A Roadmap to SVM Sequential Minimal Optimization for Classification

A Tutorial

Ruben Ramirez-Padron

Version 1.0

Deliverable for
Course EEL 6908: Independent Study
Fall 2007

School of Electrical Engineering and Computer Science
University of Central Florida
Orlando

Abstract

This document offers a concise but comprehensive introduction to the Support Vector Machine (SVM) technique applied to binary classification. The foundations of this technique are presented with a great amount of detail, including results from Numerical Optimization theory that support the SVM approach, geometrical interpretation of the SVM variables and its parameters, and a brief explanation of the SVM double dual optimization problem.

The Sequential Minimal Optimization (SMO) algorithm designed by John Platt at Microsoft Research is explained in detail. Two modifications to the original SMO algorithm, proposed by S. Sathiya Keerthi and other researchers, are explained as well. Finally, several references are suggested for those readers interested on obtaining a deeper knowledge of SVM techniques in general.

TABLE OF CONTENTS

I. Introduction 1
II. The Support Vector Machine Classifier 2
2.1. The context 2
2.2. The derivation of the SVM classification model. 4
2.3. The double dual optimization problem. 11
III. Sequential Minimal Optimization 15
3.1. Analytic optimization for two multipliers 16
3.2. Choosing two multipliers to optimize 19
3.3. Updating the bias 19
3.4. Platt's SMO pseudo-code 20
IV. An Improvement to Platt's SMO Algorithm. 24
4.1. Overcoming Platt's SMO inefficiency 24
4.2. The Modifications of SMO 26
V. Conclusions and Further Readings 27
Acknowledgements 28
References 29

I. Introduction

Binary classification of patterns is an old problem from Pattern Recognition theory. Several techniques have being devised to solve it. Support Vector Machines (SVM) is one of the latest techniques applied to classification problems. This document is a concise but comprehensive introduction to SVM applied to binary classification, and in particular to the Sequential Minimal Optimization (SMO) implementation derived from the original work of John Platt at Microsoft Research. No knowledge about SVM or other kernel learning methods is required to understand the content of this document, although basic knowledge from Linear Algebra and the Lagrange multipliers technique from Optimization Theory is assumed. A basic understanding of the concepts behind Pattern Recognition is also assumed.

The SVM binary classifier is a successful and widely used classifier (Barabino et al, 1999; Roobaert and Van Hulle 1999; Onoda et al, 2000; Hua and Sun, 2001), based on the idea of obtaining a maximum margin classification function to adjust the capacity of the model to the size and complexity of the training data. There are some good introductory reviews to the SVM classification technique. Two very good introductions are (Burges, 1998) and chapters 1 and 7 of (Scholkpof and Smola, 2002). The last reference also provides chapters on theoretical properties and implementation issues of this and other kernel machines. Another interesting but very brief introduction is (Boser et. al., 1992).

The SVM SMO is an original implementation for the SVM classifier designed to avoid the likely large QP optimization problem that appears as part of the SVM model. It was proposed originally in (Platt, 1999a). Two interesting variations to the original SMO have been proposed in (Keerthi, et al, 2001). A brief review of those variants will be offered here.

The main purpose of this work is to gather fundamental knowledge about SVM and SVM SMO for classification from different sources and put it all together in a document that could be useful to newcomers to the interesting and very active research area of Machine Learning. Furthermore, this work does not claim any original results in the field, but only pedagogical value for those readers interested on a basic but comprehensive starting-up in the field of SVM classification.

II. The Support Vector Machine Classifier

2.1. The context

The binary classification problem consists on obtaining a classification function (often called "the model") that determines whether certain given patterns belong to one of two distinct classes, denoted here by class A and class B. The model must be trained beforehand, and this learning process can be either unsupervised or supervised. In this document a supervised approach is assumed, therefore previously classified data is given for training purposes. The training data consists of a set of *n* samples $\{(x_i, y_i) \mid x_i \in \chi,\ y_i \in \{+1,-1\},\ i = 1,...,n\}$, where the set χ denotes a domain of patterns with a finite dimension, and the y_i components denote the class to which the corresponding x_i patterns belong to. For example, if for the sample (x_1, y_1) we have $y_1 = -1$ then x_1 belongs to class A; on the contrary, x_1 belongs to class B if $y_1 = +1$. The training data are assumed to be generated from some unknown i.i.d. probability distribution P(*x, y*). This assumption will not be of much use in this document, but it is essential for Statistical Learning Theory (Vapnik, 1998), the theoretical framework encompassing SVM and other kernel methods.

In the SVM approach the input patterns x_i are usually mapped to a so called **feature space** *H* using a mapping function $\phi : \chi \to H$, then a linear classification model is trained on the transformed data in *H*, and the separating hyperplane so obtained is reinterpreted as a likely non-linear separating surface in the original space χ. The whole approach is graphically shown in figure 1. This is a common process in several areas of Mathematics, where data from a difficult problem in a given space is transformed into another space where a feasible well-known technique can be applied and the solution so obtained is then interpreted back as a solution in the original space.

The hyperplane obtained in *H* should separate the corresponding feature patterns in a way that the margin (i.e. the distance) between the hyperplane and the patterns is maximized and as many training patterns as possible are correctly classified by the model as well. The feature space *H* must be a vector space and it usually has a very much larger dimension than χ. The motivation behind this particular mapping technique is that representing the training data in the higher dimensional space *H* it's more likely for the data to be linearly separable (Cover, 1965). The solution obtained turns out to be also the desired maximum margin but likely non-linear surface in χ.

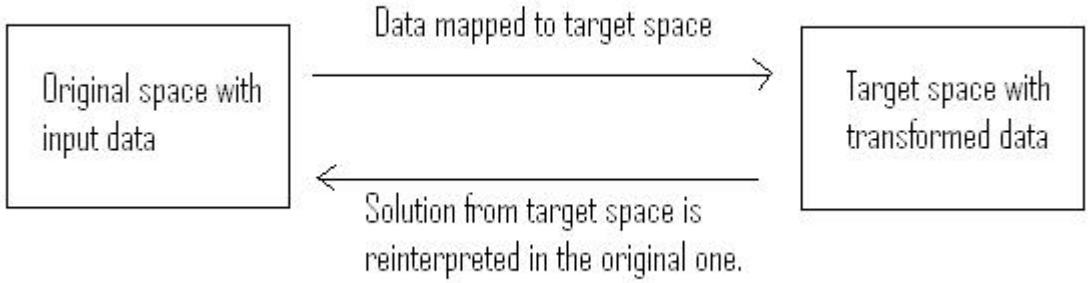

Figure 1. Transformation of data for easily solving the problem in another space.

At this point it's important to note two important cases of the mapping function ϕ:

- It could be the identity function, and so $H = \chi$. If that is the case, obviously the decision function will be linear in χ, and χ must be a vector space.
- It could map the training patterns into an infinite dimensional space *H*. As explained below, SVM deals with this case through the concept of kernel function.

As stated before, the SVM binary classifier model is expressed by a hyperplane in *H*:

$$f(x) = \text{sgn}(\langle w, \phi(x) \rangle + b) = \text{sgn}(d(\phi(x))) \qquad (1)$$

Where $d(r) = \langle w, r \rangle + b$ is the linear decision surface; $w, \phi(x) \in H$; and b is a real number. For avoiding an intricate notation the values of the feature patterns $\phi(x_i)$ will be denoted sometimes by ϕ_i for convenience.

The fundamental idea behind the SVM classifier is to fit the linear model to the mapped training data by maximizing the margin of the classifier; i.e. to find appropriate values for the parameters **w** and *b,* such that the distance from the hyperplane to the nearest training patterns from both classes is maximized and also $f(x_i) = y_i$ for as many training patterns as possible.

For realizing why it's important to look for a maximum margin hyperplane, let us analyze the simple linearly separable classification problem that appears in figure 2. Here, three feasible decision surfaces are drawn. A very intuitive rationale behind the convenience of the maximum margin classifier approach is based in the presence of noise in almost all practical problems. Let's assume that the points represented in figure 2 are the actual training patterns, but because the presence of noise the training data will be located in a very small neighborhood of these actual patterns. It's clear from the figure that after applying a certain small amount of noise to the input patterns, the maximum margin hyperplane depicted in the middle has a grater probability of still correctly separating both classes despite noise. An equivalent interpretation considers that most of the testing data lies in neighborhoods of the training patterns, and it's based on the assumption of both training and testing data being generated under the same i.i.d. distribution P(*x, y*). More formal arguments supporting this approach and its relationship to generalization properties of learning machines are offered in section 7.2 of (Scholkpof and Smola, 2002).

