Electronic Transactions on Numerical Analysis.
Volume 31, pp. 156-177, 2008.
Copyright © 2008, Kent State University.
ISSN 1068-9613.

ADAPTIVE CONSTRAINT REDUCTION FOR TRAINING SUPPORT VECTOR MACHINES[*]

JIN HYUK JUNG[†], DIANNE P. O'LEARY[‡], AND ANDRÉ L. TITS[§]

Abstract. A support vector machine (SVM) determines whether a given observed pattern lies in a particular class. The decision is based on prior training of the SVM on a set of patterns with known classification, and training is achieved by solving a convex quadratic programming problem. Since there are typically a large number of training patterns, this can be expensive. In this work, we propose an adaptive constraint reduction primal-dual interior-point method for training a linear SVM with ℓ_1 penalty (hinge loss) for misclassification. We reduce the computational effort by assembling the normal equation matrix using only a well-chosen subset of patterns. Starting with a large portion of the patterns, our algorithm excludes more and more unnecessary patterns as the iteration proceeds. We extend our approach to training nonlinear SVMs through Gram matrix approximation methods. We demonstrate the effectiveness of the algorithm on a variety of standard test problems.

Key words. Constraint reduction, column generation, primal-dual interior-point method, support vector machine.

AMS subject classifications. 90C20, 90C51, 90C59, 68W01.

1. Introduction.

Characteristics such as gill placement, coloring, and habitat can predict whether or not a mushroom is edible. Pattern recognition tasks such as this can be automated by use of a *support vector machine* (SVM). Given a *pattern* (set of observed characteristics) $\mathbf{x}$ in some domain set $\mathcal{X}$, the SVM decides whether or not the pattern is in a particular class, e.g., "edible". In the case of a *linear* SVM, the machine makes the decision by testing whether the point in $\mathcal{X}$ specified by the pattern is above or below a hyperplane

$$\{\mathbf{x} : \langle \mathbf{w}, \mathbf{x} \rangle - \gamma = 0\}.$$

Here $\langle \cdot, \cdot \rangle$ denotes an inner product. Before the SVM can be put to use, a *training process* determines the parameters $\mathbf{w}$ and γ, based on a set of *training patterns* $\mathbf{x}_i$ each having a predetermined classification label $y_i = \pm 1$, $i = 1, ..., m$, e.g., "edible" or "not edible". The goal is to set the parameters so that

$$\mathrm{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle - \gamma) = y_i, \quad \text{for } i = 1, ..., m.$$

Thus, the machine is trained to correctly identify the patterns with known classifications and is then used for classifying future patterns. If no hyperplane separates the two classes, then a loss function is included in the training to add a penalty for misclassification. In either case, this training process can be formulated as a convex quadratic program (CQP).

Often, the number of training patterns m is very much larger than the dimension of $\mathbf{x}$ (and $\mathbf{w}$), and it is well known that $\mathbf{w}$ and γ are determined by a small subset of the training patterns. In this paper, we develop an efficient primal-dual interior-point method (PDIPM) for solving this CQP. The novelty of our algorithm is the way that we iteratively identify the small subset of critical training patterns and ignore the rest.

[*] Received November 29, 2007. Accepted September 26, 2008. Published online on February 23, 2009. Recommended by Martin H. Gutknecht. This work was supported by the US Department of Energy under Grant DEFG0204ER25655.

[†] Department of Computer Science, University of Maryland, College Park, MD 20742 (jjung@cs.umd.edu).

[‡] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (oleary@cs.umd.edu).

[§] Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland, College Park, MD 20742 (andre@umd.edu).

Other authors have also exploited the fact that the number of critical patterns is small. Osuna et al. [25] proposed a decomposition algorithm for the dual SVM formulation, solving a sequence of reduced problems, and Joachims [21] proposed some improvements. Platt [26] proposed a sequential minimal optimization (SMO) algorithm that allows only two variables to change at a time; see the four essays in [20] for further discussion of the literature. Ferris and Munson [14] considered training linear SVMs with ℓ_1 and ℓ_2 hinge loss functions. They efficiently applied the Sherman-Morrison-Woodbury (SMW) formula to solving the normal equations for training the SVM, the most expensive operation in the PDIPM. Gertz and Griffin [16] proposed using either a parallel direct solver or a preconditioned conjugate gradient solver tailored for the PDIPM normal equations in training an SVM with ℓ_1 hinge loss.

In this work the focus is again on the normal equations for the ℓ_1 hinge loss formulation. Like Osuna et al., we reduce computational cost by omitting unnecessary constraints or patterns in assembling the matrix for the normal equations. However, in contrast to the decomposition based algorithms, we solve only *one* optimization problem, using constraint selection only to determine the search direction at each iteration of a PDIPM. Our algorithm is closely related to a PDIPM proposed for solving a general CQP with many inequality constraints [22].

Reducing the computational cost of linear and convex programming by using only a small number of the constraints has been actively studied. Ye [38] pioneered a “build-down” scheme for linear programming (LP), proposing a rule that can safely eliminate inequality constraints that will not be active at the optimum. Dantzig and Ye [9] proposed a “build-up” IPM of dual affine-scaling form. A potential reduction algorithm proposed by Ye [39] allows column generation for linear feasibility problems, and Luo and Sun [23] proposed a similar scheme for convex quadratic feasibility problems, to which CQPs can be transformed. Den Hertog et al. proposed “build-up” and “build-down” IPM variants [11, 12]. They also proposed a path-following cutting plane algorithm for convex programming, where they used the “build-up and -down” IPM for LP to solve a sequence of LP relaxations of the convex programming [10].

In our algorithm the constraints at each step are chosen based only on the current iterate, rather than by building up or down. Related algorithms for LP were considered in [32] and [34].

To present our algorithm, we begin in Section 2 by formulating the SVM training problem as a CQP. In Section 3, we describe our adaptive constraint reduction approach for a quadratic programming version of Mehrotra’s predictor-corrector (MPC) algorithm (see [24], [17], and [16]). We extend our results to certain nonlinear SVMs in Section 4. In Section 5, numerical results are presented. Finally, concluding remarks are provided in Section 6.

Throughout this paper we denote matrices by upper-case bold letters and column vectors by lower-case bold letters. The entries of a vector $\mathbf{x}$ are x_i, while those for a matrix $\mathbf{K}$ are k_{ij}. The i^{th} row of the matrix $\mathbf{X}$ is denoted by $\mathbf{x}_i^T$. Given a vector $\mathbf{y}$, the matrix $\mathbf{Y}$ is the diagonal matrix formed from the entries in the vector: $\mathbf{Y} = \mathrm{diag}(\mathbf{y})$. The cardinality of a set S is denoted by $|S|$.

2. Support vector machines.

An SVM is a binary classifier, answering ‘yes’ or ‘no’ to an input pattern, based on whether or not the pattern lies in a particular half space. In this section we review the formulation of SVMs and their training.

2.1. Data representation, classifier, and feature space.

The training data patterns are defined as

$$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m) \in \mathcal{X} \times \{-1, +1\}, \tag{2.1}$$

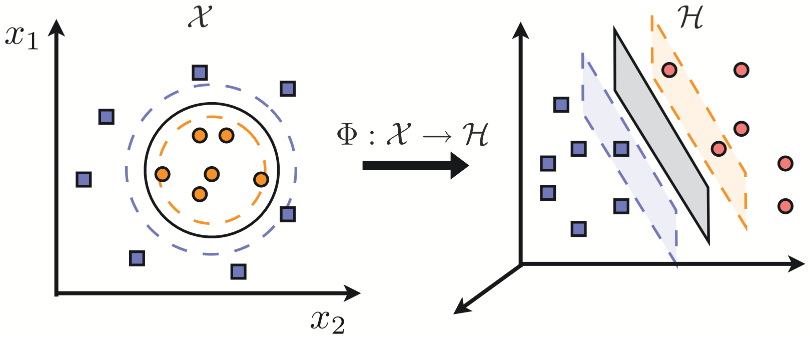


FIGURE 2.1: *By mapping the patterns in the pattern space (x_1, x_2) to a higher dimensional feature space $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$, the SVM constructs an ellipsoidal classifier in the original pattern space by finding a linear classifier in the feature space.*

where m is the number of training patterns.

A *linear classifier* is a hyperplane $\{\mathbf{x} : \langle \mathbf{w}, \mathbf{x} \rangle = \gamma\}$ in $\mathcal{X}$ that separates the "$-$" patterns ($y_i = -1$) from the "$+$" patterns ($y_i = +1$). For a pattern $\mathbf{x} \in \mathcal{X}$, the decision or prediction y of the classifier is

$$y = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle - \gamma). \tag{2.2}$$

To find a good classifier, it may be necessary to use a *feature map*

$$\Phi : \mathcal{X} \to \mathcal{H} \tag{2.3}$$
$$\mathbf{x} \mapsto \Phi(\mathbf{x}), \tag{2.4}$$

to map the training patterns into a *feature space* $\mathcal{H}$ (possibly of higher dimension) endowed with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. We define the length or norm of a vector $\mathbf{a} \in \mathcal{H}$ to be

$$\|\mathbf{a}\|_{\mathcal{H}} = \sqrt{\langle \mathbf{a}, \mathbf{a} \rangle_{\mathcal{H}}}.$$

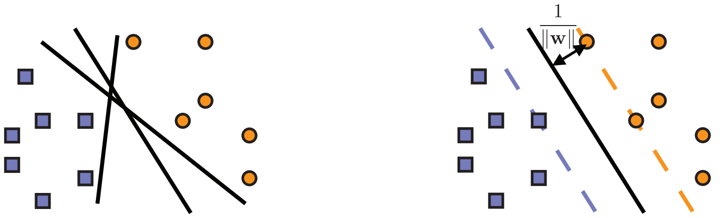
A *linear* classifier determined in the feature space may induce a *nonlinear* classifier in the original pattern space. For example, define a feature map from $\mathbb{R}^2$ to $\mathbb{R}^3$ as

$$\begin{aligned} \Phi : \mathbb{R}^2 &\to \mathbb{R}^3, \\ (x_1, x_2)^T &\mapsto (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T. \end{aligned} \tag{2.5}$$

Then the inner-product in the feature space $\mathbb{R}^3$ can be used to define the separator in $\mathbb{R}^2$ illustrated in Figure 2.1. Accordingly, we first limit our study to linear SVMs in $\mathbb{R}^n$, and then consider nonlinear SVMs in Sections 4 and 5.2.

2.2. Maximizing the separation margin. We now focus on finding a linear classifier, where $\mathbf{x}_i \in \mathbb{R}^n$ for $i = 1, ..., m$. We use the standard inner product $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \mathbf{x}_1^T \mathbf{x}_2$.

If the training patterns are strictly separable, then there are infinitely many hyperplanes that can correctly classify the patterns, as illustrated in Figure 2.2. To choose a desirable separating hyperplane, we seek one that maximizes the *separation margin*, defined to be the minimal distance from the hyperplane to the "$+$" patterns ($y_i = 1$) plus the minimal distance to the "$-$" patterns ($y_i = -1$).



