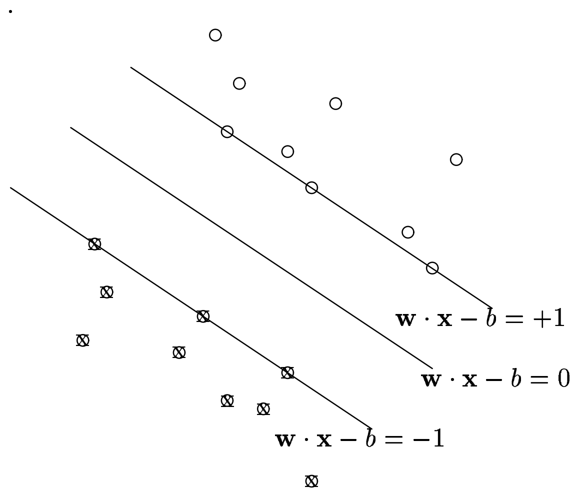
Sequential Minimal Optimization for SVM

Contents


$\mathbf{w} \cdot \mathbf{x} - b = +1$
$\mathbf{w} \cdot \mathbf{x} - b = 0$
$\mathbf{w} \cdot \mathbf{x} - b = -1$

Abstract

This is a C++ implementation of John C. Platt's sequential minimal optimization (SMO) for training a support vector machine (SVM). This program is based on the pseudocode in Platt (1998).

This is both the documentation and the C++ code. It is a NUWEB document from which both the LaTeX file and the C++ file can be generated. The documentation is essentially my notes when reading the papers (most of them being *cut-and-paste* from the papers).

1 Introduction to Support Vector Machine (SVM)

This introductio to Support Vector Machine for binary classification is based on Burges (1998).

1.1 Linear SVM

First let us look at the linear support vector machine. It is based on the idea of hyperplane classifier, or linearly separability.

Suppose we have N training data points $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_i \in \mathcal{R}^d$ and $y_i \in \{\pm 1\}$. We would like to learn a linear separating hyperplane classifier:

$$f(\mathbf{x}) = sgn(\mathbf{w} \cdot \mathbf{x} - b).$$

Furthermore, we want this hyperplane to have the maximum separating margin with respect to the two classes. Specifically, we want to find this hyperplane $H: \ y = \mathbf{w} \cdot \mathbf{x} - b = 0$ and two hyperplanes parallel to it and with equal distances to it,

$$H_1: \ y = \mathbf{w} \cdot \mathbf{x} - b = +1,$$

$$H_2: \; y = \mathbf{w} \cdot \mathbf{x} - b = -1,$$

with the condition that there are no data points between H_1 and H_2, and the distance between H_1 and H_2 is maximized.

For any separating plane H and the corresponding H_1 and H_2, we can always "normalize" the coefficients vector $\mathbf{w}$ so that H_1 will be $y = \mathbf{w} \cdot \mathbf{x} - b = +1$, and H_2 will be $y = \mathbf{w} \cdot \mathbf{x} - b = -1$. See Appendix A for details.

We want to maximize the distance between H_1 and H_2. So there will be some positive examples on H_1 and some negative examples on H_2. These examples are called *support vectors* because only they participate in the definition of the separating hyperplane, and other examples can be removed and/or moved around as long as they do not cross the planes H_1 and H_2.

Recall that in 2-D, the distance from a point (x_0, y_0) to a line $Ax+By+C = 0$ is $\frac{|Ax_0+By_0+C|}{\sqrt{A^2+B^2}}$. Similarly, the distance of a point on H_1 to $H: \; \mathbf{w} \cdot \mathbf{x} - b = 0$ is $\frac{|\mathbf{w} \cdot \mathbf{x} - b|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$, and the distance between H_1 and H_2 is $\frac{2}{\|\mathbf{w}\|}$. So, in order to maximize the distance, we should minimize $\|\mathbf{w}\| = \mathbf{w}^T\mathbf{w}$ with the condition that there are no data points between H_1 and H_2:

$$\mathbf{w} \cdot \mathbf{x} - b \geq +1, \quad \text{for positive examples } y_i = +1,$$

$$\mathbf{w} \cdot \mathbf{x} - b \leq -1, \quad \text{for negative examples } y_i = -1.$$

These two conditions can be combined into

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1.$$

So our problem can be formulated as

$$\min_{\mathbf{w}, b} \frac{1}{2}\mathbf{w}^T\mathbf{w} \quad \text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1.$$

This is a convex, quadratic programming problem (in $\mathbf{w}, b$), in a convex set.