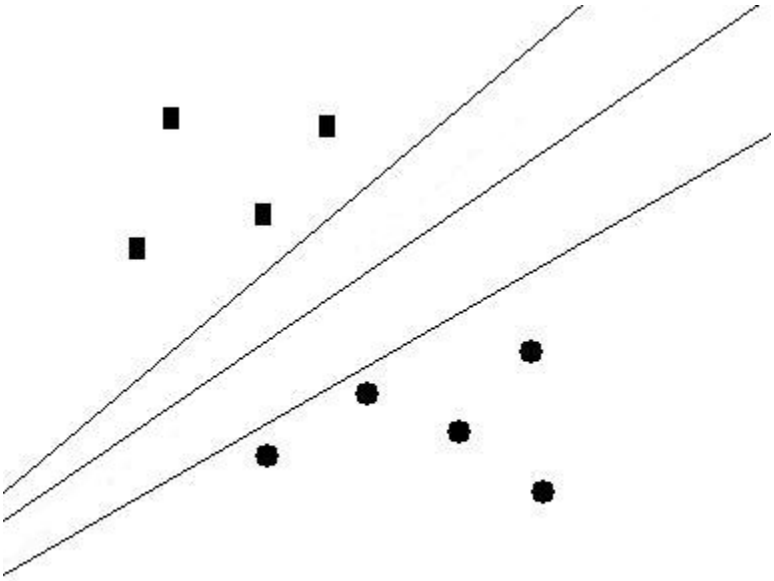

Figure 2. Why to use maximum margin hyperplanes.

If after training our classifier we had $f(x_i) = y_i$ for all the training data then the feature patterns $\{\phi(x_i)\}$ turned out to be linearly separable in H. Note that this could be the case even if the given training patterns $\{x_i\}$ are not linearly separable in the original space χ.

2.2. The derivation of the SVM classification model.

The primary optimization problem

The following two facts are very well known from Geometry:

- The vector $\mathbf{w}$ in any hyperplane $\{\mathbf{r} \in H \mid d(\mathbf{r}) = \langle \mathbf{w}, \mathbf{r} \rangle + b = 0\}$ is orthogonal to the hyperplane.
- The distance from the hyperplane to any point $\mathbf{c} \in H$ is obtained as $\frac{|d(c)|}{\|w\|}$

It's important to note that there is a freedom we need to get rid off here: by multiplying the equation $d(\mathbf{r}) = 0$ by any non-zero constant we end essentially with the same hyperplane. In order to obtain a unique representation for every different hyperplane in our model, the following definition is stated, as it appears in chapter 7 of (Scholkpof and Smola, 2002):

Definition 1: The pair $(\mathbf{w}, b) \in H \times R$ is called a **canonical hyperplane** with respect to the set of mapped training patterns $\{\phi(x_i)\}$ if it is scaled such that

$$\min_{i=1..n} \left| \langle \mathbf{w}, \phi(x_i) \rangle + b \right| = 1$$

If we only deal with the canonical representation of the feasible hyperplanes then the distance from any hyperplane to its nearest point will be always equal to $\frac{1}{\|w\|}$. Therefore, to find the hyperplane with maximum margin the norm of **w** must be as small as possible. At the same time, we should try to correctly classify any training data while allowing some bad classified patterns if the problem is not linearly separable in *H*. This situation is graphically shown in figure 3, where the maximum margin hyperplane for a linearly separable problem is drawn.

The previous arguments lead us to state the following optimization problem in order to find the maximum margin classifier:

$$\begin{aligned} &\min \tau(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \\ &\textit{subject to} \\ &\quad y_i\left(\langle \mathbf{w}, \phi_i \rangle + b\right) \geq 1 - \xi_i \\ &\quad \xi_i \geq 0 \end{aligned} \qquad (2)$$

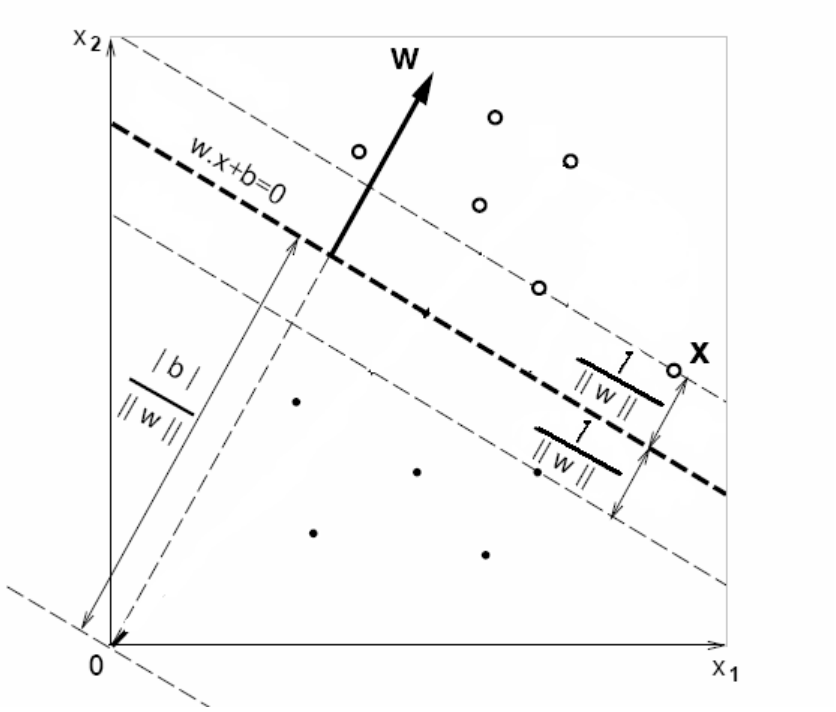

Figure 3. Example of a maximum margin hyperplane separating two classes (figure modified from (Boser et. al., 1992)).

Problem (2) is called the **primal optimization problem**. The constant C is a non-negative regularization parameter fixed by the user. The components of ξ are slack variables introduced

to make it possible to solve non-linearly separable classification problems (Cortes and Vapnik, 1995). Note that, because the positive slack variables tend to increase the objective function $\tau(w,\xi)$, if the training data is linearly separable all the slack variables should be cero at the optimal value, especially for large values of C. On the other hand, if C has a very small value then the solution will be more focused on minimizing $\|\mathbf{w}\|$ and more misclassifications are allowed. The C parameter thus establishes a trade-off between margin maximization and the acceptance of patterns such that $\langle w,\phi(x)\rangle + b < 1$.

The dual optimization problem

The primal optimization problem can be transformed using the Lagrange multipliers technique (Arfken, 1985). Introducing the Lagrange multipliers α_i and μ_i, $i = 1,..,n$, the problem can be restated as follows:

$$\min_{w,b,\xi}\max_{\alpha,\mu} L(\mathbf{w},b,\boldsymbol{\xi},\alpha_i,\mu_i) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\langle \mathbf{w},\phi_i\rangle + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n}\mu_i\xi_i$$

subject to

$$\begin{aligned} &\xi_i \geq 0 \\ &y_i\left(\langle \mathbf{w},\phi_i\rangle + b\right) - 1 + \xi_i \geq 0 \\ &\alpha_i \geq 0; \quad \mu_i \geq 0 \end{aligned} \tag{3}$$

A local solution to this optimization problem is called a saddle point because it is a minimum of the function L with respect to the primal variables $\mathbf{w}$, b and ξ, and a maximum with respect to the Lagrange multipliers. It follows that the partial derivatives of (3) with respect to the primal variables are equal to cero at the solution. Calculating those derivatives the following equations are obtained:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \phi_i = 0 \tag{4.1}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{4.2}$$

$$\frac{\partial L}{\partial \xi_k} = C - \alpha_k - \mu_k = 0 \tag{4.3}$$

Substituting the equations (4) into problem (3) the Lagrangian is transformed as follows:

$$
\begin{aligned}
L(w,b,\xi,\alpha_i,\mu_i) &= \frac{1}{2}\left\|\sum_{i=1}^{n}\alpha_i y_i \phi_i\right\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\left\langle\sum_{j=1}^{n}\alpha_j y_j \phi_j, \phi_i\right\rangle + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n}\mu_i\xi_i \\
&= \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \langle\phi_i,\phi_j\rangle + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i\left(\sum_{j=1}^{n}\alpha_j y_j \langle\phi_j,\phi_i\rangle + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n}(C-\alpha_i)\xi_i \\
&= \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \langle\phi_i,\phi_j\rangle - \sum_{i=1}^{n}\alpha_i y_i\left(\sum_{j=1}^{n}\alpha_j y_j \langle\phi_j,\phi_i\rangle + b\right) + \sum_{i=1}^{n}\alpha_i \\
&= \sum_{i=1}^{n}\alpha_i - b\sum_{i=1}^{n}\alpha_i y_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \langle\phi_i,\phi_j\rangle \\
&= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \langle\phi_i,\phi_j\rangle
\end{aligned}
$$