(A) *Available planes.* (B) *The plane with maximal margin.*

FIGURE 2.2: *The SVM is trained to find a hyperplane with maximal separation margin. The hyperplane can classify data according to the predetermined labels. Circles and squares denote positive and negative patterns, respectively.*

How can we find this hyperplane? If the patterns are separable, then there exists at least one "$+$" pattern and one "$-$" pattern closest to any separating hyperplane. Define the *"$+$" and "$-$" class boundary planes* to be the two hyperplanes parallel to the separating hyperplane that contain these two patterns. Then the distance between these class boundary planes is the separation margin, and we define $\mathbf{w}$ and γ so that $\{\mathbf{x} : \langle \mathbf{w}, \mathbf{x}\rangle = \gamma\}$ is the plane halfway between them. Since the patterns are separable, there is no pattern in between the boundary planes, and we can scale $\mathbf{w}$ and γ so that, for all $i \in \{1, ..., m\}$,

$$\langle \mathbf{w}, \mathbf{x}_i\rangle - \gamma \geq y_i, \qquad \text{if } y_i = +1, \tag{2.6}$$
$$\langle \mathbf{w}, \mathbf{x}_i\rangle - \gamma \leq y_i, \qquad \text{if } y_i = -1, \tag{2.7}$$

or equivalently

$$y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle - \gamma) \geq 1.$$

So, the boundaries of the half spaces defined by (2.6) and (2.7) are the "$+$" and "$-$" class boundary planes.

Since the distance between the boundary planes is $\frac{2}{\|\mathbf{w}\|}$, where $\|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w}\rangle$, the problem can now be modeled as an optimization problem, called the *hard-margin SVM*:

$$\min_{\mathbf{w},\gamma} \frac{1}{2}\|\mathbf{w}\|_2^2 \tag{2.8}$$
$$\text{s.t.} \quad \mathbf{Y}(\mathbf{X}\mathbf{w} - \mathbf{e}\gamma) \geq \mathbf{e}, \tag{2.9}$$

where $\mathbf{X} = [\mathbf{x}_1, ...\mathbf{x}_m]^T \in \mathbb{R}^{m\times n}$ and $\mathbf{e} = [1, ..., 1]^T$. Notice that this problem has one constraint per pattern. Typically $m \gg n$, and that is the case we consider here.

If the data are not separable, there is no solution to the hard-margin optimization problem. To cope with this situation, we add a *misclassification penalty* in the objective function (2.8). We introduce a nonnegative relaxation variable $\boldsymbol{\xi}$ in order to measure misclassification, obtaining relaxed constraints

$$y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle - \gamma) \geq 1 - \xi_i. \tag{2.10}$$

Adding an ℓ_1 misclassification penalty to the objective function (2.8), we get the (primal) *soft-margin SVM* (with ℓ_1 hinge loss) proposed by Cortes and Vapnik[1] [8]:

$$\min_{\mathbf{w},\gamma,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \boldsymbol{\tau}^T\boldsymbol{\xi} \tag{2.11}$$

$$\text{s.t.} \quad \mathbf{Y}(\mathbf{X}\mathbf{w} - \mathbf{e}\gamma) + \boldsymbol{\xi} \geq \mathbf{e}, \tag{2.12}$$

$$\boldsymbol{\xi} \geq \mathbf{0}, \tag{2.13}$$

where $\boldsymbol{\tau}$ is an m-dimensional vector of positive penalty parameters for the trade-off between the separation margin maximization and the error minimization. This soft-margin formulation is often preferred to the hard-margin formulation even when the training patterns are strictly classifiable [4]. Notice this formulation is a CQP with m nontrivial constraints (2.12), m bound constraints (2.13), m relaxation variables $\boldsymbol{\xi}$, and n variables $\mathbf{w}$, where $m \gg n$.

2.3. Dual formulation, Gram matrix, and support vectors. The dual of the CQP given by formulae (2.11)-(2.13) is

$$\max_{\boldsymbol{\alpha}} \; -\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{YKY}\boldsymbol{\alpha} + \mathbf{e}^T\boldsymbol{\alpha} \tag{2.14}$$

$$\text{s.t.} \quad \mathbf{y}^T\boldsymbol{\alpha} = 0, \tag{2.15}$$

$$\mathbf{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{\tau}, \tag{2.16}$$

where the symmetric positive semidefinite Gram matrix $\mathbf{K} \in \mathbb{R}^{m\times m}$ has entries

$$k_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle,$$

(i.e., $\mathbf{K} = \mathbf{X}\mathbf{X}^T$) and where α_i is the dual variable associated with the i^{th} constraint in (2.12).

If $\boldsymbol{\alpha}^*$ solves this dual problem, then we can compute the solution to the primal problem (2.11)-(2.13) from it:

$$\mathbf{w}^* = \mathbf{X}^T\mathbf{Y}\boldsymbol{\alpha}^* = \sum_{i\in S}\alpha_i^* y_i \mathbf{x}_i, \qquad \text{for } S = \{i : 0 < \alpha_i^*\}, \tag{2.17}$$

$$\gamma^* = \frac{1}{|S_{\text{on}}|}\sum_{i\in S}\left(\langle \mathbf{w}^*, \mathbf{x}_i\rangle - y_i\right), \qquad \text{for } S_{\text{on}} = \{i : 0 < \alpha_i^* < \tau_i\}, \tag{2.18}$$

$$\xi_i^* = \max\left\{1 - y_i(\langle \mathbf{w}^*, \mathbf{x}_i\rangle - \gamma^*), 0\right\}, \qquad \text{for } i = 1, ..., m. \tag{2.19}$$

Note that (2.18) is obtained from (2.12), noticing that

$$y_i(\langle \mathbf{w}_i^*, \mathbf{x}_i\rangle - \gamma^*) = 1 \quad \text{for all } i \text{ such that } 0 < \alpha_i^* < \tau_i.$$

The subscript "on" will be explained in the next paragraph. While $\gamma^* = (\langle \mathbf{w}^*, \mathbf{x}_i\rangle - y_i)$ for every $i \in S_{\text{on}}$, the averaging in (2.18) provides somewhat better accuracy than using a single equation to determine γ^*. Equation (2.19) is obtained from (2.12) and (2.13). In view of (2.17), the Lagrange multiplier α_i can be interpreted as the weight of the i^{th} pattern in defining the classifier.

Support vectors (SVs) are the patterns that contribute to defining the classifier, i.e., those associated with positive weight α_i^*. The *on-boundary* support vectors have weight strictly between the lower bound 0 and the upper bound τ_i, and, geometrically, lie on their class boundary plane, i.e., both (2.12) and (2.13) are active. The *off-boundary* support vectors have

[1] They use a scalar τ.

Pattern Type	α_i^*	s_i^*	ξ_i^*
Off-boundary support vector	τ_i	0	$(0,\infty)$
On-boundary support vector	$(0,\tau_i)$	0	0
Nonsupport vector	0	$(0,\infty)$	0

TABLE 2.1: *Classification of support vectors and nonsupport vectors. Here s_i^* is the optimal slack variable, defined as $s_i^* := y_i^*(\langle \mathbf{w}^*, \mathbf{x}_i\rangle - \gamma^*) + \xi_i^* - 1$, associated with the i^{th} constraint in (2.12).*

the maximal allowable weight $\alpha_i^* = \tau_i$ and lie on the wrong side of the class boundary plane, i.e., (2.12) is active but (2.13) is inactive [28].[2] We summarize this classification in Table 2.1.

We now have formulated our optimization problem and turn our attention to solving it.

3. Adaptive constraint (pattern) reduction.

In this section we present a standard primal-dual interior-point method for training our SVM, and then improve the efficiency of the method by adaptively ignoring some patterns. Since each pattern corresponds to a primal constraint, this is called constraint reduction.

3.1. Primal-dual interior-point method.

Since the soft-margin formulation for the SVM (2.11)-(2.13) is a CQP, every solution to the associated Karush-Kuhn-Tucker (KKT) conditions is a global optimum. Therefore, training the machine is equivalent to finding a solution to the KKT conditions for the primal (2.11)-(2.13) and the dual (2.14)-(2.16) problems [16]:

$$\mathbf{w} - \mathbf{X}^T\mathbf{Y}\boldsymbol{\alpha} = \mathbf{0}, \tag{3.1}$$

$$\mathbf{y}^T\boldsymbol{\alpha} = 0, \tag{3.2}$$

$$\boldsymbol{\tau} - \boldsymbol{\alpha} - \mathbf{u} = \mathbf{0}, \tag{3.3}$$

$$\mathbf{Y}\mathbf{X}\mathbf{w} - \gamma\mathbf{y} + \boldsymbol{\xi} - \mathbf{e} - \mathbf{s} = \mathbf{0}, \tag{3.4}$$

$$\mathbf{S}\boldsymbol{\alpha} = \mathbf{0}, \tag{3.5}$$

$$\mathbf{U}\boldsymbol{\xi} = \mathbf{0}, \tag{3.6}$$

$$\mathbf{s}, \mathbf{u}, \boldsymbol{\alpha}, \boldsymbol{\xi} \geq \mathbf{0},$$

where $\mathbf{s}$ is a slack variable vector for the inequality constraints (2.12), and $\mathbf{u}$ is a slack for the upper bound constraints (2.16) and a vector of multipliers for the non-negativity constraints (2.13). Conditions (3.1)-(3.3) relate the gradient of the objective function to the constraints that are active at an optimal solution, while (3.4) is the primal feasibility condition. Conditions (3.5) and (3.6) enforce complementary slackness.

In order to find a solution, we use a PDIPM. We apply a Newton-like method to solving the KKT conditions, with perturbations added to the complementarity conditions (3.5) and (3.6). For the variant of the MPC algorithm discussed in [16] (analogous to that in [36]), the search direction is obtained by solving the system of equations

$$\Delta\mathbf{w} - \mathbf{X}^T\mathbf{Y}\Delta\boldsymbol{\alpha} = -(\mathbf{w} - \mathbf{X}^T\mathbf{Y}\boldsymbol{\alpha}) \equiv -\mathbf{r}_{\mathbf{w}},$$
$$\mathbf{y}^T\Delta\boldsymbol{\alpha} = -\mathbf{y}^T\boldsymbol{\alpha} \equiv -r_{\boldsymbol{\alpha}},$$

[2] In many articles discussing the decomposition method, the terms "in-bound" and "bound support vectors" are used to denote on-boundary and off-boundary support vectors, respectively. The former terms are based on the bound constraints (2.16), whereas the latter terms are based on geometry.

$$\begin{aligned}
-\Delta\boldsymbol{\alpha} - \Delta\mathbf{u} &= -(\boldsymbol{\tau} - \boldsymbol{\alpha} - \mathbf{u}) \equiv -\mathbf{r}_{\mathbf{u}},\\
\mathbf{YX}\Delta\mathbf{w} - \mathbf{y}\Delta\gamma + \Delta\boldsymbol{\xi} - \Delta\mathbf{s} &= -(\mathbf{YXw} - \gamma\mathbf{y} + \boldsymbol{\xi} - \mathbf{e} - \mathbf{s}) \equiv -\mathbf{r}_{\mathbf{s}},\\
\mathbf{S}\Delta\boldsymbol{\alpha} + \operatorname{diag}(\boldsymbol{\alpha})\Delta\mathbf{s} &= -\mathbf{r}_{\mathbf{sv}},\\
\operatorname{diag}(\boldsymbol{\xi})\Delta\mathbf{u} + \mathbf{U}\Delta\boldsymbol{\xi} &= -\mathbf{r}_{\boldsymbol{\xi}\mathbf{u}}.
\end{aligned}$$