Introducing Lagrange multipliers $\alpha_1, \alpha_2, \ldots, \alpha_N \geq 0$, we have the following Lagrangian:

$$\mathscr{L}(\mathbf{w}, b, \boldsymbol{\alpha}) \equiv \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i (\mathbf{w} \cdot \mathbf{x}_i - b) + \sum_{i=1}^{N} \alpha_i.$$

1.2 The dual problem

We can solve the Wolfe dual instead: *maximize* $\mathscr{L}(\mathbf{w}, b, \boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$, subject to the constraints that the gradient of $\mathscr{L}(\mathbf{w}, b, \boldsymbol{\alpha})$ with respect to the primal variables $\mathbf{w}$ and b vanish:

$$\frac{\partial \mathscr{L}}{\partial \mathbf{w}} = \mathbf{0}, \tag{1}$$

$$\frac{\partial \mathscr{L}}{\partial b} = 0, \tag{2}$$

and that

$$\boldsymbol{\alpha} \geq \mathbf{0}.$$

From Equations 1 and 2, we have

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i,$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0.$$

Substitute them into $\mathscr{L}(\mathbf{w}, b, \boldsymbol{\alpha})$, we have

$$\mathscr{L}_D \equiv \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j,$$

in which the primal variables are eliminated.

When we solve α_i, we can get $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$, (we will later show how to compute the threshold b), and we can classify a new object $\mathbf{x}$ with

$$\begin{aligned} f(\mathbf{x}) &= sgn(\mathbf{w} \cdot \mathbf{x} + b) \\ &= sgn((\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i) \cdot \mathbf{x} + b) \\ &= sgn(\sum_{i=1}^{N} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b). \end{aligned}$$

Please note that in the objective function and the solution, the training vectors $\mathbf{x}_i$ occur only in the form of dot product.

Before going into the details to how to solve this quadratic programming problem, let's extend it in two directions.

1.3 Non-linear SVM

What if the surface separating the two classes are not linear? Well, we can transform the data points to another high dimensional space such that the data points will be linearly separable. Let the transformation be $\Phi(\cdot)$. In the high dimensional space, we solve

$$\mathscr{L}_D \equiv \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

Suppose, in addition, $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$. That is, the dot product in that high dimensional space is equivalent to a *kernel* function of the input space. So we need not be explicit about the transformation $\Phi(\cdot)$ as long as we know that the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ is equivalent to the dot product of some other high dimensional space. There are many kernel functions that can be used this way, for example, the radial basis function (Gaussian kernel)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2}.$$

The Mercer's condition can be used to determine if a function can be used as a kernel function:

There exists a mapping Φ and an expansion

$$K(\mathbf{x}, \mathbf{y}) = \sum_i \Phi(\mathbf{x})_i \Phi(\mathbf{y})_i,$$

if and only if, for any $g(\mathbf{x})$ such that

$$\int g(\mathbf{x})^2 d\mathbf{x} \text{ is finite,}$$

then

$$\int K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0.$$

1.4 Imperfect separation

The other direction to extend SVM is to allow for noise, or imperfect separation. That is, we do not strictly enforce that there be no data points between H_1 and H_2, but we definitely want to penalize the data points that cross the boundaries. The penalty C will be finite. (If $C = \infty$, we come back to the original perfect separating case.)

We introduce non-negative slack variables $\xi_i \geq 0$, so that

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq +1 - \xi_i \quad \text{for } y_i = +1,$$

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 + \xi_i \quad \text{for } y_i = -1,$$

$$\xi_i \geq 0, \quad \forall i.$$

and we add to the objective function a penalizing term:

$$\underset{\mathbf{w}, b, \boldsymbol{\xi}}{\text{minimize}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C (\sum_i \xi_i)^m$$

where m is usually set to 1, which gives us

$$\begin{array}{lll} \underset{\mathbf{w}, b, \xi_i}{\text{minimize}} & \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi_i & \\ \text{subject to} & y_i(\mathbf{w}^T \mathbf{x}_i - b) + \xi_i - 1 \geq 0, & 1 \leq i \leq N \\ & \xi_i \geq 0, & 1 \leq i \leq N \end{array}$$