The optimization problem has been now transformed into another (commonly called the **dual optimization problem**) that depends only on the Lagrange parameters $\boldsymbol{\alpha}$:

$$
\begin{aligned}
&\max_{\alpha} W(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \langle\phi_i,\phi_j\rangle \\
&subject\ to: \\
&\quad 0 \le \alpha_i \le C \\
&\quad \sum_{i=1}^{n}\alpha_i y_i = 0
\end{aligned}
\qquad (5)
$$

Because an expression for **w** in terms of $\boldsymbol{\alpha}$ is already available from equation (4.1)**,** the classifier can be expressed as shown below:

$$
f(x) = \operatorname{sgn}\left(\sum_{i=1}^{n}\alpha_i y_i \langle\phi_i,\phi(x)\rangle + b\right) \qquad (6)
$$

The training patterns with corresponding $\alpha_i \neq 0$ are called "support patterns" or "support vectors" (the latter is not very accurate in cases where χ is not a vector space, but it is the most commonly used term in literature), because only those patterns are employed in the final expression for the classifier. This fact has several interesting implications. First, the algorithm establishes a distinction between those patterns that are important for the classification problem and those that aren't. This is very important information not available in algorithms that uses the mean squared error approach instead of the maximum margin. One immediate consequence is that all non-support patterns can be removed from the training data without affecting the results of both the training process and the classification of new data. Usually the set of support vectors represents a very small percentage of the training data, thus improving the execution time of the classification phase compared to other non-parametric techniques (e.g. Radial Basis Functions). Another important consequence derives from the fact, as will be shown below, that some of the support vectors for which $\alpha_i = C$ could be misclassified by the trained model, and so they are

likely outliers. This property facilitates the supervised or automatic detection and removal of outliers, as shown in (Matic et al, 1992).

Geometrical interpretation of the Lagrange multipliers

In this section the geometrical interpretations of the possible values of the α_i Lagrange multipliers will be considered. For that purpose the following theorem, known as Karush-Kuhn-Tucker (KKT) conditions (Fletcher, 1987; Nocedal and Wright, 2006), will be extremely useful:

Theorem 1. KKT conditions (first-order optimality conditions)*: Consider the following general constrained optimization problem:*

$$\min_{x \in R^n} f(x)$$
$$subject\ to$$
$$c_i(x) = 0 \quad i \in E$$
$$c_i(x) \geq 0 \quad i \in I$$

where f and c_i are smooth, real valued, and continuously differentiable functions, and E and I are finite sets of indices for the equality and inequality constraints respectively. The Lagrangian of f is expressed as

$$L(x, \alpha) = f(x) - \sum_{i \in E \cup I} \alpha_i c_i(x)$$

If x^* is a local solution of the optimization problem and the LICQ[1] holds at x^*, then there is a Lagrange multiplier vector $\boldsymbol{\alpha}^*$ such that the following conditions are satisfied at $(\mathbf{x}^*, \boldsymbol{\alpha}^*)$:

$$\nabla_x L(\mathbf{x}^*, \boldsymbol{\alpha}^*) = 0$$
$$c_i(\mathbf{x}^*) = 0 \quad for\ i \in E$$
$$c_i(\mathbf{x}^*) \geq 0 \quad for\ i \in I$$
$$\alpha_i^* \geq 0 \quad for\ i \in I$$
$$\alpha_i^* c_i(\mathbf{x}^*) = 0 \quad for\ i \in E \cup I \quad (complementarity\ conditions)$$

Following the result of this theorem, let us state the complementary KKT conditions at the solution of the primal problem (3):

$$\alpha_i \left[y_i \left(\langle w, \phi_i \rangle + b \right) - 1 + \xi_i \right] = 0 \qquad i = 1,..,n \qquad (7.1)$$
$$\mu_i \xi_i = 0 \qquad i = 1,..,n \qquad (7.2)$$

[1] LICQ stands for Linear Independence Constraint Qualification. It basically means that the active constraint gradients are linearly independent. For details please refer to chapter 12 of (Nocedal and Wright, 2006).

Using conditions (4) and (7), the implications for the following three possible cases of every α_i multiplier will be analyzed:

- $\alpha_i = 0$
- $0 < \alpha_i < C$
- $\alpha_i = C$

If $\alpha_i = 0$, it is obvious from (4.3) that $\mu_i = C$, and so from (7.2) it's concluded that $\xi_i = 0$ and subsequently $y_i\left(\langle w, \phi_i \rangle + b\right) \geq 1$. Obviously, the training pattern is correctly classified and it could either lie or not lie on the margin.

Assuming that $0 < \alpha_i < C$, condition (7.1) implies that $y_i\left(\langle w, \phi_i \rangle + b\right) - 1 + \xi_i = 0$. Because condition (4.3), $\mu_i > 0$ and consequently $\xi_i = 0$ as before. As a result, $y_i\left(\langle \mathbf{w}, \phi_i \rangle + b\right) = 1$. The clear geometrical interpretation in this case is that the corresponding support pattern is correctly classified and it lies on the margin.

Finally, if $\alpha_i = C$ we again have $y_i\left(\langle w, \phi_i \rangle + b\right) - 1 + \xi_i = 0$; however, now condition (4.3) implies that $\mu_i = 0$, and so $\xi_i \geq 0$. As a result, $y_i\left(\langle \mathbf{w}, \phi_i \rangle + b\right) \leq 1$. The geometric interpretation in this case is that either the support pattern is correctly classified and lies on the margin, or it is correctly classified but it lies between the margin and the hyperplane, or it is incorrectly classified. If the latter is the case, it could be worthy to consider the possibility of this pattern to be an outlier.

These are important results offering a great insight into the SVM approach, so they are summarized in the following table:

Case	Implications	Further Implication	Interpretation
$\alpha_i = 0$	$\mu_i = C; \xi_i = 0$	$y_i d\left(\phi(x_i)\right) \geq 1$	x_i is correctly classified (it can lies or not on the margin).
$0 < \alpha_i < C$	$\mu_i > 0; \xi_i = 0$	$y_i d\left(\phi(x_i)\right) = 1$	x_i is correctly classified and it lies on the margin.
$\alpha_i = C$	$\mu_i = 0; \xi_i \geq 0$	$y_i d\left(\phi(x_i)\right) \leq 1$	One of the following options: • x_i is correctly classified and it lies on the margin. • x_i is correctly classified and it lies between the decision surface and the margin. • x_i is incorrectly classified (probably an outlier)

Table 1. Geometrical interpretation of the α Lagrange multipliers.

Estimating the bias

We already know how to calculate the value of the normal vector **w**, but the value of the bias *b* remains unknown. There are two approaches to deal with this problem, and both appear in (Vapnik, 1982). The first approach consists on fixing *b* a priori. This variant is called "Generalized Portrait Technique". However, if *b* is fixed a priori there is no guarantee that the maximum margin hyperplane is always obtained. The second approach consists on solving the dual optimization problem to obtain a solution for **w** independent of the bias, and then estimate *b* in a subsequent step.

To estimate the value of *b* corresponding to a certain value of **w** the geometrical interpretations from previous section will be helpful: Let us consider α_i such that $0 < \alpha_i < C$. As before, we have the following equation:

$$y_i\left(\langle \mathbf{w}, \phi_i \rangle + b\right) - 1 = 0 \qquad (8)$$

Therefore *b* can be estimated from any support vector such that $\alpha_i < C$ as follows:

$$b = \frac{1}{y_i} - \langle w, \phi_i \rangle = \frac{1}{y_i} - \sum_{j=1}^{n} \alpha_j y_j \langle \phi_j, \phi_i \rangle \qquad (9)$$

In (Vapnik, 1982) it is suggested to take the average of *b* values estimated from two support vectors, each from one different class, to have a more accurate estimation. If we denote the two support vectors by $\mathbf{x}_A$ and $\mathbf{x}_B$, it's straightforward how to reach from (9) the expression for *b* in that case:

$$b = -\frac{1}{2} \sum_{j=1}^{n} \alpha_j y_j \left(\langle \phi_j, \phi(\mathbf{x}_A) \rangle + \langle \phi_j, \phi(\mathbf{x}_B) \rangle \right) \qquad (10)$$

Introducing the mapping kernels

If ϕ is the identity function (i.e. $H = \chi$) or it is easy to deal with, then all previous calculations are straightforward. However, the mapping function ϕ is usually unknown or difficult to deal with. Because only the dot products in *H* are involved in the training and evaluation of the SVM classifier, kernel functions $k : \chi \times \chi \to R$ such that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ are used to avoid the direct evaluation of ϕ. By introducing a kernel function we end with the following expression for the SVM classifier:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b \right) \qquad (11)$$

Consequently, the SVM classifier can be trained and evaluated using the original training data and any appropriate kernel function, without explicitly working with the representations in the

feature space H. In the following paragraphs $k_{i,j}$ will be used sometimes to denote the value of $k(x_i, x_j)$ in order to simplify notation.