First, an affine scaling (predictor) direction $(\Delta\mathbf{w}^{\text{aff}}, \Delta\gamma^{\text{aff}}, \Delta\boldsymbol{\xi}^{\text{aff}}, \Delta\mathbf{s}^{\text{aff}}, \Delta\boldsymbol{\alpha}^{\text{aff}}, \Delta\mathbf{u}^{\text{aff}})$ is computed by setting

$$\mathbf{r}_{\mathbf{sv}} = \mathbf{S}\boldsymbol{\alpha}, \tag{3.7}$$

$$\mathbf{r}_{\boldsymbol{\xi}\mathbf{u}} = \mathbf{U}\boldsymbol{\xi}. \tag{3.8}$$

Then, the combined affine-scaling and corrector step is obtained by setting

$$\mathbf{r}_{\mathbf{sv}} = \mathbf{S}\boldsymbol{\alpha} - \sigma\mu\mathbf{e} + \Delta\mathbf{S}^{\text{aff}}\Delta\boldsymbol{\alpha}^{\text{aff}}, \tag{3.9}$$

$$\mathbf{r}_{\boldsymbol{\xi}\mathbf{u}} = \operatorname{diag}(\boldsymbol{\xi})\mathbf{u} - \sigma\mu\mathbf{e} + \Delta\mathbf{U}^{\text{aff}}\Delta\boldsymbol{\xi}^{\text{aff}}, \tag{3.10}$$

where the *complementarity measure*[3] is defined by

$$\mu = \frac{\mathbf{s}^T\boldsymbol{\alpha} + \boldsymbol{\xi}^T\mathbf{u}}{2m},$$

$\sigma > 0$ is a *centering parameter* defined later, and $\Delta\mathbf{S}^{\text{aff}}$ and $\Delta\mathbf{U}^{\text{aff}}$ are diagonal matrices formed from $\Delta\mathbf{s}^{\text{aff}}$ and $\Delta\mathbf{u}^{\text{aff}}$ respectively. These equations can be reduced to the *normal equations*

$$\mathbf{M}\Delta\mathbf{w} \equiv \left(\mathbf{I} + \mathbf{X}^T\mathbf{Y}\boldsymbol{\Omega}^{-1}\mathbf{YX} - \frac{\bar{\mathbf{y}}\bar{\mathbf{y}}^T}{\mathbf{y}^T\boldsymbol{\Omega}^{-1}\mathbf{y}}\right)\Delta\mathbf{w} = -\bar{\mathbf{r}}_{\mathbf{w}} - \frac{1}{\mathbf{y}^T\boldsymbol{\Omega}^{-1}\mathbf{y}}\bar{r}_{\boldsymbol{\alpha}}\bar{\mathbf{y}}, \tag{3.11}$$

where

$$\begin{aligned}
\boldsymbol{\Omega} &= \operatorname{diag}(\boldsymbol{\alpha})^{-1}\mathbf{S} + \mathbf{U}^{-1}\operatorname{diag}(\boldsymbol{\xi}),\\
\bar{\mathbf{y}} &= \mathbf{X}^T\mathbf{Y}\boldsymbol{\Omega}^{-1}\mathbf{y} = \mathbf{X}^T\boldsymbol{\Omega}^{-1}\mathbf{e},
\end{aligned}$$

$\bar{\mathbf{r}}_{\mathbf{w}} = \mathbf{r}_{\mathbf{w}} + \mathbf{X}^T\mathbf{Y}\boldsymbol{\Omega}^{-1}\mathbf{r}_{\boldsymbol{\Omega}}$, and $\bar{r}_{\boldsymbol{\alpha}} = r_{\boldsymbol{\alpha}} - \mathbf{y}^T\boldsymbol{\Omega}^{-1}\mathbf{r}_{\boldsymbol{\Omega}}$. Once (3.11) has been solved for $\Delta\mathbf{w}$, the increments $\Delta\gamma$, $\Delta\boldsymbol{\alpha}$, $\Delta\boldsymbol{\xi}$, $\Delta\mathbf{u}$, and $\Delta\mathbf{s}$ are obtained by solving

$$\Delta\gamma = \frac{1}{\mathbf{y}^T\boldsymbol{\Omega}^{-1}\mathbf{y}}\left(-\bar{r}_{\boldsymbol{\alpha}} + \bar{\mathbf{y}}^T\Delta\mathbf{w}\right), \tag{3.12}$$

$$\Delta\boldsymbol{\alpha} = -\boldsymbol{\Omega}^{-1}(\mathbf{r}_{\boldsymbol{\Omega}} + \mathbf{YX}\Delta\mathbf{w} - \mathbf{y}\Delta\gamma), \tag{3.13}$$

$$\Delta\boldsymbol{\xi} = -\mathbf{U}^{-1}\operatorname{diag}(\boldsymbol{\xi})(\bar{\mathbf{r}}_{\mathbf{u}} - \Delta\boldsymbol{\alpha}), \tag{3.14}$$

$$\Delta\mathbf{u} = -\operatorname{diag}(\boldsymbol{\xi})^{-1}(\mathbf{r}_{\boldsymbol{\xi}\mathbf{u}} + \mathbf{U}\Delta\boldsymbol{\xi}), \tag{3.15}$$

$$\Delta\mathbf{s} = -\operatorname{diag}(\boldsymbol{\alpha})^{-1}(\mathbf{r}_{\mathbf{sv}} + \mathbf{S}\Delta\boldsymbol{\alpha}), \tag{3.16}$$

where $\bar{\mathbf{r}}_{\mathbf{u}} = \mathbf{r}_{\mathbf{u}} + \operatorname{diag}(\boldsymbol{\xi})^{-1}\mathbf{r}_{\boldsymbol{\xi}\mathbf{u}}$ and $\mathbf{r}_{\boldsymbol{\Omega}} = \mathbf{r}_{\mathbf{s}} + \operatorname{diag}(\boldsymbol{\alpha})^{-1}\mathbf{r}_{\mathbf{sv}} - \mathbf{U}^{-1}\operatorname{diag}(\boldsymbol{\xi})\bar{\mathbf{r}}_{\mathbf{u}}$; see [16] for detailed derivation.

Forming and solving the normal equations (3.11) is the most time consuming task in a step of the predictor-corrector algorithm, so we now focus on how to speed up this process. Fine and Scheinberg [15] and Ferris and Munson [14] use the dual formulation (2.14)-(2.16),

[3] It is sometimes called the *duality measure*.

so their normal equations involve an $m \times m$ matrix, considerably larger than our $n \times n$ matrix $\mathbf{M}$. They use the Sherman-Morrison-Woodbury (SMW) formula to reduce computational complexity from $O(m^3)$ to $O(mn^2)$. In contrast, Gertz and Griffin [16] solve the normal equations (3.11) iteratively, using a matrix we call $\mathbf{M}_{(Q)}$ as a preconditioner. The approach of Woodsend and Gondzio [35], with ℓ_1 hinge loss, results in the same KKT system and search direction equations as those of Gertz and Griffin. However, they apply a different sequence of block eliminations, resulting in different normal equations. Chapelle discusses relations between the primal and dual based approaches [6]. He shows that when using the SMW formula, finding the search direction has the same computational complexity for the primal as for the dual, but he argues that the primal approach is superior, because it directly attempts to maximize the separation margin. We choose to speed up the computation by forming an approximation to the matrix $\mathbf{M}$.

3.2. Constraint reduction. In [22] we developed and analyzed an algorithm for solving CQPs by replacing the matrix $\mathbf{M}$ in the normal equations (3.11) by an approximation to it. In this section we see how this idea can be applied to the SVM problem.

Since $y_i = \pm 1$ and $\mathbf{\Omega}$ is diagonal, we see that

$$\mathbf{Y}\mathbf{\Omega}^{-1}\mathbf{Y} = \mathbf{\Omega}^{-1} \quad \text{and} \quad \mathbf{y}^T\mathbf{\Omega}^{-1}\mathbf{y} = \mathbf{e}^T\mathbf{\Omega}^{-1}\mathbf{e}.$$

Now, consider the matrix of the normal equations (3.11),

$$\begin{aligned}\mathbf{M} &= \mathbf{I} + \mathbf{X}^T\mathbf{Y}\mathbf{\Omega}^{-1}\mathbf{Y}\mathbf{X} - \frac{\bar{\mathbf{y}}\bar{\mathbf{y}}^T}{\mathbf{y}^T\mathbf{\Omega}^{-1}\mathbf{y}} \\ &= \mathbf{I} + \sum_{i=1}^{m} \omega_i^{-1}\mathbf{x}_i\mathbf{x}_i^T - \frac{(\sum_{i=1}^m \omega_i^{-1}\mathbf{x}_i)(\sum_{i=1}^m \omega_i^{-1}\mathbf{x}_i)^T}{\sum_{i=1}^m \omega_i^{-1}},\end{aligned}$$

and the matrix

$$\mathbf{M}_{(Q)} \equiv \mathbf{I} + \sum_{i\in Q} \omega_i^{-1}\mathbf{x}_i\mathbf{x}_i^T - \frac{(\sum_{i\in Q} \omega_i^{-1}\mathbf{x}_i)(\sum_{i\in Q} \omega_i^{-1}\mathbf{x}_i)^T}{\sum_{i\in Q} \omega_i^{-1}}, \tag{3.17}$$

where $Q \subseteq \{1, \ldots, m\}$ and

$$\omega_i^{-1} = \frac{\alpha_i u_i}{s_i\alpha_i + \xi_i u_i}.$$

If $Q = \{1, \ldots, m\}$, then $\mathbf{M}_{(Q)} = \mathbf{M}$. If $Q \subset \{1, \ldots, m\}$, then $\mathbf{M}_{(Q)}$ is an approximation, accurate if the neglected terms are small relative to those included. The approximation reduces the computational cost for the matrix assembly, which is the most expensive task in a PDIPM, from $O(mn^2)$ to $O(|Q|n^2)$.

How do we obtain a good approximation? Strict complementarity typically holds between the variables $\mathbf{s}$ and $\boldsymbol{\alpha}$ and between $\mathbf{u}$ and $\boldsymbol{\xi}$, so we make use of this assumption. The last term in $\mathbf{M}_{(Q)}$ is a rank-one matrix, so the bulk of the work is in handling the second term. Patterns associated with larger ω_i^{-1} make a larger contribution to this term. Since $\alpha_i + u_i = \tau_i$, the quantity ω_i^{-1} becomes very large exactly when both s_i and ξ_i are close to zero. Therefore, as seen in Table 2.1, the important terms in the first summation in (3.17) are associated with the on-boundary support vectors. Identifying on-boundary support vectors is not possible until we find the maximal margin classifier and its class boundaries. At each iteration, however, we have an approximation to the optimal value of ω_i, so we find prospective on-boundary support vectors by choosing the patterns with small ω_i. As described in [16], a growing ω_i^{-1}

diverges to infinity at an $O(\mu^{-1})$ rate and a vanishing ω_i^{-1} converges to zero at an $O(\mu)$ rate. Therefore, as the intermediate classifier approaches the maximal margin classifier, it becomes clearer which patterns are more likely to be on-boundary support vectors. This enables us to adaptively reduce the index set size used in (3.17). To measure how close the intermediate classifier is to the optimal one, we can use the complementarity measure μ, which converges to zero. At each iteration, we set the size q of our index set Q to be a value between two numbers q_L and q_U:

$$q = \max\{q_L, \min\{\lceil \rho m \rceil, q_U\}\}, \tag{3.18}$$

where the heuristic choice

$$\rho = \mu^{\frac{1}{\beta}}$$

tends to synchronize decrease of $|Q|$ with convergence of the optimality measure. Here $\beta > 0$ is a parameter for controlling the rate of decrease, and q_U is also an algorithm parameter, but q_L changes from iteration to iteration. At the first iteration we randomly choose q_U indices, because we have no information about the classifier. Fast clustering algorithms may improve the initial selection [3].

