Section 2, they differ only in finding the second element j: WSS 1 checks first order approximation of the objective function, but WSS 3 uses second order information.

5.1 Data and Experimental Settings

First, some small data sets (around 1,000 samples) including ten binary classification and five regression problems are investigated under various settings. Secondly, observations are further confirmed by using four large (more than 30,000 instances) classification problems. Data statistics are in Tables 1 and 3.

Problems german.numer and australian are from the Statlog collection (Michie et al., 1994). We select space_ga and cadata from StatLib (`http://lib.stat.cmu.edu/datasets`). The data sets image, diabetes, covtype, breast-cancer, and abalone are from the UCI machine learning repository (Newman et al., 1998). Problems a1a and a9a are compiled in Platt (1998) from the UCI "adult" data set. Problems w1a and w8a are also from Platt (1998). The tree data set was originally used in Bailey et al. (1993). The problem mg is a Mackey-Glass time series. The data sets cpusmall and splice are from the Delve archive (`http://www.cs.toronto.edu/~delve`). Problem fourclass is from Ho and Kleinberg (1996) and we further transform it to a two-class set. The problem IJCNN1 is from the first problem of IJCNN 2001 challenge (Prokhorov, 2001).

For most data sets each attribute is linearly scaled to $[-1, 1]$. We do not scale a1a, a9a, w1a, and w8a as they take two values 0 and 1. Another exception is covtype, in which 44 of 54 features have 0/1 values. We scale only the other ten features to $[0, 1]$. All data are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/`. We use LIBSVM (version 2.71) (Chang and Lin, 2001), an implementation of WSS 1, for experiments. An easy modification to WSS 3 ensures that two codes differ only in the working set implementation. We set $\tau = 10^{-12}$ in WSS 3.

Different SVM parameters such as C in (1) and kernel parameters affect the training time. It is difficult to evaluate the two methods under every parameter setting. To have a fair comparison, we simulate how one uses SVM in practice and consider the following procedure:

1. "Parameter selection" step: Conduct five-fold cross validation to find the best one within a given set of parameters.

2. "Final training" step: Train the whole set with the best parameter to obtain the final model.

For each step we check time and iterations using the two methods of working set selection. For some extreme parameters (e.g., very large or small values) in the "parameter selection" step, the decomposition method converges very slowly, so the comparison shows if the proposed WSS 3 saves time under difficult situations. On the other hand, the best parameter usually locates in a more normal region, so the "final training" step tests if WSS 3 is competitive with WSS 1 for easier cases.

The behavior of using different kernels is a concern, so we thoroughly test four commonly used kernels:

1. RBF kernel:
$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}.$$

2. Linear kernel:
$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j.$$

3. Polynomial kernel:
$$K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma(\mathbf{x}_i^T \mathbf{x}_j + 1))^d.$$

4. Sigmoid kernel:
$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + d).$$
Note that this function cannot be represented as $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ under some parameters. Then the matrix Q is not positive semi-definite. Experimenting with this kernel tests if our extension to indefinite kernels in Section 2.3 works well or not.

Parameters used for each kernel are listed in Table 2. Note that as SVR has an additional parameter ϵ, to save the running time, for other parameters we may not consider as many values as in classification.

It is important to check how WSS 3 performs after incorporating shrinking and caching strategies. Such techniques may effectively save kernel evaluations at each iteration, so the higher cost of WSS 3 is a concern. We consider various settings:

1. With or without shrinking.

2. Different cache size: First a 40MB cache allows the whole kernel matrix to be stored in the computer memory. Second, we allocate only 100K, so cache misses may happen and more kernel evaluations are needed. The second setting simulates the training of large-scale sets whose kernel matrices cannot be stored.

Kernel	Problem type	$\log_2 C$	$\log_2 \gamma$	$\log_2 \epsilon$	d
RBF	Classification	$-5, 15, 2$	$3, -15, -2$		
	Regression	$-1, 15, 2$	$3, -15, -2$	$-8, -1, 1$	
Linear	Classification	$-3, 5, 2$			
	Regression	$-3, 5, 2$		$-8, -1, 1$	
Polynomial	Classification	$-3, 5, 2$	$-5, -1, 1$		$2, 4, 1$
	Regression	$-3, 5, 2$	$-5, -1, 1$	$-8, -1, 1$	$2, 4, 1$
Sigmoid	Classification	$-3, 12, 3$	$-12, 3, 3$		$-2.4, 2.4, 0.6$
	Regression	$-3, 9, 3$	$\gamma = \frac{1}{\#\text{features}}$	$-8, -1, 3$	$-2.4, 2.4, 0.6$

Table 2: Parameters used for various kernels: values of each parameter are from a uniform discretization of an interval. We list the left, right end points and the space for discretization. For example, $-5, 15, 2$ for $\log_2 C$ means $\log_2 C = -5, -3, \ldots, 15$.

5.2 Results

For each kernel, we give two figures showing results of "parameter selection" and "final training" steps, respectively. We further separate each figure to two scenarios: without/with shrinking, and present three ratios between using WSS 3 and using WSS 1:

$$\begin{aligned}
\text{ratio 1} &\equiv \frac{\text{\# iter. by Alg. 2 with WSS 3}}{\text{\# iter. by Alg. 2 with WSS 1}}, \\
\text{ratio 2} &\equiv \frac{\text{time by Alg. 2 (WSS 3, 100K cache)}}{\text{time by Alg. 2 (WSS 1, 100K cache)}}, \\
\text{ratio 3} &\equiv \frac{\text{time by Alg. 2 (WSS 3, 40M cache)}}{\text{time by Alg. 2 (WSS 1, 40M cache)}}.
\end{aligned}$$

Note that the number of iterations is independent of the cache size. For the "parameter selection" step, time (or iterations) of all parameters is summed up before calculating the ratio. In general the "final training" step is very fast, so the timing result may not be accurate. Hence we repeat this step several times to obtain more reliable timing values. Figures 1-8 present obtained ratios. They are in general smaller than one, so using WSS 3 is really better than using WSS 1. Before describing other results, we explain an interesting observation: In these figures, if shrinking is not used, in general

$$\text{ratio 1} \le \text{ratio 2} \le \text{ratio 3}. \tag{30}$$

Under the two very different cache sizes, one is too small to store the kernel matrix, but the other is large enough. Thus, roughly we have

$$\begin{aligned}
&\text{time per Alg. 2 iteration (100K cache)} \approx \text{Calculating two } Q \text{ columns} + \text{Selection}, \\
&\text{time per Alg. 2 iteration (40M cache)} \approx \text{Selection}.
\end{aligned} \tag{31}$$

If shrinking is not used, the optimization problem is not reduced and hence

$$\frac{\text{time by Alg. 2}}{\text{\# iter. of Alg. 2}} \approx \text{cost per iteration} \approx \text{ constant}. \tag{32}$$

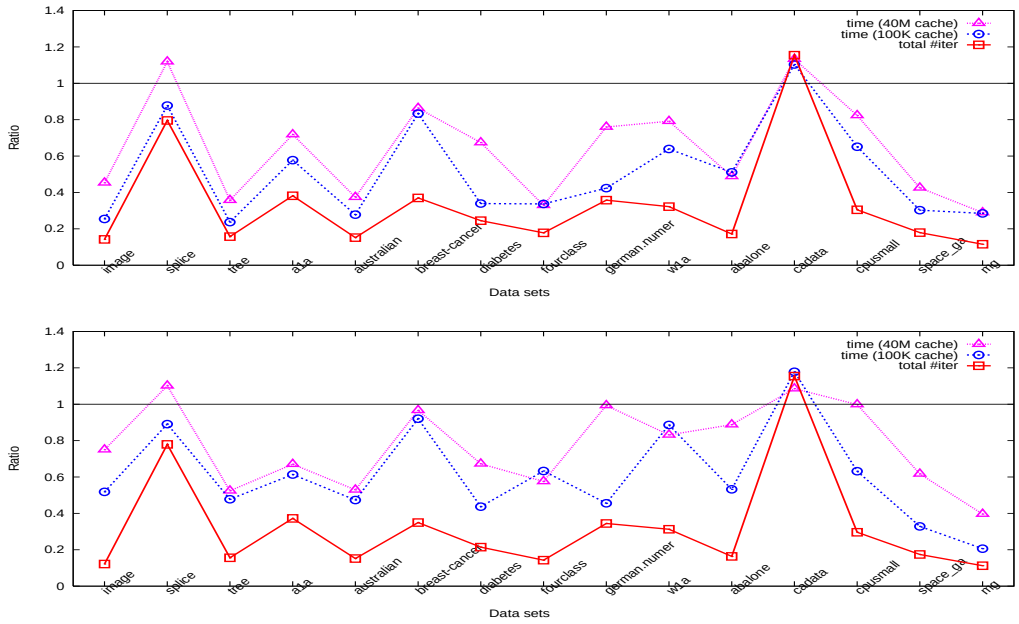


Figure 1: Iteration and time ratios between WSS 3 and 1 using the RBF kernel for the "parameter selection" step (top: without shrinking, bottom: with shrinking).

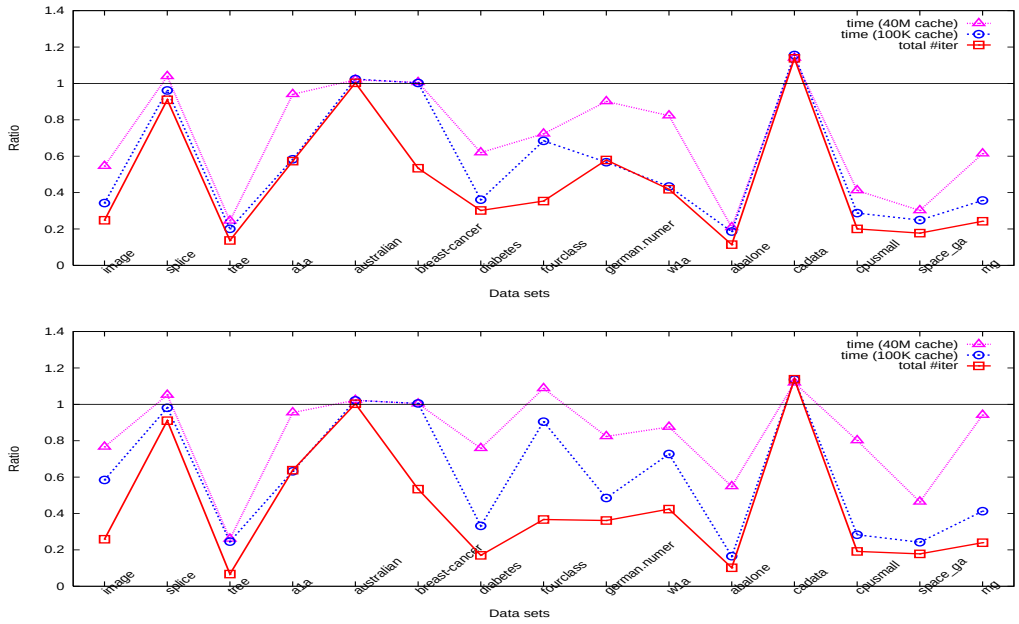


Figure 2: Iteration and time ratios between WSS 3 and 1 using the RBF kernel for the "final training" step (top: without shrinking, bottom: with shrinking).

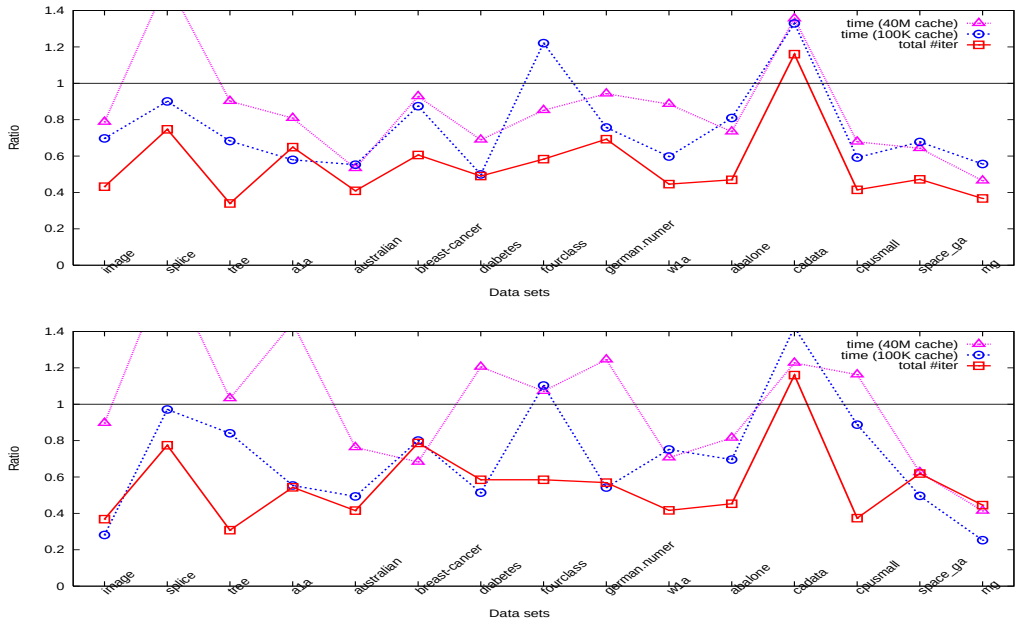


Figure 3: Iteration and time ratios between WSS 3 and 1 using the linear kernel for the "parameter selection" step (top: without shrinking, bottom: with shrinking).

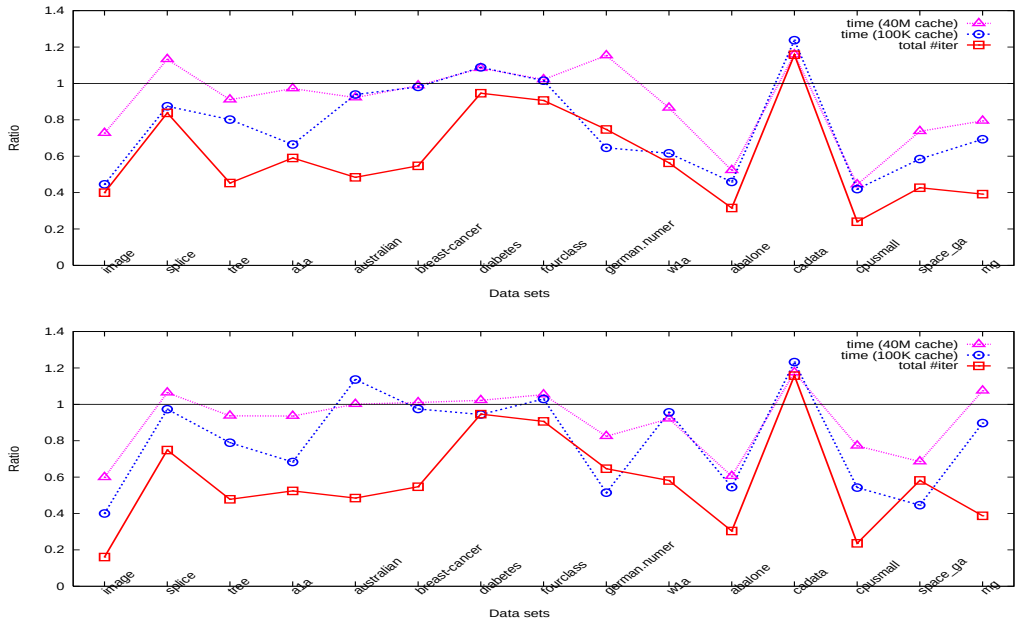


Figure 4: Iteration and time ratios between WSS 3 and 1 using the linear kernel for the "final training" step (top: without shrinking, bottom: with shrinking).

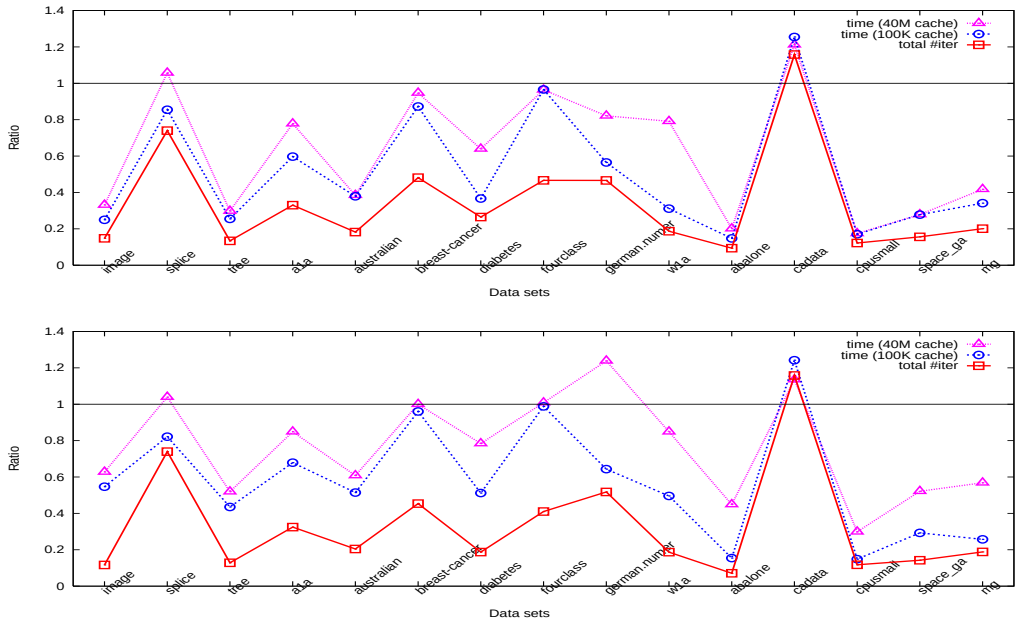


Figure 5: Iteration and time ratios between WSS 3 and 1 using the polynomial kernel for the "parameter selection" step (top: without shrinking, bottom: with shrinking).

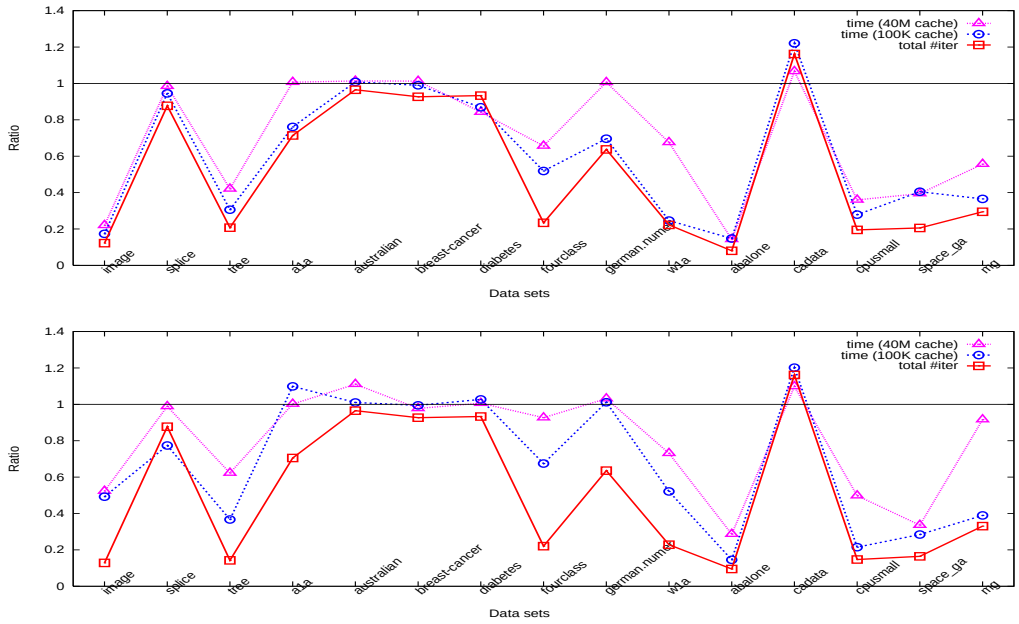


Figure 6: Iteration and time ratios between WSS 3 and 1 using the polynomial kernel for the "final training" step (top: without shrinking, bottom: with shrinking).

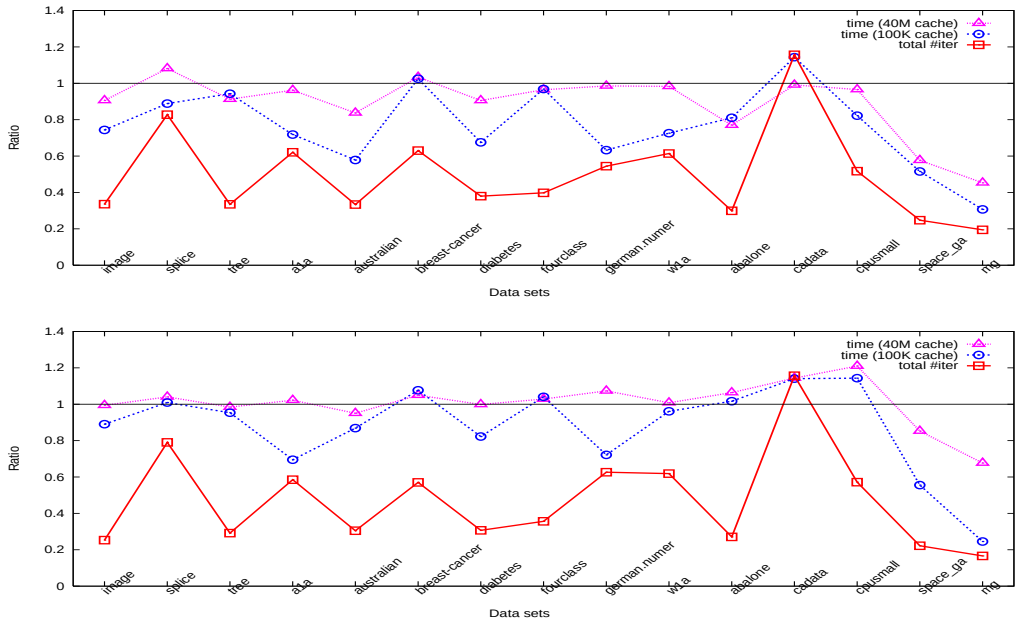


Figure 7: Iteration and time ratios between WSS 3 and 1 using the sigmoid kernel for the "parameter selection" step (top: without shrinking, bottom: with shrinking).

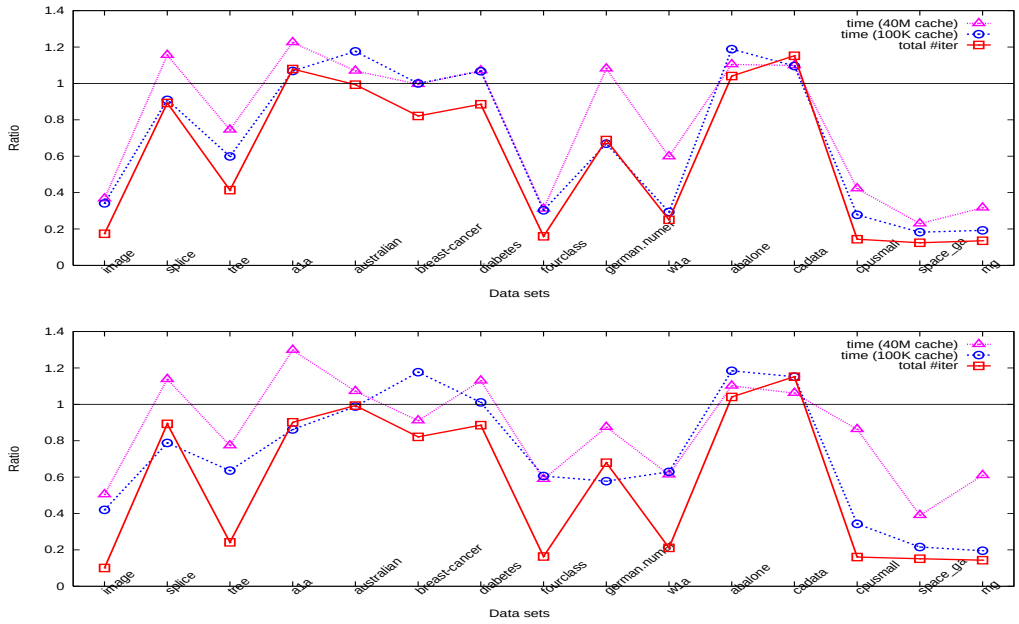


Figure 8: Iteration and time ratios between WSS 3 and 1 using the sigmoid kernel for the "final training" step (top: without shrinking, bottom: with shrinking).

			RBF kernel				Linear kernel			
			Shrinking		No-Shrinking		Shrinking		No-Shrinking	
Problem	#data	#feat.	Iter.	Time	Iter.	Time	Iter.	Time	Iter.	Time
a9a	32,561	123	0.73	0.92	0.75	0.93	0.86	0.92	0.88	0.95
w8a	49,749	300	0.48	0.72	0.50	0.81	0.47	0.90	0.53	0.79
IJCNN1	49,990	22	0.09	0.68	0.11	0.43	0.37	0.91	0.41	0.74
covtype*	100,000	54	0.37	0.90	0.37	0.76	0.19	0.59	0.22	0.52

Table 3: Large problems: Iteration and time ratios between WSS 3 and WSS 1 for the 16-point parameter selection. *: subset of a two-class data transformed from the original multi-class problem.

Since WSS 3 costs more than WSS 1, with (31),

$$1 \leq \frac{\text{time per Alg. 2 iteration (WSS 3, 100K cache)}}{\text{time per Alg. 2 iteration (WSS 1, 100K cache)}} \leq \frac{\text{time per Alg. 2 iteration (WSS 3, 40M cache)}}{\text{time per Alg. 2 iteration (WSS 1, 40M cache)}}.$$

This and (32) then imply (30). When shrinking is incorporated, the cost per iteration varies and (32) may not hold. Thus, though the relationship (30) in general still holds, there are more exceptions.

With the above analysis, our main observations and conclusions from Figures 1-8 are in the following:

1. Using WSS 3 significantly reduces the number of iterations. The reduction is more dramatic for the "parameter selection" step, where some points have slow convergence.

2. The new method is in general faster. Using a smaller cache gives better improvement. When the cache is not enough to store the whole kernel matrix, kernel evaluations are the main cost per iteration. Thus the time reduction is closer to the iteration reduction. This property hints that WSS 3 is useful on large-scale sets for which kernel matrices are too huge to be stored.

3. The implementation without shrinking gives better timing improvement than that with, even though they have similar iteration reduction. Shrinking successfully reduces the problem size and hence the memory use. Then similar to having enough cache, the time reduction does not match that of iterations due to the higher cost on selecting the working set per iteration. Therefore, results in Figures 1-8 indicate that with effective shrinking and caching implementations, it is difficult to have a new selection rule systematically surpassing WSS 1. The superior performance of WSS 3 thus makes important progress in training SVMs.

Next we experiment with large classification sets by a similar procedure. As the parameter selection is time consuming, we first use 10% training instances to identify a small region of good parameters. Then a 16-point search using the whole set is performed. The cache

size is 350M except 800M for covtype. We experiment with RBF and linear kernels. Table 3 gives iteration and time ratios of conducting the 16-point parameter selection. Similar to results for small problems, the number of iterations using WSS 3 is much smaller than that of using WSS 1. The training time of using WSS 3 is also shorter.

6. Maintaining Feasibility in Sub-problems for Working Set Selections

In Section 2, both the linear sub-problem (8) and quadratic sub-problem (11) do not require $\boldsymbol{\alpha}^k + \mathbf{d}$ to be feasible. One may wonder if enforcing the feasibility gives a better working set and hence leads to faster convergence. In this situation, the quadratic sub-problem becomes

$$\begin{aligned} \text{Sub}(B) \equiv \min_{\mathbf{d}_B} \quad & \frac{1}{2}\mathbf{d}_B^T \nabla^2 f(\boldsymbol{\alpha}^k)_{BB}\mathbf{d}_B + \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B \\ \text{subject to} \quad & \mathbf{y}_B^T \mathbf{d}_B = 0, \\ & -\alpha_t^k \le d_t \le C - \alpha_t^k, \forall t \in B. \end{aligned} \tag{33}$$

For example, from some candidate pairs, Lai et al. (2003a,b) select the one with the smallest value of (33) as the working set. To check the effect of using (33), here we replace (11) in WSS 2 with (33) and compare it with the original WSS 2.

From (9), a nice property of using (33) is that $\text{Sub}(B)$ equals the decrease of the objective function f by moving from $\boldsymbol{\alpha}^k$ to another feasible point $\boldsymbol{\alpha}^k + \mathbf{d}$. In fact, once B is determined, (33) is also the sub-problem (2) used in Algorithm 1 to obtain $\boldsymbol{\alpha}^{k+1}$. Therefore, we use the same sub-problem for both selecting the working set and obtaining the next iteration $\boldsymbol{\alpha}^{k+1}$. One may think that such a selection method is better as it leads to the largest function value reduction while maintaining the feasibility. However, solving (33) is more expensive than (11) since checking the feasibility requires additional effort. To be more precise, if $B = \{i, j\}$, using $\hat{d}_j = -\hat{d}_i = y_j d_j = -y_i d_i$ and a derivation similar to (13), we now must minimize $\frac{1}{2}\bar{a}_{ij}\hat{d}_j^2 + b_{ij}\hat{d}_j$ under the constraints

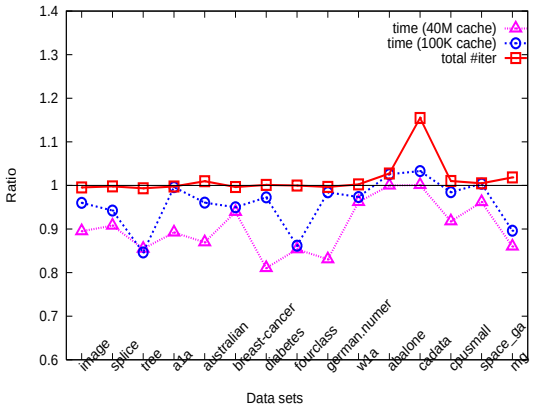
$$-\alpha_j^k \le d_j = y_j \hat{d}_j \le C - \alpha_j^k \text{ and } -\alpha_i^k \le d_i = -y_i \hat{d}_j \le C - \alpha_i^k. \tag{34}$$

As the minimum of the objective function happens at $-b_{ij}/\bar{a}_{ij}$, to have a solution satisfying (34), we multiply it by $-y_i$ and check the second constraint. Next, by $d_j = -y_i y_j d_i$, we check the first constraint. Equation (20) in WSS 3 is thus modified to

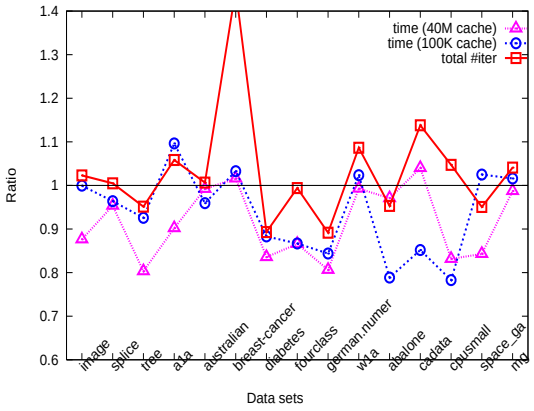
$$\begin{aligned} j \in \arg\min_t \Big\{ & \frac{1}{2}\bar{a}_{it}\hat{d}_t^2 + b_{it}\hat{d}_t \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i, \\ & \hat{d}_t = y_t \max(-\alpha_t^k, \min(C - \alpha_t^k, -y_t y_i \max(-\alpha_i^k, \min(C - \alpha_i^k, y_i b_{it}/\bar{a}_{it}))))\Big\}. \end{aligned} \tag{35}$$

Clearly (35) requires more operations than (20).

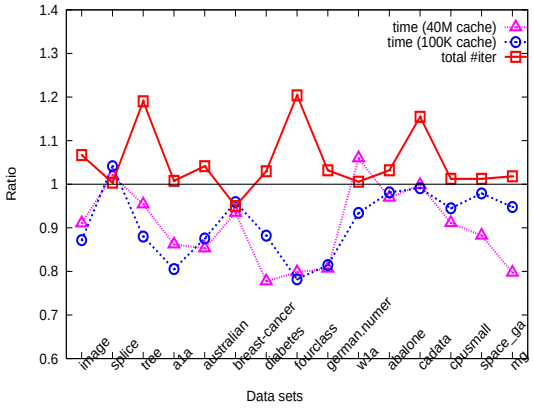
In this section, we prove that under some minor assumptions, in final iterations, solving (11) in WSS 2 is the same as solving (33). This result and experiments then indicate that there is no need to use the more sophisticated sub-problem (33) for selecting working sets.



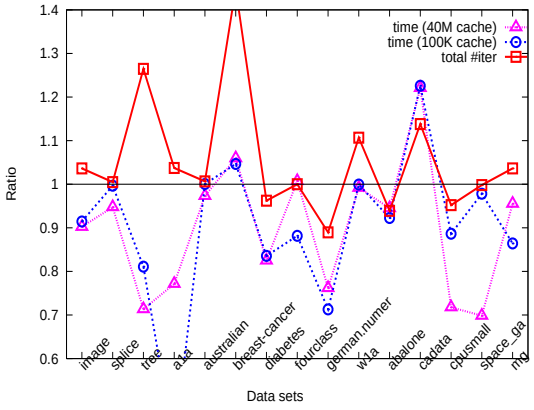
(a) The "parameter selection" step without shrinking



(b) The "final training" step without shrinking



(c) The "parameter selection" step with shrinking



(d) The "final training" step with shrinking

Figure 9: Iteration and time ratios between using (11) and (33) in WSS 2. Note that the ratio (y-axis) starts from 0.6 but not 0.

6.1 Solutions of (11) and (33) in final iterations

Theorem 7 *Let $\{\boldsymbol{\alpha}^k\}$ be the infinite sequence generated by the SMO-type decomposition method using WSS 2. Under the same assumptions of Theorem 6, there is $\bar{k}$ such that for $k \geq \bar{k}$, WSS 2 returns the same working set by replacing* (11) *with* (33).

Proof Since K is assumed to be positive definite, problem (1) has a unique optimal solution $\bar{\boldsymbol{\alpha}}$. Using Theorem 4,

$$\lim_{k \to \infty} \boldsymbol{\alpha}^k = \bar{\boldsymbol{\alpha}}. \tag{36}$$

Since $\{\boldsymbol{\alpha}^k\}$ is an infinite sequence, Theorem 5 shows that $m(\bar{\boldsymbol{\alpha}}) = M(\bar{\boldsymbol{\alpha}})$. Hence we can define the following set

$$I' \equiv \{t \mid -y_t \nabla f(\bar{\boldsymbol{\alpha}})_t = m(\bar{\boldsymbol{\alpha}}) = M(\bar{\boldsymbol{\alpha}})\}.$$

As $\bar{\boldsymbol{\alpha}}$ is a non-degenerate point, from (27),

$$\delta \equiv \min_{t \in I'}(\bar{\alpha}_t, C - \bar{\alpha}_t) > 0.$$

Using 1) Eq. (36), 2) $\nabla f(\bar{\boldsymbol{\alpha}})_i = \nabla f(\bar{\boldsymbol{\alpha}})_j, \forall i, j \in I'$, and 3) Eq. (25) of Theorem 5, there is $\bar{k}$ such that for all $k \geq \bar{k}$,

$$|\alpha_i^k - \bar{\alpha}_i| < \frac{\delta}{2}, \forall i \in I', \tag{37}$$

$$\frac{|-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j|}{K_{ii} + K_{jj} - 2K_{ij}} < \frac{\delta}{2}, \forall i, j \in I', K_{ii} + K_{jj} - 2K_{ij} > 0, \tag{38}$$

and

$$\text{all violating pairs come from } I'. \tag{39}$$

For any given index pair B, let $\text{Sub}_{(11)}(B)$ and $\text{Sub}_{(33)}(B)$ denote the optimal objective values of (11) and (33), respectively. If $\bar{B} = \{i, j\}$ is a violating pair selected by WSS 2 at the kth iteration, (37)-(39) imply that d_i and d_j defined in (15) satisfy

$$0 < \alpha_i^{k+1} = \alpha_i^k + d_i < C \text{ and } 0 < \alpha_j^{k+1} = \alpha_j^k + d_j < C. \tag{40}$$

Therefore, the optimal $\mathbf{d}_{\bar{B}}$ of (11) is feasible for (33). That is,

$$\text{Sub}_{(33)}(\bar{B}) \leq \text{Sub}_{(11)}(\bar{B}). \tag{41}$$

Since (33)'s constraints are stricter than those of (11), we have

$$\text{Sub}_{(11)}(B) \leq \text{Sub}_{(33)}(B), \forall B. \tag{42}$$

From WSS 3,

$$j \in \arg\min_t \{\text{Sub}_{(11)}(\{i, t\}) \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i\}.$$

With (41) and (42), this j satisfies

$$j \in \arg\min_t \{\text{Sub}_{(33)}(\{i, t\}) \mid t \in I_{\text{low}}(\boldsymbol{\alpha}^k), -y_t \nabla f(\boldsymbol{\alpha}^k)_t < -y_i \nabla f(\boldsymbol{\alpha}^k)_i\}.$$

Therefore, replacing (11) in WSS 3 with (33) does not affect the selected working set. ■

This theorem indicates that the two methods of working set selection in general lead to a similar number of iterations. As (11) does not check the feasibility, the implementation of using it should be faster.

6.2 Experiments

Under the framework WSS 2, we conduct experiments to check if using (11) is really faster than using (33). The same data sets in Section 5 are used under the same setting. For simplicity, we consider only the RBF kernel.

Similar to figures in Section 5, here Figure 9 presents iteration and time ratios between using (11) and (33):

$$\frac{\#\text{ iter. by using (11)}}{\#\text{ iter. by using (33)}}, \frac{\text{time by using (11) (100K cache)}}{\text{time by using (33) (100K cache)}}, \frac{\text{time by using (11) (40M cache)}}{\text{time by using (33) (40M cache)}}.$$

Without shrinking, clearly both approaches have very similar numbers of iterations. This observation is expected due to Theorem 7. Then as (33) costs more than (11) does, the time ratio is in general smaller than one. Especially when the cache is large enough to store all kernel elements, selecting working sets is the main cost and hence the ratio is lower.

With shrinking, in Figures 9(c) and 9(d), the iteration ratio is larger than one for several problems. Surprisingly, the time ratio, especially that of using a small cache, is even smaller than that without shrinking. In other words, (11) better incorporates the shrinking technique than (33) does. To analyze this observation, we check the number of removed variables along iterations, and find that (11) leads to more aggressive shrinking. Then the reduced problem can be stored in the small cache (100K), so kernel evaluations are largely saved. Occasionally the shrinking is too aggressive so some variables are wrongly removed. Then recovering from mistakes causes longer iterations.

Note that our shrinking implementation is by removing bounded elements not in the set (25). Thus, the smaller the interval $[M(\alpha^k), m(\boldsymbol{\alpha}^k)]$ is, the more variables are shrunk. In Figure 10, we show the relationship between the maximal violation $m(\boldsymbol{\alpha}^k) - M(\boldsymbol{\alpha}^k)$ and iterations. Clearly using (11) reduces the maximal violation more quickly than using (33). A possible explanation is that (11) has less restriction than (33): In early iterations, if a set $B = \{i, j\}$ is associated with a large violation $-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j$, then $\mathbf{d}_B$ defined in (15) has large components. Hence though it minimizes the quadratic functions (9), $\boldsymbol{\alpha}^k_B + \mathbf{d}_B$ is easily infeasible. To solve (33), one thus changes $\boldsymbol{\alpha}^k_B + \mathbf{d}_B$ back to the feasible region as (35) does. As a reduced step is taken, the corresponding Sub(B) may not be smaller than those of using other sets. On the other hand, (11) does not require $\boldsymbol{\alpha}^k_B + \mathbf{d}_B$ to be feasible, so a large step is taken. The resulting Sub(B) thus may be small enough so that B is selected. Therefore, using (11) tend to select working sets with large violations and hence may more quickly reduce the maximal violation.

Discussion here shows that (11) is better than (33). They lead to similar numbers of iterations, but the cost per iteration is less by using (11). Moreover, (11) better incorporates the shrinking technique.

6.3 Sub-problems Using First Order Information

Under first order approximation, we can also modify the sub-problem (8) to the following form, which maintains the feasibility:

$$\begin{aligned} \text{Sub}(B) \equiv \min_{\mathbf{d}_B} \quad & \nabla f(\boldsymbol{\alpha}^k)_B^T \mathbf{d}_B \\ \text{subject to} \quad & \mathbf{y}_B^T \mathbf{d}_B = 0, \\ & 0 \le \alpha_i + d_i \le C, i \in B. \end{aligned} \tag{43}$$