Introducing Lagrange multipliers $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, the Lagrangian is

$$\begin{aligned}
\mathscr{L}(\mathbf{w}, b, \xi_i; \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\
&\quad - \sum_{i=1}^{N}\alpha_i\left[y_i(\mathbf{w}^T\mathbf{x}_i - b) + \xi_i - 1\right] - \sum_{i=1}^{N}\mu_i\xi_i \\
&= \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}(C - \alpha_i - \mu_i)\,\xi_i \\
&\quad - \left(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i^T\right)\mathbf{w} - \left(\sum_{i=1}^{N}\alpha_i y_i\right)b + \sum_{i=1}^{N}\alpha_i
\end{aligned}$$

Neither the ξ_i's, nor their Lagrange multipliers, appear in the Wolfe dual problem:

$$\underset{\boldsymbol{\alpha}}{\text{maximize}}\, \mathscr{L}_D \equiv \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

subject to:

$$0 \le \alpha_i \le C,$$

$$\sum_i \alpha_i y_i = 0.$$

The only difference from the perfectly separating case is that α_i is now bounded above by C instead of ∞. For details, see Appendiex B.

The solution is again given by

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i$$

To train the SVM, we search through the feasible region of the dual problem and maximize the objective function. The optimal solution can be checked using the KKT conditions.

1.5 The KKT conditions

The KKT optimality conditions of the primal problem are, the gradient of $\mathscr{L}(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta})$ with respect to the primal variables $\mathbf{w}$, b, $\boldsymbol{\xi}$ vanishes (this must always be satisfied by the dual problem), and that for $1 \le i \le N$,

$$\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i - b) + \xi_i - 1) = 0, \tag{3}$$

$$\mu_i\xi_i = 0. \tag{4}$$

Depending on the value of α_i, we have three cases to consier:

1. If $\alpha_i = 0$, then $\mu_i = C - \alpha_i = C > 0$. From Equation 4, $\xi_i = 0$. so we have

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1 \geq 0.$$

2. If $0 < \alpha_i < C$, from Equation 3, we have

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) + \xi_i - 1 = 0 \tag{5}$$

Note that $\mu_i = C - \alpha_i > 0$, so $\xi_i = 0$ (Equation 4). Substituting into Equation 5, we have

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1 = 0.$$

3. If $\alpha_i = C$, then from Equation 3, we have

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) + \xi_i - 1 = 0 \tag{6}$$

Note that $\mu_i = C - \alpha_i = 0$, we have $\xi_i \geq 0$. So

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1 \leq 0.$$

The quantity $y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1$ can be computed as

$$R_i = y_i(\mathbf{w}^T\mathbf{x}_i - b) - y_i^2 = y_i(\mathbf{w}^T\mathbf{x}_i - b - y_i) = y_iE_i$$

where $E_i = \mathbf{w}^T\mathbf{x}_i - b - y_i = u_i - y_i$ is the prediction error.

To summarize, the KKT condition implies:

$$\begin{aligned} \alpha_i = 0 \quad &\Rightarrow \quad R_i \geq 0, \\ 0 < \alpha_i < C \quad &\Rightarrow \quad R_i \approx 0, \\ \alpha_i = C \quad &\Rightarrow \quad R_i \leq 0. \end{aligned}$$

In the following two cases, the KKT condition is violated:

- $\alpha_i < C$ and $R_i < 0$,
- $\alpha_i > 0$ and $R_i > 0$.

1.6 Checking KKT condition without using threshold b

As the dual problem does not solve for the threshold b directly, it would be beneficial to check the KKT condition without using threshold b. This technique is due to Keerthi et al. (2001).

The quantity $y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1$ (which must ≥ 0 for all i if the KKT condition is satisfied) can also be written as

$$\begin{aligned} & y_i(\mathbf{w}^T\mathbf{x}_i - b) - 1 \\ = \quad & y_i(\mathbf{w}^T\mathbf{x}_i - b) - y_i^2 \\ - \quad & y_i(\mathbf{w}^T\mathbf{x}_i - y_i - b) \\ = \quad & y_i(F_i - b), \end{aligned}$$

where $F_i \equiv \mathbf{w}^T\mathbf{x}_i - y_i$.

Note for $E_i = F_i - b$, we have $E_i - E_j = F_i - F_j$. (This equality is useful, as Platt's SMO algorithm uses $E_i - E_j$ when optimization the two Lagrange multipliers α_i, α_j.)