We now show how to choose patterns and determine q_L at each iteration. Based on our examination of ω_i^{-1}, there are several reasonable choices. Define $\mathcal{Q}(\mathbf{z}, q)$, the set of all subsets of $M = \{1, \ldots, m\}$ that contain the indices of q smallest components of $\mathbf{z} \in \mathbb{R}^m$:

$$\mathcal{Q}(\mathbf{z}, q) = \{Q \,|\, Q \subseteq M,\ |Q| = q \text{ and } z_i \le z_j\ \forall i \in Q, j \notin Q\}. \tag{3.19}$$

If there are no ties for the q^{th} smallest component, then $\mathcal{Q}(\mathbf{z}, q)$ contains a single index set. It is also possible to include additional indices in these sets for heuristic purposes, as allowed by [32]. Then, we have the following choices of patterns:

- **Rule 1:** $\mathcal{Q}(\mathbf{YXw} - \gamma\mathbf{y} + \boldsymbol{\xi} - \mathbf{e}, q)$. This rule selects the patterns $\mathbf{x}_i$ corresponding to indices in these sets with the smallest "one-sided" distance to its class boundary plane, meaning that we only consider patterns that are on the wrong side of their class boundary plane. Assuming primal feasibility, this measures the slacks of the primal constraints (2.12). This choice is most intuitive because support vectors contribute to defining the classifier, which is the underlying idea for most decomposition-based algorithms. Inspired by the rate of convergence or divergence of ω_i^{-1}, as noted above, we define the lower bound q_L on the index set size by

$$q_L = \left| \left\{ i : \frac{\alpha_i}{s_i} \ge \theta\sqrt{\mu} \ \text{ or } \ s_i \le \sqrt{\mu} \right\} \right|,$$

  where θ is a prescribed parameter, so that we include terms with small values of s_i.
- **Rule 2:** $\mathcal{Q}(\boldsymbol{\Omega}\mathbf{e}, q)$. The patterns $\mathbf{x}_i$ chosen by these sets have the smallest ω_i, and thus the largest contributions to the second term in the definition of the matrix $\mathbf{M}$. Again, considering the rate of convergence or divergence of ω_i^{-1}, we define the lower bound q_L on the index set size by counting the number of large ω_i^{-1}:

$$q_L = \left| \{ i : \omega_i^{-1} \ge \theta\sqrt{\mu} \} \right|, \tag{3.20}$$

  where θ is a prescribed parameter. Under our strict complementarity assumption, the value of q_L will eventually converge to the number of diverging ω_i^{-1}, or, equivalently, the number of on-boundary support vectors.

Either of these choices, however, may have an imbalance between the number of patterns selected from the "$+$" and "$-$" classes. In the worst case, we might select patterns from only one of the classes, whereas on-boundary support vectors are typically found in both classes. To avoid this unfavorable situation, we might want to use a *balanced* choice of patterns, specifying the number q^+ and q^- chosen from each class as

$$q^+ = \max\left(q_L^+, \min\left(\left\lceil \frac{\min\left(\lceil \rho m \rceil, q_U\right)}{2} \right\rceil, m^+\right)\right), \tag{3.21}$$

$$q^- = \max\left(q_L^-, \min\left(\left\lceil \frac{\min\left(\lceil \rho m \rceil, q_U\right)}{2} \right\rceil, m^-\right)\right), \tag{3.22}$$

where m^+ and m^- are the number of "$+$" and "$-$" patterns, respectively. The values q_L^+ and q_L^- are lower bounds defined for each class; for example, instead of (3.20) we use

$$\begin{aligned} q_L^+ &= |\{i : \omega_i^{-1} \geq \theta\sqrt{\mu} \text{ and } y_i = 1\}|, \\ q_L^- &= |\{i : \omega_i^{-1} \geq \theta\sqrt{\mu} \text{ and } y_i = -1\}|. \end{aligned}$$

We adjust either q^+ or q^- so that $q^+ + q^- = q$, the desired number of patterns. Given q^+ and q^-, we define the sets

$$\begin{aligned} \mathcal{Q}^+(\mathbf{z}, q^+) &= \{Q \,|\, Q \subseteq M,\ |Q| = q^+ \text{ and } z_i \leq z_j\ \forall i \in Q, j \notin Q \text{ and } d_i = d_j = +1\}, \\ \mathcal{Q}^-(\mathbf{z}, q^-) &= \{Q \,|\, Q \subseteq M,\ |Q| = q^- \text{ and } z_i \leq z_j\ \forall i \in Q, j \notin Q \text{ and } d_i = d_j = -1\}. \end{aligned}$$

The set Q is the union of one set in $\mathcal{Q}^+(\mathbf{z}, q^+)$ and another in $\mathcal{Q}^-(\mathbf{z}, q^-)$:

$$Q \in \mathcal{Q}(\mathbf{z}, q) = \{Q \,|\, Q = Q^+ \cup Q^-,\ Q^+ \in \mathcal{Q}^+(\mathbf{z}, q^+) \text{ and } Q^- \in \mathcal{Q}^-(\mathbf{z}, q^-)\}.$$

In the notation, we have suppressed the dependence of $\mathcal{Q}(\mathbf{z}, q)$ on q^+ and q^-, to be consistent with notation for the non-balanced choice (3.19). Having determined Q, we construct the reduced normal equation for one step of our interior-point method by assembling the matrix for the normal equation using a subset of the patterns, and solving

$$\mathbf{M}_{(Q)}\Delta\mathbf{w} = -\bar{\mathbf{r}}_{\mathbf{w}} - \frac{1}{\mathbf{y}^T\mathbf{\Omega}^{-1}\mathbf{y}}\bar{r}_{\boldsymbol{\alpha}}\bar{\mathbf{y}}. \tag{3.23}$$

Then we solve (3.12)-(3.16) for $\Delta\gamma$, $\Delta\boldsymbol{\alpha}$, $\Delta\boldsymbol{\xi}$, $\Delta\mathbf{u}$, and $\Delta\mathbf{s}$. Before we proceed, we have to ensure that the reduced matrix $\mathbf{M}_{(Q)}$ is positive definite.

PROPOSITION 3.1. *The matrix $\mathbf{M}_{(Q)}$ is symmetric and positive definite.*

Proof. See Proposition 1 in [16]. □

The following proposition explains the asymptotic convergence of the reduced matrix to the unreduced one.

PROPOSITION 3.2. *For q defined in (3.18) and for all $Q \in \mathcal{Q}(\mathbf{\Omega}\mathbf{e}, q)$, there exists a positive constant $C_{\mathbf{M}}$ satisfying $\|\mathbf{M} - \mathbf{M}_{(Q)}\|_2 \leq C_{\mathbf{M}}\sqrt{\mu}$.*

Proof. See Proposition 5 in [16]. □

Winternitz et al. presented convergence results for a MPC constraint reduction algorithm for LP [34]. In recent work [22], we provided a convergence proof for a constraint-reduced affine-scaling primal-dual interior-point method for convex quadratic programming. Typically, MPC algorithms require less work than affine scaling, especially when the initial point is not near the central path. Therefore, we state in Algorithm 1 a variant of a Mehrotra-type predictor-corrector algorithm with a constraint reduction mechanism to determine $\mathbf{M}_{(Q)}$.

ALGORITHM 1 (Constraint reduced SVM (CRSVM)).

```
  *Parameters*: $\beta > 0$, $\boldsymbol{\tau} > \mathbf{0}$, $\theta > 0$, integer $q_U$ satisfying $q_U \le m$, $Bal \in \{\text{false}, \text{true}\}$,
  $CC \in \{\text{`one-sided dist'}, \text{`omega'}\}$.
  Given a starting point $(\mathbf{w}, \gamma, \boldsymbol{\xi}, \mathbf{s}, \boldsymbol{\alpha}, \mathbf{u})$ with $(\boldsymbol{\xi}, \mathbf{s}, \boldsymbol{\alpha}, \mathbf{u}) > 0$.
  **for** $k = 0, 1, 2, \ldots$ **do**
    Terminate if convergence is detected:

      $$\frac{\max\{\|\mathbf{r_w}\|_\infty, |r_{\boldsymbol{\alpha}}|, \|\mathbf{r_u}\|_\infty, \|\mathbf{r_s}\|_\infty\}}{\max\{\|A\|_\infty, \|\boldsymbol{\tau}\|_\infty, 1\}} \le tol_r \quad \text{and} \quad \mu \le tol_\mu,$$

    or iteration count is reached.

    If $Bal$ is false, then determine $q$ according to (3.18); otherwise determine $q^+$ and $q^-$
    from (3.21) and (3.22).
    Pick $Q$ from $\mathcal{Q}(\mathbf{YXw} - \gamma\mathbf{y} - \mathbf{e} + \boldsymbol{\xi}, q)$ if $CC$ is `one-sided dist', or from $\mathcal{Q}(\boldsymbol{\Omega}\mathbf{e}, q)$
    if $CC$ is `omega'.

    Solve (3.23) and (3.12)-(3.16) for $(\Delta\mathbf{w}^{\text{aff}}, \Delta\gamma^{\text{aff}}, \Delta\boldsymbol{\xi}^{\text{aff}}, \Delta\mathbf{s}^{\text{aff}}, \Delta\boldsymbol{\alpha}^{\text{aff}}, \Delta\mathbf{u}^{\text{aff}})$ using
    affine-scaling residuals from (3.7)-(3.8).
    Determine predictor step length:

      $$\alpha_{\text{aff}} := \max_{\alpha\in[0,1]} \{\alpha : (\boldsymbol{\xi}, \mathbf{s}, \boldsymbol{\alpha}, \mathbf{u}) + \alpha(\Delta\boldsymbol{\xi}^{\text{aff}}, \Delta\mathbf{s}^{\text{aff}}, \Delta\boldsymbol{\alpha}^{\text{aff}}, \Delta\mathbf{u}^{\text{aff}}) \ge 0\}. \tag{3.24}$$

    Set

      $$\mu_{\text{aff}} := \frac{(\mathbf{s} + \alpha_{\text{aff}}\Delta\mathbf{s}^{\text{aff}})^T(\boldsymbol{\alpha} + \alpha_{\text{aff}}\Delta\boldsymbol{\alpha}^{\text{aff}}) + (\boldsymbol{\xi} + \alpha_{\text{aff}}\Delta\boldsymbol{\xi}^{\text{aff}})^T(\mathbf{u} + \alpha_{\text{aff}}\Delta\mathbf{u}^{\text{aff}})}{2m}.$$

    Set $\sigma := (\mu_{\text{aff}}/\mu)^3$.

    Solve (3.23) and (3.12)-(3.16) for $(\Delta\mathbf{w}, \Delta\gamma, \Delta\boldsymbol{\xi}, \Delta\mathbf{s}, \Delta\boldsymbol{\alpha}, \Delta\mathbf{u})$ using combined step
    residuals from (3.9)-(3.10).
    Determine the step length for the combined step:

      $$\alpha := 0.99 \max_{\alpha\in[0,1]} \{\alpha : (\boldsymbol{\xi}, \mathbf{s}, \boldsymbol{\alpha}, \mathbf{u}) + \alpha(\Delta\boldsymbol{\xi}, \Delta\mathbf{s}, \Delta\boldsymbol{\alpha}, \Delta\mathbf{u}) \ge 0\}. \tag{3.25}$$

    Set

      $$(\mathbf{w}, \gamma, \boldsymbol{\xi}, \mathbf{s}, \boldsymbol{\alpha}, \mathbf{u}) := (\mathbf{w}, \gamma, \boldsymbol{\xi}, \mathbf{s}, \boldsymbol{\alpha}, \mathbf{u}) + \alpha(\Delta\mathbf{w}, \Delta\gamma, \Delta\boldsymbol{\xi}, \Delta\mathbf{s}, \Delta\boldsymbol{\alpha}, \Delta\mathbf{u}).$$

  **end for**
```

