A Proximity Algorithm for Support Vector Machine Classification

Athanasios Sideris and Sílvia Estévez Castella
Department of Mechanical and Aerospace Engineering
University of California, Irvine
Irvine, CA 92697
{asideris, sestevez}@uci.edu

Abstract— **We propose a new algorithm for Support Vector Machine classification based on a geometric interpretation of the problem as finding the minimum distance between the polytopes defined by the points of the two classes. This geometric formulation applies to the hard margin, and the soft margin classification problem with quadratic violations. Our approach is based on Wolfe's classical proximity algorithm and our results show that the computational and storage requirements per iteration are relatively modest.**

I. Introduction

Support Vector Machines (SVMs) [1] provide a powerful tool for solving data mining tasks such as classification and regression. For example, the generalization capabilities of SVMs have been firmly established on inference principles such *Structural Risk Minimization* and *Regularization Theory*, with significant implications that allow the separation of model complexity and generalization [2]. At the same time, training of SVMs involves the solution of a convex quadratic problem that, albeit usually of large scale, circumvents problems of getting trapped in local minima as, for example, in training Feedforward Neural Networks. A lot of research has been focused on efficiently solving the quadratic optimization problem in SVMs. The size of this problem is determined by the amount of available data for training and it can be huge. Thus general purpose algorithms are usually inappropriate for solving it and special methods have been proposed that take advantage of the simple structure in the constraints of the dual problem [3], [4], [5], [6], [7]. Of such algorithms, Platt's *Sequential Minimal Optimization (SMO)* algorithm [5] is distinguished for its simplicity and surprisingly overall fast performance. Some notable computational improvements of this algorithm are reported in [8] and a C^{++} implementation is available in [9].

In this paper, we propose a new algorithm for SVM two-class classification based on a geometric interpretation of the problem as finding the minimum distance between the polytopes defined by the points of the two classes [10], [7]. Such a geometric algorithm has been proposed in [7] and it is based on combining the two classical algorithms of Gilbert [11], and of Mitchell-Dem'yanov-Malozemov (MDM) [12] to solve this problem. It turns out that the geometric algorithm of [7] and SMO, although developed in separate contexts, share many common elements and thus exhibit similar behaviors. Our proposed geometric approach is based on another classical proximity algorithm, namely Wolfe's algorithm [13]. The latter is considered as the method of choice for solving proximity (nearest point) problems, but it was dismissed in [7] as unsuitable for the SVM problem on grounds of excessive computational and storage requirements. However, we demonstrate in this paper, that the increased computational requirements per iteration of the Wolfe approach over the nearest point algorithm of [7] and SMO are well offset in many problems by the dramatically reduced number of iterations. Thus the proposed algorithm turns out to be faster than the previous algorithms on many standard test problems.

II. Problem Formulation and Background Results

In two-class classification (or pattern recognition), the objective is to decide whether a given pattern $x \in \mathbb{R}^p$ belongs to the positive class $\mathcal{C}^+$ or negative class $\mathcal{C}^-$. A classifier is a function $f(x)$ selected from a specified set $\mathcal{F}$ of possible functions (learning machines), so that $f(x) > 0$ if x is deemed to be in $\mathcal{C}^+$, and $f(x) < 0$ if x is deemed to be in $\mathcal{C}^-$. In supervised training, a classifier from $\mathcal{F}$ is selected so as to classify a set of given patterns $x_i \in \mathbb{R}^p$, $i = 1, \dots, l$, and corresponding class labels $y_i \in \{-1, 1\}$, in some optimal way. A *separable* training set is one that $f \in \mathcal{F}$ exists that classifies it without error; otherwise the training set is *non-separable*. *Linear Classifiers* are hyperplanes $f(x) = w^T x + b$ where $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$ are the parameters of the classifier. For a linearly separable training set, a *Linear SVM* is in fact a maximum margin classifier, where the margin is the gap between two parallel hyperplanes that separate the set. It can be shown [1], that the parameters of linear SVM can be found by solving the following convex quadratic problem:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \\ s.t. \quad & y_i(w^T x_i + b) \geq 1 - \xi_i\ , \quad i = 1, \dots, l \\ & \xi_i \geq 0, \quad i = 1, \dots, l \end{aligned} \tag{1}$$

In (1), ξ_i signifies the degree of violation for the i^{th} pattern. The cost function attempts to trade-off the objectives of maximizing the margin and minimizing the l_1 norm of such violations. The user parameter C controls this trade-off and is usually picked by cross-validation techniques that aim at optimizing the generalization properties of the learning machine. The learning machine resulting from (1) is referred to as a *soft margin classifier*. A *hard margin classifier* does not allow for constraint violations and is obtained (for a linearly separable problem) by setting $\xi_i = 0$ in (1) (equivalently by taking $C = \infty$.)

