Journal of Machine Learning Research 7 (2006) 733–769 Submitted 12/05; Revised 4/06; Published 5/06

QP Algorithms with Guaranteed Accuracy and Run Time for Support Vector Machines

Don Hush DHUSH@LANL.GOV
Patrick Kelly KELLY@LANL.GOV
Clint Scovel JCS@LANL.GOV
Ingo Steinwart INGO@LANL.GOV
Modeling, Algorithms and Informatics Group, CCS-3, MS B265
Los Alamos National Laboratory
Los Alamos, NM 87545 USA

Editor: Bernhard Schölkopf

Abstract

We describe polynomial–time algorithms that produce approximate solutions with guaranteed accuracy for a class of QP problems that are used in the design of support vector machine classifiers. These algorithms employ a two–stage process where the first stage produces an approximate solution to a dual QP problem and the second stage maps this approximate dual solution to an approximate primal solution. For the second stage we describe an $O(n\log n)$ algorithm that maps an approximate dual solution with accuracy $(2\sqrt{2K_n}+8\sqrt{\lambda})^{-2}\lambda\varepsilon_p^2$ to an approximate primal solution with accuracy ε_p where n is the number of data samples, K_n is the maximum kernel value over the data and $\lambda > 0$ is the SVM regularization parameter. For the first stage we present new results for *decomposition* algorithms and describe new decomposition algorithms with guaranteed accuracy and run time. In particular, for τ*–rate certifying* decomposition algorithms we establish the optimality of $\tau = 1/(n-1)$. In addition we extend the recent $\tau = 1/(n-1)$ algorithm of Simon (2004) to form two new *composite* algorithms that also achieve the $\tau = 1/(n-1)$ iteration bound of List and Simon (2005), but yield faster run times in practice. We also exploit the τ–rate certifying property of these algorithms to produce new stopping rules that are computationally efficient and that guarantee a specified accuracy for the approximate dual solution. Furthermore, for the dual QP problem corresponding to the standard classification problem we describe operational conditions for which the Simon and composite algorithms possess an upper bound of $O(n)$ on the number of iterations. For this same problem we also describe general conditions for which a matching lower bound exists for *any* decomposition algorithm that uses working sets of size 2. For the Simon and composite algorithms we also establish an $O(n^2)$ bound on the overall run time for the first stage. Combining the first and second stages gives an overall run time of $O(n^2(c_k+1))$ where c_k is an upper bound on the computation to perform a kernel evaluation. Pseudocode is presented for a complete algorithm that inputs an accuracy ε_p and produces an approximate solution that satisfies this accuracy in low order polynomial time. Experiments are included to illustrate the new stopping rules and to compare the Simon and composite decomposition algorithms.

Keywords: quadratic programming, decomposition algorithms, approximation algorithms, support vector machines

1. Introduction

Solving a quadratic programming (QP) problem is a major component of the support vector machine (SVM) training process. In practice it is common to employ algorithms that produce *approximate* solutions. This introduces a trade-off between computation and accuracy that has not been thoroughly explored. The accuracy, as measured by the difference between the criterion value of the approximate solution and the optimal criterion value, is important for learning because it has a direct influence on the generalization error. For example, since the optimal criterion value plays a key role in establishing the SVM performance bounds in (Steinwart and Scovel, 2004, 2005; Scovel et al., 2005b) the influence of the accuracy can be seen directly through the proofs of these bounds. Since the primal QP problem can be prohibitively large and its Wolfe dual QP problem is considerably smaller it is common to employ a two–stage training process where the first stage produces an approximate solution to the dual QP problem and the second stage maps this approximate dual solution to an approximate primal solution. Existing algorithms for the first stage often allow the user to trade accuracy and computation for the dual QP problem through the choice of a tolerance value that determines when to stop the algorithm, but it is not known how to choose this value to achieve a desired accuracy or run time. Furthermore existing algorithms for the second stage have been developed largely without concern for accuracy and therefore little is known about the accuracy of the approximate primal solutions they produce. In this paper we describe algorithms that accept the accuracy ϵ_p of the primal QP problem as an input and are guaranteed to produce an approximate solution that satisfies this accuracy in low order polynomial time. To our knowledge these are the first algorithms of this type for SVMs. In addition our run time analysis reveals the effect of the accuracy on the run time, thereby allowing the user to make an informed decision regarding the trade–off between computation and accuracy.

Algorithmic strategies for the dual QP problem must address the fact that when the number of data samples n is large the storage requirements for the kernel matrix can be excessive. This barrier can be overcome by invoking algorithmic strategies that solve a large QP problem by solving a sequence of smaller QP problems where each of the smaller QP problems is obtained by fixing a subset of the variables and optimizing with respect to the remaining variables. Algorithmic strategies that solve a QP problem in this way are called *decomposition* algorithms and a number have been developed for dual QP problems: (Balcazar et al., 2001; Chen et al., 2005, 2006; Cristianini and Shawe-Taylor, 2000; Hsu and Lin, 2002; Hush and Scovel, 2003; Joachims, 1998; Keerthi et al., 2000, 2001; Laskov, 2002; Liao et al., 2002; List and Simon, 2004, 2005; Mangasarian and Musicant, 1999, 2001; Osuna et al., 1997; Platt, 1998; Simon, 2004; Vapnik, 1998).

The key to developing a successful decomposition algorithm is in the method used to determine the *working sets*, which are the subsets of variables to be optimized at each iteration. To guarantee stepwise improvement each working set must contain a *certifying pair* (Definition 3 below). Stronger conditions are required to guarantee convergence: (Chang et al., 2000; Chen et al., 2006; Hush and Scovel, 2003; Lin, 2001a,b; List and Simon, 2004) and even stronger conditions appear necessary to guarantee rates of convergence: (Balcazar et al., 2001; Hush and Scovel, 2003; Lin, 2001a). Indeed, although numerous decomposition algorithms have been proposed few are known to possess polynomial run time bounds. Empirical studies have estimated the run time of some common decomposition algorithms to be proportional to n^p where p varies from approximately 1.7 to approximately 3.0 depending on the problem instance: (Joachims, 1998; Laskov, 2002; Platt, 1998). Although these types of studies can provide useful insights they have limited utility in predicting the run time for future problem instances. In addition these particular studies do not appear to be calibrated with respect to the accuracy of the final criterion value and so their relevance to the framework considered here is not clear. Lin (2001a) performs a convergence rate analysis that may eventually be used to establish run time bounds for a popular decomposition algorithm, but these results hold under rather restrictive assumptions and more work is needed before the tightness and utility of these bounds is known (a more recent version of this analysis can be found in (Chen et al., 2006)). Balcazar et al. (2001) present a randomized decomposition algorithm whose expected run time is $O\big((n+r(\mathtt{k}^2d^2))\ \mathtt{k}d\log n\big)$ where n is the number of samples, d is the dimension of the input space, $1 \le \mathtt{k} \le n$ is a data dependent parameter and $r(\mathtt{k}^2d^2)$ is the run time required to solve the dual QP problem over $\mathtt{k}^2d^2$ samples. This algorithm is very attractive when $\mathtt{k}^2d^2 \ll n$, but in practice the value of $\mathtt{k}$ is unknown and it may be large when the Bayes error is not close to zero. Hush and Scovel (2003) define a class of *rate certifying algorithms* and describe an example algorithm that uses $O\left(\frac{K_n n^5 \log n}{\varepsilon}\right)$ computation to reach an approximate dual solution with accuracy ε, where K_n is the maximum value of the kernel matrix. Recently Simon (2004) introduced a new rate certifying algorithm which can be shown, using the results in (List and Simon, 2005), to use $O\left(\frac{nK_n}{\lambda\varepsilon} + n^2\log\left(\frac{\lambda n}{K_n}\right)\right)$ computation to reach an approximate dual solution with accuracy ε, where $\lambda > 0$ is the SVM regularization parameter. In this paper we combine Simon's algorithm with the popular *Generalized SMO* algorithm of Keerthi et al. (2001) to obtain a *composite* algorithm that possesses the same computation bound as Simon's algorithm, but appears to use far less computation in practice (as illustrated in our experiments). We also extend this approach to form a second *composite* algorithm with similar properties. In addition we introduce operational assumptions on K_n and the choice of λ and ε that yield a simpler computation bound of $O(n^2)$ for these algorithms. Finally to guarantee that actual implementations of these algorithms produce approximate solutions with accuracy ε we introduce two new stopping rules that terminate the algorithms when an adaptively computed upper bound on the accuracy falls below ε.

The second stage of the design process maps an approximate dual solution to an approximate primal solution. In particular this stage determines how the approximate dual solution is used to form the normal vector and offset parameter for the SVM classifier. It is common practice to use the approximate dual solution as coefficients in the linear expansion of the data that forms the normal vector, and then use a heuristic based on approximate satisfaction of the Karush-Kuhn-Tucker (KKT) optimality conditions to choose the offset parameter. This approach is simple and computationally efficient, but it produces an approximate primal solution whose accuracy is unknown. In this paper we take a different approach based on the work of Hush et al. (2005). This work studies the accuracy of the approximate primal solution as a function of the accuracy of the approximate dual solution and the map from approximate dual to approximate primal. In particular for the SVM problem it appears that choosing this map involves a trade–off between computation and accuracy. Here we employ a map described and analyzed in (Hush et al., 2005) that guarantees an accuracy of ε_p for the primal QP problem when the dual QP problem is solved with accuracy $(2\sqrt{2K_n}+8\sqrt{\lambda})^{-2}\lambda\varepsilon_p^2$. This map resembles current practice in that it performs a direct substitution of the approximate dual solution into a linear expansion for the normal vector, but differs in the way that it determines the offset parameter. We develop an $O(n\log n)$ algorithm that computes the offset parameter according to this map.

The main results of this paper are presented in Sections 2 and 3. Proofs for all the theorems, lemmas, and corollaries in these sections can be found in Section 6, except for Theorem 2 which is established in (Hush et al., 2005). Section 2 describes the SVM formulation, presents algorithms for the first and second stages, and provides theorems that characterize the accuracy and run time for these algorithms. Section 3 then determines specific run time bounds for decomposition algorithms applied to the standard classification problem and the density level detection problem. Section 4 describes experiments that illustrate the new stopping rules and compare the run time of different decomposition algorithms. Section 5 provides a summary of results and establishes an overall run time bound. A complete algorithm that computes an ϵ_p–optimal solution to the primal QP problem is provided by (Procedure 1, Section 2) and Procedures 3–8 in the appendix.

2. Definitions, Algorithms, and Main Theorems

Let X be a pattern space and $k : X \times X \to \mathbb{R}$ be a kernel function with Hilbert space H and feature map $\phi : X \to H$ so that $k(x_1,x_2) = \phi(x_1)\cdot\phi(x_2), \forall x_1,x_2 \in X$. Define $Y := \{-1,1\}$. Given a data set $((x_1,y_1),...,(x_n,y_n)) \in (X\times Y)^n$ the *primal* QP problem that we consider takes the form

$$\begin{array}{ll} \min_{\psi,b,\xi} & \lambda\|\psi\|^2 + \sum_{i=1}^n u_i\xi_i \\ \text{s.t.} & y_i(\phi(x_i)\cdot\psi + b) \geq 1 - \xi_i \\ & \xi_i \geq 0,\ \ i = 1,2,...,n \end{array} \tag{1}$$

where $\lambda > 0$, $u_i > 0$ and $\sum_i u_i = 1$. This form allows a different weight u_i for each data sample. Specific cases of interest include:

1. the *L1–SVM* for the standard supervised classification problem which sets $u_i = 1/n$, $i = 1,...,n$,

2. the *DLD–SVM* for the density level detection problem described in (Steinwart et al., 2005) which sets

   $$u_i = \begin{cases} \frac{1}{(1+\rho)n_1}, & y_i = 1 \\ \frac{\rho}{(1+\rho)n_{-1}}, & y_i = -1 \end{cases}$$

   where n_1 is the number of samples distributed according to P_1 and assigned label $y = 1$, n_{-1} is the number of samples distributed according to P_{-1} and assigned label $y = -1$, $h = dP_1/dP_{-1}$ is the density function, and $\rho > 0$ defines the ρ–level set $\{h > \rho\}$ that we want to detect.

The *dual* QP problem is

$$\begin{array}{ll} \max_a & -\frac{1}{2}a\cdot \mathsf{Q}a + a\cdot 1 \\ \text{s.t.} & y\cdot a = 0 \\ & 0 \leq a_i \leq u_i\ \ i = 1,2,...,n. \end{array} \tag{2}$$

where

$$\mathsf{Q}_{ij} = y_i y_j k(x_i,x_j)/2\lambda.$$

The change of variables defined by

$$\alpha_i := y_i a_i + l_i, \quad l_i = \begin{cases} 0 & y_i = 1 \\ u_i & y_i = -1 \end{cases} \tag{3}$$

gives the *canonical dual* QP problem

$$\begin{array}{ll} \max_\alpha & -\frac{1}{2}\alpha\cdot Q\alpha + \alpha\cdot w + w_0 \\ \text{s.t.} & 1\cdot\alpha = c \\ & 0 \leq \alpha_i \leq u_i\ \ i = 1,2,...,n \end{array} \tag{4}$$

where

$$Q_{ij} = k(x_i, x_j)/2\lambda, \;\; c = l \cdot 1, \;\; w = Ql + y, \;\; w_0 = -l \cdot y - \frac{1}{2} l \cdot Ql. \tag{5}$$

We denote the canonical dual criterion by

$$R(\alpha) := -\frac{1}{2}\alpha \cdot Q\alpha + \alpha \cdot w + w_0.$$

Note that this change of variables preserves the criterion value. Also note that the relation between a and α is one–to–one. Most of our work is with the canonical dual because it simplifies the algorithms and their analysis.

We define the set of ε–optimal solutions of a constrained optimization problem as follows.

Definition 1 *Let P be a constrained optimization problem with parameter space Θ, criterion function $G : \Theta \to \mathbb{R}$, feasible set $\tilde{\Theta} \subseteq \Theta$ of parameter values that satisfy the constraints, and optimal criterion value G^* (i.e. $G^* = \sup_{\theta \in \tilde{\Theta}} G(\theta)$ for a maximization problem and $G^* = \inf_{\theta \in \tilde{\Theta}} G(\theta)$ for a minimization problem). Then for any $\varepsilon \geq 0$ we define*

$$\mathcal{O}_\varepsilon(P) := \{\theta \in \tilde{\Theta} : |G(\theta) - G^*| \leq \varepsilon\}$$

to be the set of ε–optimal solutions for P.

We express upper and lower computation bounds using $O(\cdot)$ and $\Omega(\cdot)$ notations defined by

$$O(g(n)) = \{f(n) : \exists \text{ positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\},$$
$$\Omega(g(n)) = \{f(n) : \exists \text{ positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\}.$$

We now describe our algorithm for the primal QP problem. It computes an approximate canonical dual solution $\hat{\alpha}$ and then maps to an approximate primal solution $(\hat{\psi}, \hat{b}, \hat{\xi})$ using the map described in the following theorem. This theorem is derived from (Hush et al., 2005, Theorem 2 and Corollary 1) which is proved using the result in (Scovel et al., 2005a).