When the matrix $\mathbf{X}$ is sparse, the first two terms in $\mathbf{M}_{(Q)}$ are probably also sparse, but adding the third term makes the matrix dense. Therefore, in this case we solve the normal equations (3.23) by computing a sparse Cholesky factor (with full pivoting) for the sum of the first two terms, and then applying the SMW formula to incorporate the rank 1 matrix. When $\mathbf{X}$ is dense, we fully assemble $\mathbf{M}_{(Q)}$ and obtain its dense Cholesky factor.

4. Kernelization. So far we have considered the case in which the Gram matrix $\mathbf{K}$ is defined as $k_{ij} = \mathbf{x}_i^T \mathbf{x}_j$, corresponding to the standard inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^T \mathbf{x}_j$. More generally, we might define $\mathbf{K}$ using a symmetric and positive semidefinite *kernel*:[4]

$$\begin{aligned} k : \mathcal{X} \times \mathcal{X} &\rightarrow \mathbb{R} \\ (\mathbf{x}, \bar{\mathbf{x}}) &\mapsto k(\mathbf{x}, \bar{\mathbf{x}}), \end{aligned}$$

setting $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.[5] We assume that the dimension of the space $\mathcal{X}$ is ℓ, reserving n for the rank of the approximation to $\mathbf{K}$ used in the optimization problem derived below.

If a data set has an enormous number of training patterns, building $\mathbf{K}$ may not be feasible. For example, if the kernel is Gaussian, $\mathbf{K}$ is usually dense even when the matrix $\mathbf{X}$ of training patterns is sparse. Even worse, the matrix $\mathbf{M}$ is now $m \times m$, and our constraint reduction is not effective. The reason for this is that if $\mathbf{K}$ has full rank, our KKT conditions become

$$\begin{aligned} \mathbf{K}\mathbf{w} - \mathbf{K}\mathbf{Y}\boldsymbol{\alpha} &= \mathbf{0}, \\ \mathbf{y}^T \boldsymbol{\alpha} &= 0, \\ \boldsymbol{\tau} - \boldsymbol{\alpha} - \mathbf{u} &= \mathbf{0}, \\ \mathbf{Y}\mathbf{K}\mathbf{w} - \gamma \mathbf{y} + \boldsymbol{\xi} - \mathbf{e} - \mathbf{s} &= \mathbf{0}, \\ \mathbf{S}\boldsymbol{\alpha} &= \mathbf{0}, \\ \mathbf{U}\boldsymbol{\xi} &= \mathbf{0}, \\ \mathbf{s}, \mathbf{u}, \boldsymbol{\alpha}, \boldsymbol{\xi} &\geq \mathbf{0}, \end{aligned}$$

corresponding to the primal problem

$$\begin{aligned} &\min_{\mathbf{w}, \gamma, \boldsymbol{\xi}} \frac{1}{2} \mathbf{w}^T \mathbf{K} \mathbf{w} + \boldsymbol{\tau}^T \boldsymbol{\xi} \\ &\text{s.t.} \quad \mathbf{Y}(\mathbf{K}\mathbf{w} - \mathbf{e}\gamma) + \boldsymbol{\xi} \geq \mathbf{e}, \\ &\qquad\quad \boldsymbol{\xi} \geq \mathbf{0}. \end{aligned}$$

The resulting normal equations matrix is

$$\mathbf{M} = \mathbf{K} + \mathbf{K}^T \mathbf{Y} \boldsymbol{\Omega}^{-1} \mathbf{Y} \mathbf{K} - \frac{\bar{\mathbf{y}} \bar{\mathbf{y}}^T}{\mathbf{y}^T \boldsymbol{\Omega}^{-1} \mathbf{y}}, \tag{4.1}$$

which is difficult to approximate. Several researchers [6, 15, 30] have proposed to replace the Gram matrix with a low rank approximation $\mathbf{K} \approx \mathbf{V}\mathbf{G}^2\mathbf{V}^T$, where $\mathbf{G}$ is an $n \times n$ symmetric and positive definite matrix and $\mathbf{V}$ is an $m \times n$ matrix, for $m \gg n$. If $\mathbf{V}$ has full rank, then $\mathbf{K}$ has rank n. Such approximations have been computed using the truncated eigenvalue decomposition [7, 18], low rank Cholesky factorization with symmetric pivoting [2, 15], the Nyström method [13, 33], and kernel PCA map [19, 31]. For some kernels, the fast multipole method [27, 37] can be employed to compute the truncated eigenvalue decomposition.

Substituting a low rank approximation for $\mathbf{K}$ allows constraint reduction to be effective in training nonlinear SVMs, if we are careful in our problem formulation. Consider an approximate dual CQP with $\mathbf{K}$ in (2.14)-(2.16) replaced by $\mathbf{V}\mathbf{G}^2\mathbf{V}^T$. What primal problem

[4] Researchers in the machine learning field often use “positive definite” to denote “positive semidefinite” and “strictly positive definite” for “positive definite”. Here we use the more common definitions from mathematics.

[5] The kernel is symmetric if $k(\mathbf{x}, \bar{\mathbf{x}}) = k(\bar{\mathbf{x}}, \mathbf{x})$ for every $\mathbf{x}, \bar{\mathbf{x}} \in \mathcal{X}$. It is positive semidefinite if the Gram matrix is positive semidefinite for *every* finite collection of pattern vectors. In this case, there is an associated reproducing kernel map that allows us to define $\mathbf{K}$ in the way we have indicated; see [4] and [30] for more details.

would induce this dual? It is the problem obtained by substituting $\mathbf{VG}$ for $\mathbf{X}$ in the primal (2.11)-(2.13). Therefore, to take advantage of constraint reduction in training nonlinear SVMs, we use the data $\mathbf{VG}$ in place of $\mathbf{X}$ and apply Algorithm 1.

If $\mathbf{G}$ is only readily available in its squared inverse form $\mathbf{G}^{-2}$, we could think of letting $\bar{\mathbf{w}} = \mathbf{G}\mathbf{w} \in \mathbb{R}^n$, which leads to the following problem:

$$\begin{aligned} \min_{\bar{\mathbf{w}},\gamma,\boldsymbol{\xi}} \; & \frac{1}{2}\bar{\mathbf{w}}^T\mathbf{G}^{-2}\bar{\mathbf{w}} + \boldsymbol{\tau}^T\boldsymbol{\xi} \\ \text{s.t.} \; & \mathbf{Y}(\mathbf{X}\bar{\mathbf{w}} - \mathbf{e}\gamma) + \boldsymbol{\xi} \geq \mathbf{e}, \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned}$$

where $\mathbf{X} = \mathbf{V}$. This formulation would be useful if computing $\mathbf{G}$ is not desirable. For instance, $\mathbf{G}^{-2}$ is a submatrix of $\mathbf{K}$ when the empirical kernel map [29, 31] is employed. Applying the constraint reduction to this formulation is straightforward. A simple change of $\mathbf{M}_{(Q)}$ from (3.17) to

$$\mathrm{M}_{(Q)} = \mathbf{G}^{-2} + \mathbf{X}_Q{}^T\mathbf{Y}_{Q^2}\boldsymbol{\Omega}_{Q^2}^{-1}\mathbf{Y}_{Q^2}\mathbf{X}_Q - \frac{\bar{\mathbf{y}}_{(Q)}\bar{\mathbf{y}}_{(Q)}^T}{\mathbf{y}_Q{}^T\boldsymbol{\Omega}_{Q^2}^{-1}\mathbf{y}_Q}, \tag{4.2}$$

where $\bar{\mathbf{y}}_{(Q)} = \mathbf{X}_Q^T\mathbf{Y}_{Q^2}\Omega_{Q^2}^{-1}\mathbf{y}_Q$, and the substitution of $\bar{\mathbf{w}}$ for $\mathbf{w}$, $\Delta\bar{\mathbf{w}}$ for $\Delta\mathbf{w}$, and

$$\mathbf{r}_{\bar{\mathbf{w}}} = \mathbf{G}^{-2}\bar{\mathbf{w}} - \mathbf{X}^T\mathbf{Y}\boldsymbol{\alpha}$$

for $\mathbf{r}_{\mathbf{w}}$, are all the required modifications.

5. Numerical results.

We tested Algorithm 1 using MATLAB version R2007a on a machine running Windows XP SP2 with an Intel Pentium IV 2.8GHz processor with 16 KB L1 cache, 1 MB L2 cache, 2×1 GB DDR2-400MHz configured as dual channel, with Hyper Threading enabled.

Both tol_r and tol_μ were set to 10^{-8}. The iteration limit was set to 200. We set the parameters as $\beta = 4$ to control the rate of decrease of q, $\theta = 10^2$ to determine q_L, and $\tau_i = 1$ for $i = 1, ..., m$ to penalize misclassification. In our experiments we vary q_U. The initial starting point was set as in [16]:

$$\mathbf{w} = \mathbf{0},\ \gamma = 0,\ \boldsymbol{\xi} = \mathbf{s} = \boldsymbol{\alpha} = \mathbf{u} = 2\mathbf{e}.$$

We compared Algorithm 1 CRSVM to LIBSVM [5], and SVM^{light} [21]. We set their termination tolerance parameters as their default value 10^{-3}.