An alternative soft margin formulation, that is important for the geometric approach followed in [7] and this paper, considers the l_2 norm of the constraint violations in the cost. Namely, the SVM classifier is obtained by solving:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + \frac{\widetilde{C}}{2}\sum_{i=1}^{l}\xi_i^2 \\ s.t. \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \ , \ \ i = 1, \ldots, l \end{aligned} \tag{2}$$

The capabilities of linear SVMs can be greatly extended by using kernel induced feature spaces. Specifically, patterns vectors are first mapped to feature vectors $\phi_i \equiv \phi(x_i)$, where the function $\phi(\cdot)$ is such that inner products $\phi(x_i) \cdot \phi(x_j) \equiv K(x_i, x_j)$, i.e. they can be computed by means of a kernel function $K(\cdot,\cdot)$. Then linear classification is effected in the feature space. Most commonly used kernel functions other than the linear one: $K(x, y) = x^T y$, are the Gaussian or Radial Basis Function (RBF) kernel: $K(x, y) = \exp(-\|x - y\|^2/(2\sigma^2))$, and the polynomial kernel: $K(x, y) = (x^T y + 1)^d$, with $d > 0$ an integer.

The quadratic program (1) (with x_i replaced by $\phi(x_i)$ for Nonlinear SVMs) is typically tackled by solving its dual problems that has a simpler constraint structure. Thus the dual of (1) is given by:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j \ K(x_i, x_j) \\ s.t. \quad & \sum_{i=1}^{l}\alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0, \ \ i = 1, \ldots, l \end{aligned} \tag{3}$$

The dual problem for hard margin classification is obtained by setting $C = \infty$ in (3), i.e. by discarding the upper bound constraints on the dual variables α_i, $i = 1, \ldots, l$. Each α_i represents the Lagrange multiplier for the corresponding constraint (pattern) in the primal problem. Patterns with $\alpha_i > 0$ are the so-called *support vectors* and are the only patterns needed to determine the learning machine. Indeed, the optimal classifier is expressed as:

$$f(x) = sign(\sum_{i=1}^{l} y_i\alpha_i \ K(x, x_i) + b) \tag{4}$$

where b is computed from any support vector constraint since by the Karush-Kuhn-Tucker (KKT) optimality conditions such constraints must be satisfied with equality.

Note that since for the solution of the quadratic program (3) and the application of the resulting decision function (4) only inner products between feature vectors are required, the kernel implementation of inner products allows high dimensional (even infinite dimensional) feature spaces with a computational load independent of the dimension of the feature space.

Let us now consider the soft margin classification problem with quadratic violations (2). The latter can be equivalently transformed to an instance of a hard margin classification problem in an extended input space by considering patterns $\tilde{x}_i$ and hyperplane parameters $\tilde{w}$ and $\tilde{b}$, defined as follows.

$$\widetilde{w} = \begin{pmatrix} w \\ \sqrt{\widetilde{C}}\xi \end{pmatrix}; \quad \widetilde{x}_i = \begin{pmatrix} x_i \\ \frac{1}{\sqrt{\widetilde{C}}} y_i e_i \end{pmatrix}; \quad \widetilde{b} = b, \tag{5}$$

where $\xi \equiv [\xi_1, \ldots, \xi_l]^T$ and e_i denotes the i^{th} unit vector in $\mathbb{R}^l$. In the case of Nonlinear SVMs, feature vectors $\phi(x_i)$ rather than patterns x_i are augmented in (5) and we obtain extended feature vectors $\tilde{\phi}_i \equiv [\phi(x_i)^T \ \frac{1}{\sqrt{\tilde{C}}} y_i e_i^T]^T$. Inner products in this space can be computed from:

$$\tilde{\phi}_i^T \tilde{\phi}_j = \phi(x_i)^T\phi(x_j) + \frac{1}{\tilde{C}} y_i y_j e_i^T e_j = K(x_i, x_j) + \frac{1}{\tilde{C}}\delta_{ij}, \tag{6}$$

where $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ otherwise. Thus, from the viewpoint of the dual problem (3), the soft margin SVM problem with quadratic violations (1) is reduced to the hard margin problem with only a minor modification in the kernel function, namely by adding $1/\tilde{C}$ to the diagonal elements of the kernel matrix $K(x_i, x_j)$ as indicated by (6).

A. Hard Margin Classification as a Proximity Problem

The hard margin classification problem in the linear separable case has an interesting geometric interpretation as a minimum distance (*Proximity or Nearest Point*) problem [10], [7]. More specifically, consider the (convex) positive $\mathcal{P}^+ \equiv co(\{x_1^+, \ldots, x_{l^+}^+\})$ and negative $\mathcal{P}^- \equiv co(\{x_1^-, \ldots, x_{l^-}^-\})$ polytopes defined as the *convex hulls* of the positive and negative training patterns respectively. Then the maximum margin hyperplane SVM is found as the bisector of the minimum distance segment with end points $u \in \mathcal{P}^+$ and $v \in \mathcal{P}^-$. This observation is key to applying proximity algorithms to the solution of the hard margin SVM problem (1) with $C = \infty$ and also of the soft margin SVM problem with quadratic violations (2) after using transformation (5) discussed above. Specifically, it can be shown that (3), is equivalent to following proximity problem:

$$\min_{u\in\mathcal{P}^+, \ v\in\mathcal{P}^-} \|u - v\|. \tag{7}$$

Once the solution (u^*, v^*) of (7) has been found, the optimal solution (w^*, b^*) of the maximum margin classifier is obtained from:

$$w^* = \frac{2(u^* - v^*)}{\| u^* - v^* \|^2}; \quad b^* = \frac{\| v^* \|^2 - \| u^* \|^2}{\| u^* - v^* \|^2},$$

and the optimal margin is $M^* = 2/ \| w^* \| = \| u^* - v^* \| /2$ [7].