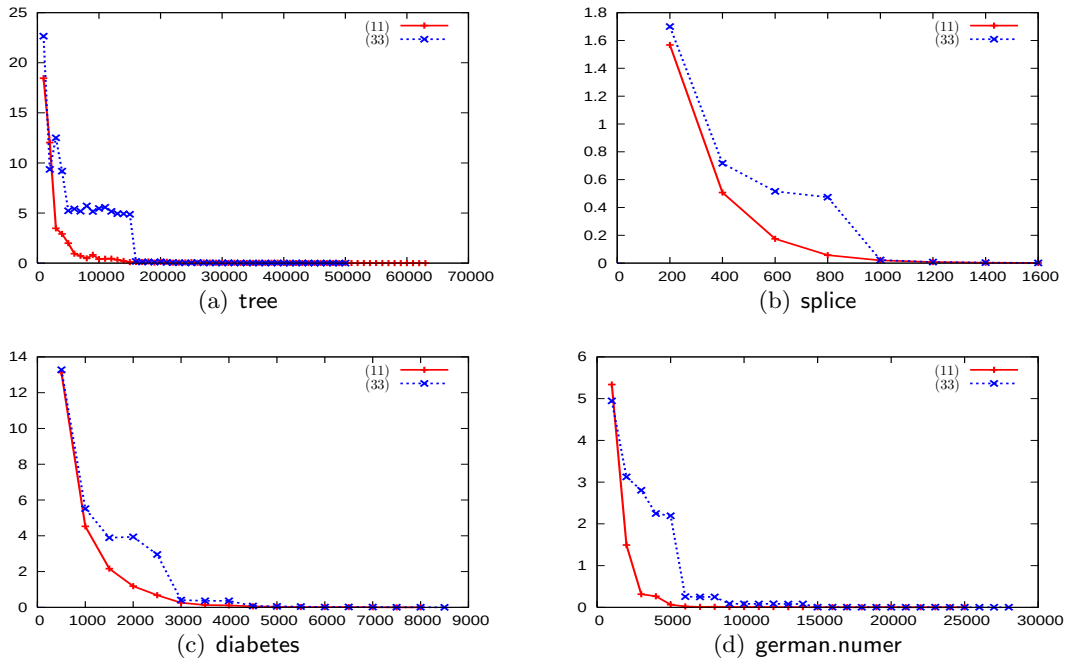


Figure 10: Iterations (x-axis) and maximal violations (y-axis) of using (11) and (33).

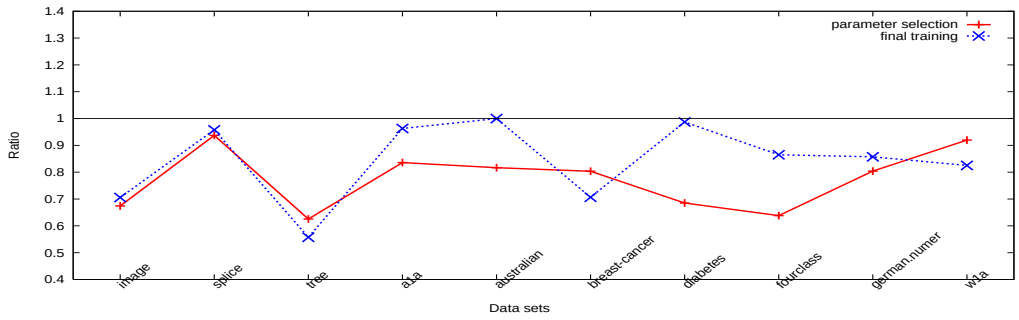
Section 2 discusses that a maximal violating pair is an optimal solution of $\min_{B:|B|=2} \text{Sub}(B)$, where $\text{Sub}(B)$ is (8). If (43) is used instead, Simon (2004) has shown an $O(l)$ procedure to obtain a solution. Thus the time complexity is the same as that of using (8).

Note that Theorem 7 does not hold for these two selection methods. In the proof, we use the small changes of α_i^k in final iterations to show that certain α_i^k never reaches bounds 0 and C. Then the sub-problem (2) to find $\boldsymbol{\alpha}^{k+1}$ is indeed the best sub-problem obtained in the procedure of working set selection. Now no matter (8) or (43) is used for selecting the working set, we still use (2) to find $\boldsymbol{\alpha}^{k+1}$. Therefore, we cannot link the small change between $\boldsymbol{\alpha}^k$ and $\boldsymbol{\alpha}^{k+1}$ to the optimal $\mathbf{d}_B$ in the procedure of working set selection. Without an interpretation like Theorem 7, the performance difference between using (8) and (43) remains unclear and is a future research issue.

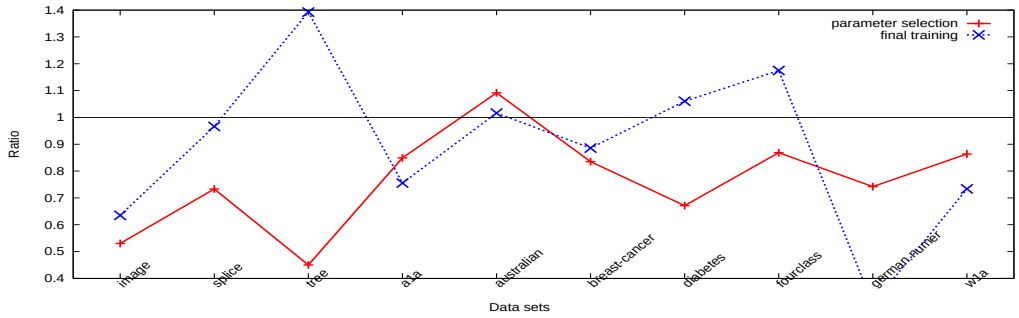
7. Discussion and Conclusions

In Section 2, the selection (10) of using second order information may involve checking $\binom{l}{2}$ pairs of indices. This is not practically viable, so in WSS 2 we heuristically fix $i \in \arg m(\boldsymbol{\alpha}^k)$ and examine $O(l)$ sets to find j. It is interesting to see how well this heuristic performs and whether we can make further improvements. By running the same small classification problems used in Section 6, Figure 11 presents the iteration ratio between using two selection methods:

$$\frac{\#\text{ iter. by Alg. 2 and checking } \binom{l}{2} \text{ pairs}}{\#\text{ iter. by Alg. 2 and WSS 2}}.$$



(a) RBF kernel



(b) Linear kernel

Figure 11: Iteration ratios between using two selection methods: checking all $\binom{l}{2}$ pairs and WSS 2. Note that the ratio (y-axis) starts from 0.4 but not 0.

We do not use shrinking and consider both RBF and linear kernels. Figure 11 clearly shows that a full check of all index pairs causes fewer iterations. However, as the average of ratios for various problems is between 0.7 and 0.8, this selection reduces iterations of using WSS 2 by only 20% to 30%. Therefore, WSS 2, an $O(l)$ procedure, successfully returns a working set nearly as good as that by an $O(l^2)$ procedure. In other words, the $O(l)$ sets heuristically considered in WSS 2 are among the best in all $\binom{l}{2}$ candidates.

Experiments in this paper fully demonstrate that using the proposed WSS 2 (and hence WSS 3) leads to faster convergence (i.e., fewer iterations) than using WSS 1. This result is reasonable as the selection based on second order information better approximates the objective function in each iteration. However, this argument explains only the behavior per iteration, but not the global performance of the decomposition method. A theoretical study showing that the proposed selection leads to better convergence rates is a difficult but interesting future issue.

In summary, we have proposed a new and effective working set selection WSS 3. The SMO-type decomposition method using it asymptotically converges and satisfies other useful theoretical properties. Experiments show that it is better than a commonly used selection WSS 1, in both the training time and iterations.

WSS 3 has replaced WSS 1 in the software LIBSVM (after version 2.8).

Acknowledgments

This work was supported in part by the National Science Council of Taiwan via the grant NSC 93-2213-E-002-030.