Theorem 2 *Consider the primal QP problem P_{SVM} in (1) with $\lambda > 0$ and $|\phi(x_i)|^2 \leq K, i = 1,..,n$, and its corresponding canonical dual QP problem D_{SVM} in (4) with criterion R. Let $\varepsilon_p > 0$, $\varepsilon = (2\sqrt{2K} + 8\sqrt{\lambda})^{-2}\lambda\varepsilon_p^2$ and suppose that $\hat{\alpha} \in \mathcal{O}_\varepsilon(D_{SVM})$ and $R(\hat{\alpha}) \geq 0$. If*

$$\hat{\psi} = \frac{1}{2\lambda}\sum_{i=1}^{n}(\hat{\alpha}_i - l_i)\phi(x_i)$$

$$\hat{\xi}_i(b) = \max\left(0, 1 - y_i(\hat{\psi} \cdot \phi(x_i) + b)\right), \; i = 1,..,n$$

and

$$\hat{b} \in \arg\min \sum_{i=1}^{n} u_i \hat{\xi}_i(b)$$

then $(\hat{\psi}, \hat{b}, \hat{\xi}(\hat{b})) \in \mathcal{O}_{\varepsilon_p}(P_{SVM})$.

This theorem gives an expression for $\hat{\psi}$ that coincides with the standard practice of replacing an optimal dual solution α^* by an approximate dual solution $\hat{\alpha}$ in the expansion for the optimal normal vector determined by the KKT conditions. The remaining variables $\hat{\xi}$ and $\hat{b}$ are obtained by substituting $\hat{\psi}$ into the primal optimization problem, optimizing with respect to the slack variable ξ, and then minimizing with respect to b [1]. To guarantee an accuracy ϵ_p for the primal problem this theorem stipulates that the value of the dual criterion at the approximate solution be non–negative and that the accuracy for the dual solution satisfy $\epsilon = (2\sqrt{2K} + 8\sqrt{\lambda})^{-2}\lambda\epsilon_p^2$. The first condition is easily achieved by algorithms that start with $\alpha = l$ (so that the initial criterion value is 0) and continually improve the criterion value at each iteration. We will guarantee the second condition by employing an appropriate stopping rule for the decomposition algorithm.

Procedure 1 shows the primal QP algorithm that produces an ϵ_p–optimal solution $(\hat{\alpha}, \hat{b})$ that defines the SVM classifier

$$\text{sign}\left(\sum_{i=1}^{n}\left(\frac{\hat{\alpha}_i - l}{2\lambda}\right)k(x_i,x) + \hat{b}\right).$$

This algorithm inputs a data set $T_n = ((x_1,y_1),...,(x_n,y_n))$, a kernel function k, and parameter values λ, u and ϵ_p. Lines 3–6 produce an exact solution for the degenerate case where all the data samples have the same label. The rest of the routine forms an instance of the canonical dual QP according to (5), sets ϵ according to Theorem 2, sets $\alpha^0 = l$ so that $R(\alpha^0) = 0$, uses the routine `Decomposition` to compute an ϵ–approximate canonical dual solution $\hat{\alpha}$, and uses the routine `Offset` to compute the offset parameter $\hat{b}$ according to Theorem 2. The parameter g, which is defined in the next section, is a temporary value computed by `Decomposition` that allows a more efficient computation of $\hat{b}$ by `Offset`. The next three sections provide algorithms and computational bounds for the routines `Decomposition` and `Offset`.

Procedure 1 The algorithm for the primal QP problem.

```
1:  PrimalQP $(T_n, k, \lambda, u, \epsilon_p)$
2:
3:  if $(y_i = y_1, \forall i)$ then
4:      $\hat{\alpha} \leftarrow l$, $\hat{b} \leftarrow y_1$
5:      Return$(\hat{\alpha}, \hat{b})$
6:  end if
7:  Form canonical dual: $Q_{ij} \leftarrow \frac{k(x_i,x_j)}{2\lambda}$,  $l_i \leftarrow \frac{(1-y_i)u_i}{2}$,  $w \leftarrow Ql + y$,  $c \leftarrow l \cdot 1$
8:  Compute Desired Accuracy of Dual: $\epsilon \leftarrow \frac{\lambda\epsilon_p^2}{(2\sqrt{2K}+8\sqrt{\lambda})^2}$
9:  Initialize canonical dual variable: $\alpha^0 \leftarrow l$
10: $(\hat{\alpha}, g) \leftarrow$ Decomposition$(Q, w, c, u, \epsilon, \alpha^0)$
11: $\hat{b} \leftarrow$ Offset$(g, y, u)$
12: Return$(\hat{\alpha}, \hat{b})$
```

2.1 Decomposition Algorithms

We begin with some background material that describes: optimality conditions for the canonical dual, a model decomposition algorithm, necessary and sufficient conditions for convergence to a

1. This method for choosing the offset was investigated briefly in (Keerthi et al., 2001, Section 4).

solution, and sufficient conditions for rates of convergence. In many cases this background material extends a well known result to the slightly more general case considered here where each component of u may have a different value.

Consider an instance of the canonical dual QP problem given by (Q, w, w_0, c, u). Define the set of feasible values

$$\mathcal{A} := \{\alpha : (0 \le \alpha_i \le u_i) \text{ and } (\alpha \cdot 1 = c)\},$$

and the set of optimal solutions

$$\mathcal{A}^* := \arg\max_{\alpha \in \mathcal{A}} R(\alpha).$$

Also define the optimal criterion value $R^* := \sup_{\alpha \in \mathcal{A}} R(\alpha)$ and the gradient at α

$$g(\alpha) := \nabla R(\alpha) = -Q\alpha + w. \tag{6}$$

The optimality conditions established by Keerthi et al. (2001) take the form,

$$\alpha \in \mathcal{A}^* \quad \Leftrightarrow \quad g_j(\alpha) \le g_k(\alpha) \text{ for all } j : \alpha_j < u_j,\ k : \alpha_k > 0. \tag{7}$$

These conditions motivate the following definition from (Keerthi et al., 2001; Hush and Scovel, 2003).

Definition 3 *A* certifying pair *(also called a* violating pair*) for $\alpha \in \mathcal{A}$ is a pair of indices that witness the non–optimality of α, i.e. it is a pair of indices $j : \alpha_j < u_j$ and $k : \alpha_k > 0$ such that $g_j(\alpha) > g_k(\alpha)$.*

Using the approach in (Hush and Scovel, 2003, Section 3) it can be shown that the requirement that working sets contain a certifying pair is both necessary and sufficient to obtain a stepwise improvement in the criterion value. Thus, since certifying pairs are defined in terms of the gradient component values it appears that the gradient plays an essential role in determining members of the working sets. To compute the gradient at each iteration using (6) requires $O(n^2)$ operations. However since decomposition algorithms compute a sequence of feasible points $(\alpha^m)_{m \ge 0}$ using working sets of size p, the sparsity of $(\alpha^{m+1} - \alpha^m)$ means that the update

$$g(\alpha^{m+1}) = g(\alpha^m) - Q(\alpha^{m+1} - \alpha^m) \tag{8}$$

requires only $O(pn)$ operations. A model decomposition algorithm that uses this update is shown in Procedure 2. After computing an initial gradient vector this algorithm iterates the process of determining a working set, solving a QP problem restricted to this working set, updating the gradient vector, and testing a stopping condition.

The requirement that working sets contain a certifying pair is necessary but not sufficient to guarantee convergence to a solution (e.g. see the examples in Chen et al., 2006; Keerthi and Ong, 2000). However Lin (2002b) has shown that including a *max–violating pair* defined by

$$(j^*, k^*) \;:\; j^* \in \arg\max_{i:\alpha_i < u_i} g_i(\alpha), \quad k^* \in \arg\min_{i:\alpha_i > 0} g_i(\alpha) \tag{9}$$

in each working set does guarantee convergence to a solution. Once the gradient has been computed a max–violating pair can be determined in one pass through the gradient components and therefore requires $O(n)$ computation. The class of *max–violating pair algorithms* that include a max–violating

Procedure 2 A model decomposition algorithm for the canonical dual QP problem.

```
1:  ModelDecomposition($Q, w, c, u, \varepsilon, \alpha^0$)
2:
3:  Compute initial gradient $g^0 \leftarrow -Q\alpha^0 + w$
4:  $m \leftarrow 0$
5:  repeat
6:     Compute a working set $W^m$
7:     Compute $\alpha^{m+1}$ by solving the restricted QP determined by $\alpha^m$ and $W^m$
8:     Update the gradient: $g^{m+1} \leftarrow g^m - Q(\alpha^{m+1} - \alpha^m)$
9:     $m \leftarrow m + 1$
10: until (stopping condition is satisfied)
11: Return($\alpha^m, g^m$)
```

pair in each working set includes many popular algorithms (e.g. Chang and Lin, 2001; Joachims, 1998; Keerthi et al., 2001). Although asymptotic convergence to a solution is guaranteed for these algorithms, their convergence rate is unknown. In contrast we now describe algorithms based on alternative pair selection strategies that have the same $O(n)$ computational requirements (once the gradient has been computed) but possess known rates of convergence to a solution.

Consider the model decomposition algorithm in Procedure 2. The run time of the main loop is the product of the number of iterations and the computation per iteration, and both of these depend heavily on the size of the working sets and how they are chosen. The smallest size that admits a convergent algorithm is 2 and many popular algorithms adopt this size. We refer to these as *W2* decomposition algorithms. A potential disadvantage of this approach is that the number of iterations may be larger than it would be otherwise. On the other hand adopting working sets of size 2 allows us to solve each 2–variable QP problem in constant time (e.g. see Platt, 1998). In addition *W2* decomposition algorithms require only $O(n)$ computation to update the gradient and have the advantage that the overall algorithm can be quite simple (as demonstrated by the *W2* max–violating pair algorithm). Furthermore adopting size 2 working sets will allow us to implement our new stopping rules in constant time. Thus, while most of the algorithms we describe below allow the working sets to be larger than 2, our experiments will be performed with their *W2* variants.

In addition to their size, the content of the working sets has a significant impact on the run time through its influence on the convergence rate of the algorithm. Hush and Scovel (2003) prove that convergence rates can be guaranteed simply by including a *rate certifying pair* in each working set. Roughly speaking a *rate certifying pair* is a certifying pair that, when used as the working set, provides a sufficient stepwise improvement. To be more precise we start with the following definitions. Define a working set to be a subset of the index set of the components of α, and let W denote a working set of unspecified size and W_p denote a working set of size p. In particular $W_n = \{1, 2, ..., n\}$ denotes the entire index set. The set of feasible solutions for the canonical dual QP sub–problem defined by a feasible value α and a working set W is defined

$$\mathcal{A}(\alpha, W) := \{\acute{\alpha} \in \mathcal{A} : \acute{\alpha}_i = \alpha_i \; \forall i \notin W\}.$$

Define

$$\sigma(\alpha|W) := \sup_{\acute{\alpha} \in \mathcal{A}(\alpha, W)} g(\alpha) \cdot (\acute{\alpha} - \alpha)$$

to be the optimal value of the linear programming (LP) problem at α. The following definition is adapted from (Hush and Scovel, 2003).

Definition 4 *For $\tau > 0$ an index pair W_2 is called a τ–*rate certifying pair *for α if $\sigma(\alpha|W_2) \geq \tau\sigma(\alpha|W_n)$. A decomposition algorithm that includes a τ–rate certifying pair in the working set at every iteration is called a τ–*rate certifying algorithm.

For a τ–rate certifying algorithm Hush and Scovel (2003) provide an upper bound on the number of iterations as a function of τ. An improved bound can be obtained as a special case of (List and Simon, 2005, Theorem 1). The next theorem provides a slightly different bound that does not depend on the size of the working sets and therefore slightly improves the bound obtained from (List and Simon, 2005, Theorem 1) when the size of the working sets is larger than 2.

Theorem 5 *Consider the canonical dual QP problem in (4) with criterion function R and Gram matrix Q. Let $L \geq \max_i Q_{ii}$ and $S \geq \max_i u_i$. A τ–rate certifying algorithm that starts with α^0 achieves $R^* - R(\alpha^m) \leq \varepsilon$ after $\lceil \acute{m} \rceil$ iterations of the main loop where*

$$\acute{m} = \begin{cases} \left[\frac{2}{\tau}\ln\left(\frac{R^* - R(\alpha^0)}{\varepsilon}\right)\right]_+, & \varepsilon \geq \frac{4LS^2}{\tau} \\ \frac{2}{\tau}\left(\frac{4LS^2}{\tau\varepsilon} - 1 + \left[\ln\left(\frac{\tau(R^* - R(\alpha^0))}{4LS^2}\right)\right]_+\right), & \varepsilon < \frac{4LS^2}{\tau} \end{cases},$$

$\lceil\theta\rceil$ denotes the smallest integer greater than or equal to θ, and $[\theta]_+ = \max(0,\theta)$.

Chang et al. (2000) have shown that for every $\alpha \in \mathcal{A}$ there exists a τ–rate certifying pair with $\tau \geq 1/n^2$. This result can be used to establish the existence of decomposition algorithms with polynomial run times. The first such algorithm was provided by Hush and Scovel (2003) where the rate certifying pairs satisfied $\tau \geq 1/n^2$. However the value τ can be improved and the bound on the number of iterations reduced if the rate certifying pairs are determined differently. Indeed List and Simon (2005) prove that $\tau \geq 1/n$ for a *max–lp2* pair

$$W_2^* \in \arg\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2)$$

which is a pair with the maximum linear program value. The next theorem provides a slightly better result of $\tau \geq 1/(n-1)$ for this pair and establishes the optimality of this bound [2].

Theorem 6 *For $\alpha \in \mathcal{A}$*

$$\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2) \geq \frac{\sigma(\alpha|W_n)}{n-1}.$$

Furthermore, there exist problem instances for which there exist $\alpha \in \mathcal{A}$ such that

$$\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2) = \frac{\sigma(\alpha|W_n)}{n-1}.$$

2. This result provides a negligible improvement over the $\tau \geq 1/n$ result of List and Simon but is included here because it establishes optimality and because its proof, which is quite different from that of List and Simon, provides additional insight into the construction of certifying pairs that achieve $\tau \geq 1/(n-1)$.

Since a max–lp2 pair gives the largest value of $\sigma(\alpha|W_2)$ it follows from Definition 4 and Theorem 6 that the largest single value of τ that can be valid for all iterations of all problem instances is $1/(n-1)$. Thus a max–lp2 pair is optimal in that it achieves the minimum iteration bound in Theorem 5 with respect to τ. Furthermore Simon (2004) has introduced an algorithm for computing a max–lp2 pair that requires only $O(n)$ computation and therefore coincides with the $O(n)$ computation required to perform the other steps in the main loop. However, in spite of the promise suggested by this analysis experimental results suggest that there is much room to improve the convergence rates achieved with max–lp2 pairs (e.g. see Section 4). The result below provides a simple way to determine pair selection methods whose convergence rates are at least as good as those guaranteed by the max–lp2 pair method and possibly much better. This result is stated as a corollary since it follows trivially from the proof of Theorem 5.