B. Review of Classical Proximity Algorithms

The problem of finding the minimum distance between two convex sets $\mathcal{P}^+$ and $\mathcal{P}^-$ is one with a long history. It is readily reduced to the problem of finding the minimum norm point of the difference set $\mathcal{D} \equiv \mathcal{P}^+ - \mathcal{P}^-$ defined as all points

$u - v$ with $u \in \mathcal{P}^+$ and $v \in \mathcal{P}^-$, which is also convex. For the latter problem a number of classical algorithms are available. In the following, we give a brief overview of three such algorithms, namely *Gilbert's algorithm* [11], the *Mitchell-Dem'yanov-Malozemov (MDM) algorithm* [12], and *Wolfe's algorithm* [13]. The approach in [7] is based on combining Gilbert's and MDM algorithms, which makes it a lot similar to the SMO algorithm conceived from a purely algebraic perspective. The objective of this paper is to investigate the potential of Wolfe's algorithm for solving the dual optimization problem (3) for hard margin or soft margin SVM's with quadratic violations.

Consider a bounded convex set $\mathcal{D}$ in a Hilbert space $\mathbb{H}$, which can be infinite-dimensional. Proximity algorithms are concerned with finding the nearest point to the origin, or equivalently the minimum norm point in $\mathcal{D}$. Specifically, they solve the following problem:

Given $\epsilon > 0$ and $0 < \rho < 1$, find a point $z^* \in \mathcal{D}$ such that either

$$\| z^* \| < \epsilon \tag{8}$$

or

$$\| z^* \| - \inf_{z \in \mathcal{D}} \|z\| < \rho \| z^* \| \tag{9}$$

Condition (8) implies the origin is in $\mathcal{D}$ within a given tolerance ϵ. In our application to SVMs, since $\mathcal{D}$ represents the difference of the polytopes defined by the training points (or corresponding feature points) in the positive and negative classes, stopping with (8) means that the two classes are not linearly separable. On the other hand condition (9) implies that the minimum norm point z^* is found with a relative precision specified by ρ. All of the three classical proximity algorithms produce sequences of points $z^k \in \mathcal{D}$, that satisfy (8) or (9) within a finite number of iterations. These points can be represented as a convex combination of a *finite* number of points of $\mathcal{D}$. In case of $\mathcal{D} = co(\mathcal{S})$ being the convex hull of a finite number of points $S = \{s_1, \ldots, s_L\}$ as it is the case in the proximity problem resulting from SVM optimization, we get

$$z^k = \sum_{i=1}^{L} \beta_i^k s_i, \quad \sum_{i=1}^{L} \beta_i^k = 1, \quad \beta_i^k \geq 0 \tag{10}$$

with possibly many of the β_i^k's being equal to zero.

At the k^{th} iteration, all three approaches require the *contact point* $g^k \in \mathcal{D}$ (in fact $g^k \in \mathcal{S}$) of the hyperplane having z^k as its normal with $\mathcal{D}$. The contact point g^k has the following important property:

$$\frac{\langle g^k,\ z\rangle}{\|z\|} \leq \inf_{z \in \mathcal{D}} \|z\| \leq \|z\| \tag{11}$$

for any $z \in \mathcal{D}$ and in particular $z \equiv z^k$. The notation $\langle \cdot, \cdot \rangle$ denotes as usual the inner product in $\mathbb{H}$. Since points in $\mathbb{H}$ are of the form $u - v$ with u, v feature vectors, we can compute $\langle u_i - v_i,\ u_j - v_j\rangle = K(u_i, u_j) - K(u_i, v_j) - K(v_i, u_j) + K(v_i, v_j)$. From (11), it follows that (9) is satisfied when

$$\|z^k\| - \frac{\langle g^k,\ z^k\rangle}{\|z^k\|} < \rho \|z^k\| \tag{12}$$

and (12) can be taken as a convenient stopping condition.

a) Gilbert's Algorithm: In this approach the next estimate z^{k+1} is found as the minimum norm point on the line segment with end points z^k and g^k. Since the latter is expressed as $z(\mu) = (1-\mu) z^k + \mu g^k$, $0 \leq \mu \leq 1$, we can easily find z^{k+1} by first considering $\mu^* = \frac{\langle z^k,\ z^k - g^k\rangle}{\|z^k - g^k\|^2}$ solving $\min_\mu z(\mu)$ with μ unrestricted, and then by taking $z^{k+1} = z^k + \mu^*(g^k - z^k)$ if $\mu^* < 1$, or $z^{k+1} = g^k$ if $\mu^* \geq 1$ (Note that $\mu^* > 0$ from (11).)

b) MDM Algorithm: The MDM algorithm utilizes besides z^k and g^k a third point m^k. Let I_k be the set of points in S with $\beta_i^k \neq 0$ in the representation (10) of z^k. Then m^k is defined as the element of $\mathcal{S}$ that satisfies

$$\langle z^k,\ m^k\rangle = \max_{i \in I_k} \langle z^k,\ s_i\rangle. \tag{13}$$