This notation is useful because the KKT conditions

$$\begin{aligned} \alpha_i = 0 \quad &\Rightarrow \quad y_i(F_i - b) \geq 0 \\ 0 < \alpha_i < C \quad &\Rightarrow \quad y_i(F_i - b) \approx 0 \\ \alpha_i = C \quad &\Rightarrow \quad y_i(F_i - b) \leq 0 \end{aligned}$$

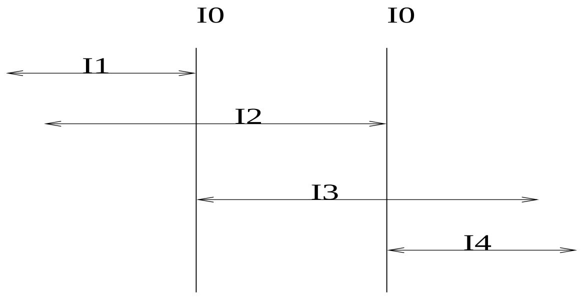
can be written as

$$\begin{aligned} i \in I_0 \cup I_1 \cup I_2 \quad &\Rightarrow \quad F_i \geq b \\ i \in I_0 \cup I_3 \cup I_4 \quad &\Rightarrow \quad F_i \leq b, \end{aligned}$$

where

$$\begin{aligned} I_0 &\equiv \{i : 0 < \alpha_i < C\} \\ I_1 &\equiv \{i : y_i = +1, \alpha_i = 0\} \\ I_2 &\equiv \{i : y_i = -1, \alpha_i = C\} \\ I_3 &\equiv \{i : y_i = +1, \alpha_i = C\} \\ I_4 &\equiv \{i : y_i = -1, \alpha_i = 0\}. \end{aligned}$$

So that $\forall i \in I_0 \cup I_1 \cup I_2$, and $\forall j \in I_0 \cup I_3 \cup I_4$, we should have $F_i \geq F_j$, if KKT condition is satisfied.



To check if this condition holds, we define

$$\begin{aligned} b_{\text{up}} &= \min\{F_i : i \in I_0 \cup I_1 \cup I_2\}, \\ b_{\text{low}} &= \max\{F_i : i \in I_0 \cup I_3 \cup I_4\}. \end{aligned}$$

The KKT condition implies $b_{\text{up}} \geq b_{\text{low}}$, and similarly, $\forall i \in I_0 \cup I_1 \cup I_2$, $F_i \geq b_{\text{low}}$, and $\forall i \in I_0 \cup I_3 \cup I_4$, $F_i \leq b_{\text{up}}$.

These comparisons do not use the threshold b.

As an added benefit, given the first α_i, these comparisons automatically finds the second α_i for joint optimization in SMO.

2 SMO Algorithm

2.1 Optimize two α_i's

The SMO algorithm searches through the feasible region of the dual problem and maxmizes the objective function

$$\mathscr{L}_D \equiv \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j,$$

$$0 \le \alpha_i \le C, \quad \forall i.$$

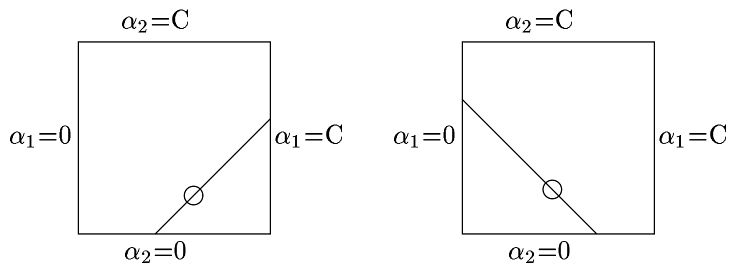
It works by optimzing two α_i's at a time (with the other α_i's fixed). It uses heuristics to choose the two α_i' for optimization. This is essentially a hill-climbing.

Without loss of generality, suppose we are optimizing α_1, α_2, from an old set of feasible solution: α_1^{old}, α_2^{old}, α_3, ..., α_N. (For initialization, we can set $\boldsymbol{\alpha}^{\text{old}} = \mathbf{0}$.)

