TECHNICAL REPORT

**No.** CCLS-11-03

**Title: Conic SMO**

**Authors:** Ilia Vovsha

## Abstract

We derive an SMO-like algorithm for the optimization problem arising from the Second Order Cone Programming formulation of support vector machines (SVMs) for classification. Due to the square root term in the objective of the form, we cannot easily compute the location of the extrema using Newton's formula. Instead we modify the analytic solution proposed for SMO (Platt, 1999), to account for the particular objective function and partial derivatives. We then establish some theoretical properties of the algorithm based on the results of (Bordes et al., 2005), and develop practical working set selection similar to (Fan et al., 2005). In addition, we propose an efficient implementation, and compare the convergence rate of Conic-SMO and SMO on synthetic data.


**Keywords:** support vector machines, second order cone program, SMO, working set selection

## 1. Introduction

Suppose we have a set of $\ell$ observations $(x_1, \ldots, x_\ell)$ with corresponding labels $(y_1, \ldots, y_\ell)$ where, $x_i \in \Re^n$, $y_i \in \{-1, 1\}$. Then to find the $\Delta$-margin hyperplane for the linearly non-separable case, we could solve the following *Second Order Cone Program* (SOCP) (Vapnik, 1998, chapter 10.2):

$$
\begin{aligned}
(P_\Delta) \qquad & \min_{\mathbf{w},b,\xi} \sum_{i=1}^{\ell} \xi_i \\
& \forall i,\ y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geq 1-\xi_i,\ \xi_i \geq 0 \\
& (\mathbf{w}\cdot\mathbf{w}) \leq \frac{1}{\Delta^2}
\end{aligned} \tag{1.1}
$$

The standard approach for solving $P_\Delta$ is to construct the Lagrangian, and find the saddle point:

$$
\begin{aligned}
(D_\Delta) \qquad & \max_{\boldsymbol{\alpha}} D_\Delta(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{\Delta}\sqrt{\sum_{i,j=1}^{\ell} \alpha_i\alpha_j y_i y_j (\mathbf{x}_i\cdot\mathbf{x}_j)} \\
& \sum_{i=1}^{\ell} y_i\alpha_i = 0 \\
& \forall i,\ 0 \leq \alpha_i \leq 1
\end{aligned} \tag{1.2}
$$

If we denote $A = \frac{1}{\Delta}$ $(A > 0)$, generalize to the nonlinear case, and assume that summands are over the indices $\{1, \ldots, \ell\}$, unless specified otherwise, then we obtain the optimization problem:

$$
\begin{aligned}
(D_A) \qquad & \max_{\boldsymbol{\alpha}} D_A(\boldsymbol{\alpha}) = \sum_{i} \alpha_i - A\sqrt{\sum_{i,j} \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)} \\
& \sum_{i} y_i\alpha_i = 0,\ 0 \leq \alpha_i \leq 1
\end{aligned} \tag{1.3}
$$

where vectors are boldface, K is a *positive semi-definite* (p.s.d) kernel matrix, and variable bounds hold $\forall i$. The solution vector $\boldsymbol{\alpha}^0$ of $D_A$ defines the generalized optimal hyperplane

(1.4), and the threshold $b$ is chosen to satisfy the *Karush-Kuhn-Tucker* (KKT) conditions (1.5).

$$f_A(\mathbf{x}) = \frac{A}{\sqrt{\sum_{i,j} \alpha_i^0 \alpha_j^0 y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)}} \sum_i \alpha_i^0 y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{1.4}$$

$$\forall t,\ \alpha_t^0 \left( \frac{A}{\sqrt{\sum_{i,j} \alpha_i^0 \alpha_j^0 y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)}} \sum_i \alpha_i^0 y_i K(\mathbf{x}_t, \mathbf{x}_i) + b \right) = 0 \tag{1.5}$$

Having obtained $\boldsymbol{\alpha}^0$ (assuming $\boldsymbol{\alpha}^0 \neq 0$) for some $A$, we could define the parameter,

$$C = \frac{A}{\sqrt{\sum_{i,j} \alpha_i^0 \alpha_j^0 y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)}} \tag{1.6}$$

and transform $D_A$ to the normalized quadratic form $D_C$. In this case, the solutions of both problems $D_A, D_C$, coincide (Vapnik, 1998, chapter 10.2).

$$(D_C) \qquad \max_{\boldsymbol{\alpha}} D_C(\boldsymbol{\alpha}) = \sum_i \alpha_i - C \cdot \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\sum_i y_i \alpha_i = 0,\ 0 \leq \alpha_i \leq 1 \tag{1.7}$$

$$f_C(\mathbf{x}) = C \cdot \sum_{i=1}^{\ell} \alpha_i^0 y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{1.8}$$

The normalized form (1.7), and the corresponding decision function (1.8), can be obtained directly from the standard dual (1.9) (Cortes and Vapnik, 1995), by factoring the parameter C out of the constraints and into the objective, and absorbing the constant (1/2) into the parameter.

$$\max_{\boldsymbol{\alpha}} D_C(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\sum_i y_i \alpha_i = 0,\ 0 \leq \alpha_i \leq C \tag{1.9}$$

### 1.1 Motivation

Considerable effort has been devoted to the implementation of efficient optimization methods for the *Quadratic Programming* (QP) dual (1.9) (Bottou and Lin, 2007). However, we were not able to identify a discussion in the literature of decomposition methods for either the specific SOCP $D_A$, or more general SOCP problems (Lobo et al., 1998; Alizadeh and Goldfarb, 2003). Nevertheless, decomposition is a compelling approach for large scale optimization problems with a square-root term in the objective, since the techinques for QPs are well developed and can be adapted. Furthermore, for the SVM form, the parameter $A$ has a concrete meaning (unlike the parameter C which lacks this intuition). The VC dimension (h) is bounded by a quantity that depends on the dimensionality (n) of the data,

the radius of the data sphere (R), and the margin ($\Delta$) of the separating hyperplane, which is inversly related to $A$ (Vapnik, 1998, theorem 10.3):

$$h \leq \min\left(\left[\frac{R^2}{\Delta^2}\right], n\right) + 1 \tag{1.10}$$

Since the radius can be computed (Schlkopf et al., 1995; Tax and Duin, 1999), solving $D_A$ (rather than $D_C$) should be preferable at least theoretically. Whether the bound on $A$ is too loose in practice, remains an open question.

There exists another SVM form, named $\nu$-SVM (Schölkopf et al., 2000; Chang and Lin, 2001a), which has a parameter ($\nu$) with a clear interpretation. However, while $\nu$ is a bound on the fraction of margin errors and number of support vectors, unlike $A$, it is only indirectly related to the VC dimension.

### 1.2 Outline

In this report we derive an SMO-like algorithm for the particular, and equivalent to (1.3), SOCP formulation:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \; & D_A(\boldsymbol{\alpha}) = \textstyle\sum_i \alpha_i y_i - A\sqrt{\sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)} \\ & \textstyle\sum_i \alpha_i = 0,\; L_i \leq \alpha_i \leq H_i \\ & L_i = \min(0, y_i),\; H_i = \max(0, y_i) \end{aligned} \tag{1.11}$$

Due to the square root term in the objective, we cannot easily compute the location of the extrema using Newton's formula. Therefore, it is necessary to modify the analytic solution proposed for SMO (Platt, 1999), and the derivation is not as straightforward as it is for the QP-SVM form (Bottou and Lin, 2007). We give the gory details in Section 2. In Section 3, we establish some theoretical properties of the algorithm based on the results of Bordes et al. (2005). We then develop practical working set selection methods similarly to Fan et al. (2005) in Section 4, and give essential implementation details in Section 5. Since the SOCP and QP formulations of SVM are equivalent for a proper choice of parameters, it is possible to compare the convergence rate of Conic-SMO and SMO empirically. We provide a numerical comparison in Section 6. Finally, we summarize our work in Section 7.