Thus m^k is the point in $\mathcal{S}$ that projects the farthest among the points participating in the representation of the current solution candidate z^k along the direction defined by z^k. Since in the representation of the minimum norm point all points must project the same, m^k can be thought as the worst point in the representation of z^k. In the MDM approach, the next estimate z^{k+1} is found as the minimum norm point in $\mathcal{D}$ and on the half line $z(\mu) \equiv z^k + \mu(g^k - m^k)$, $\mu > 0$. Let i_g, i_m be the indices of g^k, m^k respectively in $\mathcal{S}$, i.e. $g^k \equiv s_{i_g}$ and $m^k \equiv s_{i_m}$. Clearly, from (10),

$$z(\mu) = \beta_1^k s_1 + \ldots + (\beta_{i_m}^k - \mu) m^k + \ldots + (\beta_{i_g}^k + \mu) g^k + \ldots + \beta_m^k s_m,$$

and $\mu \leq \beta_{i_m}^k$ so that $z(\mu) \in \mathcal{D}$. Then z^{k+1} can be found by first considering $\mu^* = \frac{\langle z^k,\ m^k - g^k\rangle}{\|m^k - g^k\|^2} > 0$ such that $z(\mu^*)$ is the minimum norm point along the half line without restricting $z(\mu^*)$ being in $\mathcal{D}$, and next by taking $z^{k+1} = z^k + \min\{\mu^*,\ \beta_{i_m}^k\}(g^k - m^k)$.

c) Wolfe's Algorithm: A key concept that is made explicit in Wolfe's algorithm but is present in all algorithms solving proximity problems and thus SVM optimization problems is that of a *corral*. First, given a set $\mathcal{S}$, the *affine hull* $\mathit{aff}(\mathcal{S})$ of $\mathcal{S}$ is defined as the set of all affine combinations of elements in $\mathcal{S}$, i.e.

$$\mathit{aff}(\mathcal{S}) \equiv \left\{ z \middle/ z = \sum_{i=1}^{L} \beta_i s_i\ ,\ s_i \in \mathcal{S},\ \sum_{i=1}^{L} \beta_i = 1 \right\} \tag{14}$$

A finite set $\mathcal{S} = \{s_1, \ldots, s_L\}$ of points in $\mathbb{H}$ is said to be *affinely independent* if no point of $\mathcal{S}$ belongs to the affine hull of the remaining points. Also the *relative interior* $ri(\mathcal{S})$ of $\mathcal{S}$ is the interior of $\mathcal{S}$ with respect to $\mathit{aff}(\mathcal{S})$. Then, we have the following definition:

Definition A *corral* is a set of affinely independent points $\mathcal{Q} = \{q_1, \ldots q_m\} \in \mathbb{H}$ such that the nearest point in the affine hull *aff*(Q) is contained in the relative interior of the convex hull $co(\mathcal{Q})$.

A singleton is a corral. In case that the origin is outside $\mathcal{D}$, the minimum norm point z^* is in the relative interior of a *face* of $\mathcal{D}$ (i.e. the intersection of D and a supporting hyperplane.) This face is clearly a corral with z^* being its nearest point. It is uniquely determined and thus we will refer to it as the *optimal corral*. Note that the optimal corral is defined by the points in $\mathcal{S}$ with nonzero coefficients in the representation (10) for z^*. Also note that had the optimal corral $\mathcal{Q}^*$ been known, we could find z^* simply by discarding all points in $\mathcal{S}$ other than the vertices of $\mathcal{Q}^*$ and finding the minimum norm point in $\mathit{aff}(\mathcal{Q}^*)$. The latter problem amounts to solving a linear system of equation of dimension equal to the number of the vertices that define $\mathcal{Q}^*$. Knowledge of the optimal corral is equivalent with knowing the nonzero α's in problem (3).

In Wolfe's approach, z^k is the nearest point in a corral $\mathcal{Q}^k$ formed from points in $\mathcal{S}$. $\mathcal{Q}^k$ is an approximation of the optimal corral constructed by Wolfe's algorithm at the k^{th} iteration. Thus we can think of $\mathcal{Q}^k$ as being the main object of the algorithm with z^k obtained as the nearest point in $\mathcal{Q}^k$. The next corral $\mathcal{Q}^{k+1}$ is obtained as in Algorithm 1.

Algorithm 1 Corral Updating in Wolfe's Algorithm

```
Given: a corral $\mathcal{Q}^k = \{q_1, \dots q_m\} \subset \mathcal{S}$ together with $z^k$ the nearest point in $co(\mathcal{Q}^k)$ and a contact point $g^k \in \mathcal{S}$ such that $\langle g^k\ ,\ z^k\rangle < \langle z\ ,\ z^k\rangle$ for all $z \in \mathcal{S} = \{s_1, \dots, s_L\}$.
Step 1: Consider the set $\hat{\mathcal{Q}}^{k+1} = \mathcal{Q}^k \bigcup \{g^k\}$ obtained by appending the contact point $g^k$ to $\mathcal{Q}^k$.
Step 2: Compute $\hat{q}^{k+1}$ as the nearest point in $\mathit{aff}(\hat{\mathcal{Q}}^{k+1})$.
Step 3: If $\hat{q}^{k+1} \in ri(co(\hat{\mathcal{Q}}^{k+1}))$ then set $\mathcal{Q}^{k+1} \equiv \hat{\mathcal{Q}}^{k+1}$, $z^{k+1} \equiv \hat{q}^{k+1}$ and return.
Step 4: Otherwise compute the nearest point $\hat{z}^{k+1}$ in $[z^k,\ \hat{q}^{k+1}] \cap co(\hat{\mathcal{Q}}^{k+1})$, set $\hat{\mathcal{Q}}^{k+1}$ equal to the face of $\hat{\mathcal{Q}}^{k+1}$ that contains $\hat{z}^{k+1}$ and go back to step 2.
```