Because $\sum_{i=1}^{N} y_i\alpha_i = 0$, we have

$$y_1\alpha_1 + y_2\alpha_2 = y_1\alpha_1^{\text{old}} + y_2\alpha_2^{\text{old}}.$$

This confines the optimization to be on a line, as shown in the following figure:



$y_1 \neq y_2 \Rightarrow \alpha_1 - \alpha_2 = \gamma$ $\qquad$ $y_1 = y_2 \Rightarrow \alpha_1 + \alpha_2 = \gamma$

Let $s = y_1 y_2$. Multiply

$$y_1\alpha_1 + y_2\alpha_2 = \text{Const.}$$

by y_1, and we have

$$\alpha_1 = \gamma - s\alpha_2.$$

where $\gamma \equiv \alpha_1 + s\alpha_2 = \alpha_1^{\text{old}} + s\alpha_2^{\text{old}}$.

Fixing the other α_i's, the objective function can be written as

$$\begin{aligned} \mathscr{L}_D \quad = \quad & \alpha_1 + \alpha_2 + \text{Const.} \\ & -\frac{1}{2}\Big(y_1 y_1 \mathbf{x}_1^T \mathbf{x}_1 \alpha_1^2 + y_2 y_2 \mathbf{x}_2^T \mathbf{x}_2 \alpha_2^2 + 2 y_1 y_2 \mathbf{x}_1^T \mathbf{x}_2 \alpha_1 \alpha_2 \\ & \quad +2 \left(\sum_{i=3}^{N} \alpha_i y_i \mathbf{x}_i^T \right) (y_1 \mathbf{x}_1 \alpha_1 + y_2 \mathbf{x}_2 \alpha_2) + \text{Const.} \Big) \end{aligned}$$

Let $K_{11} = \mathbf{x}_1^T\mathbf{x}_1$, $K_{22} = \mathbf{x}_2^T\mathbf{x}_2$, $K_{12} = \mathbf{x}_1^T\mathbf{x}_2$, and

$$\begin{aligned}
v_j &\equiv \sum_{i=3}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_j \\
&= \mathbf{x}_j^T \mathbf{w}^{\text{old}} - \alpha_1^{\text{old}} y_1 \mathbf{x}_1^T \mathbf{x}_j - \alpha_2^{\text{old}} y_2 \mathbf{x}_2^T \mathbf{x}_j \\
&= (\mathbf{x}_j^T \mathbf{w}^{\text{old}} - b^{\text{old}}) + b^{\text{old}} - \alpha_1^{\text{old}} y_1 \mathbf{x}_1^T \mathbf{x}_j - \alpha_2^{\text{old}} y_2 \mathbf{x}_2^T \mathbf{x}_j \\
&= u_j^{\text{old}} + b^{\text{old}} - \alpha_1^{\text{old}} y_1 \mathbf{x}_1^T \mathbf{x}_j - \alpha_2^{\text{old}} y_2 \mathbf{x}_2^T \mathbf{x}_j,
\end{aligned}$$

where $u_j^{\text{old}} = \mathbf{x}_j^T \mathbf{w}^{\text{old}} - b^{\text{old}}$ is the output of $\mathbf{x}_j$ under old parameters.

$$\begin{aligned}
\mathscr{L}_D &= \alpha_1 + \alpha_2 - \frac{1}{2}\Big(K_{11}\alpha_1^2 + K_{22}\alpha_2^2 + 2sK_{12}\alpha_1\alpha_2 \\
&\quad +2y_1v_1\alpha_1 + 2y_2v_2\alpha_2\Big) + \text{Const.} \\
&= \gamma - s\alpha_2 + \alpha_2 - \frac{1}{2}\Big(K_{11}(\gamma - s\alpha_2)^2 + K_{22}\alpha_2^2 \\
&\quad +2sK_{12}(\gamma - s\alpha_2)\alpha_2 \\
&\quad +2y_1v_1(\gamma - s\alpha_2) + 2y_2v_2\alpha_2\Big) + \text{Const.} \\
&= (1-s)\alpha_2 - \frac{1}{2}K_{11}(\gamma - s\alpha_2)^2 - \frac{1}{2}K_{22}\alpha_2^2 - sK_{12}(\gamma - s\alpha_2)\alpha_2 \\
&\quad -y_1v_1(\gamma - s\alpha_2) - y_2v_2\alpha_2 + \text{Const.} \\
&= (1-s)\alpha_2 - \frac{1}{2}K_{11}\gamma^2 + sK_{11}\gamma\alpha_2 - \frac{1}{2}K_{11}s^2\alpha_2^2 - \frac{1}{2}K_{22}\alpha_2^2 \\
&\quad -sK_{12}\gamma\alpha_2 + s^2K_{12}\alpha_2^2 - y_1v_1\gamma + sy_1v_1\alpha_2 - y_2v_2\alpha_2 \\
&\quad +\text{Const.} \\
&= (1-s)\alpha_2 + sK_{11}\gamma\alpha_2 - \frac{1}{2}K_{11}\alpha_2^2 - \frac{1}{2}K_{22}\alpha_2^2 \\
&\quad -sK_{12}\gamma\alpha_2 + K_{12}\alpha_2^2 + y_2v_1\alpha_2 - y_2v_2\alpha_2 \\
&\quad +\text{Const.} \\
&= \left(-\frac{1}{2}K_{11} - \frac{1}{2}K_{22} + K_{12}\right)\alpha_2^2 \\
&\quad + \left(1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2v_1 - y_2v_2\right)\alpha_2 \\
&\quad +\text{Const.} \\
&= \frac{1}{2}\left(2K_{12} - K_{11} - K_{22}\right)\alpha_2^2 \\
&\quad + \left(1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2v_1 - y_2v_2\right)\alpha_2 \\
&\quad +\text{Const.}
\end{aligned}$$