Corollary 7 *Let* `DECOMP` *be a realization of the model decomposition algorithm for the canonical dual QP in Procedure 2 and let (α^m) represent a sequence of feasible points produced by this algorithm. At each iteration m let $\acute{W}_2^m$ be a τ–rate certifying pair and let $\acute{\alpha}^{m+1}$ be the feasible point determined by solving the restricted QP determined by α^m and $\acute{W}_2^m$. If for every $m \geq 0$ the stepwise improvement satisfies $R(\alpha^{m+1}) - R(\alpha^m) \geq R(\acute{\alpha}^{m+1}) - R(\alpha^m)$ then* `DECOMP` *will achieve $R^* - R(\alpha^m) \leq \varepsilon$ after $\lceil \acute{m} \rceil$ iterations of the main loop where $\acute{m}$ is given by Theorem 5.*

This theorem implies that any pair whose stepwise improvement is at least as good as that produced by a max–lp2 pair yields a decomposition algorithm that inherits the iteration bound in Theorem 5 with $\tau = 1/(n-1)$. An obvious example is a *max–qp2* pair, which is a pair with the *largest* stepwise improvement. However since determining such a pair may require substantial computation we seek alternatives. In particular Simon's algorithm visits several good candidate pairs in its search for a max–lp2 pair and can therefore be easily extended to form an alternative pair selection algorithm that is computationally efficient and satisfies this stepwise improvement property. To see this we start with a description of Simon's algorithm.

First note that when searching for a max–lp2 pair it is sufficient to consider only pairs (j,k) where $g_j(\alpha) > g_k(\alpha)$. For such a pair it is easy to show that (e.g. see the proof of Theorem 6)

$$\sigma(\alpha|\{j,k\}) = \min(u_j - \alpha_j, \alpha_k)(g_j(\alpha) - g_k(\alpha)) = \Delta_{jk}\ (g_j(\alpha) - g_k(\alpha)) \tag{10}$$

where u_j is the upper bound on α_j specified in (4) and $\Delta_{jk} := \min(u_j - \alpha_j, \alpha_k)$. The key to Simon's algorithm is the recognition that among the $O(n^2)$ index pairs there are at most $2n$ distinct values for Δ:

$$u_1 - \alpha_1,\ \alpha_1,\ u_1 - \alpha_2,\ \alpha_2,\ \ldots,\ u_n - \alpha_n,\ \alpha_n. \tag{11}$$

Consider searching this list of values for one that corresponds to a maximum value of σ. For an entry of the form $u_j - \alpha_j$ for some j, an index k that maximizes $\sigma(\alpha|\{j,k\})$ satisfies

$$k \ \in\ \arg\max_{l:\alpha_l \geq u_j - \alpha_j} (g_j(\alpha) - g_l(\alpha)) \ = \ \arg\min_{l:\alpha_l \geq u_j - \alpha_j} g_l(\alpha)\ .$$

Similarly for an entry of the form α_k for some k, an index j that maximizes $\sigma(\alpha|\{j,k\})$ satisfies

$$j \ \in\ \arg\max_{l:u_l - \alpha_l \geq \alpha_k} (g_l(\alpha) - g_k(\alpha)) \ = \ \arg\max_{l:u_l - \alpha_l \geq \alpha_k} g_l(\alpha)\ .$$

Now suppose we search the list of values from largest to smallest and keep track of the maximum gradient component value for entries of the form $u_j - \alpha_j$ and the minimum gradient component value for entries of the form α_k as we go. Then as we visit each entry in the list the index pair that maximizes σ can be computed in constant time. Thus a max–lp2 pair can be determined in one pass through the list. A closer examination reveals that only the nonzero values at the front of the list need to be scanned, since entries with zero values cannot form a certifying pair (i.e. they correspond to pairs for which there is no feasible direction for improvement). In addition, since nonzero entries of the form $u_j - \alpha_j$ correspond to components j where $\alpha_j < u_j$, and nonzero entries of the form α_k correspond to components k where $\alpha_k > 0$, once the scan reaches the last nonzero entry in the list the indices of the maximum and minimum gradient component values correspond to a max–violating pair. Pseudocode for this algorithm is shown in Procedure 4 in Appendix 6. This algorithm requires that the ordered list of values be updated at each iteration. If the entries are stored in a linear array this can be accomplished in $O(pn)$ time by a simple *search and insert* algorithm, where p is the size of the working set. However, with the appropriate data structure (e.g. a red–black tree) this list can be updated in $O(p\log n)$ time. In this case the size of the working sets must satisfy $p = O(n/\log n)$ to guarantee an $O(n)$ run time for the main loop.

Simon's algorithm computes both a max–lp2 pair and a max–violating pair at essentially the same cost. In addition the stepwise improvement for an individual pair can be computed in constant time. Indeed with $W_2^m = \{j,k\}$ and $g(\alpha_j^m) \geq g(\alpha_k^m)$ the stepwise improvement δ_R^m takes the form

$$\delta_R^m = \begin{cases} \Delta\delta_g - \Delta^2 q/2, & \delta_g > q\Delta \\ \frac{\delta_g^2}{2q}, & \text{otherwise} \end{cases} \tag{12}$$

where $\delta_g = g(\alpha_j^m) - g(\alpha_k^m)$, $q = Q_{jj} + Q_{kk} - 2Q_{jk}$ and $\Delta = \min\left(u_j - \alpha_j^m, \alpha_k^m\right)$. Thus we can efficiently compute and compare the stepwise improvements of the max–violating and max–lp2 pairs and choose the one with the largest improvement. We call this the *Composite–I* pair selection method. It adds a negligible amount of computation to the main loop and its stepwise improvement cannot be worse than either the max–violating pair or max–lp2 algorithm alone. We can extend this idea further by computing the stepwise improvement for all certifying pairs visited by Simon's algorithm and then choosing the best. We call this the *Composite–II* pair selection method. This methods adds a *non–negligible* amount of computation to the main loop, but may provide even better stepwise updates. It is worth mentioning that other methods have been recently introduced which examine a subset of pairs and choose the one with the largest stepwise improvement (e.g. see Fan et al., 2005; Lai et al., 2003). The methods described here are different in that they are designed specifically to satisfy the condition in Corollary 7.

We have described four pair selection methods; max–lp2, Composite–I (best of max–violating and max–lp2), Composite–II (best of certifying pairs visited by Simon's algorithm), and max–qp2 (largest stepwise improvement) which all yield decomposition algorithms that satisfy the iteration bound in Theorem 5 with $\tau = 1/(n-1)$, but whose *actual* computational requirements on a specific problem may be quite different. In Section 4 we perform experiments to investigate the actual computational requirements for these methods.

2.2 Stopping Rules

Algorithms derived from the model in Procedure 2 require a stopping rule. Indeed to achieve the run time guarantees described in the previous section the algorithms must be terminated properly. The most common stopping rule is based on the observation that, prior to convergence, a max–violating pair (j^*, k^*) represents the most extreme violator of the optimality conditions in (7). This suggests the stopping rule: *stop at the first iteration $\acute{m}$ where*

$$g_{j^*}(\alpha^{\acute{m}}) - g_{k^*}(\alpha^{\acute{m}}) \le tol \tag{13}$$

where $tol > 0$ is a user defined parameter. This stopping rule is employed by many existing decomposition algorithms (e.g. see Chang and Lin, 2001; Chen et al., 2006; Keerthi et al., 2001; Lin, 2002a) and is especially attractive for max–violating pair algorithms since the rule can be computed in constant time once a max–violating pair has been computed. Lin (2002a) justifies this rule by proving that the gap $g_{j^*}(\alpha^m) - g_{k^*}(\alpha^m)$ converges to zero asymptotically for the sequence of feasible points generated by a particular class of decomposition algorithms. In addition Keerthi and Gilbert (2002) prove that (13) is satisfied in a finite number of steps for a specific decomposition algorithm. However the efficacy of this stopping rule is not yet fully understood. In particular we do not know the relation between this rule and the accuracy of the approximate solution it produces, and we do not know the convergence rate properties of the sequence $(g_{j^*}(\alpha^m) - g_{k^*}(\alpha^m))$ on which the rule is based. In contrast we now introduce new stopping rules which guarantee a specified accuracy for the approximate solutions they produce, and whose convergence rate properties are well understood. In addition we will show that these new stopping rules can be computed in constant time when coupled with the pair selection strategies in the previous section.

The simplest stopping rule that guarantees an ε–optimal solution for a τ–rate certifying algorithm is to stop after $\acute{m}$ iterations where $\acute{m}$ is given by Theorem 5 with $R^* - R(\alpha^0)$ replaced by a suitable upper bound (e.g. 1). We call this *Stopping Rule 0*. However the bound in Theorem 5 is conservative. For a typical problem instance the algorithm may reach the accuracy ε in far fewer iterations. We introduce stopping rules that are tailored to the problem instance and therefore may terminate the algorithm much earlier. These rules compute an upper bound on $R^* - R(\alpha)$ *adaptively* and then stop the algorithm when this upper bound falls below ε. There are many ways to determine an upper bound on $R^* - R(\alpha)$. For example the primal-dual gap, which is the difference between the primal criterion value and the dual criterion value, provides such a bound and therefore could be used to terminate the algorithm. However, computing the primal-dual gap would add significant computation to the main loop and so we do not pursue it here. Instead we develop stopping rules that, when coupled with one of the pair selection methods in the previous section, are simple to compute. These rules use the bound $R^* - R(\alpha) \le \sigma(\alpha|W_2)/\tau$ which was first established by Hush and Scovel (2003) and is reestablished as part of the theorem below. The theorem and corollary below establish the viability of these rules by proving that this bound converges to zero as $R(\alpha^m) \to R^*$, and that if $R(\alpha^m) \to R^*$ at a certain rate then the bound converges to zero at a similar rate.

Theorem 8 *Consider the canonical dual QP problem in (4) with Gram matrix Q, constraint vector u, feasible set $\mathcal{A}$, criterion function R, and optimal criterion value R^*. Let $\alpha \in \mathcal{A}$ and let W_p be a size p working set. Then the gap $R^* - R(\alpha)$ is bounded below and above as follows:*

1. *Let $L \ge max_i Q_{ii}$ and*

$$\sup_{\{V_p : V_p \subseteq W_n\}} \sum_{i \in V_p} u_i^2 \;\le\; U_p$$

where the supremum is over all size p subsets of W_n. Then

$$\frac{\sigma(\alpha|W_p)}{2} \min\left(1,\ \frac{\sigma(\alpha|W_p)}{pLU_p}\right) \;\le\; R^* - R(\alpha). \tag{14}$$

2. *If W_p includes a τ–rate certifying pair for α then*

$$R^* - R(\alpha) \;\leq\; \frac{\sigma(\alpha|W_p)}{\tau}. \tag{15}$$

The next corollary follows trivially from Theorem 8.

Corollary 9 *Consider the canonical dual QP problem in (4) with criterion function R. For any sequence of feasible points (α^m) and corresponding sequence of working sets (W^m) that include τ–rate certifying pairs the following holds:*

$$R(\alpha^m) \to R^* \quad \Leftrightarrow \quad \sigma(\alpha^m|W^m) \to 0.$$

In addition, rates for $R(\alpha^m) \to R^*$ imply rates for $\sigma(\alpha^m|W^m) \to 0$.

This corollary guarantees that the following stopping rule will eventually terminate a τ–rate certifying algorithm, and that when terminated at iteration $\acute{m}$ it will produce a solution $\alpha^{\acute{m}}$ that satisfies $R(\alpha^{\acute{m}}) - R^* \leq \varepsilon$.

Definition 10 (Stopping Rule 1) *For a τ–rate certifying algorithm with τ–rate certifying pair sequence (W_2^m), stop at the first iteration $\acute{m}$ where $\sigma(\alpha^{\acute{m}}|W_2^{\acute{m}}) \leq \tau\varepsilon$.*

This rule can be implemented in constant time using (10). The effectiveness of this rule will depend on the tightness of the upper bound in (15) for values of α near the optimum. We can improve this stopping rule as follows. Define

$$\delta_R^m := R(\alpha^{m+1}) - R(\alpha^m)$$

and suppose we have the following bound at iteration m

$$R^* - R(\alpha^m) \leq s.$$

Then at iteration $m+1$ we have

$$R^* - R(\alpha^{m+1}) \leq \min\left(\frac{\sigma(\alpha^{m+1}|W_2^{m+1})}{\tau},\; s - \delta_R^m\right).$$

Thus an initial bound s^0 (e.g. $s^0 = \sigma^0/\tau$) can be improved using the recursion

$$s^{m+1} = \min\left(\frac{\sigma(\alpha^{m+1}|W_2^{m+1})}{\tau}, s^m - \delta_R^m\right)$$

which leads to the following stopping rule:

Definition 11 (Stopping Rule 2) *For a τ–rate certifying algorithm with τ–rate certifying pair sequence (W_2^m), stop at the first iteration $\acute{m}$ where $s^{\acute{m}} \leq \varepsilon$.*

This rule is at least as good as Stopping Rule 1 and possibly better. However it requires that we additionally compute the stepwise improvement $\delta_R^m = R(\alpha^{m+1}) - R(\alpha^m)$ at each iteration. In the worst case, since the criterion can be written $R(\alpha) = \frac{1}{2}\alpha \cdot (g(\alpha) + w) + w_0$, the stepwise improvement δ_R^m can be computed in $O(n)$ time (assuming $g(\alpha^m)$ has already been computed). However for *W2* variants this value can be computed in constant time using (12). In Section 4 we describe experimental results that compare all three stopping rules.

2.3 Computing the Offset

We have concluded our description of algorithms for the Decomposition routine in Procedure 1 and now proceed to describe an algorithm for the Offset routine. According to Theorem 2 this routine must solve

$$\hat{b} \in \arg\min_b \sum_{i=1}^n u_i \max\left(0, 1 - y_i(\hat{\psi}\cdot\phi(x_i) + b)\right).$$

An efficient algorithm for determining $\hat{b}$ is enabled by using (5) and (6) to write

$$\begin{aligned} 1 - y_i\hat{\psi}\cdot\phi(x_i) &= 1 - y_i\left(\frac{1}{2\lambda}\sum_{j=1}^n(\hat{\alpha}_j - l_j)k(x_j,x_i)\right) \\ &= 1 - y_i(Q(\hat{\alpha} - l))_i \;=\; y_i\big(w_i - (Q\hat{\alpha})_i\big) \;=\; y_i g_i(\hat{\alpha})\ . \end{aligned}$$

This simplifies the problem to

$$\hat{b} \in \arg\min_b \sum_{i=1}^n u_i \max\big(0,\ y_i\big(g_i(\hat{\alpha}) - b\big)\big)\,.$$

The criterion $\sum_{i=1}^n u_i \max\big(0,\ y_i\big(g_i(\hat{\alpha}) - b\big)\big)$ is the sum of hinge functions with slopes $-u_i y_i$ and b–intercepts $g_i(\hat{\alpha})$. It is easy to verify that the finite set $\{g_i(\hat{\alpha}), i = 1,...,n\}$ contains an optimal solution $\hat{b}$. To see this note that the sum of hinge functions creates a piecewise linear surface where minima occur at corners, and also possibly along flat spots that have a corner at each end. Since the corners coincide with the points $g_i(\hat{\alpha})$ the set $\{g_i(\hat{\alpha}), i = 1,...,n\}$ contains an optimal solution. The run time of the algorithm that performs a brute force computation of the criterion for every member of this set is $O(n^2)$. However this can be reduced to $O(n\log n)$ by first sorting the values $g_i(\hat{\alpha})$ and then visiting them in order, using constant time operations to update the criterion value at each step. The details are shown in Procedure 8 in Appendix 6.