We now discuss in more detail the implementation of the steps in Algorithm 1. Step 2 requires solving the following optimization problem on a set $\mathcal{Q} = \{q_1, \dots, q_m, q_{m+1}\}$, where we identify $q_{m+1} \equiv g^k$. Namely:

$$\hat{q} = \arg\min_{\beta}\{\|q\|^2/q = \sum_{i=1}^{m+1} \beta_i q_i, \sum_{i=1}^{m+1} \beta_i = 1\}, \qquad (15)$$

where $\beta = [\beta_1, \dots, \beta_m, \beta_{m+1}]^T$. The necessary (here also sufficient) conditions for optimality give the system of linear equations

$$\left\{ \begin{array}{c} e^T\beta = 1 \\ e\rho + M\beta = 0 \end{array} \right\} \qquad (16)$$

where M is the matrix with elements $M_{ij} = \langle q_i, q_j\rangle$, the inner products between the q_i's computed using the kernel function and e identified here with $e(m+1) \equiv [1, \dots, 1]^T \in \mathbb{R}^{m+1}$. Also ρ is the Langrange multiplier corresponding to the equality constraint in (15). System (16) can be shown to be equivalent to:

$$\left\{ \begin{array}{c} (ee^T + M)\bar{\beta} = e \\ \beta = \bar{\beta}/e^T\bar{\beta}. \end{array} \right\} \qquad (17)$$

Furthermore, affine independence of $\mathcal{Q}$ implies that vectors $[1, q_i^T]^T$ are linearly independent and therefore the symmetric matrix $ee^T + M$ is positive definite and the solution of (17) is well defined. By employing the *Cholesky factorization* $R^TR = ee^T + M$ with R upper triangular, the system in (17) is efficiently solved via two triangular systems [13]. Moreover, since $\mathcal{Q}$ always changes by adding one point (Step 1) or deleting one point (Step 4), R can be updated at a low computational cost (see also Section III-B.)

In Step 3, $\hat{q}^{k+1}$ (identified with $\hat{q}$ in (15)) is in the relative interior of $co(\hat{\mathcal{Q}}^{k+1})$ (identified with $\mathcal{Q}$ in (15)) if and only if the corresponding weights $\beta_i > 0$ for all $i = 1, \dots, m+1$. In Step 4, we have $\hat{q}^{k+1} \notin ri(co(\hat{\mathcal{Q}}^{k+1}))$ and thus $\hat{q}^{k+1} = \sum_{i=1}^{m+1} \hat{\beta}_i^{k+1} q_i$, $\sum_{i=1}^{m+1} \hat{\beta}_i^{k+1} = 1$ and $\hat{\beta}_j^{k+1} \leq 0$ for some $j \in \{1, \dots, m+1\}$. We have $z^k \in co(\mathcal{Q})$ and the algorithm maintains a description $z^k = \sum_{i=1}^{m+1} \beta_i^k q_i$ with $\sum_{i=1}^{m+1} \beta_i^k = 1$ and $\beta_i^k \geq 0$, for all $i = 1, \dots, m+1$ (observe that $\beta_{m+1}^k = 0$.) Then points on the line segment $[z^k, \hat{q}^{k+1}]$ that are in $co(\hat{\mathcal{Q}}^{k+1})$, are represented as $\bar{q} = \sum_{i=1}^{m+1} \bar{\beta}_i q_i$ with $\bar{\beta}_i = (1-\mu)\beta_i^k + \mu\hat{\beta}_i^{k+1} \geq 0$, $0 \leq \mu \leq 1$, for all $i = 1, \dots, m+1$. The last condition requires that

$$0 \leq \mu \leq \min_{i \in \{1, \dots, m+1\}} \{\frac{\beta_i^k}{\beta_i^k - \hat{\beta}_i^{k+1}},\ \ \hat{\beta}_i^{k+1} \leq 0\}. \qquad (18)$$

Since the distance from the origin monotonically decreases along the line segment from z^k to $\hat{q}^{k+1}$, the solution is clearly obtained for μ equal to the upper bound in (18). Note that for this μ, some $\bar{\beta}_i = 0$ and the corresponding point gets eliminated from $\hat{\mathcal{Q}}^{k+1}$, which is then reduced to one of its faces towards $\hat{q}^{k+1}$.

III. Main Results

A. Application of Wolfe's Algorithm to the Hard Margin Classification Problem

As discussed in Section II, the SVM optimization problem (3) is transformed to the problem of finding the minimum distance between the positive and negative polytopes $\mathcal{P}^+$ and $\mathcal{P}^-$ respectively. The latter problem is equivalent with finding the minimum norm point in the difference polytope

$$\mathcal{D} \equiv \mathcal{P}^+ - \mathcal{P}^- \equiv \{z = u - v/u \in \mathcal{P}^+,\ \ v \in \mathcal{P}^-\}. \qquad (19)$$

We apply Wolfe's Algorithm to find $u^* \in \mathcal{P}^+$ and $v^* \in \mathcal{P}^-$, such $\|u^* - v^*\| = \min\{u - v/u \in \mathcal{P}^+,\ \ v \in \mathcal{P}^-\}$. We remark that in our implementation, we do not explicitly form $\mathcal{D}$ that can have as many points as $l^+ \cdot l^-$ with l^+ and l^- the numbers of positive and negative training points respectively, but we represent each point in $\mathcal{D}$ by its corresponding points in $\mathcal{P}^+$ and $\mathcal{P}^-$. We also work with points in the feature space that might be of infinite dimension. As we only need inner products between feature points $\phi(x_i)$ that are computed through kernel functions, we do not need the feature points themselves, and indeed, ϕ may be unknown. Thus in our algorithm, we represent the feature points $\phi(x_i)$ by their index i and use x_i in the original space in the kernel computations.

At each iteration Wolfe's algorithm maintains the corral $\mathcal{Q}^k$ and the minimum norm point z^k in $co(\mathcal{Q}^k)$. We keep a representation of $\mathcal{Q}^k$ as:

$$\mathcal{Q}^k \equiv \{(i_1^k, j_1^k), \dots, (i_{m_k}^k, j_{m_k}^k)\} \qquad (20)$$

where i_l^k, j_l^k are the indices of the points in $\mathcal{P}^+$, $\mathcal{P}^-$ respectively that define the vertices of $\mathcal{Q}^k$. Note that it is possible for some index i_l^k (or j_l^k) to be repeated if the corresponding vertex in $\mathcal{P}^+$ (or $\mathcal{P}^-$) participates in more than one vertices of $\mathcal{D}$, but every pair of indices in (20) is distinct. The unique indices i_l^k (j_l^k) induce sets $\mathcal{Q}_+^k \in \mathcal{P}^+$ and $\mathcal{Q}_-^k \in \mathcal{P}^-$ that we call positive and negative corral respectively. Also z^k is represented by its barycentric coordinates $\beta^k = [\beta_1^k, \beta_2^k, \ldots, \beta_{m_k}^k]^T$, $\beta_l^k \geq 0$, $\sum_{i=1}^{m_k} \beta_i^k = 1$, with respect to the corral vertices. We then easily obtain from β^k, the barycentric coordinates γ^k and δ^k of $u^k \in \mathcal{P}^+$ and $v^k \in \mathcal{P}^-$ with respect to the positive and negative corrals respectively where $z^k = u^k - v^k$. (However, note that u^k and v^k are not minimum norm points in $\mathcal{Q}_+^k$ and $\mathcal{Q}_-^k$, rather they define the estimate for the optimal hyperplane at the k^{th} iteration.)

In summary, our implementation employs two integer arrays indexing the vertices from $\mathcal{P}^+$ and $\mathcal{P}^-$ participating in forming the corral $\mathcal{Q}^k$ in (20) and the arrays β^k and γ^k, δ^k with the coordinates of z^k and u^k, v^k points defining the SVM hyperplane.

We next give details about the operations involved in Wolfe's Algorithm as applied to the SVM optimization problem.

1) Contact point Calculation: The contact point $g^k = \arg\min_{z\in\mathcal{D}} \langle z^k, z\rangle$. Since $\mathcal{D}$ is a polytope, i.e. $\mathcal{D} = co(\{s_1, \ldots, s_L\})$, g^k is one of the s_i's, the vertices of $\mathcal{D}$. Let $z^k = u^k - v^k$ and $z = p - r$ with $u^k, p \in \mathcal{P}^+$ and $v^k, r \in \mathcal{P}^-$. Then,

$$\begin{aligned}
\min_{z\in\mathcal{D}}\{\langle z^k, z\rangle\} &= \min_{p\in\mathcal{P}^+,\ r\in\mathcal{P}^-}\{\langle z^k, p-r\rangle\}\\
&= \min_{p\in\mathcal{P}^+}\{\langle z^k, p\rangle\} - \max_{r\in\mathcal{P}^-}\{\langle z^k, r\rangle\}\\
&= \min_{p\in\mathcal{P}^+}\{\langle u^k, p\rangle - \langle v^k, p\rangle\} - \max_{r\in\mathcal{P}^-}\{\langle u^k, r\rangle - \langle v^k, r\rangle\}\\
&= \min_{p\in\mathcal{P}^+}\{K(u^k, p) - K(v^k, p)\} - \max_{r\in\mathcal{P}^-}\{K(u^k, r) - K(v^k, r)\}. \quad (21)
\end{aligned}$$

The optimizations in (21) can be further expanded by noting that $u^k = \sum_{i=1}^{m_k^+} \gamma_i^k q_i^+$ and $v^k = \sum_{i=1}^{m_k^+} \delta_i^k q_i^-$ where q_i^+ and q_i^- are the vertices of the positive $\mathcal{Q}_+^k$ and negative $\mathcal{Q}_-^k$ corrals respectively (dependence of q_i^+ and q_i^- on the iteration index k is omitted for ease of notation.) Thus from (21), it can be seen that the contact point g^k can be identified by two searches over the vertices of the positive and negative polytopes, i.e. the training samples in class $\mathcal{C}^+$ and class $\mathcal{C}^-$ respectively.

2) Updating the Cholesky factor: In Step 2 of Algorithm 1, (see Section II-B) the Cholesky factor $R^T R = ee^T + M$ is used to solve efficiently system (17). R needs to be updated when a new point (such as g^k) is added to the corral, or when a corral point is eliminated as in Step 3 of Algorithm 1 to obtain the next corral.

In the following, we let $Q = [q_1, \ldots, q_m]$ denote the matrix with columns the vertices q_i defining the corral at the k^{th} iteration. Again we omit dependence on k for notational convenience. We remark that the q_i's being feature vectors can be infinite dimensional, but we only use inner products among such vectors that are computed through the kernel function. In particular, we have $M \equiv Q^T Q$. Let us first consider the case that a new point $q \in \mathcal{D}$ is added to the corral. Then $Q_{\text{new}} = [Q\ q]$ and if R_{new} denotes the updated Cholesky factor, we have

$$R_{\text{new}}^T R_{\text{new}} = e(m+1)e^T(m+1) + Q_{\text{new}}^T Q_{\text{new}} \qquad (22)$$

where $e(m) = [1, \ldots, 1]^T \in \mathbb{R}^m$. Consider the following form for R_{new}:

$$R_{\text{new}} = \begin{bmatrix} R & \eta \\ 0 & \rho_o \end{bmatrix} \qquad (23)$$

where η and ρ_o have to be found. Substituting (23) in (22), we obtain

$$\begin{bmatrix} R^T R & R^T \eta \\ \eta^T R & \rho_o^2 + \eta^T \eta \end{bmatrix} = \begin{bmatrix} e(m)e^T(m) + Q^T Q & e(m) + Q^T q \\ e^T(m) + q^T Q & 1 + q^T q \end{bmatrix}$$

and it follows

$$\begin{aligned}
\eta &= R^{-T}[\langle q_1, q\rangle, \ldots \langle q_m, q\rangle]^T \qquad (24)\\
\rho_o &= (\langle q, q\rangle - \eta^T \eta)^{-1/2}, \qquad (25)
\end{aligned}$$

where we replaced $Q^T q$ and $q^T q$ with the indicated inner products that can be computed via kernel evaluations. We remark that besides these kernel evaluations, the other major computation involved in updating the corral is the solution of the triangular system in (24).

Next we discuss the case of eliminating a point q_j from the corral. In this case we define $Q_{\text{new}} = [q_1, \ldots, q_{j-1}, q_{j+1}, \ldots, q_m]$ with obvious modifications if $j = 1$ or $j = m$. By introducing the permutation matric $E_j = [e_1, \ldots, e_{j-1}, e_{j+1}, \ldots, e_m, e_j]$, where e_i denotes the i^{th} column of the $m \times m$ identity matrix, we have $QE_j = [Q_{\text{new}},\ q_j]$. Then from $R^T R = e(m+1)e^T(m+1) + Q^T Q \Rightarrow E_j^T R^T R E_j = e(m+1)e^T(m+1) + E_j^T Q^T Q E_j$ we obtain

$$\tilde{R}^T \tilde{R} = e(m)e^T(m) + Q_{\text{new}}^T Q_{\text{new}}, \qquad (26)$$

where $\tilde{R}$ in (26) denotes the upper-left $(m-1)\times(m-1)$ block of R. By construction, $\tilde{R}$ is in upper Hessenberg form and the updated Cholesky factor R_{new} can be efficiently computed by using row operations, such as Givens rotations, to make $\tilde{R}$ upper triangular without affecting the right-hand-side of (26).

B. Computational Considerations

Kernel evaluations are among the most costly in this and any SVM algorithm. Thus we can improve the efficiency of the proposed algorithm considerably by caching the kernel products that are most likely to be reused. Our implementation of Wolfe's algorithm for solving the SVM optimization problem requires $o(ml + m^2)$ operations per iteration versus $o(l)$ for the SMO algorithm. Our algorithm is clearly computationally more expensive per iteration but it typically requires a much smaller number of iterations to converge than that required by the SMO algorithm. Therefore, our approach can indeed outperform the SMO algorithm, especially in problems with the final corral size being relatively small with respect to the sample size. This analysis is supported by the numerical experiments discussed in Section IV.

Data Set	Pattern Size	Sample Size	Kernel Type
Checkers	2	1000	Gaussian ($\sigma^2 = 0.1$)
Rand. Lin. Sep.	300	10000	Linear
Adult 3	123	3185	Gaussian ($\sigma^2 = 10$)

TABLE I

DATA SET PROPERTIES.