Let $\eta \equiv 2K_{12} - K_{11} - K_{12}$. The coefficient of α_2 is

$$\begin{aligned}
&1 - s + sK_{11}\gamma - sK_{12}\gamma + y_2v_1 - y_2v_2 \\
= \; &1 - s + sK_{11}(\alpha_1^{\text{old}} + s\alpha_2^{\text{old}}) - sK_{12}(\alpha_1^{\text{old}} + s\alpha_2^{\text{old}}) \\
&+y_2(u_1^{\text{old}} + b^{\text{old}} - \alpha_1^{\text{old}} y_1 K_{11} - \alpha_2^{\text{old}} y_2 K_{12})
\end{aligned}$$

$$\begin{aligned}
& -y_2(u_2^{\text{old}} + b^{\text{old}} - \alpha_1^{\text{old}} y_1 K_{12} - \alpha_2^{\text{old}} y_2 K_{22}) \\
= \;& 1 - s + sK_{11}\alpha_1^{\text{old}} + K_{11}\alpha_2^{\text{old}} - sK_{12}\alpha_1^{\text{old}} - K_{12}\alpha_2^{\text{old}} \\
& +y_2 u_1^{\text{old}} + y_2 b^{\text{old}} - sK_{11}\alpha_1^{\text{old}} - K_{12}\alpha_2^{\text{old}} \\
& -y_2 u_2^{\text{old}} - y_2 b^{\text{old}} + sK_{12}\alpha_1^{\text{old}} + K_{22}\alpha_2^{\text{old}} \\
= \;& 1 - s + (sK_{11} - sK_{12} - sK_{11} + sK_{12})\alpha_1^{\text{old}} \\
& +(K_{11} - 2K_{12} + K_{22})\alpha_2^{\text{old}} \\
& +y_2(u_1^{\text{old}} - u_2^{\text{old}}) \\
= \;& y_2^2 - y_1 y_2 + (K_{11} - 2K_{12} + K_{22})\alpha_2^{\text{old}} + y_2(u_1^{\text{old}} - u_2^{\text{old}}) \\
= \;& y_2(y_2 - y_1 + u_1^{\text{old}} - u_2^{\text{old}}) - \eta\alpha_2^{\text{old}} \\
= \;& y_2((u_1^{\text{old}} - y_1) - (u_2^{\text{old}} - y_2)) - \eta\alpha_2^{\text{old}} \\
= \;& y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}}.
\end{aligned}$$

So the objective function is

$$\mathscr{L}_D = \frac{1}{2}\eta\alpha_2^2 + (y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}})\alpha_2 + \text{Const.}$$

The first and second derivatives are

$$\frac{\mathrm{d}\mathscr{L}_D}{\mathrm{d}\alpha_2} = \eta\alpha_2 + (y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}}),$$

$$\frac{\mathrm{d}^2\mathscr{L}_D}{\mathrm{d}\alpha_2^2} = \eta.$$

Note that $\eta = 2K_{12} - K_{11} - K_{22} \le 0$. Proof: Let $K_{11} = \mathbf{x}_1^T\mathbf{x}_1$, $K_{12} = \mathbf{x}_1^T\mathbf{x}_2$, $K_{22} = \mathbf{x}_2^T\mathbf{x}_2$. Then $\eta = -(\mathbf{x}_2 - \mathbf{x}_1)^T(\mathbf{x}_2 - \mathbf{x}_1) = -\|\mathbf{x}_2 - \mathbf{x}_1\|^2 \le 0$.