A natural question is what functions k are feasible kernels for using in the SVM framework. It turns out that a function $k : \chi \times \chi \to R$ can be used as a kernel if for all $m \in \mathrm{N}$ and all $x_1, x_2, ..., x_m \in \chi$ the resulting mxm matrix K with elements $K_{i,j} = k(x_i, x_j)$ is a positive semidefinite and symmetric matrix.[2] It is interesting to note that this condition implies $k(x, x) \geq 0$ for all $x \in \chi$ (positivity on the diagonal).

A theoretical approach to this requirement on positive semidefiniteness and two different interpretations on mapping kernels can be read in chapter 2 of (Scholkpof and Smola, 2002). For the introductory purposes of this document a list of the most common kernels and their equivalences with formulation of other machine learning models will suffice:

Kernel	Equivalent model	Notes
$k(x_i, x) = \langle x_i, x \rangle$	Linear classifier (Perceptrons).	
$k(x_i, x) = (\langle x_i, x \rangle + 1)^d$	Polynomial of degree *d*.	
$k(x_i, x) = \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right)$	Gaussian Radial Basis Functions.	The support patterns correspond to class centers.
$k(x_i, x) = \tanh(\langle x_i, x \rangle - \theta)$	Multi Layer Perceptron with one hidden layer (Sigmoid kernel).	The number of support vector corresponds to the number of units in the hidden layer. The support patterns are the weights of the first layer and the values of α_i are the weights of the second layer.

Table 2. Most commonly used kernels.

2.3. The double dual optimization problem.

The Lagrange technique can be applied again, this time to the dual optimization problem. For avoiding writing large expressions with sum signs the dual problem is restated here as a minimization problem in matrix terms:

[2] The expression "positive semidefinite" is used here as defined in Matrix Theory, i.e. $\mathbf{x}^T K \mathbf{x} \geq 0$ for all $\mathbf{x} \in R^m$. Please, take care that in (Scholkpof and Smola, 2002) and other references as well the term "positive definite" is used instead with the same meaning.

$$\begin{aligned}
&\min_{\alpha} \; -W(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{1} \\
&\textit{subject to}: \\
&\quad \alpha_i \geq 0 \qquad\qquad i = 1,..,n \\
&\quad C - \alpha_i \geq 0 \qquad i = 1,..,n \\
&\quad \boldsymbol{\alpha}^T \mathbf{y} = 0
\end{aligned} \tag{12}$$

where:

$$\begin{aligned}
&\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n)^T \\
&\mathbf{1} = \textit{identity vector of size } n \\
&\mathbf{0} = \textit{zero vector of size } n \\
&Q_{i,j} = y_i y_j k(x_i, x_j) = \mathbf{y}^T K \mathbf{y} \\
&\mathbf{y} = (y_1, y_2, ..., y_n)^T
\end{aligned}$$

The Lagrangian for this problem can be written as:

$$L(\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\mu}, \beta) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{1} - \boldsymbol{\delta}^T \boldsymbol{\alpha} - \boldsymbol{\mu}^T (C\mathbf{1} - \boldsymbol{\alpha}) - \beta \boldsymbol{\alpha}^T \mathbf{y}$$

where

$$\begin{aligned}
&\boldsymbol{\delta} = (\delta_1, \delta_2, ..., \delta_n)^T \\
&\boldsymbol{\mu} = (\mu_1, \mu_2, ..., \mu_n)^T \\
&\beta \in \Re
\end{aligned}$$

Consequently, a new equivalent optimization problem (called the **double dual problem**) can be stated by minimizing the Lagrangian $L(\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\mu}, \beta)$ with respect to $\boldsymbol{\alpha}$ and maximizing it with respect to the new Lagrange multipliers:

$$\min_{\alpha} \max_{\delta,\mu,\beta} L(\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\mu}, \beta) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{1} - \boldsymbol{\delta}^T \boldsymbol{\alpha} - \boldsymbol{\mu}^T (C\mathbf{1} - \boldsymbol{\alpha}) - \beta \boldsymbol{\alpha}^T \mathbf{y}$$

subject to:

$$\begin{aligned}
&\alpha_i \geq 0 \qquad i = 1,..,n \\
&C - \alpha_i \geq 0 \qquad i = 1,..,n \\
&\boldsymbol{\alpha}^T \mathbf{y} = 0 \\
&\delta_i \geq 0 \qquad i = 1,..,n \\
&\mu_i \geq 0 \qquad i = 1,..,n \\
&\beta \in \Re
\end{aligned} \tag{13}$$

The following KKT conditions are satisfied at an optimum solution of this problem:

$$
\begin{aligned}
&\nabla_{\alpha} L(\boldsymbol{\alpha}, \boldsymbol{\delta}, \boldsymbol{\mu}, \beta) = 0 \\
&\alpha_i \geq 0 \qquad\qquad i = 1..n \\
&C - \alpha_i \geq 0 \qquad i = 1..n \\
&\boldsymbol{\alpha}^T \mathbf{y} = 0 \\
&\delta_i \alpha_i = 0; \qquad i = 1..n \\
&\mu_i (C - \alpha_i) = 0; \quad i = 1..n \\
&\beta \boldsymbol{\alpha}^T \mathbf{y} = 0
\end{aligned}
\tag{14}
$$

The gradient of the Lagrangian is easily obtained as:

$$\nabla_{\alpha} L(\alpha, \delta, \mu, \beta) = Q\alpha - 1 - \delta + \mu - \beta y = 0$$

Every component of the above gradient is expressed by:

$$
\begin{aligned}
\frac{\partial L(\alpha, \delta, \mu, \beta)}{\partial \alpha_i} &= \sum_{j=1}^{n} \alpha_j y_i y_j k_{i,j} - 1 - \delta_i + \mu_i - \beta y_i \\
&= \sum_{j=1}^{n} \alpha_j y_i y_j k_{i,j} - y_i^2 - \delta_i + \mu_i - \beta y_i \\
&= \left(\sum_{j=1}^{n} \alpha_j y_j k_{i,j} - y_i - \beta \right) y_i - \delta_i + \mu_i \\
&= (F_i - \beta) y_i - \delta_i + \mu_i = 0
\end{aligned}
\tag{15}
$$

where

$$F_i = \sum_{j=1}^{n} \alpha_j y_j k_{i,j} - y_i = d(\phi_i) - b - y_i \tag{16}$$

There are three different important cases for the values of each Lagrange multiplier α_i. Every case will give us a clear geometric interpretation for the relationship between the corresponding training pattern and the hyperplane, using the previous KKT conditions.

To start this analysis let us assume that a particular α_i is equal to cero. It then follows that $\mu_i = 0$. Taking into account the possible values for δ_i and μ_i in the expression of the corresponding component of the gradient we reach into $(F_i - \beta) y_i \geq 0$.

In the other extreme, if $\alpha_i = C$, then $\delta_i = 0$ and the component of the gradient now fulfills $(F_i - \beta) y_i \leq 0$.

If $0 \le \alpha_i \le C$ then $\mu_i = 0$ and $\delta_i = 0$. It then follows that $F_i - \beta = 0$. This expression can be rewritten as $d(\phi_i) - y_i - b - \beta = 0$. As seen in the previous section, in the case of a non-bounded α_i the corresponding pattern lies on the margin. This means that $d(\phi_i) = y_i$ and it's thus concluded that $\beta = -b$ when $0 \le \alpha_i \le C$. Because the value of β is unique for the whole model, it is then concluded that the bias b is equal to $-\beta$ in general. This is an important result that will be useful in the following sections of this tutorial.

The results are summarized in the following table:

Case	Implications	Further implications
$\alpha_i = 0$	$\delta_i \ge 0; \mu_i = 0$	$(F_i - \beta) y_i \ge 0$
$0 < \alpha_i < C$	$\delta_i = 0; \mu_i = 0$	$F_i - \beta_i = 0$ $\beta = -b$
$\alpha_i = C$	$\delta_i = 0; \mu_i \ge 0$	$(F_i - \beta) y_i \le 0$

Table 3. Interpretation of the α Lagrange multipliers.