2.4 A Complete Algorithm

We have now described a complete algorithm for computing an ϵ_p–optimal solution to the primal QP problem. A specific realization is provided by (Procedure 1,Section 2) and Procedures 3–8 in Appendix 6. Multiple options exist for the Decomposition routine depending on the choice of working set size, pair selection method, and stopping rule. The realization in the appendix implements a *W2* variant of the Composite–I decomposition algorithm with Stopping Rule 2 (and is easily modified to implement the Composite–II algorithm). In the next two sections we complete our run time analysis of decomposition algorithms.

3. Operational Analysis of Decomposition Algorithms

In this section we use Theorem 5 and Corollary 7 to determine run time bounds for rate certifying decomposition algorithms that are applied to the L1–SVM and DLD–SVM canonical dual QP problems. It is clear from Theorem 5 that these bounds will depend on the parameters τ, S, L, R^* and ϵ. Let us consider each of these in turn. In the algorithms below each working set contains either a max–lp2 pair or a pair whose stepwise improvement is at least as good as that of a max–lp2 pair. Thus by Corollary 7 we can set $\tau = 1/(n-1)$. Instead however we set $\tau = 1/n$ since this value is also valid and it greatly simplifies the iteration bounds without changing their basic nature. The parameter S will take on a different, but known, value for the L1–SVM and DLD–SVM problems as described below. Using the definition of L in Theorem 5 and the definition of Q in (5) we set $L = \frac{K}{2\lambda}$ where $K \geq \max_{1 \leq i \leq n} k(x_i, x_i)$. We consider two possibilities for K. The first is the value

$$K_n = \max_{1 \leq i \leq n} k(x_i, x_i)$$

which is used to bound the run time for a specific problem instance and the second is the constant

$$\bar{K} = \sup_{x \in X} k(x, x)$$

which is used to bound the run time for classes of problem instances that use the same kernel, e.g. SVM learning problems where the kernel is fixed. In the second case we are interested in problems where $\bar{K}$ is finite. For example for the Gaussian RBF kernel $k(x, x') = e^{-\sigma\|x-x'\|^2}$ we obtain $\bar{K} = 1$. The optimal criterion value R^* is unknown but restricted to $[0, 1]$. To see this we use (5) to obtain

$$R(\alpha) \;=\; -\frac{1}{2}\alpha \cdot Q\alpha + \alpha \cdot w + w_0 \;=\; -\frac{1}{2}(\alpha - l) \cdot Q(\alpha - l) + (\alpha - l) \cdot y.$$

Then since $l \in \mathcal{A}$ it follows that $R^* \geq R(l) = 0$. Furthermore, using the positivity of Q and the definition of l in (3) we obtain that for any $\alpha \in \mathcal{A}$ the bound

$$R(\alpha) \;=\; -\frac{1}{2}(\alpha - l) \cdot Q(\alpha - l) + (\alpha - l) \cdot y \;\leq\; (\alpha - l) \cdot y \;\leq\; u \cdot 1 = 1$$

holds. We have now considered all the parameters that determine the iteration bound except λ and ε which are chosen by the user.

Recent theoretical results by Steinwart and Scovel (2004, 2005); Scovel et al. (2005b) indicate that with a suitable choice of kernel and mild assumptions on the distribution the trained classifier's generalization error will approach the Bayes error at a fast rate if we choose $\lambda \propto n^{-\beta}$, where the rate is determined (in part) by the choice of $0 < \beta < 1$. Although these results hold for exact solutions to the primal QP problem it is likely that similar results will hold for approximate solutions as long as $\varepsilon_p \to 0$ at a sufficiently fast rate in n. However in practice there is little utility in improving the performance once it is sufficiently close to the Bayes error. This suggests that once we reach a suitably large value of n there may be no need to decrease λ and ε_p below some fixed values $\bar{\lambda}$ and $\bar{\varepsilon}_p$. Thus, for fixed values $\bar{\lambda} > 0$ and $\bar{\varepsilon}_p > 0$ we call any (λ, ε_p) that satisfies $\lambda \geq \bar{\lambda}$ and $\varepsilon_p \geq \bar{\varepsilon}_p$ an *operational* choice of these parameters. When $\bar{K}$ is finite Theorem 2 gives a corresponding fixed value $\bar{\varepsilon} = (2\sqrt{2\bar{K}} + 8\sqrt{\bar{\lambda}})^{-2}\bar{\lambda}\bar{\varepsilon}_p^{\,2} > 0$ that we use to define an operational choice of the dual accuracy ε.

We begin our analysis by considering decomposition algorithms for the L1–SVM problem. Although our emphasis is on rate certifying decomposition algorithms, our first theorem establishes a lower bound on the number of iterations for *any W2* decomposition algorithm.

Theorem 12 *Consider the L1–SVM canonical dual with optimal criterion value R^*. Any W2 variant of Procedure 2 that starts with $\alpha^0 = l$ will achieve $R^* - R(\alpha^m) \leq \varepsilon$ in no less than $\lceil \bar{m} \rceil$ iterations where*

$$\bar{m} \;=\; \max\left(0,\; \frac{n(R^* - \varepsilon)}{2}\right).$$

Remark 13 *When $R^* > \varepsilon$ the minimum number of iterations is proportional to n and increases linearly with R^*. Thus it is important to understand the conditions where R^* is significantly larger than ε. Under very general conditions it can be shown that, with high probability, $R^* \geq e^* - \varepsilon_n$ where e^* is the Bayes classification error and ε_n is a term that tends to 0 for large n. Thus, for large n, R^* will be significantly larger than ε when e^* is significantly larger than ε, which we might expect to be common in practice.*

We briefly outline a path that can be used to establish a formal proof of these claims. Since the duality gap for the L1–SVM primal and dual QP problems is zero, R^* is the optimal value of the primal QP problem (e.g. for finite and infinite dimensional problems respectively see Cristianini and Shawe-Taylor, 2000; Hush et al., 2005). Furthermore it is easy to show that R^* is greater than or equal to the corresponding empirical classification error (i.e. the training error). Therefore the error deviance result in (Hush et al., 2003) can be used to establish general conditions on the data set $T_n = ((x_1,y_1),\ldots,(x_n,y_n))$, the kernel k, and the regularization parameter λ such that the bound $R^* \geq e^* - \varepsilon_n$ holds with probability $1-\delta$, where $\varepsilon_n = O\left(\sqrt{\ln(\sqrt{n}/\delta)/n}\right)$. Since e^* is a constant it can be further shown that with a suitably chosen constant $c > 0$ and a sufficiently large value n_0, then $Pr\left(\text{ number of iterations } \geq \frac{n(e^*-\varepsilon)}{2+c}, \forall n \geq n_0)\right) \geq 1 - \delta_{n_0}$ where $\delta_{n_0} \to 0$ at a rate that is exponential in n_0. Thus when $e^* > \varepsilon$ we can prove that the number of iterations is $\Omega(n)$ with probability 1.

We now continue our analysis by establishing upper bounds on the computation required for rate certifying decomposition algorithms applied to the L1–SVM and the DLD–SVM problems. In the examples below we establish two types of computation bounds: *generic bounds* which hold for any value of n, any choice of $\lambda > 0$, and either value of K; and *operational bounds* that hold when $K = \bar{K}$ is finite and operational choices are made for ε and λ. In the latter case we obtain bounds that are uniform in λ and ε and whose constants depend on the operational limits $\underline{\varepsilon}$ and $\underline{\lambda}$. These bounds are expressed using $O(\cdot)$ notation which suppresses their dependence on $\bar{K}$, $\underline{\varepsilon}$ and $\underline{\lambda}$ but reveals their dependence on n. In both examples we first consider a general class of rate certifying decomposition algorithms whose working sets may be larger than 2. For these algorithms we establish generic and operational bounds on the number of iterations. Then we consider the *W2* variants of these algorithms and establish operational bounds on their overall run time.

Example 1 *Consider solving the L1–SVM canonical dual using a decomposition algorithm where each working set includes a certifying pair whose stepwise improvement is at least as good as that produced by a max–lp2 pair. This includes algorithms where each working set includes a max–lp2, Composite–I, Composite–II or max–qp2 pair. Applying Theorem 5 with $S = 1/n$, $L = K/2\lambda$, $R^* - R(\alpha^0) \leq 1$, $\tau = 1/n$ and $\varepsilon < 1$ gives the generic bound*

$$\acute{m} \leq \begin{cases} 2n \ln\left(\frac{1}{\varepsilon}\right), & \varepsilon \geq \frac{2K}{\lambda n} \\ \\ 2n\left(\frac{2K}{\lambda\varepsilon n} - 1 + \ln\left(\frac{\lambda n}{2K}\right)\right), & \varepsilon < \frac{2K}{\lambda n} \end{cases} \tag{16}$$

on the number of iterations. With $K = K_n$ this expression gives a bound on the number of iterations for a specific problem instance. When $K = \bar{K}$ is finite, operational choices are made for ε and λ, *and n is large the number of iterations is determined by the first case and is $O(n)$. This matches the lower bound in Remark 13 and is therefore optimal in this sense. For a W2 variant that uses an algorithm from Section 2.1 to compute a max–lp2, Composite–I or Composite–II pair at each iteration the main loop requires $O(n)$ computation to determine the pair, $O(\log n)$ computation to update the ordered list M, $O(1)$ computation to update α, and $O(n)$ computation to update the gradient. Thus the main loop requires a total of $O(n)$ computation. Combining the bounds on the number of iterations and the computation per iteration we obtain an overall computational requirement of $O(n^2)$. In contrast, for a W2 variant that computes a max–qp2 pair at each iteration the main loop computation will increase. Indeed the current best algorithm for computing a max–qp2 pair is a brute force search which requires $O(n^2)$ computation and we strongly suspect that this cannot be reduced to the $O(n)$ efficiency of Simon's algorithm. Combining this with the lower bound on the number of iterations in Remark 13 demonstrates that there are cases where the overall run time of the max–qp2 variant is inferior.*

Example 2 *Consider solving the DLD–SVM canonical dual using a decomposition algorithm where each working set includes a certifying pair whose stepwise improvement is at least as good as that produced by a max–lp2 pair. In this case we can determine a value for S as follows,*

$$\max_i u_i = \max\left(\frac{1}{(1+\rho)n_1}, \frac{\rho}{(1+\rho)n_{-1}}\right) \le \max\left(\frac{1}{n_1}, \frac{1}{n_{-1}}\right) = \frac{1}{\min(n_1, n_{-1})} := S$$

where n_1 and n_{-1} are the number of samples with labels $y = 1$ and $y = -1$ respectively as described in Section 2. Suppose that $n_1 \le n_{-1}$ (results for the opposite case are similar). Applying Theorem 5 with $L = K/2\lambda$, $R^* - R(\alpha^0) \le 1$, and $\tau = 1/n$ gives the generic bound

$$\acute{m} \le \begin{cases} 2n \ln\left(\frac{1}{\varepsilon}\right), & \varepsilon \ge \frac{2Kn}{\lambda n_1^2} \\ 2n\left(\frac{2Kn}{\varepsilon\lambda n_1^2} - 1 + \ln\left(\frac{\lambda n_1^2}{2Kn}\right)\right), & \varepsilon < \frac{2Kn}{\lambda n_1^2} \end{cases} \tag{17}$$

on the number of iterations. The dependence on n_1 distinguishes this bound from the bound in (16). With $K = K_n$ (17) gives a bound on the number of iterations for a specific problem instance. Suppose that $n_1 = \Omega(n)$. Then when $K = \bar{K}$ is finite, operational choices are made for ε and λ, and n is large the number of iterations is determined by the first case and is $O(n)$. For a W2 *variant that uses an algorithm from Section 2.1 to compute a max–lp2, Composite–I or Composite–II pair at each iteration the main loop requires $O(n)$ computation. Thus the overall computational requirement is $O(n^2)$.*

4. Experiments

The experiments in this section are designed to accomplish three goals: to investigate the utility of Stopping Rules 1 and 2 by comparing them with Stopping Rule 0, to compare actual versus worst case computational requirements, and to investigate the computational requirements of *W2* decomposition algorithms that use different pair selection methods. Our focus is on the computational requirements of the main loop of the decomposition algorithm since this loop contributes a dominating term to our run time analysis, and since the computational requirements of the other algorithmic components can be determined very accurately without experimentation. We compare the four rate certifying pair selection methods (max–qp2, max–lp2, Composite–I, Composite–II) described in Section 2.1, and a max–violating pair method that we call *max–vps*. This max–vps algorithm is identical to the Composite–I algorithm, except that when choosing between a max–lp2 and max–violating pair we always choose the max–violating pair. To provide objective comparisons all algorithms use the same stopping rule. This means that the max–vps algorithm uses a different stopping rule than existing max–violating algorithms. Nevertheless including the max–vps algorithm in our experiments helps provide insight into how the algorithms developed here might compare with existing algorithms.

Our experiments are based on two different problems: a DLD–SVM problem formed from the **Cyber–Security** data set described in (Steinwart et al., 2005) and an L1–SVM problem formed from the **Spambase** data set from the UCI repository (Blake and Merz, 1998). All experiments employ SVMs with a Gaussian RBF kernel $k(x,x') = e^{-\sigma\|x-x'\|^2}$. Since a value of the regularization parameter (λ,σ) that optimizes performance is usually not known ahead of time, the value that is ultimately used to design the classifier is usually determined through some type of search that requires running the algorithm with different values of (λ,σ). Thus it is important to understand how different values, optimal and otherwise, affect the run time. To explore this effect we present results for two different values, (λ^*,σ^*) and $(\bar{\lambda},\bar{\sigma})$, obtained as follows. We train the SVM at a set of grid values and choose (λ^*,σ^*) to be a value that gives the best performance on an independent validation data set [3]. Then $(\bar{\lambda},\bar{\sigma})$ is chosen to be some other grid value encountered during the search that yielded non–optimal but nontrivial performance (i.e. it achieves some separation of the training data). For the DLD–SVM the performance is defined by the risk function $\mathcal{R}$ in (Steinwart et al., 2005) and for the L1–SVM it is the average classification error.

The **Cyber–Security** data set was derived from network traffic collected from a single computer over a 16-month period. The goal is to build a detector that will recognize anomalous behavior from the machine. Each data sample is a 12–dimensional feature vector whose components represent real valued measurements of network activity over a one-hour window (e.g. "average number of bytes per session"). Anomalies are defined by choosing a uniform reference distribution and a density level $\rho = 1$. The parameter values $(\lambda^*,\sigma^*) = (10^{-7},10^{-1})$ and $(\bar{\lambda},\bar{\sigma}) = (.05,.05)$ were obtained by employing a grid search with n_1:n_{-1} = 4000:10,000 training samples and 2000:100,000 validation samples. The solution obtained with parameter values (λ^*,σ^*) separated the training data and gave a validation risk of $\mathcal{R} = 0.00025$. The corresponding *alarm rate* (i.e. the rate at which anomalies are predicted by the classifier once it is placed in operation) is 0.0005.