## 2. Conic SMO

We solve (1.11) analytically for two Lagrange multipliers. As in the original SMO derivation (Platt, 1999), using the linear constraint $\sum_i \alpha_i = 0$, we first express the objective function as a function of a single Lagrange multiplier, and a constant term which does not depend on the chosen multipliers (Section 2.1). By defining suitable constants, we transform the derived objective function (and its partial derivatives) into a simpler form, which is easier to analyze (Section 2.2). We then consider the possible cases and show how the objective is maximized for each (Section 2.3). Due to their exhaustive nature, most of the algebraic manipulations are detailed in the Appendix.

### 2.1 Function of a Single Lagrange Multiplier

Without loss of generality (w.l.o.g), let the two Lagrange multipliers be $\{\alpha_1, \alpha_2\}$. Then $D_A(\boldsymbol{\alpha})$, can be expressed as (see derivation in Appendix A.1):

$$D_A(\alpha_1, \alpha_2, \boldsymbol{\alpha}_{const}) = \alpha_1 y_1 + \alpha_2 y_2 + \boldsymbol{\alpha}_{const}^T \mathbf{y}_{const} - A\sqrt{D^*(\alpha_1, \alpha_2) + D^*_{const}} \tag{2.1}$$

$$D^*(\alpha_1, \alpha_2) = K_{11}\alpha_1^2 + K_{22}\alpha_2^2 + 2K_{12}\alpha_1\alpha_2 + 2v_1\alpha_1 + 2v_2\alpha_2 \tag{2.2}$$

$$v_i = \sum_{j=3}^{n} a_j K_{ij} \tag{2.3}$$

where $K_{ij} = K(i, j)$, and $D^*_{const}$, $\boldsymbol{\alpha}^T_{const}$, $\mathbf{y}_{const}$, are terms that do not depend on $\{\alpha_1, \alpha_2\}$.

Each Conic-SMO iteration will find the maximum of $D_A(\boldsymbol{\alpha})$ along the line defined by the constraint $\sum_i \alpha_i = 0$. Let,

$$\Delta\alpha = \alpha_1 + \alpha_2 = \alpha_1^{old} + \alpha_2^{old} \tag{2.4}$$

where the superscript 'old' indicates values at the end of the previous iteration, and due to the constraint, $\Delta\alpha$ is constant for a given iteration. The objective function can now be expressed in terms of $\{\alpha_1\}$ alone (see Appendix A.2):

$$D_A(\alpha_1, \boldsymbol{\alpha}^{old}) = (y_1 - y_2)(\alpha_1 - \alpha_1^{old}) + (\boldsymbol{\alpha}^{old})^T \mathbf{y} - A\sqrt{D^*(\alpha_1) + D^*_{const}} \tag{2.5}$$

$$\begin{aligned} D^*(\alpha_1) &= K_{11}\alpha_1^2 + K_{22}(\Delta\alpha - \alpha_1)^2 + 2K_{12}\alpha_1(\Delta\alpha - \alpha_1) + 2v_1\alpha_1 + 2v_2(\Delta\alpha - \alpha_1) \\ &= \eta\alpha_1^2 - 2\alpha_1(\alpha_1^{old}\eta + (U_2 - U_1)) + \Delta\alpha(\Delta\alpha K_{22} + 2v_2) \end{aligned} \tag{2.6}$$

$$U_i = \sum_j \alpha_j^{old} K_{ij} \tag{2.7}$$

$$\eta = K_{11} + K_{22} - 2K_{12} \tag{2.8}$$

### 2.2 Objective in Simpler Form

Since we need to compute the first and second partial derivatives of $D_A(\boldsymbol{\alpha})$, it is desirable to express the objective function (2.5)–(2.8) in a simpler form first. To keep the derivation as clear as possible, we shall assume that the matrix $K$ is *positive definite* (p.d) i.e., $\eta > 0$. Substitute $D(\alpha_1)$ for $D_A(\alpha_1, \boldsymbol{\alpha}^{old})$, and let $\{C_1, C_2, C_3, \delta_y, \eta^*\}$ be constants defined as follows:

$$C_1 = a_1^{old} + \frac{(U_2 - U_1)}{\eta} \tag{2.9}$$

$$C_2 = \frac{\Delta\alpha(\Delta\alpha K_{22} + 2v_2) + D^*_{const}}{\eta} \tag{2.10}$$

$$C_3 = C_2 - C_1^2 \tag{2.11}$$

$$\delta_y = y_1 - y_2 \tag{2.12}$$

$$\eta^* = \eta - (\frac{\delta_y}{A})^2 \tag{2.13}$$

Then the objective function (2.5), and the partial derivatives of $D(\alpha_1)$ with respect to $\alpha_1$, which might not be defined for $\alpha_1 = 0$, can be expressed in the form (2.14)–(2.17) (see Appendix A.3):

$$D^*(\alpha_1) + D^*_{const} = \eta((\alpha_1 - C_1)^2 + C_3) \tag{2.14}$$

$$D(\alpha_1) = \delta_y(\alpha_1 - \alpha_1^{old}) + (\boldsymbol{\alpha}^{old})^T\mathbf{y} - A\sqrt{\eta}\sqrt{(\alpha_1 - C_1)^2 + C_3} \tag{2.15}$$

$$\frac{\partial D(\alpha_1)}{\partial \alpha_1} = \delta_y - A\sqrt{\eta}\frac{(\alpha_1 - C_1)}{\sqrt{(\alpha_1 - C_1)^2 + C_3}} \tag{2.16}$$

$$\frac{\partial^2 D(\alpha_1)}{\partial \alpha_1^2} = \frac{-A\sqrt{\eta}}{\sqrt{(\alpha_1 - C_1)^2 + C_3}}(1 - \frac{(\alpha_1 - C_1)^2}{(\alpha_1 - C_1)^2 + C_3}) \tag{2.17}$$

If $\exists \alpha' : \frac{\partial D(\alpha')}{\partial \alpha_1} = 0$, then from (2.13),(2.16), and (2.17),

$$\frac{\partial^2 D(\alpha')}{\partial \alpha_1^2} = \frac{-A\sqrt{\eta}}{\sqrt{(\alpha' - C_1)^2 + C_3}} \cdot \frac{\eta^*}{\eta} \tag{2.18}$$

Since we have assumed that $K$ is p.d, the quadratic quantity under the square root must be nonnegative. Therefore, $\frac{\partial D(\alpha_1)}{\partial \alpha_1}$ may be undefined at a single point only ($\boldsymbol{\alpha} = 0$). In other words, if $C_3 = 0$, then $\frac{\partial D(\alpha_1)}{\partial \alpha_1}$ is undefined at $\alpha_1 = C_1 = 0$. This case can occur if exactly two alphas are nonzero, or if we are at zero. Otherwise $C_3 > 0$, and $\frac{\partial D(\alpha_1)}{\partial \alpha_1}$ is defined $\forall \alpha_1$ (see A.4).

### 2.3 Analytic Solution

We consider how the objective is maximized for each case in Appendix B, and summarize the results in Table 1. The solution in the table is valid for all points except zero, which we discuss below. As in SMO, the unconstrained maximum should be clipped to the ends of the line segment.

Table 1: Conic-SMO analytic solution

| Case Summary | $\eta^*$ | $C_3 > 0$ | $C_3 = 0$ |
|---|---|---|---|
| $y_1 = y_2$ | $\eta^* = \eta$ | $\alpha_1^{new} = C_1$ | N/A |
| $y_1 \neq y_2$ | $\eta^* > 0$ | $\alpha_1^{new} = C_1 + y_1 \frac{2}{A}\sqrt{\frac{C_3}{\eta^*}}$ | $\alpha_1^{new} = 0$ |
| | $\eta^* \leq 0$ | $\alpha_1^{new} = y_1$ | $\alpha_1^{new} = y_1$ |

Suppose $\boldsymbol{\alpha} = 0\,\{\forall i : \alpha_i^{old} = 0\}$ for some iteration while solving $D_A$. By definition $C_1 = C_3 = 0$, and to make progress at zero, $y_1 \neq y_2$. If w.l.o.g we let $y_1 = 1$, then the objective function (2.15) can be simplified to:

$$\begin{aligned} D_A(\boldsymbol{\alpha}) &= (y_1 - y_2)\alpha_1 - A\sqrt{\eta}\,|\alpha_1| \\ &= (2 - A\sqrt{\eta})\alpha_1 \end{aligned} \tag{2.19}$$

Suppose $\eta^* > 0$. It follows that $(2 - A\sqrt{\eta}) < 0$, and from (2.19) or Table 1, it is clear that $D_A(\boldsymbol{\alpha})$ is maximized at $\alpha_1^{new} = 0$.

$$\eta^* > 0 \Longrightarrow (2 - A\sqrt{\eta}) < 0 \Longrightarrow \max D_A(\boldsymbol{\alpha}) = D_A(0) = 0 \tag{2.20}$$

In other words, if $A$ is sufficiently large, and we ascend to zero at some iteration (or choose zero as our initial point), it is possible that no pair of indices $\{i, j\}$ would allow us to increase the objective function value, since with respect to any maximally sparse direction, $D_A(\boldsymbol{\alpha})$ is already maximized. For example, consider a radial basis function (RBF) kernel: then typically $(\forall i)\, K_{ii} = 1$, $(\forall j \neq i)\, 0 < K_{ij} < 1$, and therefore $1 < \eta < 2$ for all pairs $\{i, j\}$. It follows that an $A > 2$ is sufficiently large.

The obvious approach to bypass this problem, is to modify the constraints of $D_A$ to exclude zero from the feasible set. From a practical standpoint, we prefer to define an ad-hoc gradient and take an 'artificial' step, if the initial point is zero. We can ensure that we do not return to zero at any subsequent iteration with an appropriate *working set selection* (WSS) method. We discuss the details in Sections 4, 5.

## 3. Theoretical Properties

Let $\mathcal{F}_{socp-svm}$ denote the convex polytope defined by the constraints of $D_A$, where the zero point is excluded, and assume that the kernel matrix $K$ is p.d. Then $D_A(\boldsymbol{\alpha})$ is a concave, twice differentiable with continuous derivatives function, defined on $\mathcal{F}_{socp-svm}$ where,

$$\boldsymbol{\alpha} \in \mathcal{F}_{socp-svm} \iff \begin{array}{l} \sum_i \alpha_i = 0 \\ L_i \leq \alpha_i \leq H_i \\ \boldsymbol{\alpha} \neq 0 \end{array} \tag{3.1}$$

Therefore, the theoretical results presented in the Appendix of Bordes et al. (2005) are applicable, and we can establish various convergence properties of the Conic-SMO algorithm. In particular, we can specify a *finite witness family* (Bordes et al., 2005, definition 5) for $\mathcal{F}_{socp-svm}$, and derive the optimality conditions. If we define the gradient of $D_A$, and the sets of indices $\{I_{up}(\alpha), I_{low}(\alpha)\}$, then assertion (iii) in Theorem 2 yields the necessary and sufficient "violating pair" optimality criterion (Keerthi et al., 2001; Fan et al., 2005) for the SOCP-SVM problem.

$$\mathbf{g} = \nabla D_A(\boldsymbol{\alpha}) = \mathbf{y} - A\frac{K\boldsymbol{\alpha}}{\sqrt{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}} \tag{3.2}$$

$$g_k = \nabla D_A(\boldsymbol{\alpha})_k \tag{3.3}$$

$$I_{up}(\alpha) = \{\, t : \alpha_t < H_t\}, \quad I_{low}(\alpha) = \{\, t : \alpha_t > L_t\} \tag{3.4}$$

**Proposition 1** *Let $(e_1, \ldots, e_n)$ be the canonical basis of $\Re^n$. Set $\mathcal{U}_s = \{e_i - e_j,\, i \neq j\}$ is a (finite) witness family for the convex set $\mathcal{F}_{socp-svm}$ defined by* (3.1)*. This set is used in the Conic-SMO algorithm.*

**Proof** See Definition 5 and Proposition 7 in (Bordes et al., 2005). ■

**Theorem 2** *For $\{\mathcal{U}_s, \mathcal{F}_{socp-svm}\}$, the following assertions are equivalent:*
*i) $\boldsymbol{\alpha}$ is a solution of $D_A$.*
*ii) $\forall u \in \mathcal{U}_s$, $\max\{D_A(\boldsymbol{\alpha}+\lambda u), \boldsymbol{\alpha}+\lambda u \in \mathcal{F}_{socp-svm}\} \leq D_A(\boldsymbol{\alpha})$.*
*iii) $\forall u \in \mathcal{U}_s$ at $\boldsymbol{\alpha}$, $u^T \nabla D_A(\boldsymbol{\alpha}) \leq 0$.*

**Proof** Theorem 1 is a special case of Theorem 6 in (Bordes et al., 2005). ■

**Definition 3 (Violating Pair)** *If $i \in I_{up}(\alpha)$, $j \in I_{low}(\alpha)$, and $g_i - g_j > 0$, then $\{i, j\}$ is a "violating pair".*

**Proposition 4** *A feasible $\boldsymbol{\alpha} \in \mathcal{F}_{socp-svm}$ is optimal for $D_A$, if and only if,*

$$\forall i, j \in \{1...n\}^2,\ \alpha_i < H_i \text{ and } \alpha_j > L_j \Longrightarrow \nabla D_A(\boldsymbol{\alpha})_i - \nabla D_A(\boldsymbol{\alpha})_j \leq 0$$

*Stated differently, $\boldsymbol{\alpha} \in \mathcal{F}_{socp-svm}$ is optimal, if and only if, there does not exist a "violating pair" at $\boldsymbol{\alpha}$.*

In practice, we define $\{M_{up}(\boldsymbol{\alpha}), m_{low}(\boldsymbol{\alpha})\}$, pre-specify a small tolerance $\tau > 0$, and as in SMO, check whether the optimality condition is satisfied approximately:

$$M_{up}(\boldsymbol{\alpha}) = \max_{\mathbf{i \in I_{up}}(\alpha)} g_i, \quad m_{low}(\boldsymbol{\alpha}) = \min_{\mathbf{j \in I_{low}}(\alpha)} g_j \tag{3.5}$$

$$M_{up}(\boldsymbol{\alpha}) - m_{low}(\boldsymbol{\alpha}) \leq \tau \tag{3.6}$$

Therefore, Conic-SMO fits the framework of the *Approximate Stochastic Witness Direction Search* algorithm (Bordes et al., 2005), and the following convergence results are valid:

**Definition 5** *A vector $u \in \Re^n$ is called a $\kappa\tau$-violating direction at point $\boldsymbol{\alpha}$, if $\max\{\lambda \geq 0, \boldsymbol{\alpha}+\lambda u \in \mathcal{F}_{socp-svm}\} > \kappa$, and $u^T \nabla D_A(\boldsymbol{\alpha}) > \tau$.*

**Proposition 6** *Since $\mathcal{U}_s$ is a finite witness family for $\mathcal{F}_{socp-svm}$, any sequence of points $\{\boldsymbol{\alpha}^k\}$ generated by the Conic-SMO algorithm (with any rule used to select the working set at each iteration), converges ("$\kappa\tau$-approximately") to some point $\boldsymbol{\alpha}'$ (not necessarily the optimal $\boldsymbol{\alpha}^*$) after a finite number of steps.*

**Proof** Proposition 3 is a special case of Proposition 16 in (Bordes et al., 2005). ■

**Theorem 7 (Asymptotic convergence)** *Assume that at every iteration $k$, Conic-SMO selects a working set $\{i, j\}$ which defines a $\kappa\tau$-violating direction at $\boldsymbol{\alpha}^{k-1}$, if such direction exists. Then the sequence $\{\boldsymbol{\alpha}^k\}$ generated by Conic-SMO converges ("$\kappa\tau$-approximately") to the optimal $\boldsymbol{\alpha}^*$ for $\{D_A, \mathcal{F}_{socp-svm}\}$ after a finite number of steps.*

**Proof** Theorem 2 is a special case of Theorem 17 in (Bordes et al., 2005). ■

## 4. Working Set Selection

A practical implementation of the Conic-SMO algorithm requires an effective strategy for choosing the working set at each iteration. Under the same assumptions as in Section 3, we adopt the notation of Fan et al. (2005), and propose two selection methods. The first method is identical to WSS 1 (first proposed by Keerthi et al., 2001) in (Fan et al., 2005) It can be derived through the KKT conditions, where the gradient and $\{M_{up}(\boldsymbol{\alpha}),\, m_{low}(\boldsymbol{\alpha})\}$ are defined in (3.2)–(3.5).

---

**WSS 1 (Maximal violating pair)**

1. Select

   $i \in \arg M_{up}(\boldsymbol{\alpha}),\ j \in \arg m_{low}(\boldsymbol{\alpha})$
2. Return $B = \{i, j\}$.

---

A more effective selection method can be obtained if we consider using second order information. The goal is to solve (4.1) where $\mathbf{e}$ is a vector of ones, and the hessian matrix $\nabla^2 D_A(\boldsymbol{\alpha})$ is defined in (4.2).

$$
\begin{aligned}
(WS_2) \qquad \max_{\mathbf{d}_B} F(B) &= \frac{1}{2}\mathbf{d}_B^T \nabla^2 D_A(\boldsymbol{\alpha})_{BB}\mathbf{d}_B + \nabla D_A(\boldsymbol{\alpha})_B^T \mathbf{d}_B \\
&\text{subject to:} \\
&\mathbf{e}^T \mathbf{d}_B = 0 \\
&\alpha_t = L_t,\ t \in B \Longrightarrow d_t \geq 0 \\
&\alpha_t = H_t,\ t \in B \Longrightarrow d_t \leq 0
\end{aligned} \tag{4.1}
$$

$$
\nabla^2 D_A(\boldsymbol{\alpha}) = -\frac{A}{\sqrt{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}}\left(K - \frac{(K\boldsymbol{\alpha})(K\boldsymbol{\alpha})^T}{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}\right) \tag{4.2}
$$

Similar to Theorem 3 in (Fan et al., 2005), Theorem 8 below shows that one could efficiently solve $WS_2$. Based on this theorem, we can derive the selection method WSS 2.

**Theorem 8** *If $B = \{i, j\}$ is a violating pair, and $K$ is p.d, then the optimal objective value of $WS_2$, which depends on $a_{ij}$, is:*

$$
\begin{aligned}
&(1) \quad a_{ij} > 0 \Longrightarrow \frac{1}{2}\frac{\sqrt{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}}{A} \cdot \frac{b_{ij}^2}{a_{ij}} \\
&(2) \quad a_{ij} = 0 \Longrightarrow \infty
\end{aligned}
$$

*where $a_{ij}, b_{ij}$ are defined in* (4.6)–(4.7).

**Proof** Since $|B| = 2$,

$$
\mathbf{d}_B = [d_i, d_j]^T = [d_i, -d_i]^T \tag{4.3}
$$

from (2.7),(2.8),(3.3), (4.2) and (4.3)

$$
\mathbf{d}_B^T \nabla^2 D_A(\boldsymbol{\alpha})_{BB}\mathbf{d}_B = -\frac{A}{\sqrt{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}}\left(\eta_{ij} - \frac{(U_j - U_i)^2}{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}\right) d_i^2 \tag{4.4}
$$

$$
\nabla D_A(\boldsymbol{\alpha})_B^T \mathbf{d}_B = (g_i - g_j) d_i \tag{4.5}
$$

and if we define $a_{ij}, b_{ij}$, then $F(B)$ can be expressed as (4.8):

$$a_{ij} = \eta_{ij} - \frac{(U_j - U_i)^2}{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}} \tag{4.6}$$

$$b_{ij} = (g_i - g_j) \tag{4.7}$$

$$\begin{aligned} F(B) &= -\frac{1}{2}\frac{A}{\sqrt{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}}(\eta_{ij} - \frac{(U_j - U_i)^2}{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}})d_i^2 + (g_i - g_j)d_i \\ &= -\frac{1}{2}\frac{A}{\sqrt{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}} a_{ij} d_i^2 + b_{ij} d_i \end{aligned} \tag{4.8}$$

By definition $b_{ij} > 0$, and we can show (A.5) that

$$\begin{aligned} C_3 > 0 &\Longrightarrow a_{ij} > 0 \\ C_3 = 0 &\Longrightarrow a_{ij} = 0 \end{aligned}$$

where $C_3$ is a constant defined in (2.9)–(2.11) which depens on the chosen pair $\{i, j\}$. Hence w.l.o.g $d_i \geq 0$, and the optimal objective value of $(WS_2)$ follows from the maximum of (4.8):

$$\begin{aligned} a_{ij} > 0 &\Longrightarrow d_i = \frac{\sqrt{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}}{A}\frac{b_{ij}}{a_{ij}} \\ &\Longrightarrow \max_{\mathbf{d}_B} F(B) = \frac{1}{2}\frac{\sqrt{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}}{A}\frac{b_{ij}^2}{a_{ij}} \end{aligned} \tag{4.9}$$

$$\begin{aligned} a_{ij} = 0 &\Longrightarrow d_i = \infty \\ &\Longrightarrow \max_{\mathbf{d}_B} F(B) = \infty \end{aligned} \tag{4.10}$$

■

---

**WSS 2 (Second order information)**

1. Select

   $i \in \arg M_{up}(\boldsymbol{\alpha})$

2. Consider (4.9), where $a_{it}, b_{it}$ are defined in (4.6)–(4.7), and select

   $j \in \arg\max_t \{\frac{b_{it}^2}{a_{it}} \,|\, t \in I_{low}(\alpha),\, g_i > g_t,\, a_{it} \neq 0\}$

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

---

At the end of Section 2.2 we mentioned that the case $C_3 = 0$ ($a_{ij} = 0$) can occur if we are at zero, or if exactly two alphas $\{i, j\}$ are non-zero, and make up the working set i.e., $B = \{i, j\}$. If the initial point is zero, we take the 'artificial' step described in Section 5. By requiring $a_{it} \neq 0$ in step 2 of WSS 2, we guarantee that a violating pair is not considered unless there is at least one more non-zero alpha. Thus we do not return to zero at any iteration. WSS 2 is efficient since $a_{ij}$ can be computed from the gradient (see Section 5), and as in WSS 2 for SMO (Fan et al., 2005), we need to check only $O(l)$ possible pairs to select $\{i, j\}$.

Since $D_A(\boldsymbol{\alpha})$ is not a quadratic function, it is interesting to consider higher order information. We could use more than two terms of the Taylor series to approximate $D_A(\boldsymbol{\alpha})$. However, this would require computing the tensor product of the n-dimensional cube of derivatives $\nabla^n D_A(\boldsymbol{\alpha})$ with the search direction $\mathbf{d}_B$. For example, to obtain the third term of the Taylor series, we need to compute the 3D cube $\nabla^3 D_A(\boldsymbol{\alpha})$. Moreover, Theorem 8 holds only for a specific $F(B)$. When $F(B)$ is no longer a quadratic (cubic for the third order case), it is necessary to find the maximum of a more complicated function.

## 5. Implementation Details

We describe the pseudo-code of the Conic-SMO algorithm (Algorithm 1). Several aspects of the implementation demand particular attention:

- To take the initial step, we define an 'artificial' gradient at zero ($\mathbf{g} = \nabla D_A(\boldsymbol{\alpha}) = \mathbf{y}$).
- We use WSS 2 for SMO (Fan et al., 2005) to select the pair of indices at the first iteration (step 2 of the algorithm). For all violating pairs at zero, $g_i - g_j = 2$. Index $i$ (where $i \in I_{up}(\alpha)$) can be selected randomly, while to select index $j$, we can check for the minimum denominator in $\frac{g_i - g_j}{K_{ii}+K_{jj}-2K_{ij}}$, since the numerator is constant.
- In step 3, we update the alphas according to the default case $\eta^* \leq 0$. If $\eta^* > 0$, the update is 'artificial' since the objective value would decrease. Note that it cannot decrease on any of the subsequent iterations.
- In order to update the gradient (3.2) efficiently, we maintain an additional vector $\mathbf{U}$ (defined in (2.7), $\mathbf{U} = K\boldsymbol{\alpha}$), and a constant $aka = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$.
- A single iteration of Conic-SMO (steps 4-5) is more expensive than that of SMO. However, the additional cost is not significant, since we can compute the terms in steps 5a,5c efficiently (A.6).
- WSS 2 in step 4 is also efficient since $a_{ij}$ depends on the terms $\{U_i, U_j, aka\}$ which are available. It can also be computed from the gradient since $\frac{(U_j - U_i)^2}{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}} = (g_i - g_j - y_i + y_j)^2$.

## 6. Empirical Comparison

Since the SOCP and QP formulations of SVM, (1.3) and (1.7) respectively, are equivalent for a proper choice of parameters, we can compare the convergence rate of Conic-SMO and SMO numerically. Solving the QP problem $D_C$ using an interior point method is considerably faster than solving the corresponding SOCP problem $D_A$, assuming the same accuracy is required. Although the interior point methods converge to the solution in a similar number of iterations, each iteration for $D_A$ is computationally more expensive (see Appendix C). However, an efficient implementation of Conic-SMO, should mitigate this discrepancy.

In Table 2, we compare the Conic-SMO and SMO algorithms for various sample sizes, and $C$ ($A$) parameter values. Both methods are implemented without shrinking, and using WSS 2 with the tolerance fixed to $\tau = 10^{-3}$. We use an RBF kernel (with a different sigma

**Algorithm 1** Conic-SMO

1) Set $\boldsymbol{\alpha} \longleftarrow \mathbf{0}$ and initialize the ('artificial') gradient.
   $\forall k \in \{1, \ldots, \ell\}$, $\alpha_k \longleftarrow 0$, $U_k \longleftarrow 0$, $g_k \longleftarrow y_k$
2) Choose a $\tau$-violating pair $(i, j)$ according to WSS 2 for SMO.
   $i \longleftarrow \arg M_{up}(\boldsymbol{\alpha})$
   $j \longleftarrow \arg\min_j \{K_{ii} + K_{jj} - 2K_{ij}\}$
3) Take an artificial step.
   $\alpha_i \longleftarrow 1$, $\alpha_j \longleftarrow -1$ (update alphas)
   $\forall k$, $U_k \longleftarrow U_k + K_{ik} - K_{jk}$
   $\forall k$, $g_k \longleftarrow g_k - A \cdot \frac{U_k}{\sqrt{K_{ii}+K_{jj}-2K_{ij}}}$ (update gradient)
4) Choose a $\tau$-violating pair $(i, j)$ according to WSS 2 above. Stop if no such pair exists.
5) Take a step in the direction defined by $(i, j)$.
   5a) Compute $a_i^{new}$ from Table 2.
   5b) $\lambda \longleftarrow \min\{a_i^{new} - a_i,\, H_i - a_i,\, a_j - L_j\}$
   5c) Update term $\{aka\}$.
   5d) $\alpha_i \longleftarrow \alpha_i + \lambda$, $\alpha_j \longleftarrow \alpha_j - \lambda$ (update alphas)
   5e) $\forall k$, $U_k \longleftarrow U_k + \lambda K_{ik} - \lambda K_{jk}$
   $\forall k$, $g_k \longleftarrow y_k - A \cdot \frac{U_k}{\sqrt{aka}}$ (update gradient)
6) Return to step (4).

parameter for each sample size), on synthetic data generated according to a chess-board pattern. This is a two class problem, where points generated inside dark squares belong to one class, and the rest belong to the second class.

Table 2: SMO vs. Conic-SMO (convergence rate)

| # | C | A | SMO | Conic-SMO | $\|\Delta\boldsymbol{\alpha}^0\|_2$ | $\Delta D(\boldsymbol{\alpha}^0)$ |
|---|---|---|---|---|---|---|
| 1000 | 1e03 | 75.10 | 2760 | 3920 | 0.03 | 0 |
| 1000 | 1e04 | 648.7 | 1.12e05 | 1.38e05 | 0.08 | 0 |
| 2000 | 1e04 | 798.3 | 4.16e05 | 3.83e05 | 0.09 | 0 |
| 3000 | 1e04 | 970.7 | 3.11e05 | 3.18e05 | 0.04 | 1e-04 |
| 4000 | 1e04 | 1105 | 6.59e05 | 6.91e05 | 0.09 | 1e-04 |

The results in the table are obtained as follows: for each sample size and C parameter value, we execute the SMO algorithm as specified above. Having obtained the solution vector $\boldsymbol{\alpha}^0$ of $D_C$ we compute the parameter $A = C \cdot \sqrt{\sum_{i,j} \alpha_i^0 \alpha_j^0 y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)}$. We then solve the corresponding problem $D_A$ using Conic-SMO, and record the number of iterations the algorithms needed to converge in columns 4 and 5. To compare the solutions, we compute the norm of the difference between the solution vectors $\Delta\boldsymbol{\alpha}^0 = \boldsymbol{\alpha}_C^0 - \boldsymbol{\alpha}_A^0$, and the difference (rounded off to four decimal places) in the objective value $\Delta D(\boldsymbol{\alpha}^0) = D_C(\boldsymbol{\alpha}^0) - D_A(\boldsymbol{\alpha}^0)$ in columns 6 and 7 respectively. Suppose SMO needs x iterations to converge for some run. If conic SMO requires more iterations (which is often the case), we compute the difference in columns 6-7 using the vector obtained after x iterations rather than the solution vector. The numbers in columns 4-7 are averaged over 5 runs, each with a different random sample (but identical parameters and sample size). To give some intuition about the parameter A (which is different for each run), we record the value for one run in column 3.

The SMO method suffers from slow convergence since a large portion of iterations are spent for achieving the final digit of the required accuracy (Chang and Lin, 2001b). We observe that this issue is exacerbated for the Conic-SMO algorithm. However, the results in columns 6-7 suggest that Conic-SMO is competitive with SMO if we reduce the required accuracy. In addition, the running time per iteration (not shown) is only marginally greater, indicating that the implementation detailed in section 5 is efficient.

## 7. Conclusion

We have derived an SMO-like algorithm for the SOCP formulation of SVM. Due to the structure of the objective function and the gradient, we had to obtain the solution analytically, and treat the zero point as a special case. However, relying on existing results for SMO, it was straightforward to establish asymptotic convergence and derive an effective working set selection for Conic-SMO. As mentioned at the end of Section 4, it would be interesting to try a WSS for Conic-SMO with higher order information.

We have also provided an implementation of the algorithm in Section 5. Empirical evidence suggests that this implementation is efficient, and significantly reduces the discrepancy in the running time observed when the interior point methods are compared.

Furthermore, Table 2 demonstrates that Conic-SMO is competitive with SMO, if we do not insist on the final digit of the required accuracy (or set a lower stopping tolerance). Nevertheless, we emphasize that the motivation for solving the SOCP form is the theoretical relation between the parameter $A$ and the VC dimension. From a computational perspective, the QP form is perhaps preferable. Finally, SOCP problems arise in a variety of applications, and hence a decomposition method for the specific SVM problem may be useful as a blueprint for more general forms.

## Acknowledgments

This work was funded by NSF grant IIS-0916200. The author would like to thank Vladimir Vapnik and Dmitry Pechyony for discussing the probem, and Haimonti Dutta for thoroughly reviewing the report.

## Appendix A. Algebraic Derivations

Let "(def)" denote definition of constant/variable.

### A.1 Section 2.1, equations 2.1-2.3

$$
\begin{aligned}
(def)\ & c = 3:n,\ \alpha_c = [\alpha_3 \dots \alpha_n]^T \\
& \boldsymbol{\alpha}^T K \boldsymbol{\alpha} = \alpha_c^T K_{cc} \alpha_c + 2\alpha_1 \sum_{j=3}^{n} \alpha_j K_{1,j} + 2\alpha_2 \sum_{j=3}^{n} \alpha_j K_{2,j} + \alpha_1^2 K_{11} + 2\alpha_1\alpha_2 K_{12} + \alpha_2^2 K_{22} \\
(def)\ & D^*_{const} = \alpha_c^T K_{cc} \alpha_c = \sum_{i=3}^{n} \alpha_i \sum_{j=3}^{n} \alpha_j K_{i,j} \\
(def)\ & v_i = \sum_{j=3}^{n} \alpha_j K_{i,j} \\
& \boldsymbol{\alpha}^T K \boldsymbol{\alpha} = D^*_{const} + K_{11}\alpha_1^2 + K_{22}\alpha_2^2 + 2K_{12}\alpha_1\alpha_2 + 2v_1\alpha_1 + 2v_2\alpha_2 \\
(def)\ & D^*(\alpha_1, \alpha_2) = K_{11}\alpha_1^2 + K_{22}\alpha_2^2 + 2K_{12}\alpha_1\alpha_2 + 2v_1\alpha_1 + 2v_2\alpha_2
\end{aligned} \tag{A.1}
$$

### A.2 Section 2.1, eq. 2.5-2.8

$$
\begin{aligned}
(def)\ \Delta\alpha &= \alpha_1 + \alpha_2 \\
\alpha_2 &= \Delta\alpha - \alpha_1 \\
(def)\ D^*(\alpha_1) &= K_{11}\alpha_1^2 + K_{22}(\Delta\alpha - \alpha_1)^2 + 2K_{12}\alpha_1(\Delta\alpha - \alpha_1) + 2v_1\alpha_1 + 2v_2(\Delta\alpha - \alpha_1) \\
&= (K_{11} + K_{22} - 2K_{12})\alpha_1^2 + 2\Delta\alpha\alpha_1(K_{12} - K_{22}) + 2(v_1 - v_2)\alpha_1 \\
&\quad + \Delta\alpha\Delta\alpha K_{22} + 2\Delta\alpha v_2 \\
(def)\ \eta &= K_{11} + K_{22} - 2K_{12}
\end{aligned}
$$

$$
\begin{aligned}
D^*(\alpha_1) &= \eta\alpha_1^2 + 2\Delta\alpha\alpha_1(K_{12} - K_{22}) + 2(v_1 - v_2)\alpha_1 + \Delta\alpha\Delta\alpha K_{22} + 2\Delta\alpha v_2 \\
&= \eta\alpha_1^2 + 2\alpha_1[\Delta\alpha(K_{12} - K_{22}) + (v_1 - v_2)] + \Delta\alpha\Delta\alpha K_{22} + 2\Delta\alpha v_2 \\
&= \eta\alpha_1^2 - 2\alpha_1[(v_2 - v_1) + \Delta\alpha(K_{22} - K_{12})] + \Delta\alpha[\Delta\alpha K_{22} + 2v_2]
\end{aligned}
$$

$$
\begin{aligned}
(def)\ U_i &= \sum_j \alpha_j^{old} K_{i,j} \\
v_i &= U_i - \alpha_1^{old} K_{i,1} - \alpha_2^{old} K_{i,2} \\
v_2 - v_1 &= U_2 - U_1 - \alpha_1^{old} K_{12} - \alpha_2^{old} K_{22} + \alpha_1^{old} K_{11} + \alpha_2^{old} K_{12} \\
&= U_2 - U_1 + \alpha_1^{old}(K_{11} + K_{22} - 2K_{12}) \\
&\quad + \alpha_1^{old} K_{12} - \alpha_1^{old} K_{22} - \alpha_2^{old} K_{22} + \alpha_2^{old} K_{12} \\
&= U_2 - U_1 + \alpha_1^{old}\eta + (K_{12} - K_{22})(\alpha_1^{old} + \alpha_2^{old}) \\
&= U_2 - U_1 + \alpha_1^{old}\eta + \Delta\alpha(K_{12} - K_{22}) \\
(def)\ r_1 &= (v_2 - v_1) + \Delta\alpha(K_{22} - K_{12}) \\
&= U_2 - U_1 + \alpha_1^{old}\eta + \Delta\alpha(K_{22} - K_{12}) + \Delta\alpha(K_{12} - K_{22}) \\
&= U_2 - U_1 + \alpha_1^{old}\eta \\
D^*(\alpha_1) &= \eta\alpha_1^2 - 2\alpha_1(U_2 - U_1 + \alpha_1^{old}\eta) + \Delta\alpha[\Delta\alpha K_{22} + 2v_2] \\
&= \eta\alpha_1^2 - 2\alpha_1[\alpha_1^{old}\eta + (U_2 - U_1)] + \Delta\alpha[\Delta\alpha K_{22} + 2v_2]
\end{aligned} \tag{A.2}
$$

### A.3 Section 2.2, eq. 2.9-2.18

$$
\begin{aligned}
&Assume:\ \eta > 0 \\
D^*(\alpha_1) + D^*_{const} &= \eta\alpha_1^2 - 2\alpha_1[\alpha_1^{old}\eta + (U_2 - U_1)] + \Delta\alpha[\Delta\alpha K_{22} + 2v_2] + D^*_{const} \\
&= \eta[\alpha_1^2 - 2\alpha_1(\alpha_1^{old} + \frac{(U_2 - U_1)}{\eta}) + \frac{\Delta\alpha[\Delta\alpha K_{22} + 2v_2] + D^*_{const}}{\eta}] \\
(def)\ C_1 &= \alpha_1^{old} + \frac{(U_2 - U_1)}{\eta} \\
(def)\ C_2 &= \frac{\Delta\alpha[\Delta\alpha K_{22} + 2v_2] + D^*_{const}}{\eta} \\
(def)\ C_3 &= C_2 - C_1^2
\end{aligned}
$$

$$
\begin{aligned}
D^*(\alpha_1) + D^*_{const} &= \eta(\alpha_1^2 - 2\alpha_1 C_1 + C_2) \\
&= \eta((\alpha_1 - C_1)^2 + C_3) \\
\sum_i \alpha_i y_i &= \alpha_1 y_1 + \alpha_2 y_2 + \boldsymbol{\alpha}^T_{const}\mathbf{y}_{const} \\
&= \alpha_1 y_1 + (\Delta\alpha - \alpha_1) y_2 + \boldsymbol{\alpha}^T_{const}\mathbf{y}_{const} \\
&= \alpha_1 y_1 + (\alpha_1^{old} + \alpha_2^{old} - \alpha_1) y_2 + \boldsymbol{\alpha}^T_{const}\mathbf{y}_{const} \\
&= \alpha_1 y_1 - \alpha_1 y_2 + (y_2 - y_1)\alpha_1^{old} + (\boldsymbol{\alpha}^{old})^T\mathbf{y} \\
&= (y_1 - y_2)(\alpha_1 - \alpha_1^{old}) + (\boldsymbol{\alpha}^{old})^T\mathbf{y}
\end{aligned}
$$

$$
\begin{aligned}
(def)\ \delta_y &= y_1 - y_2 \\
D(\alpha_1, \boldsymbol{\alpha}^{old}) &= \delta_y(\alpha_1 - \alpha_1^{old}) + (\boldsymbol{\alpha}^{old})^T\mathbf{y} - A\sqrt{D^*(\alpha_1) + D^*_{const}} \\
&= \delta_y(\alpha_1 - \alpha_1^{old}) + (\boldsymbol{\alpha}^{old})^T\mathbf{y} - A\sqrt{\eta}\sqrt{(\alpha_1 - C_1)^2 + C_3} \\
\frac{\partial D(\alpha_1)}{\partial \alpha_1} &= \delta_y - \frac{1}{2}A\sqrt{\eta}\frac{2(\alpha_1 - C_1)}{\sqrt{(\alpha_1 - C_1)^2 + C_3}} \\
&= \delta_y - A\sqrt{\eta}\frac{(\alpha_1 - C_1)}{\sqrt{(\alpha_1 - C_1)^2 + C_3}} \\
\frac{\partial^2 D(\alpha_1)}{\partial \alpha_1^2} &= -A\sqrt{\eta}[\frac{1}{\sqrt{(\alpha_1 - C_1)^2 + C_3}} - \frac{1}{2}\frac{2(\alpha_1 - C_1)^2}{[(\alpha_1 - C_1)^2 + C_3]^{3/2}} \\
&= -A\sqrt{\eta}\frac{1}{\sqrt{(\alpha_1 - C_1)^2 + C_3}}[1 - \frac{(\alpha_1 - C_1)^2}{(\alpha_1 - C_1)^2 + C_3}] \\
\frac{\partial D(\alpha')}{\partial \alpha_1} &= 0 \Longrightarrow \\
&\quad \delta_y - A\sqrt{\eta}\frac{(\alpha_1 - C_1)}{\sqrt{(\alpha_1 - C_1)^2 + C_3}} = 0 \\
&\quad A\sqrt{\eta}\frac{(\alpha_1 - C_1)}{\sqrt{(\alpha_1 - C_1)^2 + C_3}} = \delta_y \\
&\quad \frac{(\alpha_1 - C_1)}{\sqrt{(\alpha_1 - C_1)^2 + C_3}} = \frac{\delta_y}{A\sqrt{\eta}} \\
\frac{\partial^2 D(\alpha')}{\partial \alpha_1^2} &= -A\sqrt{\eta}\frac{1}{\sqrt{(\alpha_1 - C_1)^2 + C_3}}[1 - (\frac{\delta_y}{A\sqrt{\eta}})^2] \\
&= -A\sqrt{\eta}\frac{1}{\sqrt{(\alpha_1 - C_1)^2 + C_3}}[1 - \frac{1}{\eta}(\frac{\delta_y}{A})^2] \\
&= -A\sqrt{\eta}\frac{1}{\sqrt{(\alpha_1 - C_1)^2 + C_3}}\frac{[\eta - (\frac{\delta_y}{A})^2]}{\eta} \\
(def)\ \eta^* &= \eta - (\frac{\delta_t}{A})^2 \\
\frac{\partial^2 D(\alpha')}{\partial \alpha_1^2} &= -A\sqrt{\eta}\frac{1}{\sqrt{(\alpha_1 - C_1)^2 + C_3}}\frac{\eta^*}{\eta}
\end{aligned}
\tag{A.3}
$$

### A.4 Section 2.2, last paragraph

$$
\begin{aligned}
\boldsymbol{\alpha}^T K \boldsymbol{\alpha} &= \eta((\alpha_1 - C_1)^2 + C_3) \\
\frac{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}{\eta} &= (\alpha_1 - C_1)^2 + C_3 \\
\forall \boldsymbol{\alpha},\, \boldsymbol{\alpha}^T K \boldsymbol{\alpha} \geq 0 &\Longrightarrow \forall \alpha_1,\, [(\alpha_1 - C_1)^2 + C_3] \geq 0 \\
&\Longrightarrow \text{One (double) root or none} \\
(def)\ F(a_1) &= (\alpha_1 - C_1)^2 + C_3 \\
F(a_1) = 0 &\Longleftrightarrow \alpha_1 = C_1 \pm \sqrt{-C_3} \\
C_3 > 0 &\Longrightarrow \text{no roots} \\
C_3 = 0 &\Longrightarrow \text{one root} \Longrightarrow C_1 = 0
\end{aligned}
\tag{A.4}
$$

### A.5 Section 4, Theorem 3

$$
\begin{aligned}
&Assume:\ \boldsymbol{\alpha} \neq 0,\, \eta > 0 \\
&(1)\ \boldsymbol{\alpha}^T K \boldsymbol{\alpha} = \eta((\alpha_i^{old} - C_1)^2 + C_3) \\
&(2)\ C_1 = \alpha_i^{old} + \frac{(U_j - U_i)}{\eta} \\
&(3)\ a_{ij} = \eta - \frac{(U_j - U_i)^2}{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}} \\
\text{From (1)-(3),}\quad C_3 &= \frac{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}{\eta} - (\alpha_i^{old} - C_1)^2 \\
&= \frac{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}{\eta} - (\alpha_i^{old} - \alpha_i^{old} - \frac{(U_j - U_i)}{\eta})^2 \\
&= \frac{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}{\eta} - \frac{(U_j - U_i)^2}{\eta^2} \\
&= \frac{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}{\eta^2}[\eta - \frac{(U_j - U_i)^2}{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}] \\
&= \frac{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}{\eta^2} a_{ij} \\
C_3 > 0 &\Longrightarrow a_{ij} > 0 \\
C_3 = 0 &\Longrightarrow a_{ij} = 0
\end{aligned}
\tag{A.5}
$$

### A.6 Section 5, algorithm steps 5a, 5c

We describe the pseudo-code to compute $a_i^{new}$ efficiently according to Table 1. The relevant terms are defined in equations (2.8)–(2.13). We actually compute $\lambda^* = a_i^{new} - a_i$.

**Algorithm 2** Pseudocode

$\eta = K_{ii} + K_{jj} - 2K_{ij}$
$dU = U_j - U_i$
$C_3 = \frac{\boldsymbol{\alpha}^T K \boldsymbol{\alpha}}{\eta} - (\frac{U_j - U_i}{\eta})^2 = \frac{aka}{\eta} - (\frac{dU}{\eta})^2$
$C_1 = a_i + \frac{(U_j - U_i)}{\eta} = a_i + \frac{dU}{\eta}$
**if** $y_i = y_j$ **then**
  $\lambda^* = a_i^{new} - a_i = \frac{dU}{\eta}$
**else**
  $\eta^* = \eta - (\frac{2}{A})^2$
  **if** $\eta^* \leq 0$ **then**
    $\lambda^* = y_i - a_i$
  **else**
    $\lambda^* = \frac{dU}{\eta} + (y_i \cdot \frac{2}{A} \cdot \sqrt{\frac{C_3}{\eta^*}})$
  **end if**
**end if**

We can update the term '$aka$' from $\{\lambda, \eta, dU, C_3\}$, where $\lambda$ is obtained in step 5b.

$$\begin{aligned} aka &= (\boldsymbol{\alpha}^{new})^T K \boldsymbol{\alpha}^{new} \\ &= \eta \cdot [(a_i^{new} - C_1)^2 + C_3] \\ &= \eta \cdot [(a_i^{new} - a_i - \frac{dU}{\eta})^2 + C_3] \\ &= \eta \cdot [(\lambda - \frac{dU}{\eta})^2 + C_3] \end{aligned} \tag{A.6}$$

## Appendix B. Analytic Solution Cases

We consider the cases $\delta_y = 0$ and $\delta_y \neq 0$ separately:

**Case 1** $\delta_y = 0$

Since $\eta^* = \eta$, the objective function and partial derivatives simplify to:

$$\begin{aligned} D(\alpha_1) &= (\alpha^{old})^T \mathbf{y} - A\sqrt{\eta}\sqrt{(\alpha_1 - C_1)^2 + C_3} \\ \frac{\partial D(\alpha_1)}{\partial \alpha_1} &= -A\sqrt{\eta}\frac{(\alpha_1 - C_1)}{\sqrt{(\alpha_1 - C_1)^2 + C_3}} \\ \exists \alpha' : \frac{\partial D(\alpha')}{\partial \alpha_1} = 0 \Longrightarrow \frac{\partial^2 D(\alpha')}{\partial \alpha_1^2} &= \frac{-A\sqrt{\eta}}{\sqrt{(\alpha' - C_1)^2 + C_3}} \end{aligned}$$

**Case 1.1** $C_3 > 0$

Since $\frac{\partial D(\alpha_1)}{\partial \alpha_1}$ is defined for $\forall \alpha_1$, the extremum of the objective function is at $\alpha_1 = C_1$. It is a maximum since $\frac{\partial^2 D(C_1)}{\partial \alpha_1^2} < 0$.

**Case 1.2** $C_3 = 0$

This case cannot occur.

**Case 2** $\delta_y \neq 0$

The objective function and partial derivatives cannot be simplified, and unlike case #1, the derivatives now depend on $\{\delta_y, \eta^*\}$.

**Case 2.1** $C_3 = 0$

$C_1 = 0$ by definition, and the objective can be simplified to:

$$\begin{aligned} D(\alpha_1) &= \delta_y(\alpha_1 - \alpha_1^{old}) + (\alpha^{old})^T\mathbf{y} - A\sqrt{\eta}\sqrt{(\alpha_1 - C_1)^2 + C_3} \\ &= 2y_1(\alpha_1 - \alpha_2^{old}) + (\alpha^{old})^T\mathbf{y} - A\sqrt{\eta}\,|\alpha_1 - C_1| \\ &= 2y_1\alpha_1 - A\sqrt{\eta}\,|\alpha_1| \\ &= |\alpha_1|\,(2 - A\sqrt{\eta}) \end{aligned}$$

$$\eta^* = 0 \Longrightarrow (2 - A\sqrt{\eta} = 0) \Longrightarrow \max D(\alpha_1) : |\alpha_1| \geq 0$$

$$\eta^* > 0 \Longrightarrow (2 - A\sqrt{\eta} < 0) \Longrightarrow \max D(\alpha_1) : |\alpha_1| = 0$$

$$\eta^* < 0 \Longrightarrow (2 - A\sqrt{\eta} > 0) \Longrightarrow \max D(\alpha_1) : |\alpha_1| = 1$$

$\eta^* = 0$ is an interesting case which doesn't occur in SMO. We choose to replace the inequality with the bound i.e. $|\alpha_1| \geq 0$ with $|\alpha_1| = 1$.

**Case 2.2** $C_3 > 0$

The extremum of the objective function depends on the sign of $\delta_y$. We consider several cases:

$$\delta_y > 0,\ \alpha_1 \leq C_1 \Longrightarrow \frac{\partial D(\alpha_1)}{\partial \alpha_1} > 0$$

$$\delta_y < 0,\ \alpha_1 \geq C_1 \Longrightarrow \frac{\partial D(\alpha_1)}{\partial \alpha_1} > 0$$

$$\frac{\partial D(\alpha_1)}{\partial \alpha_1} = 0 \Longrightarrow$$

$$\delta_y = A\sqrt{\eta}\frac{(\alpha_1 - C_1)}{\sqrt{(\alpha_1 - C_1)^2 + C_3}}$$

$$(\frac{\delta_t}{A})^2 = \eta\frac{(\alpha_1 - C_1)^2}{(\alpha_1 - C_1)^2 + C_3}$$

$$[\eta - (\frac{\delta_t}{A})^2](\alpha_1 - C_1)^2 - (\frac{\delta_t}{A})^2 C_3 = 0$$

$$\eta^*(\alpha_1 - C_1)^2 - (\frac{\delta_t}{A})^2 C_3 = 0$$

**Case 2.2.a** $\eta^* \leq 0$

$$\delta_y > 0,\ \alpha_1 > C_1 \Longrightarrow \frac{\partial D(\alpha_1)}{\partial \alpha_1} > 0$$

$$\delta_y < 0,\ \alpha_1 < C_1 \Longrightarrow \frac{\partial D(\alpha_1)}{\partial \alpha_1} < 0$$

Therefore,

$$\delta_y > 0 \Longrightarrow \alpha_1^{new} = H_1 = y_1 = 1$$
$$\delta_y < 0 \Longrightarrow \alpha_1^{new} = L_1 = y_1 = -1$$

**Case 2.2.b** $\eta^* \geq 0$

$$(\alpha_1 - C_1)^2 = \frac{1}{\eta^*}(\frac{\delta_y}{A})^2 C_3$$
$$\alpha_1 - C_1 = \sqrt{\frac{1}{\eta^*}(\frac{\delta_y}{A})^2 C_3}$$
$$\alpha_1 = C_1 \pm \frac{\delta_y}{A}\sqrt{\frac{C_3}{\eta^*}}$$

Therefore,

$$\delta_y > 0 \Longrightarrow \alpha_1^{new} = C_1 + \frac{\delta_y}{A}\sqrt{\frac{C_3}{\eta^*}}$$
$$\delta_y < 0 \Longrightarrow \alpha_1^{new} = C_1 - \frac{\delta_y}{A}\sqrt{\frac{C_3}{\eta^*}}$$

We can replace $\pm\delta_y = 2y_1$ and get the table entry.

## Appendix C. SOCP vs. QP Comparison

We compare the SOCP and QP formulations of SVM using a state of the art numerical package (MOSEK) implementing interior point methods. We use synthetic data with an RBF kernel for all rows in the table. Columns 4 and 6 in the table give the number of iterations for each interior point method, columns 5 and 7 are the running time in seconds. The accuracy and test error are the same for both forms and hence not shown.

Table 3: SOCP vs. QP

| | | | QP | | SOCP | |
|---|---|---|---|---|---|---|
| # | C | A | # ITER | TIME | # ITER | TIME |
| 200 | 137 | 50 | 12 | 0.05 | 14 | 0.34 |
| 200 | 5542 | 250 | 15 | 0.06 | 16 | 0.38 |
| 400 | 349 | 100 | 17 | 0.34 | 15 | 2.76 |
| 400 | 9315 | 400 | 17 | 0.35 | 17 | 3.18 |
| 800 | 2316 | 300 | 22 | 2.53 | 22 | 32.5 |
| 1600 | 2306 | 500 | 26 | 19.7 | 21 | 245 |
| 2500 | 11656 | 1000 | 30 | 78.4 | 30 | 1307 |

## References

F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95:3–51, 2003.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

L. Bottou and C-J. Lin. Support vector machine solvers. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

C-C. Chang and C-J. Lin. Training v-support vector classifiers: Theory and algorithms. *Neural Computation*, 13(9):2119–2147, 2001a.

C-C. Chang and C-J. Lin. *LIBSVM: a Library for Support Vector Machines*, 2001b. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

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

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Comput.*, 13:637–649, March 2001.

M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1-3):193–228, 1998.

J. C. Platt. *Fast training of support vector machines using sequential minimal optimization*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

B. Schlkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery and Data Mining, Menlo Park*, pages 252–257. AAAI Press, 1995.

D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999.

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.