The results from this section will be used in (Keerthi, et al, 2001) to establish a variation on SMO, which will be summarized on section IV.

III. Sequential Minimal Optimization

The classical approach to solve the dual optimization problem is to use some numerical QP optimization tool. This approach is cumbersome or impractical for dealing with huge data sets, both because it's time consuming and it also requires a significant amount of memory. Several techniques have been devised to deal with this limitation. Most of them reduce the size of the QP problem by iteratively working with smaller QP sub-problems. In a process called "chunking" (Vapnik, 1982), the algorithm takes advantage of the fact that the value of the quadratic part of the SVM optimization problem remains the same after removing the rows and columns corresponding to non-support vectors (because $\alpha_i = 0$ for those patterns). The chunking algorithm reduces the memory requirements to approximately the square of the number of support vectors by iteratively solving variable sized QP sub-problems. Although chunking is a useful technique it still requires a decent amount of memory if there are many support vectors (in case of huge problems). The work by (Osuna et al, 1997) proposes an iterative solving technique very similar to chunking but with the added property of maintaining a constant size for the QP sub-problems regardless of the size of the overall QP problem.

The previously mentioned variants use numerical optimization packages for solving inner QP sub-problems. The Sequential Minimal Optimization (SMO) algorithm proposed in (Platt, 1999a) avoids working with numerical QP routines by analytically solving a large number of small optimization sub-problems that involves only two Lagrange multipliers at the time. That is the smallest possible quantity of multipliers to work with because the linear equality constraint $\sum_{i=1}^{n} \alpha_i y_i = 0$ in the original QP problem.

The main idea is to depart from a feasible vecctor of Lagrange multipliers (all the multipliers equal cero at the beginning) and keep improving the overall objective function by adjusting two Lagrange multipliers at the time. At every step the SMO algorithm analytically solves a QP sub-problem for the two chosen Lagrange multipliers and updates the SVM model accordingly. The method's size is linear in the quantity of training patterns. The time complexity is reported to be between linear and quadratic in the size of the training set for various datasets.

It's important to note here that Platt assumes the following model for the classifier instead of (1):

$$d(x) = \langle w, \phi(x) \rangle - b \qquad (17)$$

Of course, the whole framework could be derived for the model as expressed in (1), but that effort would mostly confuse people that read papers on SMO, so from now on the SVM model will be fixed to the one posted by Platt.

The algorithm involves three important components:

- An analytic solution to the optimization problem for the two chosen multipliers.
- A heuristic for choosing the two multipliers to optimize at a given step.
- A step to compute the offset *b*.

A description of these three components and a pseudo-code for the whole algorithm are included in the following sections.

3.1. Analytic optimization for two multipliers

Maintining the constraints

The multipliers chosen by the heuristic criteria mentioned above are denoted as α_1 and α_2 without loss of generality. In the following, α_1^{old} and α_2^{old} denotes the values of α_1 and α_2 at the beginning of the update process, and α_1^{new} and α_2^{new} denotes their new optimized values. To maintain the linear equality constraint of the dual optimization problem (5) the expression $y_1\alpha_1 + y_2\alpha_2$ should remain constant after updating α_1 and α_2. This can be equivalently expressed by

$$\alpha_1^{old} + s\alpha_2^{old} = \alpha_1^{new} + s\alpha_2^{new} = \gamma \qquad (18)$$

where $s = y_1 y_2$.

Equation (18) together with the box constraints $0 \le \alpha_i \le C$ implies that the updated values of α_1 and α_2 must remain in a segment of a line as shown in the following figure.

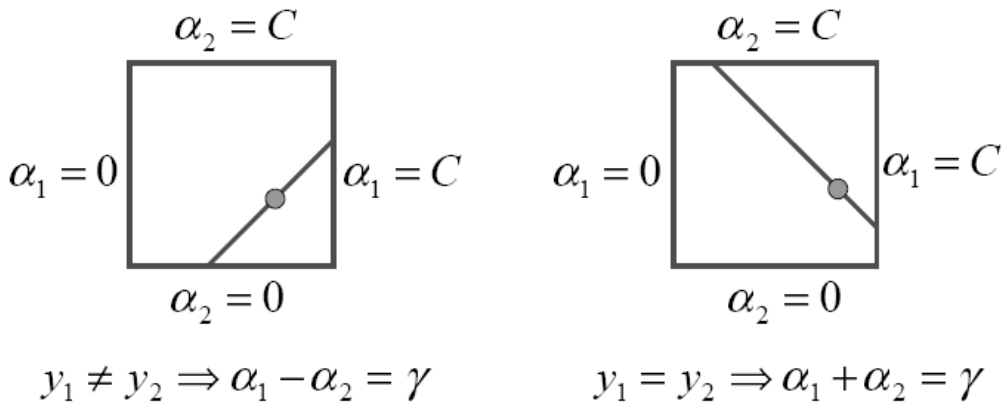

Figure 3: The segment that fulfill the linear and box constraints.
Picture copied from (Platt, 1999a)

The main idea is to solve the optimization sub-problem in terms of α_2 restricted to the whole line $\alpha_1 + s\alpha_2 = \gamma$; clip the so obtained α_2 optimal value if it is out of the interval (0, *C*); and then calculate the new optimal value for α_1.

The lower and upper bounds on the α_2 multiplier are very easy to determine using equation (18). If $y_1 \neq y_2$ the bounds are as follows:

$$L = \max\left(0, \alpha_2^{old} - \alpha_1^{old}\right)$$
$$H = \min\left(C, \alpha_2^{old} - \alpha_1^{old} + C\right) \qquad (19)$$

In the case $y_1 = y_2$ the bounds are:

$$L = \max\left(0, \alpha_2^{old} + \alpha_1^{old} - C\right)$$
$$H = \min\left(C, \alpha_2^{old} + \alpha_1^{old}\right) \qquad (20)$$

In (Platt, 1999b) two general expressions (independent of the values of y_1 and y_2) for these bounds are given:

$$L = \max\left\{0, \alpha_2^{old} + s\alpha_1^{old} - \frac{s+1}{2}C\right\}$$
$$H = \min\left\{C, \alpha_2^{old} + s\alpha_1^{old} - \frac{s-1}{2}C\right\} \qquad (21)$$

Updating the Lagrange multipliers

The SMO technique makes use of the equality (18) to reach an expression for the objective function $W(\alpha)$ that depends only on α_2. Considering that only α_1 and α_2 will be used for maximization of W, and that the kernel *k* is symmetric, we can rewrite the objective function of the dual optimization problem as follows:

$$W(\alpha) = \left[\alpha_1 + \alpha_2 + \sum_{i=3}^{n}\alpha_i\right] -$$
$$-\frac{1}{2}\left[\alpha_1^2 k_{1,1} + \alpha_2^2 k_{2,2} + 2\alpha_1\alpha_2 y_1 y_2 k_{1,2} + 2\alpha_1 y_1 \sum_{j=3}^{n}\alpha_j y_j k_{1,j} + 2\alpha_2 y_2 \sum_{j=3}^{n}\alpha_j y_j k_{2,j} + \sum_{i,j=3}^{n}\alpha_i\alpha_j y_i y_j k_{i,j}\right]$$

Because all the Lagrange multipliers except α_1 and α_2 are considered constants for a particular SMO optimization step, the objective function can be expressed as:

$$W(\alpha_1, \alpha_2) = \alpha_1 + \alpha_2 - \frac{1}{2}\alpha_1^2 k_{1,1} - \frac{1}{2}\alpha_2^2 k_{2,2} - \alpha_1\alpha_2 s k_{1,2} - \alpha_1 y_1 v_1 - \alpha_2 y_2 v_2 + const \qquad (22)$$

where

$$v_i = \sum_{j=3}^{n}\alpha_j y_j k_{i,j} = d^{old}(x_i) + b^{old} - \alpha_1^{old} y_1 k_{1,i} - \alpha_2^{old} y_2 k_{2,i} \qquad (23)$$

The main idea now is to make the objective function W to depend only on the second Lagrange multiplier to obtain its maximum across the line $\alpha_1 + s\alpha_2 = \gamma$. Substituting $\gamma - s\alpha_2$ for α_1 in $W(\alpha_1, \alpha_2)$ the following expressions are obtained for $W(\alpha_2)$ and its derivatives:

$$W(\alpha_2) = \gamma - s\alpha_2 + \alpha_2 - \frac{1}{2}(\gamma - s\alpha_2)^2 k_{1,1} - \frac{1}{2}\alpha_2^2 k_{2,2} - s(\gamma - s\alpha_2)\alpha_2 k_{1,2} - (\gamma - s\alpha_2) y_1 v_1 - \alpha_2 y_2 v_2 + const \tag{24}$$

$$\frac{\partial W(\alpha_2)}{\partial \alpha_2} = s(\gamma - s\alpha_2)k_{1,1} - \alpha_2 k_{2,2} + \alpha_2 k_{1,2} - s(\gamma - s\alpha_2)k_{1,2} + y_2 v_1 - s - y_2 v_2 + 1 = (2k_{1,2} - k_{1,1} - k_{2,2})\alpha_2 + s(k_{1,1} - k_{1,2})\gamma + (v_1 - v_2)y_2 - s + 1 \tag{25}$$

$$\frac{\partial W(\alpha_2)}{\partial^2 \alpha_2} = 2k_{1,2} - k_{1,1} - k_{2,2} \tag{26}$$

If $\frac{\partial W(\alpha_2)}{\partial^2 \alpha_2} < 0$ then a maximum exists. In that case, from $\frac{\partial W(\alpha_2)}{\partial \alpha_2} = 0$ and equation (23) the following expression is reached for an optimum value of α_2:

$$\alpha_2^* = \alpha_2^{old} - \frac{y_2(E_1 - E_2)}{2k_{1,2} - k_{1,1} - k_{2,2}} \tag{27}$$

where $E_i = d^{old}(x_i) - y_i$. Here $d^{old}(\mathbf{x})$ denotes the decision function with the old set of Lagrange multipliers.

Note that in (27) the value of α_2 that maximizes the objective function is denoted by α_2^* instead of α_2^{new}. The reason for this notation is that the box constraint $0 \le \alpha_2 \le C$ must be enforced before updating the two multipliers. To preserve the boundary constraint the optimal α_2 value is clipped as follows:

$$\alpha_2^{new} = \begin{cases} H & if\ \alpha_2^* \ge H \\ \alpha_2^* & if\ L < \alpha_2^* < H \\ L & if\ \alpha_2^* \le L \end{cases} \tag{28}$$

Finally, the new value of α_1^{new} is computed by

$$\alpha_1^{new} = \alpha_1^{old} + s(\alpha_2^{old} - \alpha_2^{new}) \tag{29}$$

If $\frac{\partial W(\alpha_2)}{\partial^2 \alpha_2} = 0$ (this could be the case if there are some repeated training patterns) then $W(\alpha_2)$ is obviously a line over α_2 and two cases may appear:

- If $|W(\alpha_2) - W(0)| > \varepsilon$, then α_2^{new} will be either 0 or C, the value that maximizes W. The value of α_1^{new} is again obtained from (29).

- If $|W(\alpha_2) - W(0)| \leq \varepsilon$, then no improvements are possible and the optimization step must be halted and initiate a search for two new multipliers. This case appears, for example, when $x_1 = x_2$, as both derivatives are then equal cero and W turns to be a constant all over the line.

3.2. Choosing two multipliers to optimize

There are two different heuristics for selecting the Lagrange multipliers to be optimized at certain step in Platt's SMO, structured as two nested loops. In this review just brief summaries of theses heuristics are given, as they are not required to conceptually understand the SMO approach and in fact different heuristics for the selection process are possible. For more details about the reasons behind this particular heuristics there is a nice explanation in (Platt, 1999a).

The outer loop searches for a multiplier that violates the conditions

$$\begin{aligned} \alpha_i = 0 \quad &\Rightarrow y_i f(\mathbf{x}_i) \geq 1 \\ 0 < \alpha_i < C &\Rightarrow y_i f(\mathbf{x}_i) = 1 \\ \alpha_i = C \quad &\Rightarrow y_i f(\mathbf{x}_i) \leq 1 \end{aligned} \tag{30}$$

within certain admissible error ε. It is interesting to note here that in (Platt, 1999a) these are called KKT conditions, although strictly speaking these expressions are in fact necessary conditions for the KKT conditions of the dual optimization problem.

The first time the outer loop runs it iterates over all the training data. After the first iteration the outer loop keeps alternating between a single pass over all the training patterns and multiple iterations over the patterns with non-bounded multipliers (i.e. $0 < \alpha_i < C$). The algorithm finishes when all the training patterns fulfill conditions (30) within the error ε.

Each time a multiplier is chosen by the outer loop then the inner loop iterates over the rest of the multipliers to find out one that should maximize the updating step of equation (27). Because evaluating the denominator of (27) for every candidate is time consuming this is done by keeping a cache of the error values for non-bounded multipliers, and choosing the multiplier that maximizes $|E_1 - E_2|$. Platt recommends for the inner loop to keep alternating between this and other approaches in case no positive progress is achieved.

3.3. Updating the bias

The value of the offset b is computed after each optimization step. For every α_i^{new}, $i \in \{1,2\}$, if $0 < \alpha_i^{new} < C$ then $d^{new}(x_i) = y_i$ (as shown in section 2), and a new value for b can be estimated as follows:

$$b_i^{new} = E_i + y_1\left(\alpha_1^{new} - \alpha_1^{old}\right)k(x_i, x_1) + y_2\left(\alpha_2^{new} - \alpha_2^{old}\right)k(x_i, x_2) + b^{old} \qquad (31)$$

Although (31) could seem a bit daunting at first, in fact it is easy to understand it if we recall that $E_i = d^{old}(x_i) - y_i$. The first two terms after E_i in (31) are just updating $d(x)$ with the new values of α_1 and α_2. The last term is removing the old bias from the decision function. Thus equation (31) can be read as $b_i^{new} = \langle w^{new}, \phi_i \rangle - y_i$, that is clearly true if the Lagrange multiplier is non-bounded. Though the expression used in (31) seems a bit complicated it is obviously faster to execute because it makes use of the error terms *E* that can be kept in a cache, so the decision function does not need to be evaluated.

If both multipliers are not at the bounds then $b_1^{new} = b_2^{new}$. If both are at the bounds but $L \neq H$, then b^{new} can be estimated by averaging the b_1^{new} and b_2^{new} obtained from formula (31).

3.4. Platt's SMO pseudo-code

To finish this section, and with the only purpose of making this tutorial fully comprehensive, the SMO pseudo-code is included. It resembles the pseudo-code shown in (Platt, 1999).

There are three fundamental functions to be implemented. Each one is listed here in an independent table. The **takeStep** function (shown in Table 4) contains the code that attempt to update the chosen pair of Lagrange multipliers. The **examineExample** function searches for a second Lagrange multiplier if the first multiplier (which it receives as a parameter) violates optimality conditions. It is described in Table 5. The main routine of SMO is a simple loop, shown in Table 6.

```
bool takeStep(i1, i2)
// Parameters i1, i2 denotes the indices of the multipliers.
{
        if (i1 = i2)
                return false

        //Check for different bounds on alpha_i2
        s ← yi1 yi2
        E1 = d(αi1) − yi1   //Check in error cache
        Compute L, H
        if (L = H)
                return 0
```

//Obtain a new value for alpha_i2

$\eta \leftarrow 2k(x_{i1}, x_{i2}) - k(x_{i1}, x_{i1}) - k(x_{i2}, x_{i2})$

if ($\eta < 0$)

{

$\alpha_{i2}^{new} \leftarrow \alpha_{i2} - \frac{y_{i2}(E_{i1} - E_{i2})}{\eta}$

if ($\alpha_{i2}^{new} < L$) $\alpha_{i2}^{new} \leftarrow L$

else if ($\alpha_{i2}^{new} > H$) $\alpha_{i2}^{new} \leftarrow H$

}

else

{

$L_{obj} \leftarrow W(\alpha_{i2} = L)$

$H_{obj} \leftarrow W(\alpha_{i2} = H)$

if ($L_{obj} > H_{obj} + \varepsilon$)

$\alpha_{i2}^{new} \leftarrow L$

else if ($L_{obj} < H_{obj} - \varepsilon$)

$\alpha_{i2}^{new} \leftarrow H$

else

$\alpha_{i2}^{new} \leftarrow \alpha_{i2}$

}

//Take care of numerical errors

if ($\alpha_{i2}^{new} < \varepsilon$)

$\alpha_{i2}^{new} = 0$

else if ($\alpha_{i2}^{new} > C - \varepsilon$)

$\alpha_{i2}^{new} = C$

if $\left|\alpha_{i2}^{new} - \alpha_{i2}\right| < \varepsilon(\alpha_{i2}^{new} + \alpha_{i2} + \varepsilon)$

return 0

//Update alpha_i1

$\alpha_{i1}^{new} = \alpha_{i1} + s(\alpha_{i2} - \alpha_{i2}^{new})$

Update bias to reflect change in Lagrange multipliers

Update weight vector to reflect change in alpha_i1 & alpha_i2, if linear SVM

Update error cache using new Lagrange multipliers

Store alpha_i1 in the alpha array

Store alpha_i2 in the alpha array

return true

}

Table 4. Function takeStep tries to update two chosen Lagrange multipliers.

```
bool examineExample(i2)
{
        E_2 = d(α_i2) − y_i2    //Check in error cache

        //If alpha_i2 violates optimality conditions then look for an appropriate alpha_i1.
        r_2 = E_i2(y_i2)
        if ((r_2 < −ε and α_i2 < C) or (r_2 > ε and α_i2 > 0))
        {
                if (number of non-zero & non-C alpha multipliers > 1)
                {
                        i1 = result of second choice heuristic
                        if takeStep(i1,i2)
                                return true
                }

                loop over all non-zero and non-C alpha, starting at random point
                {
                        i1 = identity of current alpha
                        if takeStep(i1,i2)
                                return true
                }
                loop over all possible i1, starting at a random point
                {
                        i1 = loop variable
                        if takeStep(i1,i2)
                                return true
                }
        }
        return false
}
```

Table 5. Function examineExample implements the inner loop of the SMO algorithm.

```
initialize alpha array to all zero
initialize bias to zero
numChanged ← 0;
examineAll ← 1;
while (numChanged > 0 or examineAll)
{
        numChanged = 0;
        if (examineAll)
                loop I over all training examples
                        numChanged += examineExample(I)
        else
                loop I over examples where alpha is not 0 & not C
                        numChanged += examineExample(I)
        if (examineAll = 1)
                examineAll ← 0
        else if (numChanged = 0)
                examineAll ←1
}
```

Table 6. Main routine of the SMO algorithm.

IV. An Improvement to Platt's SMO Algorithm.

Although Platt's SMO has being shown to be a fast training algorithm for the SVM classifier, there is a source of inefficiency in SMO related to the way the bias b is computed and used. This inefficiency and an approach to improve the original SMO formulation are explained in detail in (Keerthi, et al, 2001). In this section a brief summary of the main ideas exposed in that paper is offered. However, the presentation doesn't follow exactly the same line as in (Keerthi, et al, 2001), mostly because the use of the error terms E_i to justify the result presented in this subsection.

A remark is important here to easily understand this content: As it was proved at the end of section 2.3, if the classification model (1) is used then $b = -\beta$. Because in SMO the model is the one stated in (17), it turns out that in SMO the bias b of the solution is equal to the β multiplier.[3] That's, why in the SMO references, b and β are used interchangeably.

Whenever two Lagrange multipliers α_1 and α_2 are updated in the original SMO algorithm, the bias is computed. The value of the bias is used whenever the algorithm is looking for Lagrange multipliers that violate the optimality conditions stated in (30). In (Keerthi, et al, 2001) it is argued that this process is not optimal. It is possible that the original SMO algorithm could fail to detect optimality due to an incorrect estimation of b. In such situations the original SMO implementation could spend considerable time performing a wasteful search for a second index to execute the next iteration.

4.1. Overcoming Platt's SMO inefficiency

To overcome the previously stated handicap, it is possible to define new criteria for determining optimality of the Lagrange parameters. A first step in that direction is to realize that the optimality conditions (14) are equivalent to the set of conditions stated in Table 3. Those conditions are repeated below:

- If $\alpha_i = 0$, then $\delta_i \geq 0; \mu_i = 0; (F_i - \beta) y_i \geq 0$ (32a)
- If $0 < \alpha_i < C$, then $\delta_i = 0; \mu_i = 0; F_i - \beta_i = 0;$ (32b)
- If $\alpha_i = C$, then $\delta_i = 0; \mu_i \geq 0; (F_i - \beta) y_i \leq 0$ (32c)

The following index sets are defined in order to classify the previous cases according to the values of the Lagrange multipliers and the class of the corresponding patterns:

- $I_0 = \{i : 0 < \alpha_i < C\}$
- $I_1 = \{i : \alpha_i = 0; y_i = 1\}$
- $I_2 = \{i : \alpha_i = C; y_i = -1\}$

[3] Is this the reason why Platt changed the sign of the bias in the SMO model?

- $I_3 = \{i : \alpha_i = C; y_i = 1\}$
- $I_4 = \{i : \alpha_i = 0; y_i = -1\}$

Taking into account the expression for the error terms ($E_i = d(x_i) - y_i$), and the geometrical interpretations of the Lagrange multipliers (Table 1), it turns out that whenever optimality conditions hold then:

- $E_i \geq 0$ for those α_i such that $i \in I_0 \cup I_1 \cup I_2$ (33a)
- $E_i \leq 0$ for those α_i such that $i \in I_0 \cup I_3 \cup I_4$ (33b)

Because $F_i = E_i + \beta$ for every Lagrange multiplier α_i, conditions (33) implies that at optimality the following condition is true:

$$\forall i \in I_0 \cup I_3 \cup I_4, \forall j \in I_0 \cup I_1 \cup I_2,\ F_i \leq F_j \qquad (34)$$

If $b_{low} = \max\{F_i : i \in I_0 \cup I_3 \cup I_4\}$ and $b_{up} = \min\{F_i : i \in I_0 \cup I_1 \cup I_2\}$ the condition (34) can be rewritten as

$$b_{low} \leq b_{up} \qquad (35)$$

On the other hand, if condition (35) is satisfied, it is not difficult to realize that for every value of β such that $b_{low} \leq \beta \leq b_{up}$ the conditions (32) are satisfied, which in turn means that the original optimality conditions (14) are satisfied (provided that δ and μ have appropriate values in each case). The previous rationale proves the following

> **Result:** If $b_{low} = \max\{F_i : i \in I_0 \cup I_3 \cup I_4\}$ and $b_{up} = \min\{F_i : i \in I_0 \cup I_1 \cup I_2\}$, the optimality conditions (14) hold for a particular set of Lagrange parameters $\boldsymbol{\alpha}$ if and only if $b_{low} \leq b_{up}$.

The condition just obtained, besides being equivalent to the original optimality condition, has a useful characteristic: it doesn't depend on a particular value of β. The work in (Keerthi, et al, 2001) takes advantage of this fact to establish two variants of SMO that maintain two thresholds parameters (b_{low} and b_{up}), and use condition (34) to check for optimality of the Lagrange multipliers.

Because the solution obtained is numerical, an approximate condition is used in practice instead of (35), adding a very small positive tolerance parameter τ:

$$b_{low} \leq b_{up} + 2\tau \qquad (36)$$

The optimality condition in (34) is changed as well:

$$\forall i \in I_0 \cup I_3 \cup I_4, \forall j \in I_0 \cup I_1 \cup I_2,\ F_i \leq F_j + 2\tau \qquad (37)$$

4.2. The Modifications of SMO

Essentially, there are two modifications to Platt's SMO, called **SMO-Modification 1** and **SMO-Modification 2** respectively. In this subsection we will take a look first at the similarities between both variants, and afterwards the differences will be remarked. The pseudo-code for these modifications can be found in (Keerthi et al, 1999).

Similar to Platt's SMO, the outer loop, (the one that chooses α_2) spends the majority of its time examining the non-bounded ($\alpha_i \in I_0$) Lagrange multipliers; so a cache for F_i such that $\alpha_i \in I_0$ is maintained. The outer loop also resembles Platt's SMO in the way it alternates between examining all the patterns and examining only those with non-bounded Lagrange multipliers.

The values of b_{low} and b_{up}, and the corresponding indices *i_low* and *i_up*, are also maintained by the SMO modifications. These variables are updated inside the **takeStep** function but limited to the F_i values of the pair of patterns that made an improvement and the patterns in I_0 (using the cached F_i values). The F_i values of patterns not in I_0 are considered for updating b_{low} and b_{up}, and the corresponding indices, when they are tested for optimality inside the **examineExample** function.

When the outer loop is examining all the Lagrange multipliers the following procedure is followed by both modifications of SMO: A particular α_i checked in the **examineExample** function is a violator of the optimality conditions if one of the following conditions is met:

- $i \in I_0 \cup I_1 \cup I_2$ and $F_i < b_{low} - 2\tau$ (case 1)
- $i \in I_0 \cup I_3 \cup I_4$ and $F_i > b_{up} + 2\tau$ (case 2)

If case 1 is detected then α_i is paired with the multiplier at index *i_low* and the optimization process is done by calling **takeSetp**(α_i, *i_low)*. If case 2 is detected then the optimization is attempted through a call to **takeStep**(α_i, *i_up)*.