5.1. Linear SVM examples.

We tested our implementation on problems MUSHROOM, ISOLET, WAVEFORM, and LETTER, all taken from [16]. Except for the ISOLET problem, all inputs were mapped to higher dimensional feature space via the mapping associated with the second order polynomial kernel $k(\mathbf{x}, \bar{\mathbf{x}}) = (\mathbf{x}^T\bar{\mathbf{x}} + 1)^2$, as in [16]. The mapping Φ is defined as

$$\begin{aligned} & \Phi : \mathbb{R}^l \rightarrow \mathbb{R}^n \\ & \begin{bmatrix} x_1 \\ \vdots \\ x_l \end{bmatrix} \mapsto \sqrt{2}\left[\frac{x_1^2}{\sqrt{2}}, \ldots, \frac{x_l^2}{\sqrt{2}}, x_1x_2, \ldots, x_1x_l, \right. \\ & \qquad\qquad \left. x_2x_3, \ldots, x_2x_l, \ldots, x_{l-1}x_l, x_1, \ldots, x_l, \frac{1}{\sqrt{2}}\right]^T, \end{aligned} \tag{5.1}$$

Name	ISD	FSD	Patterns $(+/-)$	SVs $(+/-)$	On-SVs $(+/-)$
MUSHROOM	22	276	8124 (4208/ 3916)	2285 (1146/1139)	52 (31 / 21)
ISOLET	617	617	7797 (300 / 7497)	186 (74 / 112)	186 (74 /112)
WAVEFORM	40	861	5000 (1692/ 3308)	1271 (633 / 638)	228 (110/118)
LETTER	16	153	20000 (789 /19211)	543 (266 / 277)	40 (10 / 30)

TABLE 5.1: *Properties of the problems. ISD: Input space dimension. FSD: Feature space dimension using the map (5.1). SVs: support vectors. On-SVs: on-boundary support vectors.*

where $n = \binom{l+2}{2} = \binom{l}{2} + 2l + 1$. The i^{th} row of $\mathbf{X}$ is set to $\Phi(\mathbf{x}_i)^T$, where $\mathbf{x}_i^T$ is the i^{th} training input. We also normalized the resulting matrix using

$$x_{ij} = \frac{x_{ij}}{\max_{kl} |x_{kl}|}, \tag{5.2}$$

as directed in [16]. Properties of the problems are summarized in Table 5.1.

In our experiment, we compared our algorithms to the unreduced MPC algorithm, which uses all the constraints for every iteration. We experimented with several variants of our algorithms:

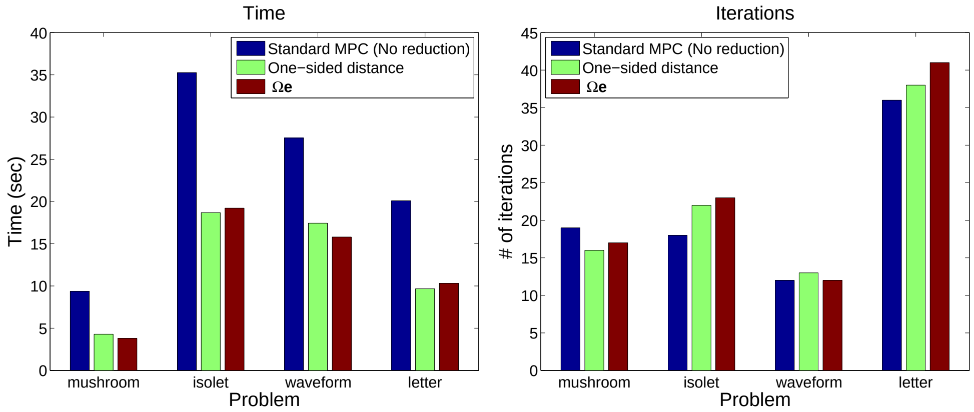
- nonadaptive balanced constraint reduction, which uses fixed q^+ and q^- throughout the iteration;
- adaptive non-balanced constraint reduction, which determines q as in (3.18);
- adaptive balanced constraint reduction, which determines q^+ and q^- as in (3.21) and (3.22).

We chose constraints using either one-sided distance ($\mathbf{YXw} - \gamma\mathbf{y} + \boldsymbol{\xi} - \mathbf{e}$) or $\mathbf{\Omega e}$ to form the set Q, as explained in Section 3.2, resulting in six algorithm variants.

In Figure 5.1a and 5.1b, the time and iteration count of the algorithm with the two constraint choices and the balanced selection scheme are compared to those of the unreduced MPC. We set $q_U = m$, $Bal =$ true, and $CC =$ 'one-sided dist' or $CC =$ 'omega'. Bar graphs are grouped by problem. All algorithms produced residuals with similar accuracy, and Figure 5.1b shows that the number of iterations is insensitive to algorithm choice. As a result, all of the constraint reduction algorithms are faster than the unreduced MPC algorithm, as seen in Figure 5.1a. In solving hard problems (MUSHROOM and WAVEFORM for instance, which have many support vectors), it is observed that the constraint choice based on $\mathbf{\Omega e}$ shows better performance than the other. This is because the number of on-boundary support vectors is nevertheless small in the hard cases; see Table 5.1.

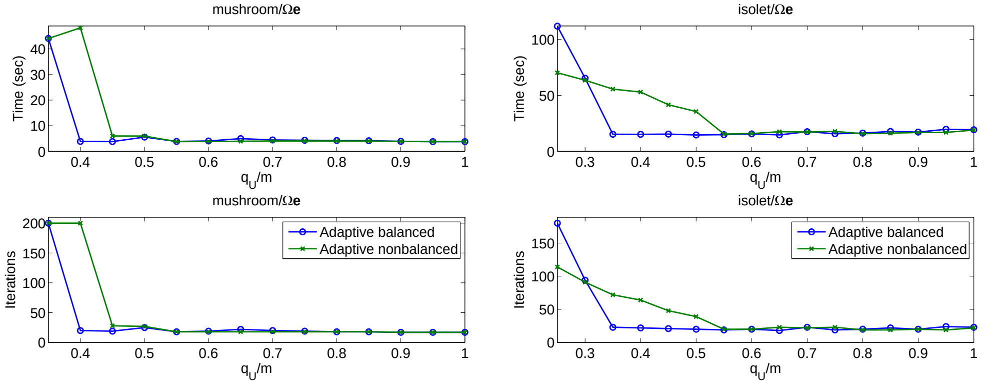
Figure 5.2 compares the balanced and nonbalanced adaptive reduction algorithms over a range of q_U. In solving well balanced problems, the two algorithms show little difference, as seen in Figure 5.2a. On the other hand, for problems such as ISOLET, having many more patterns in one class than the other, balanced selection gives more consistent performance, especially for small values of q_U, as seen in Figure 5.2b. In problems with more than two classification labels, a one-class-versus-the-rest approach is frequently employed [30, Chap. 7], so the number of "$-$" patterns is often much larger than the number of "$+$" patterns.

In Figure 5.3, we compare the adaptive and nonadaptive balanced reduction algorithms over a range of values of the upper bound on the index size q_U. Observe that there is little difference in iteration counts between the two variants over a large range of q_U values. The time taken to solve a problem decreases very slowly or remains almost invariant with the adaptive algorithm, as q_U decreases over this range, whereas the nonadaptive algorithm is more expensive for large values of q_U.



(A) *Time* (B) *Iteration count*

FIGURE 5.1: *Time and iteration count of adaptive reduction with balanced selection, compared to those for the original MPC algorithm. The value of q_U is set to $m\,(100\%)$ for all cases.*



(A) MUSHROOM *problem: in solving a well balanced problem, the two algorithms show little difference.*

(B) ISOLET *problem: in solving a poorly balanced problem, the balanced algorithm shows better robustness.*

FIGURE 5.2: *The adaptive balanced and adaptive nonbalanced algorithms are compared, with the constraint choice based on* $\mathbf{\Omega e}$.

In Figure 5.4, the two constraint choices based on one-sided distance and $\mathbf{\Omega e}$ are applied to the adaptive balanced reduction algorithm and are compared over a range of q_U values. In solving easy problems having almost all support vectors on the boundary planes, it is hard to say which constraint choice is better than the other. For hard problems, the $\mathbf{\Omega e}$ based constraint choice is capable of filtering out more patterns at later iterations and shows better performance.

In Figure 5.5, we compare the number of patterns used and the complementarity mea-

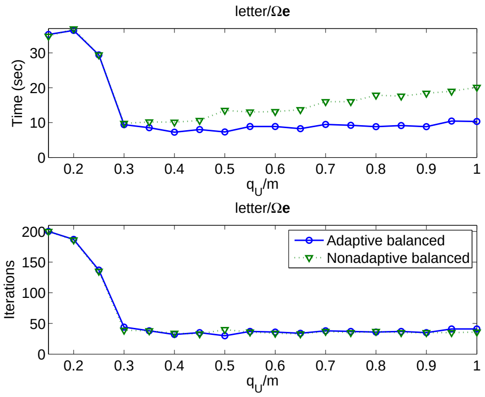
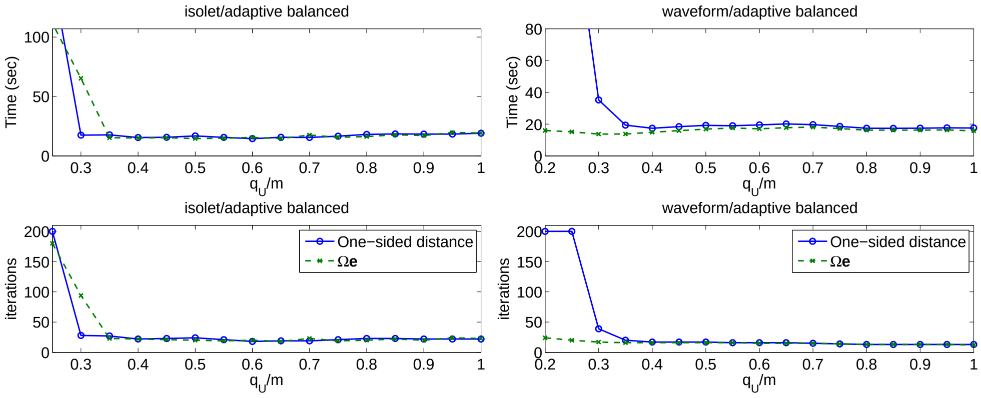


FIGURE 5.3: LETTER *problem: the adaptive and nonadaptive balanced algorithms are compared, with the constraint choice based on* $\mathbf{\Omega e}$*.*



(A) ISOLET, *an easy problem.*

(B) WAVEFORM, *a hard problem.*

FIGURE 5.4: *The two constraint choices are applied for adaptive balanced reduction.*

surement μ at every iteration for various choices of q_U. When q_U is large, the graphs of μ are quite close to each other. From these graphs we see that the search direction of the adaptive reduction algorithm is not as good as that of the unreduced algorithm at early iterations. At later iterations, however, the search direction of the adaptive reduction algorithm is as good as that of the unreduced MPC algorithm, and sometimes better.

We compared our algorithm CRSVM to LIBSVM [5] and SVM^{light} [21] on the ADULT problem of the UCI repository [1]. We obtained a formatted problem from the LIBSVM web page [5]. The problem consists of 9 sparse training sets with different numbers of sample patterns. Each training set has a corresponding testing set. For this comparison, we used the linear SVM, giving the algorithms $\mathbf{X}$ in a sparse format with no modification except the normalization (5.2). We used adaptive balanced constraint reduction, choosing patterns based on $\mathbf{\Omega e}$. Figure 5.6 shows the timing results. Observe that the timing curve of our algorithm is close to linear, while those of LIBSVM and SVM^{light} are between linear and cubic [26].

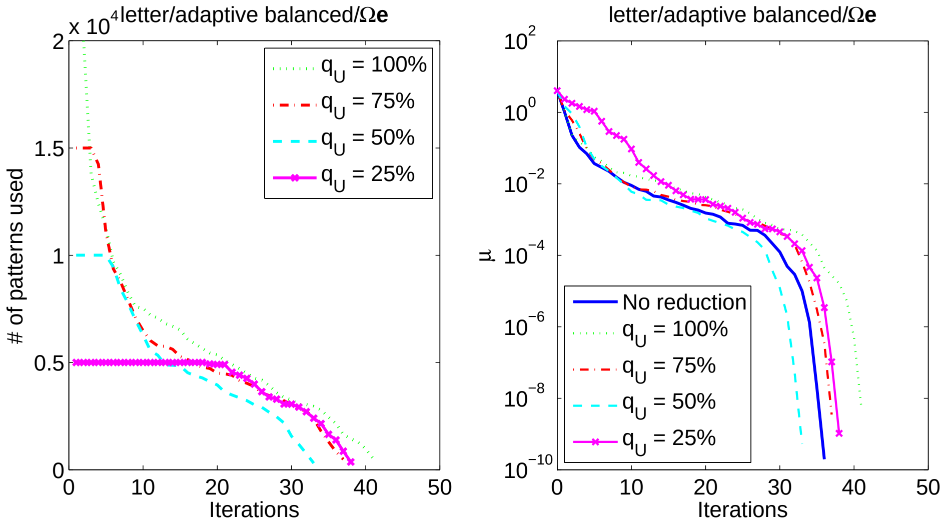
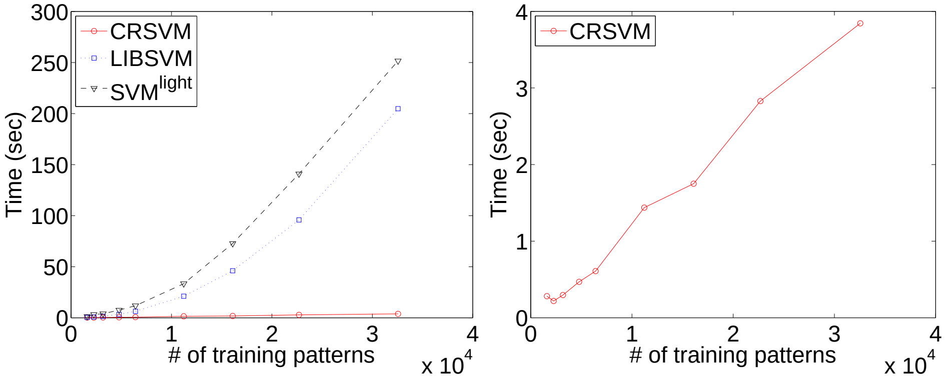


FIGURE 5.5: LETTER *problem: adaptive balanced reduction based on* $\mathbf{\Omega e}$ *constraint choice.*



(A) *Timing results.*

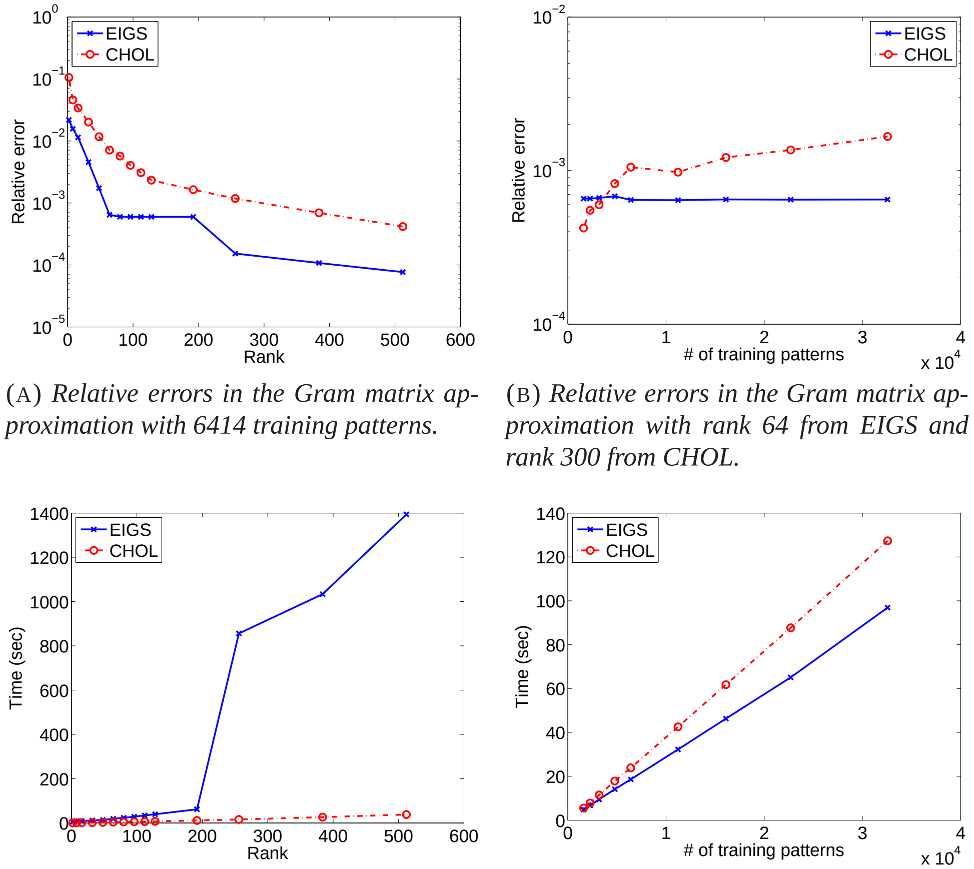
(B) *Timing complexity is almost linear in the number of training patterns.*

FIGURE 5.6: *Timing results of algorithms for linear SVM training on* ADULT *data sets.*

5.2. Nonlinear SVM examples. We compared our algorithm to LIBSVM and SVM^{light}. We used adaptive balanced constraint reduction, choosing patterns based on $\mathbf{\Omega e}$. We tested the algorithms on the ADULT problem of the UCI repository. For this comparison, we used a Gaussian kernel $k(\mathbf{x}, \bar{\mathbf{x}}) = \exp\left(-\|\mathbf{x} - \bar{\mathbf{x}}\|^2/(2\sigma^2)\right)$ with $\sigma = \sqrt{l/2}$, where l is 123, the dimension of the input patterns.[6]

We computed a low-rank approximation to $\mathbf{K}$ using both MATLAB's EIGS function and our implementation in MATLAB of a low rank Cholesky factorization with symmetric pivoting [15]. The CHOL algorithm returns a rank n Cholesky factor $\mathbf{L}$ and a permutation

[6] This is the default setting of Gaussian kernel in LIBSVM.



(A) *Relative errors in the Gram matrix approximation with 6414 training patterns.*

(B) *Relative errors in the Gram matrix approximation with rank 64 from EIGS and rank 300 from CHOL.*

(C) *Time to approximate the Gram matrix with 6414 training patterns.*

(D) *Time to approximate the Gram matrix with rank 64 from EIGS and rank 300 from CHOL.*

FIGURE 5.7: *Gram matrix approximation on* ADULT *data sets.*

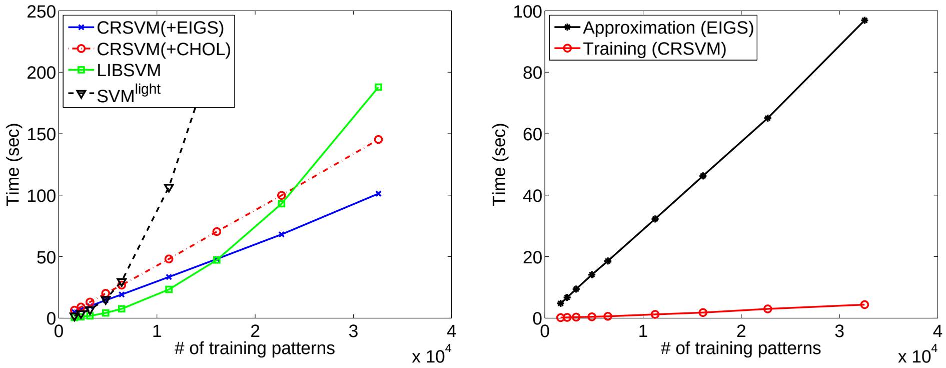
matrix $\mathbf{P}$ such that $\mathbf{P}^T\mathbf{L}\mathbf{L}^T\mathbf{P} \approx \mathbf{K}$.

Figure 5.7 shows results for these approximations on the ADULT data set.[7] Figure 5.7a and 5.7b show relative error of the low rank approximation to $\mathbf{K}$ measured by

$$\|\mathbf{K} - \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T\|_\infty / \|\mathbf{K}\|_\infty \quad \text{and} \quad \|\mathbf{K} - \mathbf{P}^T\mathbf{L}\mathbf{L}^T\mathbf{P}\|_\infty / \|\mathbf{K}\|_\infty.$$

As illustrated in Figure 5.7a, for a given rank EIGS approximates $\mathbf{K}$ much better than CHOL. However, EIGS uses a fast multipole algorithm, IFGT, to approximate matrix-vector products involving $\mathbf{K}$, and there is a limit to how large the rank can be before IFGT becomes impractically slow, since its tolerance must be tightened as the rank is increased. In our experiments we set the IFGT tolerance to be $\min(0.5, 4/\sqrt{\text{rank}})$. Figure 5.7b shows that, when the rank is fixed, errors in the Gram matrix approximation by CHOL are more sensitive to the number of training patterns. Figure 5.7c and 5.7d show the time to approximate $\mathbf{K}$. In Figure 5.7a and 5.7c, EIGS and CHOL were tested on the set of 6414 training patterns. In Figure 5.7b

[7] IFGT supports dense input only.



(A) *Timing results of algorithms on* ADULT *data sets. The CRSVM time includes Gram matrix approximation.*

(B) *Time to approximate the Gram matrix and train SVMs with a rank 64 approximation through EIGS.*

FIGURE 5.8: *Nonlinear SVM training on* ADULT *data sets.*

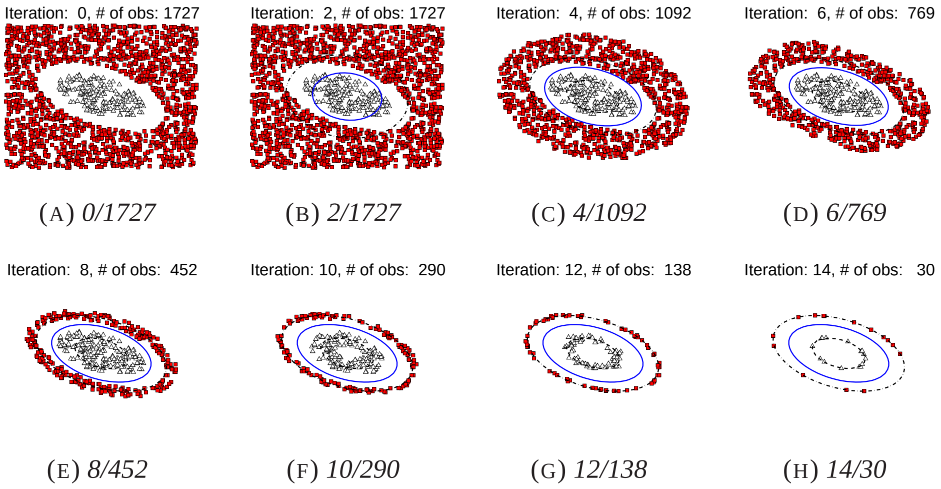
Size	LIBSVM	SVMLight	CRSVM(+EIGS)	CRSVM(+CHOL)
1605	83.57	83.57	83.62	83.60
2265	83.94	83.94	83.93	83.95
3185	83.85	83.84	83.85	83.84
4781	83.97	83.97	83.97	83.97
6414	84.15	84.15	84.17	84.19
11220	84.17	84.18	84.21	84.21
16100	84.58	84.58	84.58	84.45
22696	85.01	-	84.82	84.98
32561	84.82	-	84.92	84.85

TABLE 5.2: *Accuracy shown as percent of testing patterns correctly classified.*

and 5.7d, we requested a rank 64 approximation from EIGS and a rank 300 approximation from CHOL.

Figure 5.8a compares CRSVM with the other methods. Notice both LIBSVM and SVM^{light} are implemented in the C language. We expect we can improve CRSVM and CHOL by implementing them in C. We requested 64 eigenvalues and eigenvectors from EIGS to form a rank 64 approximation to $\mathbf{K}$. We set CHOL to form a rank 300 approximation. Figure 5.8b shows times for both the approximation and the training. Table 5.2 shows accuracy of the classifier generated by each algorithm, measured by the percentage of correctly classified testing patterns. The classifiers were tested on data sets associated with the training set. Notice that, with a proper approximation, it is possible to get a classifier performing as well as the one trained with the exact matrix.

5.3. Visualizing how the algorithm works. To illustrate how our algorithm achieves efficiency, we constructed a two dimensional toy problem. We generated 2000 uniformly distributed random points in $[-1,1] \times [-1,1]$. Then, we set an intentional ellipsoidal separation gap and deleted patterns inside the gap, resulting in 1727 remaining patterns.



(A) *0/1727* (B) *2/1727* (C) *4/1092* (D) *6/769*

(E) *8/452* (F) *10/290* (G) *12/138* (H) *14/30*

FIGURE 5.9: *Snapshots of finding a classifier using the adaptive reduction algorithm for a randomly generated toy problem in 2-dimensional input space. The mapping associated with the second order homogeneous polynomial kernel is used to find the surface. The numbers below each figure indicate* $(\textit{iteration})/(\textit{number of involving patterns})$.

Figure 5.9 shows snapshots of several iterations of the adaptive balanced reduction algorithm (with $q_U = m$) in solving the problem. Patterns are chosen based on $\mathbf{\Omega e}$. To find an ellipsoidal classifier, the mapping (2.5) (associated with a second order homogeneous polynomial kernel) is used to map the 2-dimensional input space of the problem to a 3-dimensional feature space. The dashed ellipsoids are the boundary curves, corresponding to boundary planes in the feature space. As the iteration count increases, the number of selected patterns decreases, and only the on-boundary support vectors are chosen at the end, leading to significant time savings.

6. Conclusion. We presented an algorithm for training SVMs using a constraint reduced IPM with a direct solver for the normal equations. Significant time saving is reported for all problems, since the algorithm acts as an adaptive filter for excluding unnecessary patterns. For problems in which iterative solvers are appropriate, constraint reduction would reduce the cost of matrix-vector products.

Balanced constraint selection is more robust than unbalanced. The $\mathbf{\Omega e}$ constraint choice proved to be more effective than the one-sided distance, especially for hard problems that have many off-boundary support vectors. Other constraint choice heuristics can be used provided that they include constraints that seem to be most active at the current point. Blending different constraint choices is also allowable.

We experimentally compared our algorithms with other popular algorithms, including LIBSVM and SVM^{light}. We showed the potential of our algorithms on training linear SVMs. In training nonlinear SVMs, we showed how our algorithm can be applied when using a low-rank approximation to the Gram matrix.

The algorithm has substantial potential parallelism, but load balancing could be challenging, since constraints are dynamically chosen.

Acknowledgment. We would like to thank Joshua D. Griffin for helpful conversations and for providing the SVM training samples.

REFERENCES

[1] A. Asuncion and D. Newman, *UCI machine learning repository*, 2007. Available at: http://archive.ics.uci.edu/ml.

[2] F. R. Bach and M. I. Jordan, *Predictive low-rank decomposition for kernel methods*, in ICML '05: Proc. of the 22nd International Conf. on Machine Learning (Bonn, Germany, August 07 - 11, 2005), ICML '05, vol. 119. ACM, New York, NY, pp. 33–40.

[3] D. Boley and D. Cao, *Training support vector machines using adaptive clustering*, in Proceedings of the Fourth SIAM International Conference on Data Mining, M. W. Berry, U. Dayal, C. Kamath, and D. Skillicorn, eds., SIAM Press, Philadelpha, 2004, pp. 126–137.

[4] C. Burges, *A tutorial on support vector machines for pattern recognition*, Data Min. Knowl. Discov., 2 (1998), pp. 121–167.

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

[6] O. Chapelle, *Training a support vector machine in the primal*, Neural Comput., 19 (2007), pp. 1155–1178.

[7] T.-J. Chin, K. Schindler, and D. Suter, *Incremental kernel SVD for face recognition with image sets*, in FGR '06: Proc. of the 7th International Conf. on Automatic Face and Gesture Recognition (FGR06), Washington, 2006, IEEE Computer Society, pp. 461–466.

[8] C. Cortes and V. Vapnik, *Support-vector networks*, Mach. Learn., 20 (1995), pp. 273–297.

[9] G. B. Dantzig and Y. Ye, *A build–up interior point method for linear programming: Affine scaling form*, Tech. Report, University of Iowa, Iowa City, July 1991.

[10] D. den Hertog, J. Kaliski, C. Roos, and T. Terlaky, *A logarithmic barrier cutting plane method for convex programming*, Ann. Oper. Res., 58 (1995), pp. 67–98.

[11] D. den Hertog, C. Roos, and T. Terlaky, *A build-up variant of the path-following method for LP*, Tech. Report DUT-TWI-91-47, Delft University of Technology, Delft, 1991.

[12] ———, *Adding and deleting constraints in the path–following method for linear programming*, in Advances in Optimization and Approximation (Nonconvex Optimization and Its Applications), vol. 1, Kluwer Academic Publishers, Dordrecht, 1994, pp. 166–185.

[13] P. Drineas and M. Mahoney, *On the Nyström method for approximating a Gram matrix for improved kernel-based learning*, J. Mach. Learn. Res., 6 (2005), pp. 2153–2175.

[14] M. C. Ferris and T. S. Munson, *Interior-point methods for massive support vector machines*, SIAM J. Optim., 13 (2002), pp. 783–804.

[15] S. Fine and K. Scheinberg, *Efficient SVM training using low-rank kernel representations*, J. Mach. Learn. Res., 2 (2002), pp. 243–264.

[16] E. M. Gertz and J. D. Griffin, *Support vector machine classifiers for large data sets*, Tech. Report ANL/MCS-TM-289, Argonne National Laboratories, Argonne, 2006. Available at: http://www.osti.gov/energycitations/product.biblio.jsp?osti_id=881587.

[17] E. M. Gertz and S. J. Wright, *Object-oriented software for quadratic programming*, ACM Trans. Math. Software, 29 (2003), pp. 58–81.

[18] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 1996.

[19] S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller, *Kernel-based nonlinear blind source separation*, Neural Comput., 15 (2003), pp. 1089–1124.

[20] M. A. Hearst, *Trends and controversies: Support vector machines*, IEEE Intell. Syst., 13 (1998), pp. 18–28.

[21] T. Joachims, *Making large-scale SVM learning practical*, in Advances in Kernel Methods – Support Vector Learning, B. Schölkopf, C. Burges, and A. Smola, eds., MIT Press, 1998, pp. 169–185.

[22] J. H. Jung, D. P. O'Leary, and A. L. Tits, *Adaptive constraint reduction for convex quadratic programming*, University of Maryland, 2007. Available at: http://www.optimization-online.org/DB_HTML/2007/10/1820.html.

[23] Z.-Q. Luo and J. Sun, *An analytic center based column generation algorithm for convex quadratic feasibility problems*, SIAM J. Optim., 9 (1998), pp. 217–235.

[24] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM J. Optim., 2 (1992), pp. 575–601.

[25] E. Osuna, R. Freund, and F. Girosi, *An improved training algorithm for support vector machines*, Proc. IEEE Signal Processing Society Workshop on Neural Networks and Signal Processing VII, J. Principe, L. Gile, N. Morgan, and E. Wilson, eds. IEEE Press, New York (1997), pp. 276–285. Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=622408

[26] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*, in Advances in Kernel Methods: Support Vector Learning, B. Schölkopf, C. Burges, and A. Smola, eds., MIT Press,

Cambridge, 1999, pp. 185–208.

[27] V. C. Raykar, C. Yang, R. Duraiswami, and N. Gumerov, *Fast computation of sums of Gaussians in high dimensions*, Tech. Report CS-TR-4767, Department of Computer Science, University of Maryland, College Park, 2005. Available at: http://www.umiacs.umd.edu/~vikas/Software/IFGT/IFGT_code.htm.

[28] B. Schölkopf, C. Burges, and A. Smola, *Introduction to support vector learning*, in Advances in Kernel Methods – Support Vector Learning, B. Schölkopf, C. Burges, and A. Smola, eds., MIT Press, 1998, pp. 1–22.

[29] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, and A. J. Smola, *Input space versus feature space in kernel-based methods*, IEEE Trans. Neural Networks, 5 (1999), pp. 1000–1017.

[30] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, 2001.

[31] A. J. Smola and B. Schölkopf, *Sparse greedy matrix approximation for machine learning*, in Proc. 17th International Conf. on Machine Learning, P. Langley, ed., Morgan Kaufmann, San Francisco, CA, 2000, pp. 911–918.

[32] A. L. Tits, P.-A. Absil, and W. P. Woessner, *Constraint reduction for linear programs with many inequality constraints*, SIAM J. Optim., 17 (2006), pp. 119–146.

[33] C. K. I. Williams and M. Seeger, *Using the Nyström method to speed up kernel machines*, in Neural Information Processing Systems 13, T. Leen, T. Dietterich, and V. Tresp, eds., MIT Press, 2001, pp. 682–688.

[34] L. B. Winternitz, S. O. Nicholls, A. L. Tits, and D. P. O'Leary, *A constraint-reduced variant of Mehrotra's predictor-corrector algorithm*. Submitted for publication, 2007. Available at: http://www.optimization-online.org/DB_HTML/2007/07/1734.html.

[35] K. Woodsend and J. Gondzio, *Exploiting separability in large-scale support vector machine training*, Tech. Report MS-07-002, School of Mathematics, University of Edinburgh, 2007. Available at: http://www.optimization-online.org/DB_HTML/2007/08/1750.html.

[36] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, 1997.

[37] C. Yang, R. Duraiswami, and L. Davis, *Efficient kernel machines using the improved fast Gauss transform*, in Advances in Neural Information Processing Systems 17, L. K. Saul, Y. Weiss, and L. Bottou, eds., MIT Press, Cambridge, 2005, pp. 1561–1568.

[38] Y. Ye, *A "build-down" scheme for linear programming*, Math. Program., 46 (1990), pp. 61–72.

[39] ———, *A potential reduction algorithm allowing column generation*, SIAM J. Optim., 2 (1992), pp. 7–20.