Appendix A. WSS 1 Solves Problem (7): the Proof

For any given $\{i, j\}$, we can substitute $\hat{d}_i \equiv y_i d_i$ and $\hat{d}_j \equiv y_j d_j$ to (8), so the objective function becomes

$$(-y_i \nabla f(\boldsymbol{\alpha}^k)_i + y_j \nabla f(\boldsymbol{\alpha}^k)_j)\hat{d}_j. \tag{44}$$

As $d_i = d_j = 0$ is feasible for (8), the minimum of (44) is zero or a negative number. If $-y_i \nabla f(\boldsymbol{\alpha}^k)_i > -y_j \nabla f(\boldsymbol{\alpha}^k)_j$, using the condition $\hat{d}_i + \hat{d}_j = 0$, the only possibility for (44) to be negative is $\hat{d}_j < 0$ and $\hat{d}_i > 0$. From (3), (8b), and (8c), this corresponds to $i \in I_{\text{up}}(\boldsymbol{\alpha}^k)$ and $j \in I_{\text{low}}(\boldsymbol{\alpha}^k)$. Moreover, the minimum occurs at $\hat{d}_j = -1$ and $\hat{d}_i = 1$. The situation of $-y_i \nabla f(\boldsymbol{\alpha}^k)_i < -y_j \nabla f(\boldsymbol{\alpha}^k)_j$ is similar.

Therefore, solving (7) is essentially the same as

$$\begin{aligned} & \min\Big\{\min\big(y_i \nabla f(\boldsymbol{\alpha}^k)_i - y_j \nabla f(\boldsymbol{\alpha}^k)_j, 0\big) \;\Big|\; i \in I_{\text{up}}(\boldsymbol{\alpha}^k), j \in I_{\text{low}}(\boldsymbol{\alpha}^k)\Big\} \\ = \;& \min\big(-m(\boldsymbol{\alpha}^k) + M(\boldsymbol{\alpha}^k), 0\big). \end{aligned}$$

Hence, if there are violating pairs, the maximal one solves (7).

Appendix B. Pseudo Code of Algorithm 2 and WSS 3

B.1 Main Program (Algorithm 2)

```
Inputs:
  y:   array of {+1, -1}: class of the i-th instance
  Q:   Q[i][j] = y[i]*y[j]*K[i][j]; K: kernel matrix
  len: number of instances

// parameters
eps = 1e-3  // stopping tolerance
tau = 1e-12

// main routine
initialize alpha array A to all zero
initialize gradient array G to all -1

while (1) {
  (i,j) = selectB()
  if (j == -1)
    break

  // working set is (i,j)
  a = Q[i][i]+Q[j][j]-2*y[i]*y[j]*Q[i][j]
  if (a <= 0)
    a = tau
```

```
  b = -y[i]*G[i]+y[j]*G[j]

  // update alpha
  oldAi = A[i], oldAj = A[j]
  A[i] += y[i]*b/a
  A[j] -= y[j]*b/a

  // project alpha back to the feasible region
  sum = y[i]*oldAi+y[j]*oldAj
  if A[i] > C
    A[i] = C
  if A[i] < 0
    A[i] = 0
  A[j] = y[j]*(sum-y[i]*A[i])
  if A[j] > C
    A[j] = C
  if A[j] < 0
    A[j] = 0
  A[i] = y[i]*(sum-y[j]*A[j])

  // update gradient
  deltaAi = A[i] - oldAi, deltaAj = A[j] - oldAj
  for t = 1 to len
    G[t] += Q[t][i]*deltaAi+Q[t][j]*deltaAj
}
```

B.2 Working Set Selection Subroutine (WSS 3)

```
// return (i,j)
procedure selectB
  // select i
  i = -1
  G_max = -infinity
  G_min = infinity
  for t = 1 to len {
    if (y[t] == +1 and A[t] < C) or
       (y[t] == -1 and A[t] > 0) {
      if (-y[t]*G[t] >= G_max) {
        i = t
        G_max = -y[t]*G[t]
      }
    }
  }

  // select j
  j = -1
  obj_min = infinity
  for t = 1 to len {
    if (y[t] == +1 and A[t] > 0) or
       (y[t] == -1 and A[t] < C) {
      b = G_max + y[t]*G[t]
```

```
      if (-y[t]*G[t] <= G_min)
        G_min = -y[t]*G[t]
      if (b > 0) {
        a = Q[i][i]+Q[t][t]-2*y[i]*y[t]*Q[i][t]
        if (a <= 0)
          a = tau
        if (-(b*b)/a <= obj_min) {
          j = t
          obj_min = -(b*b)/a
        }
      }
    }
  }

  if (G_max-G_min < eps)
    return (-1,-1)

  return (i,j)
end procedure
```

References

R. R. Bailey, E. J. Pettit, R. T. Borochoff, M. T. Manry, and X. Jiang. Automatic recognition of usgs land use/cover categories using statistical and neural networks classifiers. In *SPIE OE/Aerospace and Remote Sensing*, Bellingham, WA, 1993. SPIE.

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

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Pai-Hsuen Chen, Rong-En Fan, and Chih-Jen Lin. A study on SMO-type decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 17:893–908, July 2006. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/generalSMO.pdf`.

Corina Cortes and Vladimir Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

Tin Kam Ho and Eugene M. Kleinberg. Building projectable classifiers of arbitrary complexity. In *Proceedings of the 13th International Conference on Pattern Recognition*, pages 880–885, Vienna, Austria, August 1996.

Don Hush and Clint Scovel. Polynomial-time decomposition algorithms for support vector machines. *Machine Learning*, 51:51–71, 2003. URL `http://www.c3.lanl.gov/~dhush/machine_learning/svm_decomp.ps`.

Thorsten Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.

D. Lai, N. Mani, and M. Palaniswami. Increasing the step of the Newtonian decomposition method for support vector machines. Technical Report MECSE-29-2003, Dept. Electrical and Computer Systems Engineering Monash University, Australia, 2003a.

D. Lai, N. Mani, and M. Palaniswami. A new method to select working sets for faster training for support vector machines. Technical Report MESCE-30-2003, Dept. Electrical and Computer Systems Engineering Monash University, Australia, 2003b.

Chih-Jen Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12(6):1288–1298, 2001a. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/conv.ps.gz`.

Chih-Jen Lin. Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Transactions on Neural Networks*, 13(1):248–250, 2002. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/q2conv.pdf`.

Chih-Jen Lin. Linear convergence of a decomposition method for support vector machines. Technical report, Department of Computer Science, National Taiwan University, 2001b. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/linearconv.pdf`.

D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Prentice Hall, Englewood Cliffs, N.J., 1994. Data available at `http://www.ncc.up.pt/liacc/ML/statlog/datasets.html`.

D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases. Technical report, University of California, Irvine, Dept. of Information and Computer Sciences, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of CVPR'97*, pages 130–136, New York, NY, 1997. IEEE.

Laura Palagi and Marco Sciandrone. On the convergence of a modified version of SVM^{light} algorithm. *Optimization Methods and Software*, 20(2-3):315–332, 2005.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.

Danil Prokhorov. IJCNN 2001 neural network competition. Slide presentation in IJCNN'01, Ford Research Laboratory, 2001. `http://www.geocities.com/ijcnn/nnc_ijcnn01.pdf` .

B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.

Hans Ulrich Simon. On the complexity of working set selection. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT 2004)*, 2004.