The difference between SMO-Modification 1 and SMO-Modification 2 is introduced when the outer loop is examining Lagrange multipliers with only non-bounded values. In that case, the SMO-Modification 1 keeps executing the same procedure as before, but SMO-Modification 2 goes a step further. It keeps choosing the worst violating pair as the next two Lagrange multipliers to optimize (i.e. those with indices *i_low, i_up)* until optimality holds for all Lagrange multipliers in I_0.

The experimental results presented in (Keerthi, et al, 2001) showed that both SMO modifications, and particularly the second one, had better performance than the original SMO algorithm for most of the data sets under consideration.

V. Conclusions and Further Readings

The present tutorial offers a comprehensive review of the SVM technique, the original SMO algorithm developed by Platt, and modifications and/or improvements made to this original SMO variant. SVM is a hot research topic at the moment, and I hope that newcomers to this interesting topic find the information presented here useful. The main goal of this tutorial is to offer the foundations for more advanced readings in the SVM area. If the present document is at least near that goal, the hours dedicated to the preparation of this work were definitely worthy.

There are several directions to take from this point, depending on the particular interests of the readers. To finish this tutorial, some suggestions about further readings will be offered.

Besides classification, support vector machines have being successfully applied to other types of problems as well. Two problems worth to mention here are regression and principal component analysis (PCA). There is a great introduction to both types of applications in (Scholkpof and Smola, 2002). That book also provides interesting links for an in depth study of these subjects.

There is a variant of SMO already designed for the regression case. More information about it can be found in (Smola and Schölkopf, 1998). A similar improvement as the one presented in section 4 of this tutorial was established for the regression case in (Shevade et al, 1999) and (Shevade et al, 2000). Readers interested in alternative criteria for working set selection, such as choosing more than two points to optimize and the usage of second order information to obtain faster convergence, are recommended to read (Fan et al, 2005).

The parameter C, as presented in this tutorial, determines a trade-off between a maximization of the margin and a minimization of training errors. However, there isn't any intuitive interpretation for C; which typically forces researchers to follow a computationally expensive cross-validation process. One particularly interesting variant of the SVM technique designed to offer a more intuitive interpretation to the trade-off parameter is the one called ν-SVM. In this approach C is replaced by a parameter ν and a new variable ρ is added to the optimization problem. The new parameter ν has an intuitive meaning: it is an upper bound on the fraction of margin errors and a lower bound on the fraction of support vectors for a solution. A detailed description of this approach can be read at (Scho:lkopf et al, 2000).

The algorithm introduced in this tutorial only solves binary classification problems. Several approaches have being proposed for dealing with multiple-class problems. A traditional approach in the Pattern Recognition field is to break the multiple-class problem into several binary classification sub-problems, following one of two feasible ways:

- The "one against many" approach, where each class is let aside and the remaining classes are merged. A SVM machine is trained for each of such sub-problems. The final classification of a new pattern corresponds to the decision function that offered the greatest value.

- The "one against one" approach, where if N classes are considered then $\frac{N(N-1)}{2}$ models are constructed and fitted to all possible two-class data sub-sets. This approach is more computationally expensive but typically offers a more accurate classification.

Besides the above mentioned approaches to multi-class classification problems, there is a SVM model proposed to solve the problem at once using a single optimization problem (Weston and Watkins, 1999). The former models assume that the patterns belong to only one class. A generalization to problems where patterns are allowed to belong to more than one class (multi-label problems) is offered in (Elisseeff and Weston, 2001).

For those interested in the theoretical basis of SVM, (Aronszajn, 1950) is a fundamental reference about the properties of reproducing kernels. A recent review on the mathematical foundations of machine learning can be found at (Cucker and Smale, 2001).

Because the space of input patterns are only required to be a set in the case of non-linear SVM, the SVM technique can be also applied to categorical data if we manage to find kernel functions to map those categorical data into appropriate feature spaces. Unfortunately I couldn't find a good reference on this particular topic at the moment of writing this tutorial.

Acknowledgements

The author would like to thanks professors Michael Georgiopoulos and Georgios Anagnostopoulos for their support and valuable suggestions. Thanks also to Chris Sentelle and Nicholas Shorter for their comments to improve this document.

References

Arfken, G., "Lagrange Multipliers." §17.6 in Mathematical Methods for Physicists, 3rd ed. Orlando, FL: Academic Press, pp. 945-950, 1985.

Aronszajn N., Theory of Reproducing Kernels, *Transactions of the American Mathematical Society*, Vol. 68, No. 3 pp. 337-404, 1950.

Barabino N., Pallavicini M., Petrolini A., Pontil M., and Verri A., Support vector machines vs multi-layer perceptrons in particle identification. In M. Verleysen, editor, *Proceedings ESANN*, pages 257 – 262, Brussels, 1999, D Facto.

Boser B. E., Guyon I. M. and Vapnik V., A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144-152, Pittsburg, PA, ACM, July 1992.

Burges C. J. C., *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery, Vol. 2, Number 2, p. 121-167, © Kluwer Academic Publishers, 1998.

Cortes C. and Vapnik V., Support Vector Networks, Machine Learning, 20:273 – 297, 1995.

Cover T. M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Transactions on Electronic Computers, 14:326 – 334, 1965.

Cucker F. and Smale S., *On the Mathematical Foundations of Learning*, Bulletin of the American Mathematical Society, volume 39, number 1, pages 1 – 49, 2001.

Elisseeff A. and Weston J., Kernel methods for multi-labelled classification and categorical regression problems. Technical Report, Biowulf Technologies, New York, 2001.

Fan R., Chen P. and Lin C., Working Set Selection Using Second Order Information for Training Support Vector Machines, The Journal of Machine Learning Research, Vol. 6, pp. 1889 – 1918, 2005.

Fletcher R., Practical Methods of Optimization, 2nd Edition, John Wiley and Sons, 1987.

Hua S. and Sun Z., A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach, *Journal of Molecular Biology*, 308:397 – 407, 2001.

Keerthi S. S., Shevade S. K., Bhattacharyya C., and Murthy K. R. K., Improvements to Platt's SMO Algorithm for SVM Classifier Design, Neural Computation, 13: 637-649, 2001.

Keerthi S. S., Shevade S. K., Bhattacharyya C., and Murthy K. R. K., Improvements to Platt's SMO Algorithm for SVM Classifier Design, Tech. Report CD-99-14, Control Division, Dept. of Mechanical and Production Engineering, National University of Singapore, 1999.

Matic M., Guyon I., Bottou L., Denker J., and Vapnik V., Computer-aided cleaning of large databases for character recognition. In *Digest ICPR*. ICPR, Amsterdam, August 1992.

Nocedal J. and Wright S. J., Numerical Optimization, 2nd Edition, Springer, 2006.

Onoda T., Ra:tsch G., and Mu:ller K-R, A non-intrusive monitoring system for household electric appliances with inverters. *In Proceedings the Second International ICSC Symposium on Neural Computation,* Berlin, 2000.

Osuna, E., Freund, R.and Girosi, F., "Improved Training Algorithm for Support Vector Machines," Proc. IEEE NNSP '97, 1997.

Platt J. C., Fast training of Support Vector Machines using sequential minimal optimization. In B. Scho:lkopf, C. J. C. Burges and A. J. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185-208, Cambridge, MA, MIT Press, 1999a.

Platt, J.C., "Using Analytic QP and Sparseness to Speed Training of Support Vector Machines", NIPS 11, pp. 557-563, 1999b.

Roobaert D., and Van Hulle M. M., View-based 3D object recognition with support vector machines. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, IEEE, 1999.

Scho:lkopf B. and Smola A. J., *Learning with Kernels*, MIT Press, 2002.

Sho:lkopf B., Smola A. J., Williamson R. C. and Bartlett P. L., New Support Vector Algorithms, Neural Computation, Vol. 12, No. 5, pp. 1207 – 1245, 2000.

Smola A. J. and Schölkopf B., A Tutorial on Support Vector Regression, Royal Holloway College, London, U.K., NeuroCOLT Tech. Rep. TR-98-030, 1998.

Shevade S. K., Keerthi S. S., Bhattacharyya C., and Murthy K. R. K., Improvements to the SMO Algorithm for SVM regression,

Shevade S. K., Keerthi S. S., Bhattacharyya C., and Murthy K. R. K., Improvements to the SMO Algorithm for SVM regression, Tech. Report CD-99-16, Control Division, Dept. of Mechanical and Production Engineering, National University of Singapore, 1999.

Vapnik V., *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, 1982.

Vapnik V., *Statistical Learning Theory*, Wiley-Interscience, 1998.

Weston J. and Watkins C., Multi-class support vector machines, In M. Verleysen, editor, Proceedings ESANN, Brussels, 1999.