Let $\frac{\mathrm{d}\mathscr{L}_D}{\mathrm{d}\alpha_2} = 0$, and we have

$$\begin{aligned}
\alpha_2^{\text{new}} &= -\frac{y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}}}{\eta} \\
&= \alpha_2^{\text{old}} + \frac{y_2(E_2^{\text{old}} - E_1^{\text{old}})}{\eta}
\end{aligned}$$

If $\eta < 0$, the above equation gives us the unconstrained maximum point α_2^{new}. It must be checked against the feasible range. Let $s = y_1 y_2$, and $\gamma = \alpha_1^{\text{old}} + s\alpha_2^{\text{old}}$. The range of α_2 is determined as follows:

- If $s = 1$, then $\alpha_1 + \alpha_2 = \gamma$.
  - If $\gamma > C$, then $\max\alpha_2 = C$, and $\min\alpha_2 = \gamma - C$.
  - If $\gamma < C$, then $\min\alpha_2 = 0$, and $\max\alpha_2 = \gamma$.
- If $s = -1$, then $\alpha_1 - \alpha_2 = \gamma$.
  - If $\gamma > 0$, then $\min\alpha_2 = 0$, and $\min\alpha_2 = C - \gamma$.

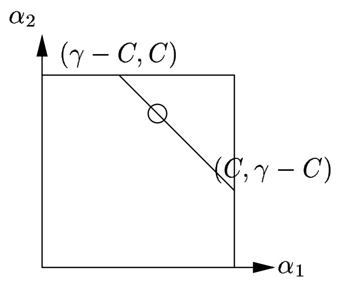


Figure 1: $\alpha_1 + \alpha_2 = \gamma$, and $\gamma > C$.

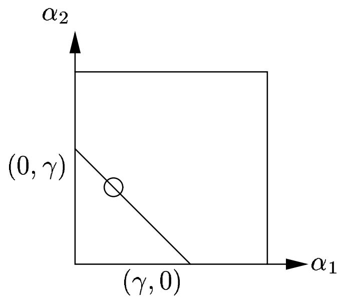


Figure 2: $\alpha_1 + \alpha_2 = \gamma$, and $\gamma < C$.

– If $\gamma < 0$, then $\min \alpha_2 = -\gamma$, and $\max \alpha_2 = C$.

Let the minimum feasible value of α_2 be L, maximum be H. Then

$$\alpha_2^{\text{new,clipped}} = \begin{cases} H, & \text{if } H < \alpha_2^{\text{new}}, \\ \alpha_2^{\text{new}}, & \text{if } L \le \alpha_2^{\text{new}} \le H \\ L, & \text{if } \alpha_2^{\text{new}} < L. \end{cases}$$

To summarize, given α_1, α_2 (and the corresponding y_1, y_2, K_{11}, K_{12}, K_{22}, $E_2^{\text{old}} - E_1^{\text{old}}$), we can optimize the two α's by the following procedure:

1. $\eta = 2K_{12} - K_{11} - K_{22}$.
2. If $\eta < 0$,

$$\Delta\alpha_2 = \frac{y_2(E_2^{\text{old}} - E_1^{\text{old}})}{\eta},$$

and clip the solution within the feasible region. Then

$$\Delta\alpha_1 = -s\Delta\alpha_2.$$

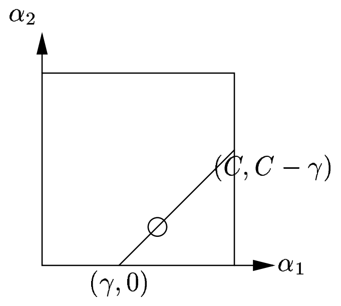


Figure 3: $\alpha_1 - \alpha_2 = \gamma$, and $\gamma > 0$.

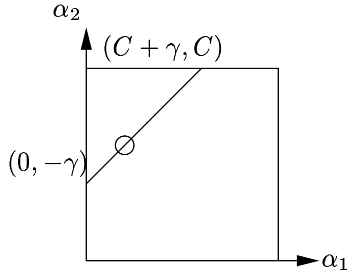


Figure 4: $\alpha_1 - \alpha_2 = \gamma$, and $\gamma < 0$.