The **Spambase** data set contains 4601 samples from $\mathbb{R}^d_+ \times \{-1,1\}$ where $d = 57$. This data set contains 1813 samples with label $y = -1$ and 2788 samples with label $y = 1$. The parameter values $(\lambda^*,\sigma^*) = (10^{-6},10^{-3})$ and $(\bar{\lambda},\bar{\sigma}) = (10^{-2},10^{-3})$ are obtained by employing a grid search with 3601 training samples and 1000 validation samples. The solution obtained with parameter values (λ^*,σ^*) did not separate the training data and gave a classification error of 0.093 on the validation set.

We present results for three experiments.

3. More specifically, for each value of $\lambda \in \{1,.5,.1,.05,\ldots,.000005,.0000001\}$ we search a grid of σ values that starts with the set $\{0.001,0.01,0.05,0.1,0.5,1,5,10,100\}$ and is refined using a golden search as described in (Steinwart et al., 2005, Section 4).

Experiment 1 *This experiment investigates the utility of Stopping Rules 1 and 2 by comparing them with Stopping Rule 0. More specifically we compare the actual criterion gap $R^* - R(\alpha^m)$ to the bounds used by these three stopping rules. We refer to the bounds for Stopping Rules 0, 1, and 2 as Bounds 0, 1, and 2 respectively. To obtain an estimate $\hat{R}^*$ of R^* we run the decomposition algorithm in Procedure 3 with $\varepsilon = 10^{-10}$ and compute the resulting criterion value. Then to obtain results for comparison we run this algorithm again and compute: the criterion gap $\hat{R}^* - R(\alpha^m)$, Bound 1 given by $n\sigma(\alpha^m|W_2^m)$, Bound 2 obtained from the recursive rule $s^m = \min(n\sigma(\alpha^m|W_2^m), s^{m-1} - \delta_R^{m-1})$, and Bound 0 given by equation (23) in the proof of Theorem 5.*

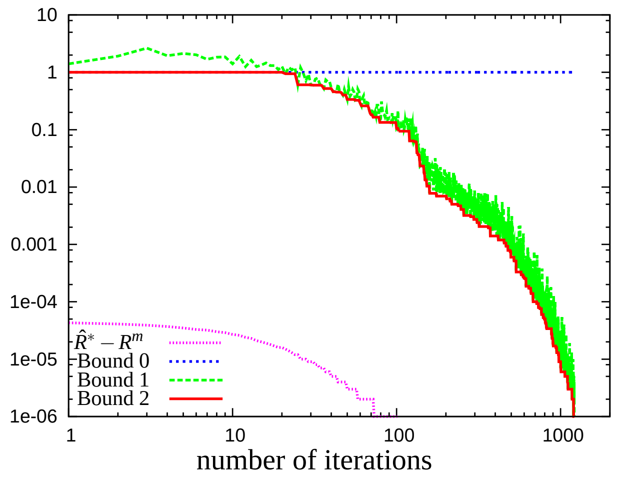

Figure 1: The criterion gap $\hat{R}^* - R^m$ and bounds on this gap employed by Stopping Rules 0, 1 and 2 for the **Cyber–Security** data. Bound 0 and 2 are indistinguishable up to about iteration 25, at which point they separate and Bound 2 becomes a monotonically decreasing lower envelope of Bound 1.

A plot of these values when the algorithm is applied to the **Cyber–Security** *data with $(\lambda^*, \sigma^*) = (10^{-7}, 10^{-1})$ and n_1:n_{-1} = 4000:10000 is shown in Figure 1. While Bound 1 is a bit erratic Bound 2 is monotonic and relatively smooth. Nevertheless both will stop the algorithm at nearly the same iteration (unless ε is very close to 1). In addition while Bounds 1 and 2 may be loose, i.e. they are often several orders of magnitude larger than the actual criterion gap, their behavior tracks that of the criterion gap relatively well and therefore the corresponding stopping rules are very effective relative to Rule 0. For example suppose we choose $\varepsilon = 10^{-5}$. Because the initial criterion gap is so small it takes only about* 25 *iterations for the algorithm to reach this accuracy. Both Stopping Rules 1 and 2 terminate the algorithm after approximately* 1000 *iterations, but Stopping Rule 0 terminates after approximately* 1.225×10^{13} *iterations (approximately* 10 *orders of magnitude more).*

Results obtained by applying the algorithm to the **Spambase** *data with $(\lambda^*, \sigma^*) = (10^{-6}, 10^{-3})$ and $n = 4601$ are shown in Figure 2. In this case the initial criterion gap is larger so the separation between the criterion gap and the bounds is smaller. Once again Bound 1 is a bit erratic, and this time there are several regions (beyond the initial region) where Bounds 1 and 2 are well separated. This suggests that the monotonic behavior of Bound 2 provides a more robust stopping rule. As before Bounds 1 and 2 are loose, but their behavior tracks that of the criterion gap relatively well and*

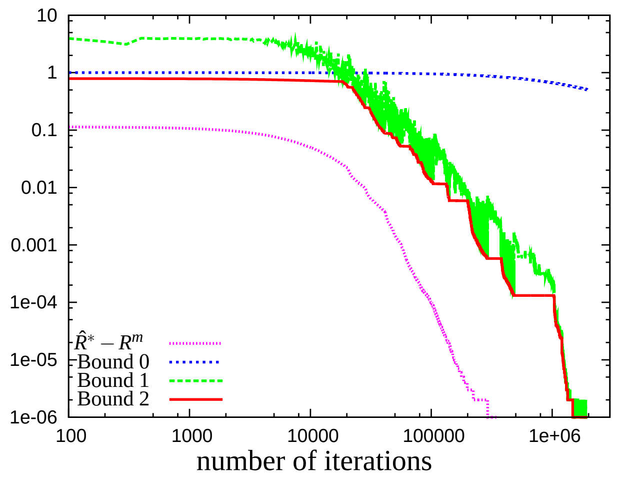

Figure 2: The criterion gap $\hat{R}^* - R^m$ and bounds on this gap employed by Stopping Rules 0, 1 and 2 for the **Spambase** data. Bound 0 and 2 are close up to about iteration 20,000, at which point they separate and Bound 2 becomes a monotonically decreasing lower envelope of Bound 1.

therefore the corresponding stopping rules are very effective. For example it takes about $200,000$ *iterations for the algorithm to reach an accuracy* $\varepsilon = 10^{-5}$*, while both Stopping Rules 1 and 2 terminate the algorithm after approximately* $2,000,000$ *iterations and Stopping Rule 0 terminates after approximately* 4×10^{11} *iterations (approximately* 5 *orders of magnitude more). More generally the number of the excess iterations for Stopping Rule 2 appears to be less than an order of magnitude for a large range of values of* ε*.*

In both cases above it is clear that Stopping Rules 1 and 2 are far superior to Stopping Rule 0.

Experiment 2 *This experiment compares actual computational requirements for the main loop of various decomposition algorithms applied to the* **Cyber–Security** *data. With density level* $\rho = 1$*, accuracy* $\varepsilon = 10^{-6}$*, parameter values* $(\lambda^*, \sigma^*) = (10^{-7}, 10^{-1})$ *and* $(\overline{\lambda}, \overline{\sigma}) = (.05, .05)$*, and five different problem sizes* n_1*:*n_{-1} *= 2000:4000, 2500:5000, 3000:6000, 3500:7000, and 4000:8000 we employed the decomposition algorithm with Stopping Rule 2 and pair selection methods max–lp2, Composite–I, Composite–II, max–vps and max–qp2. For each problem size we generated ten different training sets by randomly sampling (without replacement) the original data set. Then we ran the decomposition algorithm on each training set and recorded the number of iterations and the wallclock time of the main loop. The minimum, maximum and average values of these quantities for parameter values* $(\lambda^*, \sigma^*) = (10^{-7}, 10^{-1})$ *are shown in Figure 3* [4]*. There is much to discern from the plot on the left. It is easy to verify that for all pair selection methods the numbers of iterations are several orders of magnitude smaller than the worst case bound given in Example 2. On average the convergence rate of the max–lp2 method is much worse than the other methods. This may be partly due to the fact that this method uses only first order information to determine its pair.*

4. In Figures 3–6 the x–axis values of some points are slightly offset so that their y–axis values can be more easily visualized.

However, this is also true of the max–vps method whose convergence rate is much faster. Indeed, it is curious that the max–lp2 method, which chooses a stepwise direction based on a combination of **steepness** *and* **room to move***, has a worse convergence rate than the max–vps method, which chooses a stepwise direction based on* **steepness** *alone. By slightly modifying the max–lp2 method to obtain the Composite–I method a much faster convergence rate is observed. The Composite–I*

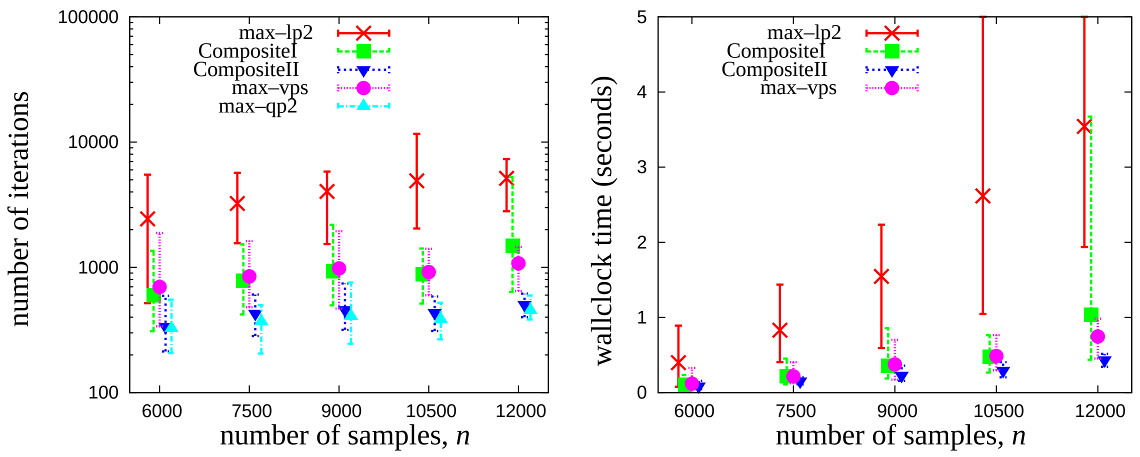

Figure 3: Main loop computation for **Cyber–Security** data with $(\lambda^*, \sigma^*) = (10^{-7}, 10^{-1})$.

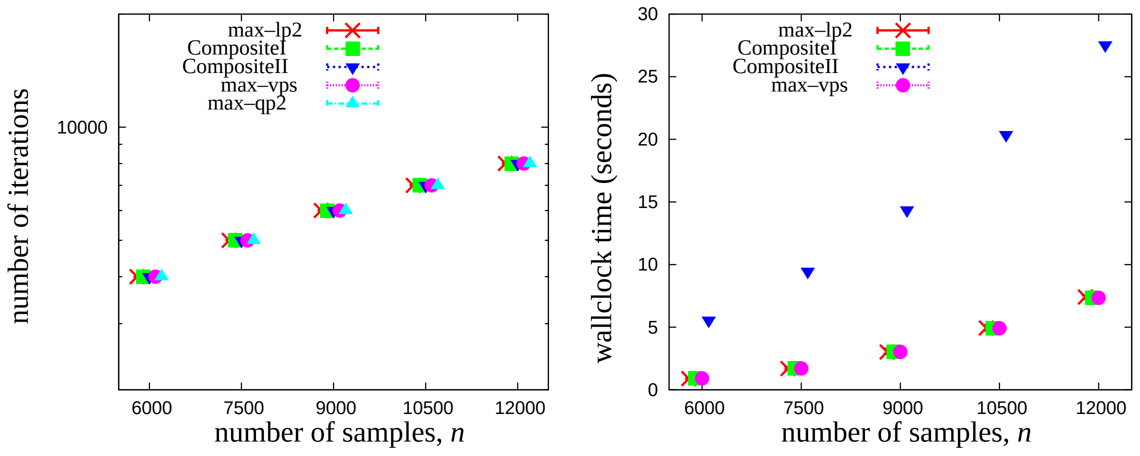

Figure 4: Main loop computation for **Cyber–Security** data with $(\bar{\lambda}, \bar{\sigma}) = (.05, .05)$. The number of iterations in the left plot is identical for all five methods for all values of n. The wallclock time in the right plot is indistinguishable for the Composite–I, max–vps and max–lp2 methods.

and max–vps methods have roughly the same convergence rate. This suggests that Composite–I may be achieving its improved rate by choosing a max–violating pair a large fraction of the time. Indeed, on a typical run of the Composite–I method we found that, among the 53% of the iterations where the max–lp2 and max–violating pairs were different, a max–violating pair was chosen 4.3 times as *often. Although a larger stepwise improvement does not guarantee a faster convergence rate the max–qp2 method, which gives the largest stepwise improvement, also gave the fastest convergence rate. However the Composite–II method, which requires far less computation than the max–qp2 method, gave nearly the same convergence rate. Quantitatively the average number of iterations for the max–lp2 method is roughly 9 times that of Composite–II, while the average number of iterations for Composite–I is roughly 2 times that of Composite–II. The variation in the number of iterations is smallest for Composite–II and max–qp2, followed by Composite–I and max–vps, and then max–lp2. This variation ranges from 2x to 8x across the different sample sizes and methods. The plot on the right shows the wallclock times. The times for the max–qp2 method are omitted because they are much larger than the rest. Indeed they are roughly n times larger than the wallclock times for Composite–II. The Composite–II method achieved the fastest average wallclock times which were roughly 6.8 times faster than the max–lp2 method and 1.6 times faster than the Composite–I and max–vps methods.*

Results for parameter value $(\bar{\lambda},\bar{\sigma}) = (.05,.05)$ *are shown in Figure 4. The computational requirements here are greater than with the previous parameter value. We attribute this primarily to the fact that* R^* *is larger so that the initial criterion gap is larger. The larger value of* λ *corresponds to a strong regularization term that produces a solution where all components of* α *are forced from their initial values at one bound to their final values at the opposite bound. To move all* $n_1 + n_{-1}$ *components of* α *to their opposite bound using working sets that contain one sample from each class requires* n_{-1} *iterations (since* $n_{-1} > n_1$*) and this is exactly what the algorithms did for all five pair selection methods on every training set. This is a quintessential example of a problem where the number of iterations must be (at least) a significant fraction of the number of training samples regardless of which algorithm is used. The resulting solution has the simple interpretation that its normal vector is the difference in class means. The wallclock times of the max–lp2, Composite–I and max–vps algorithms are roughly 5 times faster than the Composite–II algorithm because of the extra computation per iteration employed by Composite–II. The relationship between the number of iterations and the training set size is demonstrably linear, and the relationship between the wallclock times and the training set size is demonstrably quadratic. These relations coincide with the linear and quadratic forms predicted by the analysis in Section 3.*

Experiment 3 *This experiment is similar to the previous experiment except that the algorithms are applied to the* **Spambase** *data. With accuracy* $\varepsilon = 10^{-6}$*, parameter values* $(\lambda^*,\sigma^*) = (10^{-6},10^{-3})$ *and* $(\bar{\lambda},\bar{\sigma}) = (10^{-2},10^{-3})$*, and seven different problem sizes* $n = 1000, 1500, 2000, 2500, 3000, 3500, 4000$ *we employed the decomposition algorithm with Stopping Rule 2 and pair selection methods max–lp2, Composite–I, Composite–II, max–vps and max–qp2. We ran the decomposition algorithm on ten different training sets for each problem size and recorded the number of iterations and the wallclock time of the main loop. The minimum, maximum and average values of these quantities for runs with parameter values* $(\lambda^*,\sigma^*) = (10^{-6},10^{-3})$ *are shown in Figure 5. Once again it is easy to verify that for all pair selection methods the numbers of iterations in the left plot are several orders of magnitude smaller than the worst case bound given in Example 1. In addition the convergence rate is fastest for the Composite–II and max–qp2 methods, followed by the Composite–I and max–vps methods, and then the max–qp2 method. In this case it appears that the max–vps method has a slight edge on the Composite–I method. On a typical run of the Composite–I method we found that, among the 64% of the iterations where the max–lp2 and max–violating pairs were different, a max–violating pair was chosen 3.9 times as often. The variation in the number of iterations, which* *ranges from 2x to 4x across the different sample sizes and methods, is smallest for Composite–II and max–qp2, followed by Composite–I and max–vps, and then max–lp2. Quantitatively the average number of iterations for max–lp2, Composite–I and max–vps is roughly 92, 13 and 11 times that of Composite–II respectively. In addition the average wallclock times of the max–lp2, Composite–I and max–vps are roughly 23.6, 3.8 and 2.5 times that of Composite–II respectively. Once again the plot on the right does not show the wallclock times for the max–qp2 method, but they are roughly* $n/4$ *times that of the Composite–II method.*

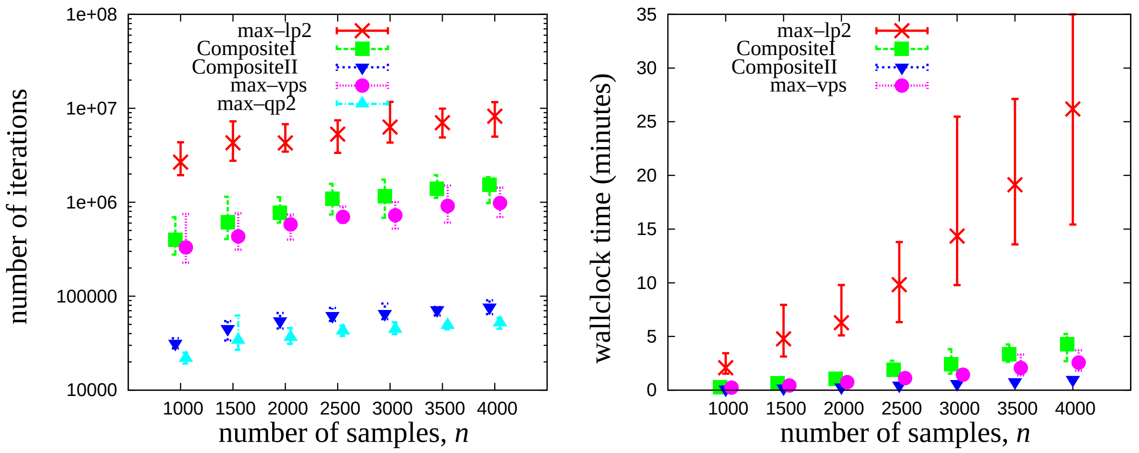

Figure 5: Main loop computation for **Spambase** data: $(\lambda^*, \sigma^*) = (10^{-6}, 10^{-3})$.

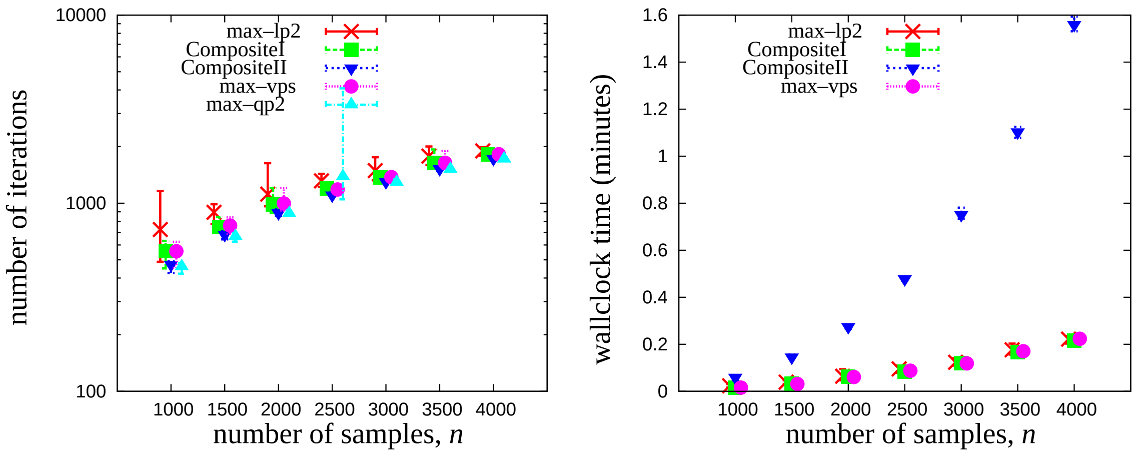

Figure 6: Main loop computation for **Spambase** data: $(\bar{\lambda}, \bar{\sigma}) = (10^{-2}, 10^{-3})$. The number of iterations in the left plot is similar for all three methods. The wallclock time in the right plot is nearly indistinguishable for the Composite–I and max–lp2 methods.

The results for parameter values $(\bar{\lambda}, \bar{\sigma}) = (10^{-2}, 10^{-3})$ *are shown in Figure 6 and indicate a significant decrease in the computational requirements. This decrease in computation as a result of a larger* λ *is opposite to what we observed in Experiment 2. We attribute this to the fact that* *the switch from (λ^*, σ^*) to $(\bar{\lambda}, \bar{\sigma})$ did not yield a big change in the initial criterion gap as it did in Experiment 2. However most other characteristics of the solutions produced here are similar to those in Experiment 2. Indeed the number of iterations is roughly the same for all five pair selection methods and the wallclock times for the max–lp2, Composite–I and max–vps algorithms are approximately 5 times faster than Composite–II. In addition the relationships between the number of iterations, the wallclock times, and the training set size coincide with the linear and quadratic forms predicted by the analysis in the previous section.*

For the L1–SVM the gap between the lower and upper iteration bounds is smaller when λ is larger. Indeed, for large λ and large n the lower bound is $\frac{n}{2}(R^* - \varepsilon)$ and the upper bound is $2n\ln\frac{R^*}{\varepsilon}$. When R^* is large these two values may differ by no more than a factor of 10. This partially explains why the computational requirements for the strongly regularized problem instances in Experiments 2 and 3 exhibit such a low variance and coincide so well with the predicted linear and quadratic forms. In these cases the max–lp2, Composite–I and max–vps algorithms are fastest because they require less computation per iteration. On the other hand, in instances where (λ, σ) give near–optimal performance the values of λ are smaller and so the gaps between the lower and upper bounds are often much larger. In these cases the actual computation is often not close to either bound, the variance is higher, and the Composite–II algorithm is the fastest because it requires far fewer iterations. In addition these near–optimal values of λ can give a smaller value for R^*, especially when they yield a solution that separates the training data. In such cases the initial criterion gap is smaller and the run times are often faster. This is the most likely explanation for the significantly lower computational requirements for the **Cyber–Security** experiments.

5. Summary

We have described SVM classifier design algorithms that allow a different weight for each training sample. These algorithms accept an accuracy ε_p of a primal QP problem as input and are guaranteed to produce an approximate solution that satisfies this accuracy in low order polynomial time. They employ a two–stage process where the first stage produces an approximate solution to a dual QP problem and the second stage maps this approximate dual solution to an approximate primal solution. For the second stage we have described a simple $O(n\log n)$ algorithm that maps an approximate dual solution with accuracy $(2\sqrt{2K} + 8\sqrt{\lambda})^{-2}\lambda\varepsilon_p^2$ to an approximate primal solution with accuracy ε_p. For the first stage we have presented new results for decomposition algorithms and we have described decomposition algorithms that employ new pair selection methods and new stopping rules.

For τ–*rate certifying* decomposition algorithms we have established the optimality of $\tau = 1/(n-1)$ and described several pair selection methods (max–qp2, max–lp2, Composite–I, Composite–II) that achieve the $\tau = 1/(n-1)$ iteration bound. We have also introduced new stopping rules that are computationally efficient and that guarantee a specified accuracy for the approximate dual solution. While these stopping rules can be used by any decomposition algorithm they are especially attractive for the algorithms developed here because they add a negligible amount of computation to the main loop.

Since the pair selection methods (max–lp2, Composite–I, Composite–II) require $O(n)$ computation they yield *W2* decomposition algorithms that require only $O(n)$ computation in the main loop. In addition, for the L1–SVM dual QP problem we have described operational conditions for which these *W2* decomposition algorithms possess an upper bound of $O(n)$ on the number of iterations.

For this same problem we have presented a lower bound for *any W2* decomposition algorithm and we have described general conditions for which this bound is $\Omega(n)$. Combining the bounds on main loop computation with the bounds on number of iterations yields an overall run time of $O(n^2)$. Our experiments suggest that the pair selection algorithms with the most promise are the Composite–I and Composite–II algorithms which were obtained through a simple extension of Simon's algorithm.

Once the run time of the decomposition algorithm has been established it is straightforward to determine the run time of the main routine in Procedure 1. Let c_k be an upper bound on the time it takes to perform a kernel evaluation. For an instance of L1–SVM where $\bar{K}$ is finite and operational choices are made for ε_p and λ Procedure 1 takes $O(c_k n^2)$ time to compute the parameters for the canonical dual on lines 7-8, $O(n)$ time to set α^0 on line 9, $O\left(n^2\right)$ time to compute an approximate dual solution on line 10, and $O(n \log n)$ time to compute the offset $\hat{b}$ on line 11. Thus, the overall run time is $O(n^2(c_k+1))$. This run time analysis assumes that the matrix Q is computed once and stored in main memory for fast (constant time) access. However the storage requirements for this matrix may exceed the size of main memory. If this issue is resolved by computing a kernel evaluation each time an element of Q is accessed then the time to compute an approximate dual solution is multiplied by c_k. On the other hand if the elements of Q are cached in a block of main memory so that the average access time for an element of Q is βc_k, where $0 < \beta \leq 1$ is determined by the size and replacement strategy for the cache, then the multiplier is reduced to βc_k for the average case. It is an interesting topic of future research to determine how the different pair selection methods affect the efficiency of the cache.

We note that algorithmic enhancements such as the shrinking heuristic in (Joachims, 1998) can easily be adapted to the algorithms presented here. In addition, the algorithms in this paper have been developed for the SVM formulation in (1), but similar algorithms with the same run time guarantees can be developed for the 1-CLASS formulation of Schölkopf et al. (2001) which has a similar form for the dual.

6. Proofs

The following lemma is used in the proofs of Theorems 5 and 8. It provides upper and lower bounds on the improvement in criterion value obtained by solving the restricted QP problem determined by a feasible point α and an *arbitrary* working set W_q.

Lemma 14 *Consider the canonical dual QP problem in (4) with Gram matrix Q, constraint vector u, feasible set $\mathcal{A}$, criterion function R, and optimal criterion value R^*. For $\alpha \in \mathcal{A}$ and a size q working set W_q let*

$$\alpha_q \in \arg \max_{\gamma \in \mathcal{A}(\alpha, W_q)} R(\gamma)$$

be a solution to the QP problem at (α, W_q). Then

$$R(\alpha_q) - R(\alpha) \; \leq \; \sigma(\alpha | W_q). \tag{18}$$

Furthermore, for $(\bar{\sigma}, L, U_q)$ satisfying $\bar{\sigma} \leq \sigma(\alpha | W_q)$, $L \geq max_i Q_{ii}$, and

$$\sup_{\{V_q : V_q \subseteq W_n\}} \sum_{i \in V_q} u_i^2 \; \leq \; U_q$$

where the supremum is over all size p subsets of W_n, the following bound holds,

$$R(\alpha_q) - R(\alpha) \ \geq \ \begin{cases} \bar{\sigma}/2, & \bar{\sigma} \geq qLU_q \\ \frac{\bar{\sigma}^2}{2qLU_q}, & \bar{\sigma} < qLU_q \end{cases} . \tag{19}$$

Proof First we prove the upper bound. From the positivity of Q and the definition of σ we obtain

$$R(\alpha_q) - R(\alpha) \ = \ g(\alpha)\cdot(\alpha_q - \alpha) - \frac{1}{2}(\alpha_q - \alpha)\cdot Q(\alpha_q - \alpha) \ \leq \ g(\alpha)\cdot(\alpha_q - \alpha) \ \leq \ \sigma(\alpha|W_q).$$

Now we prove the lower bound. Let

$$\acute{\alpha}_q \in \arg \max_{\gamma \in \mathcal{A}(\alpha, W_q)} g(\alpha)\cdot(\gamma - \alpha)$$

be a solution to the LP problem at (α, W_q) and consider the direction $d_q := \acute{\alpha}_q - \alpha$. The improvement in criterion value for any feasible point in this direction cannot be larger than the improvement for α_q, i.e.

$$R(\alpha_q) - R(\alpha) \geq R(\alpha + \omega d_q) - R(\alpha), \ \ 0 \leq \omega \leq 1. \tag{20}$$

To obtain a lower bound for the right side we start by writing

$$R(\alpha + \omega d_q) - R(\alpha) \ = \ \omega g(\alpha)\cdot d_q - \frac{\omega^2}{2} d_q \cdot Q d_q \ = \ \omega\sigma(\alpha|W_q) - \frac{\omega^2}{2} d_q \cdot Q d_q \ \geq \ \omega\bar{\sigma} - \frac{\omega^2}{2} d_q \cdot Q d_q.$$

Note that d_q has at most q nonzero components determined by the members of W_q. Let Q_q be the $q \times q$ matrix formed from the elements $Q_{ij} : i, j \in W_q$, and let $\lambda_{max}(Q_q)$ and $\mathrm{trace}(Q_q)$ be the largest eigenvalue and the trace of Q_p. Since $Q \geq 0 \Rightarrow Q_p \geq 0$ we have $\lambda_{max}(Q_q) \leq \mathrm{trace}(Q_q) \leq qL$. Thus

$$d_q \cdot Q d_q \ \leq \ \lambda_{max}(Q_q)(d_q \cdot d_q) \ \leq \ qL \sum_{i \in W_q} u_i^2 \ \leq \ qLU_q$$

and therefore

$$R(\alpha + \omega d_q) - R(\alpha) \ \geq \ \omega\bar{\sigma} - \frac{\omega^2}{2} qLU_q.$$

Choosing $\omega \in [0,1]$ to maximize the right side gives

$$\omega^* = \begin{cases} 1, & \bar{\sigma} \geq qLU_q \\ \frac{\bar{\sigma}}{qLU_q}, & \bar{\sigma} < qLU_q \end{cases}$$

so that

$$R(\alpha + \omega^* d_q) - R(\alpha) \geq \begin{cases} \bar{\sigma} - \frac{qLU_q}{2}, & \bar{\sigma} \geq qLU_q \\ \frac{\bar{\sigma}^2}{2qLU_q}, & \bar{\sigma} < qLU_q \end{cases} . \tag{21}$$

The first case satisfies

$$\bar{\sigma} - \frac{qLU_q}{2} \ \geq \ \bar{\sigma}/2$$

so that

$$R(\alpha + \omega^* d_q) - R(\alpha) \geq \begin{cases} \bar{\sigma}/2, & \bar{\sigma} \geq qLU_q \\ \frac{\bar{\sigma}^2}{2qLU_q}, & \bar{\sigma} < qLU_q \end{cases} .$$

Combining this result with (20) gives the result in (19). ■

Proof [Proof of Theorem 5] This proof is a slight modification of the proof in (List and Simon, 2005, Section 3.3) so we describe only the main differences. The basic approach is to obtain an upper bound on the number of iterations by deriving a lower bound on the stepwise improvement. The first difference is based on an idea from the proof of Hush and Scovel (2003, Theorem 5). Let $W_2^m \subseteq W^m$ be a τ–rate certifying pair for α^m. The stepwise improvement with W^m is at least as good as the stepwise improvement with W_2^m and therefore

$$R(\alpha^{m+1}) - R(\alpha^m) \geq R(\acute{\alpha}^{m+1}) - R(\alpha^m) \tag{22}$$

where $\acute{\alpha}^{m+1}$ is a solution to the two–variable QP problem at (α^m, W_2^m). Define

$$\Delta^m := R^* - R(\alpha^m).$$

Since $\sigma(\alpha^m|W_2^m) \geq \tau\Delta^m$ (see Hush and Scovel, 2003; List and Simon, 2005) we can bound the right side of (22) by applying the lower bound in (Lemma 14, Equation (19)) with $q = 2$, $\bar{\sigma} = \tau\Delta^m$, and $U_2 = 2S$ to obtain

$$R(\acute{\alpha}^{m+1}) - R(\alpha^m) \;\geq\; \begin{cases} \frac{\tau\Delta^m}{2}, & \Delta^m \geq \frac{4LS^2}{\tau} \\ \frac{(\tau\Delta^m)^2}{8LS}, & \Delta^m < \frac{4LS^2}{\tau} \end{cases} .$$

Combining this result with (22) and using $R(\alpha^{m+1}) - R(\alpha^m) = \Delta^m - \Delta^{m+1}$ we obtain

$$\Delta^{m+1} \leq \left(1 - \frac{\tau}{2}\right)\Delta^m, \quad \text{when } \Delta^m \geq \frac{4LS^2}{\tau}$$

$$\Delta^{m+1} \leq \Delta^m - \gamma(\Delta^m)^2, \quad \text{when } \Delta^m < \frac{4LS^2}{\tau}$$

where $\gamma = \tau^2/8LS^2$. This is essentially the same result obtained in (List and Simon, 2005, p. 316) except that here we have $4L$ in place of the term qL_{max} in (List and Simon, 2005) where q is the size of the working set W^m and L_{max} is the largest among the eigenvalues of all the principle $q \times q$ submatrices of Q. To complete the proof we follow the steps in (List and Simon, 2005, Section 3.3) until the bottom of page 317 where we retain the (slightly) tighter bound

$$\delta_m \geq \delta_{m_0} + \gamma(m - m_0)$$

where, in our case, $\delta_{m_0} \geq \tau/4LS^2$. This gives a bound on the criterion gap

$$\Delta^m \leq \frac{1}{\delta_{m_0} + \gamma(m - m_0)} \tag{23}$$

which leads to the "-1" term in the second part of our expression for $\acute{m}$ and ensures that the expressions in first and second parts match at the boundary where $\epsilon = \frac{4LS^2}{\tau}$. ■

Proof [Proof of Theorem 6:] We start by proving the first assertion. To simplify notation we write g as a shorthand for $g(\alpha)$. Since setting $\acute{\alpha} = \alpha$ gives $g \cdot (\acute{\alpha} - \alpha) = 0$ it follows that $\sigma(\alpha|W_n) \geq 0$. Similarly $\sigma(\alpha|W_2) \geq 0$ for all $W_2 \subseteq W_n$. Thus when $\sigma(\alpha|W_n) = 0$ it follows that the assertion is true. Therefore let us assume $\sigma(\alpha|W_n) > 0$.

Let

$$W_2^* \in \arg\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2).$$

We start by deriving an expression for $\sigma(\alpha|W_2^*)$. A two–variable problem with working set $W_2 = \{j,k\}$ satisfies

$$\begin{aligned}
\sigma(\alpha|W_2) = \sup_{\acute{\alpha}\in\mathcal{A}(\alpha,W_2)} g\cdot(\acute{\alpha}-\alpha) &= \sup_{\alpha+d\in\mathcal{A}(\alpha,W_2)} g\cdot d \\
&= \sup_{\substack{d_j=-d_k\\ -\alpha_j\le d_j\le u_j-\alpha_j\\ -\alpha_k\le d_k\le u_k-\alpha_k}} d_jg_j+d_kg_k \\
&= \sup_{\substack{-\alpha_j\le d_j\le u_j-\alpha_j\\ \alpha_k-u_k\le d_j\le \alpha_k}} d_j(g_j-g_k) \\
&= \Delta_{jk}(g_j-g_k)
\end{aligned}$$

where

$$\Delta_{jk} = \begin{cases} \min(u_j-\alpha_j,\alpha_k), & g_j>g_k \\ -\min(\alpha_j,u_k-\alpha_k), & g_j<g_k \\ 0, & g_j=g_k \end{cases}.$$

The expression for $\sigma(\alpha|W_2^*)$ is obtained by maximizing over all pairs,

$$\sigma(\alpha|W_2^*) = \max_{\{j,k\}\subseteq W_n} \Delta_{jk}(g_j-g_k). \tag{24}$$

Now write

$$\sigma(\alpha|W_n) = \sup_{\acute{\alpha}\in\mathcal{A}} g\cdot(\acute{\alpha}-\alpha) = \sup_{\alpha+d\in\mathcal{A}} g\cdot d = \sup_{d\in\mathcal{D}} g\cdot d$$

where

$$\mathcal{D} = \{d : d\cdot 1 = 0, -\alpha_i \le d_i \le u_i-\alpha_i\}.$$

Let d^* be a solution so that

$$\sigma(\alpha|W_n) = g\cdot d^*.$$

The intuition for what follows is that we will (implicitly) decompose d^* into

$$d^* = \overline{d^1}+\ldots+\overline{d^p}$$

such that $p\le n-1$, and for every $i\in\{1,\ldots,p\}$ $\overline{d^i}$ has only two non-zero components and $\alpha+\overline{d^i}$ is feasible at α. Define the index sets

$$I_+ = \{i : d_i^* > 0\}, \quad I_- = \{i : d_i^* < 0\}$$

and write

$$\sigma(\alpha|W_n) = \sum_{i\in I_+} d_i^* g_i + \sum_{i\in I_-} d_i^* g_i.$$

Note that $\sigma(\alpha|W_n) \neq 0$ and $d^* \cdot 1 = 0$ imply that both I_+ and I_- are non–empty. We decompose the right hand side into a sum of two–variable terms by applying the following recursion. Initialize with $m = 0$, $d_i^0 = d_i^*$, $I_+^0 = I_+$, $I_-^0 = I_-$, and

$$h^0 = \sum_{i\in I_+^0} d_i^0 g_i + \sum_{i\in I_-^0} d_i^0 g_i.$$

Then while $h^m \neq 0$ choose an index pair $(j_m, k_m) \in I_+^m \times I_-^m$, define $\delta_{j_m k_m} = \min(d_{j_m}^m, -d_{k_m}^m)$, and define

$$h^{m+1} = \sum_{i\in I_+^{m+1}} d_i^{m+1} g_i + \sum_{i\in I_-^{m+1}} d_i^{m+1} g_i$$

where

$$d_i^{m+1} = \begin{cases} d_i^m - \delta_{j_m k_m}, & i = j_m \\ d_i^m + \delta_{j_m k_m}, & i = k_m \\ d_i^m, & i \neq j_m \text{ or } k_m \end{cases} \tag{25}$$

and

$$I_+^{m+1} = \{i : d_i^{m+1} > 0\}, \quad I_-^{m+1} = \{i : d_i^{m+1} < 0\}.$$

This gives the recursion

$$h^{m+1} = h^m - \delta_{j_m k_m}(g_{j_m} - g_{k_m}).$$

From equation (25) it follows that

$$d^m \in D \quad \Rightarrow \quad d^{m+1} \in D.$$

Thus if $h^{m+1} \neq 0$ then both I_+^{m+1} and I_-^{m+1} are non–empty verifying the existence of an index pair for the next iteration. Furthermore, the definition of $\delta_{j_m k_m}$ implies that either $d_{j_m}^{m+1} = 0$ or $d_{k_m}^{m+1} = 0$ (or both) so that the size of the index sets decreases by at least one at each iteration, i.e.

$$|I_+^{m+1} \cup I_-^{m+1}| \leq |I_+^m \cup I_-^m| - 1.$$

Therefore at least one of the index sets becomes empty after at most $n-1$ iterations. Furthermore $d^m \cdot 1 = 0$ implies that both index sets become empty at the same iteration. Thus this recursion decomposes the original sum as follows

$$\sigma(\alpha|W_n) = h^0 = \delta_{j_1k_1}(g_{j_1} - g_{k_1}) \ + \ \delta_{j_2k_2}(g_{j_2} - g_{k_2}) \ + \ \ldots \ + \ \delta_{j_pk_p}(g_{j_p} - g_{k_p}) \tag{26}$$

where $p \leq n-1$. Let q be the index corresponding to the largest of these terms and let $\sigma_{j_qk_q} = \delta_{j_qk_q}(g_{j_q} - g_{k_q})$ be its value. Then (26) implies

$$\sigma_{j_qk_q} \geq \frac{\sigma(\alpha|W_n)}{p} \geq \frac{\sigma(\alpha|W_n)}{n-1}. \tag{27}$$

Furthermore if we combine the fact that $\sigma_{j_qk_q} > 0$ implies $g_{j_q} - g_{k_q} > 0$ with the definitions of d^q and $\Delta_{j_qk_q}$ we obtain

$$\delta_{j_qk_q} = \min(d_{j_q}^q, -d_{k_q}^q) \leq \min(u_{j_q} - \alpha_{j_q}, \alpha_{k_q}) = \Delta_{j_qk_q}.$$

Finally, combining this result with (27) and (24) gives

$$\frac{\sigma(\alpha|W_n)}{n-1} \le \sigma_{j_q k_q} = \delta_{j_q k_q}(g_{j_q} - g_{k_q}) \le \Delta_{j_q k_q}(g_{j_q} - g_{k_q}) = \sigma(\alpha|\{j_q, k_q\}) \le \sigma(\alpha|W_2^*)$$

which completes the proof of the first assertion.

To prove the second assertion it suffices to give an example of a problem instance and a value $\alpha \in \mathcal{A}$ such that the equality holds. For the primal problem in (1) let $y = (y^+, y^-)$ where $y^+ = (1, ..., 1)$ and $y^- = (-1, ..., -1)$ are vectors whose lengths are not yet specified. Let u be decomposed into corresponding components so that $u = (u^+, u^-)$. For the corresponding canonical dual problem consider the feasible value $\alpha = (0, u^-)$ (which is the initial value of α in Procedure 1). This gives $g = y$. If $u^+ \cdot 1 = u^- \cdot 1$ then it is easy to verify that

$$\begin{aligned}\sigma(\alpha|W_n) = \sup_{\acute{\alpha} \in \mathcal{A}(\alpha, W_n)} g \cdot (\acute{\alpha} - \alpha) &= y \cdot \left(\left(u^+, 0\right) - \left(0, u^-\right)\right) \\ &= \left(y^+, y^-\right) \cdot \left(u^+, -u^-\right) \\ &= 1 \cdot u = 1\ ,\end{aligned}$$

and

$$\max_{W_2 \subseteq W_n} \sigma(\alpha|W_2) = \max_{(j,k)} \left(\min(u_j - \alpha_j, \alpha_k)(g_j - g_k)\right) = 2 \min(u_*^+, u_*^-)$$

where u_*^+ is the largest component value of u^+, and u_*^- is the largest component value of u^-. Thus any problem where $u^+ \cdot 1 = u^- \cdot 1$ and $\min(u_*^+, u_*^-) = \frac{1}{2(n-1)}$ yields the relationship we seek and finishes the proof. For example this condition is satisfied by $u^+ = (1/2)$ and $u^- = \left(\frac{1}{2(n-1)}, ..., \frac{1}{2(n-1)}\right)$.
■

Proof [Proof of Corollary 7:] This proof follows directly from the proof of Theorem 5 since by assumption the stepwise improvement of DECOMP satisfies (22) and the rest of the proof follows without modification. ■

Proof [Proof of Theorem 8:] Let α_p and α_n be solutions to the QP problem at (α, W_p) and (α, W_n) respectively. Since $R(\alpha_p) \le R^*$ and $R(\alpha_n) = R^*$ we obtain

$$R(\alpha_p) - R(\alpha) \le R^* - R(\alpha) = R(\alpha_n) - R(\alpha).$$

Applying (Lemma 14, Equation (19)) with $q = p$ and $\bar{\sigma} = \sigma(\alpha|W_p)$ on the left, and (Lemma 14, Equation (18)) with $q = n$ on the right gives

$$\frac{\sigma(\alpha|W_p)}{2} \min\left(1, \frac{\sigma(\alpha|W_p)}{pLU_p}\right) \le R^* - R(\alpha) \le \sigma(\alpha|W_n).$$

If W_p contains a τ–rate certifying pair then $\sigma(\alpha|W_n) \le \frac{\sigma(\alpha|W_p)}{\tau}$ and the proof is finished. ■

Proof [Proof of Theorem 12:] Use (3) to determine the dual variables $\hat{a}$ and a^0 corresponding to $\hat{\alpha}$ and α^0 respectively. This gives $a^0 = 0$. Let s be the number of nonzero components of $\hat{a}$. Since

$a^0 = 0$ and a *W2* decomposition algorithm can change only two components at each iteration the number of iterations m required to reach $\hat{a}$ satisfies $m \geq s/2$. Furthermore since $\hat{a}_i \leq 1/n$ we obtain $s/n \geq \hat{a} \cdot 1$ and therefore

$$m \;\geq\; \frac{n\hat{a} \cdot 1}{2} \,.$$

Since $\hat{a}$ is an ε–optimal solution

$$-\frac{1}{2}\hat{a} \cdot \mathsf{Q}\hat{a} + \hat{a} \cdot 1 \;\geq\; R^* - \varepsilon$$

and since Q is positive semi–definite this implies $\hat{a} \cdot 1 \;\geq\; R^* - \varepsilon$ and therefore $m \;\geq\; \frac{n(R^*-\varepsilon)}{2}$. ■

Appendix A. Algorithms

A complete algorithm that computes an ε_p–optimal solution to the primal QP problem is provided by the (Procedure 1,Section 2) and Procedures 3–8 in this appendix. Procedure 3 implements a *W2* variant of the Composite–I decomposition algorithm with Stopping Rule 2. Procedure 4 implements Simon's algorithm where the values in (11) are stored in a list of 3–tuples of the form (μ, i, ς) where μ is a value from (11), i is the index of the corresponding component of α, and $\varsigma \in \{+,-\}$ is a symbol indicating the entry type (in particular $\varsigma_l = +$ when $\mu_l = u_{i_l} - \alpha_{i_l}$ and $\varsigma_l = -$ when $\mu_{i_l} = \alpha_{i_l}$). The algorithm scans the ordered list and saves the index pair that maximizes $\sigma(\alpha|\{j,k\})$ as described in Section 2.1. Since this algorithm tracks the indices of the maximum and minimum gradient values it also produces a max–violating pair when it exits the loop. Procedure 5 computes the initial gradient, the initial list M, and an initial upper bound $s^0 = 1$ on the criterion gap $R^* - R(\alpha^0)$. The run time of this procedure is $O(n^2)$ as determined by the gradient computation. Procedure 6 computes the stepwise improvement for the W_{mlp2} and W_{mv} pairs and then updates α according to the pair with the largest improvement. This routine runs in $O(1)$ time. Procedure 7 shows the deletions and insertions required to update the M–list. With the appropriate data structure each of these insert and delete operations can be performed in $O(\log n)$ time. Procedure 8 implements the $O(n \log n)$ algorithm described in Section 2.3.

References

Jose L. Balcazar, Yang Dai, and Osamu Watanabe. Provably fast training algorithms for support vector machines. In *Proceedings of the 1st International Conference on Data Mining ICDM*, pages 43–50, 2001. URL `citeseer.ist.psu.edu/590348.html`.

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

C.C. Chang, C.W. Hsu, and C.J. Lin. The analysis of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 11(4):1003–1008, 2000.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM : a library for support vector machines, 2001.

Procedure 3 The Composite–I Decomposition Algorithm.

```
1:  Decomposition($Q, w, c, u, \varepsilon, \alpha^0$)
2:
3:  $(g^0, M^0, s^0) \leftarrow$ Initialize $(Q, w, u, \alpha^0)$
4:  $m \leftarrow 0$
5:  **repeat**
6:      $(W^m_{mlp2}, W^m_{mv}, \sigma^m) \leftarrow$ Simon$(g^m, M^m)$
7:      **if** $(\sigma^m = 0)$ **then**
8:          Return$(\alpha^m, g^m)$
9:      **end if**
10:     $(\alpha^{m+1}, \delta^m_R, W^m) \leftarrow$ CompositeUpdate$(\alpha^m, g^m, Q, W^m_{mlp2}, W^m_{mv})$
11:     $g^{m+1} \leftarrow g^m - Q(\alpha^{m+1} - \alpha^m)$
12:     $M^{m+1} \leftarrow$ UpdateMlist$(M^m, W^m, \alpha^m, \alpha^{m+1})$
13:     $s^{m+1} \leftarrow \min((n-1)\sigma^m, s^m) - \delta^m_R$
14:     $m \leftarrow m+1$
15: **until** $\left(s^m \leq \varepsilon\right)$
16: Return$(\alpha^m, g^m)$
```

Procedure 4 This routine uses Simon's algorithm to compute a max–lp2 pair W_{mlp2}. It also computes and returns a max–violating pair W_{mv} and the value $\sigma^* = \sigma(\alpha|W_{mlp2})$. It assumes that M is an sorted list arranged in nonincreasing order by the value of first component.

```
1:  Simon($g$, $M$)      $\{ M = \left[(\mu, i, \varsigma)_1, (\mu, i, \varsigma)_2, ..., (\mu, i, \varsigma)_{2n}\right] \}$
2:
3:  $i_{max} \leftarrow 0$,  $i_{min} \leftarrow 0$,  $g_{max} \leftarrow -\infty$,  $g_{min} \leftarrow \infty$,  $\sigma^* \leftarrow 0$,  $W_{mlp2} \leftarrow \emptyset$
4:  $k \leftarrow 1$
5:  **while** $(\mu_k > 0)$ **do**
6:      **if** $((\varsigma_k = +1)$ and $(g_{i_k} > g_{max}))$ **then**
7:          $g_{max} \leftarrow g_{i_k}$,  $i_{max} \leftarrow i_k$
8:          **if** $(\mu_k(g_{max} - g_{min}) > \sigma^*)$ **then**
9:              $W_{mlp2} \leftarrow \{i_{max}, i_{min}\}$,  $\sigma^* \leftarrow \mu_k(g_{max} - g_{min})$
10:         **end if**
11:     **else if** $((\varsigma_k = -1)$ and $(g_{i_k} < g_{min}))$ **then**
12:         $g_{min} \leftarrow g_{i_k}$,  $i_{min} \leftarrow i_k$
13:         **if** $(\mu_k(g_{max} - g_{min}) > \sigma^*)$ **then**
14:             $W_{mlp2} \leftarrow \{i_{max}, i_{min}\}$,  $\sigma^* \leftarrow \mu_k(g_{max} - g_{min})$
15:         **end if**
16:     **end if**
17:     $k \leftarrow k+1$
18: **end while**
19: $W_{mv} \leftarrow \{i_{max}, i_{min}\}$
20: Return($W_{mlp2}$, $W_{mv}$, $\sigma^*$)
```

Procedure 5 This routine accepts a feasible value α and computes the corresponding gradient g, a list M of 3–tuples (μ, i, ς) sorted by μ, and a trivial bound $s = 1$ on $R^* - R(\alpha)$.

```
1: Initialize($Q, w, u, \alpha$)
2:
3: $g \leftarrow -Q\alpha + w$
4: $M \leftarrow \emptyset$
5: for $(i = 1, \ldots, n)$ do
6:    $M \leftarrow$ Insert$(M, (\alpha_i, i, -))$
7:    $M \leftarrow$ Insert$(M, (u_i - \alpha_i, i, +))$
8: end for
9: $s \leftarrow 1$
10: Return($g, M, s$)
```

Procedure 6 This routine computes the stepwise improvements for a max–lp2 pair W_{mlp2} and a max–violating pair W_{mv}, and then updates α using the pair with the largest stepwise improvement. It returns the new value of α, and the corresponding stepwise improvement value and index pair.

```
1: CompositeUpdate($\alpha^{old}$, $g$, $Q$, $W_{mlp2}$, $W_{mv}$)
2:
3: $\{i_1, i_2\} \leftarrow W_{mlp2}$
4: $\delta_g \leftarrow g_{i_1} - g_{i_2}$,   $q \leftarrow Q_{i_1 i_1} + Q_{i_2 i_2} - 2Q_{i_1 i_2}$,   $\Delta_{mlp2} = \min\left(u_{i_1} - \alpha_{i_1}^{old}, \alpha_{i_2}^{old}\right)$
5: if $(\delta_g > q\Delta_{mlp2})$ then
6:    $\delta_{mlp2} \leftarrow \Delta_{mlp2}\left(\delta_g - \frac{q\Delta_{mlp2}}{2}\right)$
7: else
8:    $\delta_{mlp2} \leftarrow \frac{\delta_g^2}{2q}$,   $\Delta_{mlp2} \leftarrow \frac{\delta_g}{q}$
9: end if
10:
11: $\{j_1, j_2\} \leftarrow W_{mv}$
12: $\delta_g \leftarrow g_{j_1} - g_{j_2}$,   $q \leftarrow Q_{j_1 j_1} + Q_{j_2 j_2} - 2Q_{j_1 j_2}$,   $\Delta_{mv} = \min\left(u_{j_1} - \alpha_{j_1}^{old}, \alpha_{j_2}^{old}\right)$
13: if $(\delta_g > q\Delta_{mv})$ then
14:    $\delta_{mv} \leftarrow \Delta_{mv}\left(\delta_g - \frac{q\Delta_{mv}}{2}\right)$
15: else
16:    $\delta_{mv} \leftarrow \frac{\delta_g^2}{2q}$,   $\Delta_{mv} \leftarrow \frac{\delta_g}{q}$
17: end if
18:
19: if $(\delta_{mlp2} > \delta_{mv})$ then
20:    $\alpha_{i_1}^{new} \leftarrow \alpha_{i_1}^{old} + \Delta_{mlp2}$,   $\alpha_{i_2}^{new} \leftarrow \alpha_{i_2}^{old} - \Delta_{mlp2}$
21:    Return($\alpha^{new}$, $\delta_{mlp2}$, $W_{mlp2}$)
22: else
23:    $\alpha_{j_1}^{new} \leftarrow \alpha_{j_1}^{old} + \Delta_{mv}$,   $\alpha_{j_2}^{new} \leftarrow \alpha_{j_2}^{old} - \Delta_{mv}$
24:    Return($\alpha^{new}$, $\delta_{mv}$, $W_{mv}$)
25: end if
```

Procedure 7 This routine updates the sorted list M.

```
UpdateMlist $(M, W, \alpha^{old}, \alpha^{new})$

$\{i_1, i_2\} \leftarrow W$
$M \leftarrow$ Delete$\big(M, (\alpha^{old}_{i_1}, i_1, -)\big)$
$M \leftarrow$ Delete$\big(M, (u_{i_1} - \alpha^{old}_{i_1}, i_1, +)\big)$
$M \leftarrow$ Delete$\big(M, (\alpha^{old}_{i_2}, i_2, -)\big)$
$M \leftarrow$ Delete$\big(M, (u_{i_2} - \alpha^{old}_{i_2}, i_2, +)\big)$
$M \leftarrow$ Insert$\big(M, (\alpha^{new}_{i_1}, i_1, -)\big)$
$M \leftarrow$ Insert$\big(M, (u_{i_1} - \alpha^{new}_{i_1}, i_1, +)\big)$
$M \leftarrow$ Insert$\big(M, (\alpha^{new}_{i_2}, i_2, -)\big)$
$M \leftarrow$ Insert$\big(M, (u_{i_2} - \alpha^{new}_{i_2}, i_2, +)\big)$
Return($M$)
```

Procedure 8 This routine determines the offset parameter according to Theorem 2. Note that the input g is the gradient vector from the canonical dual solution.

```
Offset$(g, y, u)$

$s^+ \leftarrow \sum_{i:y_i=1} u_i, \quad s^- \leftarrow 0$
$\big((\bar{g}_1, \bar{y}_1, \bar{u}_1), ..., (\bar{g}_n, \bar{y}_n, \bar{u}_n)\big) \leftarrow$ SortIncreasing$\big((g_1, y_1, u_1), ..., (g_n, y_n, u_n)\big)$
$L \leftarrow \sum_{i:\bar{y}_i=1} \bar{u}_i(\bar{g}_i - \bar{g}_1)$
$L^* \leftarrow L, \quad b \leftarrow \bar{g}_1$
for $(i = 1, ..., n-1)$ do
    if $(\bar{y}_i = 1)$ then
        $s^+ \leftarrow s^+ - \bar{u}_i$
    else
        $s^- \leftarrow s^- + \bar{u}_i$
    end if
    $L \leftarrow L - (\bar{g}_{i+1} - \bar{g}_i)(s^+ - s^-)$
    if $(L < L^*)$ then
        $L^* \leftarrow L, \quad b \leftarrow \bar{g}_{i+1}$
    end if
end for
Return($b$)
```

P.-H. Chen, R.-E. Fan, and C.-J. Lin. Training support vector machines via SMO–type decomposition methods. In *Proceedings of the 16th International Conference on Algorithmic Learning Theory*, pages 45–62, 2005.

P.-H. Chen, R.-E. Fan, and C.-J. Lin. A study on SMO-type decomposition methods for support vector machines. Technical report, 2006. URL `http://www.csie.ntu.edu.tw/∼cjlin/papers.html`. to appear in IEEE Transactions on Neural Networks.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Canbridge ; United Kingdom, 1st edition, 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.

C.-W. Hsu and C.-J. Lin. A simple decomposition algorithm for support vector machines. *Machine Learning*, 46:291–314, 2002.

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

D. Hush, C. Scovel, and I. Steinwart. Stability of unstable learning algorithms. Technical report, Los Alamos National Laboratory LA-UR-03-4845, 2003. URL `http://wwwc3.lanl.gov/ml/pubs_ml.shtml`. submitted for publication.

D. Hush, C. Scovel, and I. Steinwart. Polynomial time algorithms for computing approximate SVM solutions with guaranteed accuracy. Technical report, Los Alamos National Laboratory LA-UR 05-7738, 2005. URL `http://wwwc3.lanl.gov/ml/pubs_ml.shtml`.

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

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 and C.J. Ong. On the role of the threshold parameter in SVM training algorithms. Control Division Technical Report CD-00-09, Dept. of Mechanical and Production Engineering, National University of Singapore, 2000.

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*, 11:637–649, 2000.

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. 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, 2003.

P. Laskov. Feasible direction decomposition algorithms for training support vector machines. *Machine Learning*, 46(1–3):315–349, 2002.

S.-P. Liao, H.-T. Lin, and C.-J. Lin. A note on the decomposition methods for support vector regression. *Neural Computation*, 14:1267–1281, 2002.

C.-J. Lin. Linear convergence of a decomposition method for support vector machines. Technical Report, 2001a. URL `http://www.csie.ntu.edu.tw/∼cjlin/papers.html`.

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

C.-J. Lin. A formal analysis of stopping criteria of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 13:1045–1052, 2002a.

C.-J. Lin. Asymptotic convergence of an SMO algorithm without any assumptions. *IEEE Transactions on Neural Networks*, 13:248–250, 2002b.

N. List and H.U. Simon. A general convergence theorem for the desomposition method. In J. Shawe-Taylor and Y. Singer, editors, *17th Annual Conference on Learning Theory, COLT 2004, volume 3120 of Lecture Notes in Computer Science*, pages 363–377, 2004.

N. List and H.U. Simon. General polynomial time decomposition algorithms. In P. Auer and R. Meir, editors, *18th Annual Conference on Learning Theory, COLT 2005*, pages 308–322, 2005.

O.L. Mangasarian and D.R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001.

O.L. Mangasarian and D.R. Musicant. Successive overrelaxation for support vector machines. *IEEE Transactions on Neural Networks*, 10:1032–1037, 1999.

E.E. Osuna, R. Freund, and F. Girosi. Support vector machines: training and applications. Technical Report AIM-1602, MIT, 1997.

J.C. 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*, pages 41–64. MIT Press, Cambridge, MA, 1998.

B. Schölkopf, J.C. Platt, J. Shawe-Taylor, and A.J. Smola. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

C. Scovel, D. Hush, and I. Steinwart. Approximate duality. Technical report, Los Alamos National Laboratory LA-UR 05-6766, 2005a. URL `http://wwwc3.lanl.gov/ml/pubs_ml.shtml`. to appear in Journal of Optimization Theory and Applications.

C. Scovel, D. Hush, and I. Steinwart. Learning rates for density level detection. *Analysis and Applications*, 3(4):356–371, 2005b.

H.U. Simon. On the complexity of working set selection. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory*, 2004. URL `http://eprints.pascal-network.org/archive/00000125/`.

I. Steinwart and C. Scovel. Fast rates for support vector machines. In P. Auer and R. Meir, editors, *18th Annual Conference on Learning Theory, COLT 2005*, pages 279–294, 2005.

I. Steinwart and C. Scovel. Fast rates for support vector machines using Gaussian kernels. Technical report, Los Alamos National Laboratory LA-UR 04-8796, 2004. URL `http://www.c3.lanl.gov/∼ingo/publications/pubs.shtml`. submitted to Annals of Statistics (2004).

I. Steinwart, D. Hush, and C. Scovel. A classification framework for anomaly detection. *Journal of Machine Learning Research*, 6:211–232, 2005.

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, Inc., New York, NY, 1998.