IV. NUMERICAL EXPERIMENTS

We next empirically evaluate and compare the proposed algorithm against the popular SMO algorithm on a number of standard test problems. Our Wolfe approach for solving the SVM classification problem was implemented in MATLAB. MATLAB programs usually run slower than compiled FORTRAN or C code, and to keep the comparison fair, we also implemented SMO in MATLAB, along the lines of [9] that incorporates the improvements in [8]. By implementing our own SMO routine, we were able to tap the variables necessary for our comparison, and perhaps more importantly, to adapt SMO to solve the SVM problem with quadratic violations that our proposed algorithm solves. Finally, our SMO implementation uses identical caching as the proposed Wolfe algorithm and no shrinking.

The experiments performed are summarized in Table I. In this table, we identify the data set used, the size of the patterns, the size of the training sample, the type of kernel used (Linear or Gaussian) and for the Gaussian kernel the variance σ^2 used. We remark that we used values that are commonly used in previous experiments employing the same data sets [5], [6], [7]. The *Checkers* data set considered in [4] (see also [7]) involves random two-dimensional points arranged in a 4×4 checkerboard pattern. This problem is separable in feature space when using a Gaussian kernel. The Random Linearly Separable data set was considered in [5], and is constructed by considering random binary 300-dimensional vectors with a 10% fraction of them in Class $\mathcal{C}^+$. The Adult 3 data set comes from the UCI repository: **ftp://ftp.ics.uci.edu/pub/machine-learning-databases**. The benchmark results were obtained by running our MATLAB code on a laptop computer with an Intel Mobile Pentium 4, 1.7Ghz CPU and 1Gb RAM, running Windows XP. We report in Table II total computation time in $secs$, number of iterations, total number of kernel evaluations, number of support vectors, and achieved margin for the Wolfe-SVM algorithm proposed here and the SMO algorithm modified as explained above to solve the soft margin SVM problem with quadratic violations. Table II also gives the value of the regularization constant $\tilde{C}$ used in each experiment. These experiments (and all other experiments performed but not reported here due to space limitations) point to the following conclusions. The total kernel computations are comparable in both algorithms, and decrease as $\tilde{C}$ increases. Our Wolfe algorithm uses in general much fewer iterations than SMO. However, the difference in the number of iterations required becomes more dramatic for higher values of $\tilde{C}$. However,

	Time (secs)	Iter.	Kernel Eval.	SV #	Margin
Data Set: Checkers ($\tilde{\mathbf{C}} = \mathbf{10}$)					
Wolfe-SVM	0.44	162	79053	163	0.0424
SMO-q	0.82	1758	80913	163	0.0424
Data Set: Random Linearly Separable 4 ($\tilde{\mathbf{C}} = \infty$)					
Wolfe-SVM	166.3	1143	13200003	300	0.1085
SMO-q	940.9	159618	11340003	307	0.1085
Data Set: Adult 3 ($\tilde{\mathbf{C}} = \mathbf{10}$)					
Wolfe-SVM	348.0	1694	5736188	1782	0.0106
SMO-q	132.1	13707	5755298	1782	0.0106
Data Set: Adult 3 ($\tilde{\mathbf{C}} = \mathbf{1000}$)					
Wolfe-SVM	362.9	1957	5532348	1275	0.0024
SMO-q	631.7	168060	4892163	1276	0.0024

TABLE II

PERFORMANCE COMPARISON OF WOLFE-SVM AND SMO ON STANDARD TEST DATA SETS.

SMO has the advantage over our algorithm for smaller values of $\tilde{C}$. The previous results suggest a strategy that uses the proposed algorithm for large values, and SMO for small values of the regularization parameter $\tilde{C}$.

REFERENCES

[1] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[2] B. Schölkopf and A. Smola, *Learning with Kernels*. MIT Press Cambridge, 2002.

[3] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," *Proc. 1997 IEEE Workshop on Neural Networks for Signal Processing*, pp. 276–285, 1997.

[4] L. Kaufman, "Solving the quadratic problem arising in support vector classification," in *Advances in Kernel Methods: Support Vector Machines*. Cambridge, MA, MIT Press: Eds. B. Schölkopf, C. Burges, A. Smola, 1998.

[5] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Machines*. Cambridge, MA, MIT Press: Eds. B. Schölkopf, C. Burges, A. Smola, 1998.

[6] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods: Support Vector Machines*. Cambridge, MA, MIT Press: Eds. B. Schölkopf, C. Burges, A. Smola, 1998.

[7] 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*, vol. 11, no. 1, pp. 124–136, 2000. [Online]. Available: citeseer.nj.nec.com/keerthi99fast.html

[8] ——, "Improvements to platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, pp. 637–649, 2001.

[9] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," `http://www.csie.ntu.edu.tw/cjlin`, December 2004.

[10] K. P. Bennett and E. J. Brendensteiner, "Duality and geometry in svm classifiers," in *Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 57–64. [Online]. Available: citeseer.nj.nec.com/bennett00duality.html

[11] E. G. Gilbert, "An iterative procedure for computing the minimum of a quadratic form on a convex set," *SIAM J. Contr.*, vol. 4, pp. 61–79, 1966.

[12] B. F. Mitchell, V. F. Dem'yanov, and V. N. Malozemov, "Finding the point of a polyhedron closest to the origin," *SIAM J. Contr.*, vol. 12, pp. 19–26, 1974.

[13] P. Wolfe, "Finding the nearest point in a polytope," *Mathematical Programming*, vol. 11, pp. 128–149, 1976.