3. If $\eta = 0$, we need to evaluate the objective function at the two endpoints, and set α_2^{new} to be the one with larger objective function value. The objective function is

$$\mathscr{L}_D = \frac{1}{2}\eta\alpha_2^2 + (y_2(E_1^{\text{old}} - E_2^{\text{old}}) - \eta\alpha_2^{\text{old}})\alpha_2 + \text{Const.} \tag{7}$$

2.2 SMO Algorithm: Updating after a successful optimization step

When α_1, α_2 are changed by $\Delta\alpha_1$, $\Delta\alpha_2$, we can update E_i's, F_i's, $\mathbf{w}$ (for linear kernel), and b. Let $E(\mathbf{x}, y)$ be the prediction error on $(\mathbf{x}, y)$:

$$E(\mathbf{x}, y) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} - b - y,$$

The change in E is

$$\Delta E(\mathbf{x}, y) = \Delta\alpha_1 y_1 \mathbf{x}_1^T \mathbf{x} + \Delta\alpha_2 y_2 \mathbf{x}_2^T \mathbf{x} - \Delta b. \tag{8}$$

The change in the threshold can be computed by forcing $E_1^{\text{new}} = 0$ if $0 < \alpha_1^{\text{new}} < C$ (or $E_2^{\text{new}} = 0$ if $0 < \alpha_2^{\text{new}} < C$). From

$$0 \quad = \quad E(\mathbf{x}, y)^{\text{new}}$$

$$
\begin{aligned}
&= E(\mathbf{x}, y)^{\text{old}} + \Delta E(\mathbf{x}, y) \\
&= E(\mathbf{x}, y)^{\text{old}} + \Delta\alpha_1 y_1 \mathbf{x}_1^T \mathbf{x} + \Delta\alpha_2 y_2 \mathbf{x}_2^T \mathbf{x} - \Delta b
\end{aligned}
$$

we have

$$\Delta b = E(\mathbf{x}, y)^{\text{old}} + \Delta\alpha_1 y_1 \mathbf{x}_1^T \mathbf{x} + \Delta\alpha_2 y_2 \mathbf{x}_2^T \mathbf{x}. \tag{9}$$

If $\alpha_1 = 0$, we can only say $y_1 E_1^{\text{new}} \geq 0$; similarly, if $\alpha_1 = C$, we have $y_1 E_2^{\text{new}} \leq 0$. If both α_1 and α_2 take values 0 or C, the original SMO algorithm computes two values of the new b for α_1 and α_2 using Equation 9, and takes the average. This is regarded as problematic by Keerthi et al. (2001).

Similarly, from

$$F(\mathbf{x}, y) = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} - y$$

we have

$$\Delta F(\mathbf{x}, y) = \Delta\alpha_1 y_1 \mathbf{x}_1^T \mathbf{x} + \Delta\alpha_2 y_2 \mathbf{x}_2^T \mathbf{x}. \tag{10}$$

For the weight vector of linear kernels,

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i,$$

$$\Delta\mathbf{w} = \Delta\alpha_1 y_1 \mathbf{x}_1 + \Delta\alpha_2 y_2 \mathbf{x}_2. \tag{11}$$

2.3 SMO Algorithm: Pick two α_i's for optimization

The heuristics for picking two α_i's for optimization in the original SMO paper are as follows:

- The outer loop selects the first α_i, the inner loop selects the second α_i that maximizes $|E_2 - E_1|$.
- The outer loop alternates between one sweep through all examples and as many sweeps as possible through the non-boundary examples (those with $0 < \alpha_i < C$), selecting the example that violates the KKT condition.
- Given the first α_i, the inner loop looks for a non-boundary that maximizes $|E_2 - E_1|$. If this does not make progress, it starts a sequential scan through the non-boundary examples, starting at a random position; if this fails too, it startis a sequential scan through all the examples, also starting at a random postion.

Because the algorithm spends most of the time adjusting the non-boundary examples, the E_i's of these examples are cached.

The improvement proposed in Keerthi et al. (2001) avoids the use of the threshold b in checking KKT condition, and compares two F_i's, which also automatically selects the second α_i for joint optimization. There are two variantions when the outer loop deals only with the non-boundary examples: