# Getting to grips with Support Vector Machines: Theory

Steve Kroon[1] and Christian W. Omlin[2]

[1] Department of Statistics, University of Stellenbosch, Private Bag X1, Matieland, South Africa, 7602
kroon@sun.ac.za
[2] Department of Computer Science, University of the Western Cape, Bellville, South Africa, 7535
comlin@uwc.ac.za

**Abstract.** The support vector machine (SVM) is a technique for function estimation which was proposed in the early 1990s. The technique provided state-of-the-art performance on many problems familiar to the machine learning community, and has hence gained enormous popularity among them. Despite the solid theoretical justification for the SVM as the solution of a regularization problem, the SVM has not yet become a common topic for study in statistics. This paper aims to provide an introductory overview of the SVM. A followup paper will provide useful hints and suggestions for newcomers to the field. The focus is on the SVM classifier, although the support vector technique is also applicable to regression and other techniques.
**Keywords**: SVM, tutorial, primer, R, LIBSVM, classification

## 1 Introduction

Since its introduction in Boser, Guyon and Vapnik (1992), the SVM has proven to be a versatile technique, able to effectively handle a wide variety of tasks. Particularly in the case of classification and regression problems, good results have been achieved, often equalling or bettering the prediction accuracy of previous state-of-the-art methods (Schölkopf, 1999). From a classification perspective, it has been shown that the SVM asymptotically implements the Bayes classication rule (Lee, Lin and Wahba, 2001). This paper aims to introduce the SVM to the uninitiated. It will be followed by a paper discussing practical issues involved when applying SVMs.

The technique makes use of a number of mathematical concepts. Most concepts necessary will be described in the text. For more detail on unfamiliar mathematical concepts and techniques, see Kroon

(2003). For more on optimization with Lagrange multipliers, see e.g. Bertsekas (1982), Koo (1977), Panik (1976). Short introductions to the SVM algorithm are common in many papers utilizing the technique — some more extensive discussions can be found in Boser, Guyon and Vapnik (1992), Burges (1998), Campbell and Cristianini (1998), Cortes and Vapnik (1995), Guyon, Boser and Vapnik (1993), Haykin (1999), Schölkopf, Smola, Williamson and Bartlett (1998), Smola and Schölkopf (1998), Vapnik (1995), Vapnik, Golowich and Smola (1997), Weston (1999).

The SVM is introduced in stages. Firstly, Section 2 introduces the basic concept and derivation of the SVM for dichotomous (2-class) classification in the case of linearly separable data. After the basic algorithm is introduced, techniques for handling nonlinearity (Section 3) and inseparable data (Section 4) are considered. Here we also consider how the SVM can be viewed as the solution to a regularization problem. Polychotomous SVM classification is considered in Section 5. The application of SVMs to regression estimation is then shortly discussed in Section 6.

Note that no mention is made of the structural risk minimization (SRM) principle. This is an inductive principle from statistical learning theory which motivated the proposal of the SVM. Despite its importance from this perspective, the SVM can be considered from other viewpoints too, which we follow here. Those looking for information on the SVM from an SRM perspective are referred to Vapnik (1995).

Throughout, $a \cdot b$ denotes the inner product of (possibly infinite dimensional) vectors $a$ and $b$.

## 2 The Basic Concept

We begin by considering the 2-class classification problem. The concept of SVMs is derived from that of an optimal separating hyperplane, a hyperplane which separates the data points of two classes without error, if it is possible. Furthermore, an optimal separating hyperplane has maximal least distance to any data point of any such separating hyperplane (Boser, Guyon and Vapnik, 1992, Cortes and Vapnik, 1995, Vapnik, 1982). An hyperplane is an $n$-dimensional generalization of a line in two dimensions and a plane in three dimensions. More specifically, in an $n$-dimensional inner product space $G$, a hyperplane is the set $S$ of points $x$ satisfying

$$w \cdot x + b = 0$$

for some pair $(w, b) \in G \times \mathbb{R}$. Here $n$ can be finite, for example $G = \mathbb{R}^{12}$, or infinite. Each pair $(w, b)$ specifies a hyperplane in $G$. Note that $(cw, cb)$ specifies the same hyperplane for all non-zero values of $c$.

The SVM algorithm in the simplest case constructs a hyperplane (if possible) which separates the data in such a way that $w \cdot x + b > 0$ for all points in one class, and $w \cdot x + b < 0$ for all points in the other class. From this perspective, we can view the hyperplane specified by $(w, b)$ as a surface separating the two classes, and we call this hyperplane a separating hyperplane.

More formally, consider a training set $(x_1, y_1), \ldots, (x_l, y_l)$, where $l$ is the number of training examples, and $y_i \in \{-1, 1\}$ is the class label for $x_i \in \mathbb{R}^N$. In addition, we assume that the training data are linearly separable, i.e. that a separating hyperplane exists.

Thus, a $w \in \mathbb{R}^N$ and $b \in \mathbb{R}$ exist such that

$$y_i(w \cdot x_i + b) \geq 1, i \in \{1, \ldots, l\} \ , \tag{1}$$

with equality occuring in the case of at least one $x_i$. The equality follows since we can scale $(w, b)$ appropriately without changing the hyperplane. The set of hyperplanes $w \cdot x + b = 0$ satisfying the criteria above (called *canonical* separating hyperplanes) is the same as the set of all separating hyperplanes, but multiple representations of the same hyperplane are normalized to a single representation of the hyperplane in canonical form (Vapnik, 1995, Section 5.4.2).

It can easily be shown that even when only considering canonical separating hyperplanes, the existence of a single such hyperplane implies the existence of infinitely many other canonical separating hyperplanes. The next step is then selecting the canonical separating hyperplane for which the closest data point $x_i$ is furthest from the hyperplane. (If one requires that $b = 0$, then the $w$ generating such a hyperplane will be a Vapnik-Chervonenkis *generalized portrait* of the training set for $k = -1$ — see Vapnik (1982, Addendum I.2).)

Now, the distance of a point $x_i$ from the plane $w \cdot x + b = 0$ is, similarly to the distance of a point from a plane or line, $\frac{|w \cdot x_i + b|}{\|w\|}$. For all the canonical separating hyperplanes, the minimum of the numerator is 1 (from (1)), so the problem reduces to maximizing $\frac{1}{\|w\|}$, a quantity known as the *margin*. $\|w\|$ is also a measure of the *smoothness* (Girosi, Jones and Poggio, 1995, Smola and Schölkopf, 1998) of the function $w \cdot x + b$. Smoothness is also called flatness, or stability: the smaller $\|w\|$ is, the less $w \cdot x + b$ changes with a fixed change in $x$. From this perspective, one can view the problem as that of finding the canonical separating hyperplane exhibiting the least variance.

Now, maximizing the margin $\frac{1}{\|w\|}$ is equivalent to minimizing $\|w\|$ or $\frac{1}{2}\|w\|^2 = \frac{1}{2} w \cdot w$, which corresponds to finding the smoothest canonical separating hyperplane. Thus we have a constrained optimization problem:

$$\text{Minimize} \quad \tfrac{1}{2} w \cdot w \tag{2}$$

$$\text{subject to} \quad y_i(w \cdot x_i + b) \geq 1, i \in \{1, \ldots, l\} \ . \tag{3}$$

Introducing a vector of Lagrange multipliers $\alpha = (\alpha_1, \ldots, \alpha_l) \geq 0$ to handle the constraints (1), one obtains the primal Lagrangian

$$L_P(w, b, \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{l} \alpha_i \{ y_i(w \cdot x_i + b) - 1 \} \ .$$

Optimization theory tells us that the solution to the problem lies at the saddle point of the Lagrangian function $L_P$, which must now be minimized with respect to $w$ and $b$, and simultaneously maximized with respect to $\alpha$ (subject to $\alpha \geq 0$). Taking partial derivatives with respect to $w$ and $b$, and setting them to zero yields

$$\frac{\partial L_P(w, b, \alpha)}{\partial w} = w - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \ , \tag{4}$$

$$\frac{\partial L_P(w, b, \alpha)}{\partial b} = -\sum_{i=1}^{l} \alpha_i y_i = 0 \ . \tag{5}$$

Equation (4) shows that $w$ is a linear combination of the $x_i$'s, while (5) indicates that the sums of coefficients of data points from each class are equal. Furthermore, the Karush-Kuhn-Tucker (KKT) complementarity conditions between the $\alpha_i$ and their corresponding constraints can be exploited to find out more about the linear combination. In this case, the relevant KKT complementarity conditions, which hold at the solution, are

$$\alpha_i\{y_i(w \cdot x_i + b) - 1\} = 0, i \in \{1, \ldots, l\} \ . \tag{6}$$

Thus, $\alpha_i = 0$ unless $y_i(w \cdot x_i + b) - 1 = 0$; or, in other words, only those data points $x_i$ closest to the hyperplane (at a distance of $\frac{1}{\|w\|}$) can have nonzero Lagrange multipliers $\alpha_i$. Usually, these points form a small proportion of the data (Boser, Guyon and Vapnik, 1992), and so SVMs are said to learn a decision function with a *sparse* representation (Brown, Grundy, Lin, Cristianini, Sugnet, Manuel Ares, Jr. and Haussler, 1999, Campbell, 2000). These data points used to calculate $w$ via (4) are called the support vectors (SVs). It can furthermore be shown that removing any number of the non-SVs from the training set and retraining the SVM (solving the new optimization problem) yields the same values of $w$ and $b$ (Burges, 1998) — however, selecting these points necessitates training the SVM on the entire training set. (This does not render the result useless: it is the basis of the earliest results on generalization performance of SVMs.)

Substituting the equality constraints (4) and (5) into L, one obtains what is known as the *Wolfe dual* problem of maximizing

$$\begin{aligned}
&L_D(\alpha) \\
&= \frac{1}{2}(\{\sum_{i=1}^{l} \alpha_i y_i x_i\} \cdot \{\sum_{j=1}^{l} \alpha_j y_j x_j\}) - \sum_{i=1}^{l} \alpha_i \{y_i((\sum_{j=1}^{l} \alpha_j y_j x_j) \cdot x_i + b) - 1\} \\
&= \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j - b \sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i \\
&= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j
\end{aligned} \tag{7}$$

with respect to $\alpha$, subject to $\alpha \geq 0$ and (5) (The constraint in (4) falls away since $w$ is not present in (7)). The Wolfe dual problem is equivalent to the original optimization problem, in that the $\alpha$ obtained in both cases is identical. Since we can calculate $w$ and $b$

given $\alpha$, solving either the original or the Wolfe dual problem will suffice. Now, the expression (7) for $L_D$ is a quadratic form, and the constraints are all linear in the $\alpha_i$'s. This is then a standard quadratic programming problem, which can be optimized by various computer packages.

Suppose we have an $\alpha^o$ which maximizes $L_D$ subject to the given constraints. Let $w^o$ and $b^o$ be the parameters of the corresponding optimal hyperplane. It follows from (4) that $w^o = \sum_{i=1}^{l} \alpha_i^o y_i x_i$. In order to calculate $b^o$, however, one must make use of the KKT complementarity conditions (6). For any nonzero $\alpha_i^o$, it follows that

$$y_i(w^o \cdot x_i + b^o) = 1 \ ,$$

so that

$$b^o = y_i - w^o \cdot x_i \ , \tag{8}$$

since $y_i = \pm 1$. Solutions to the quadratic programming problem are usually determined by computers, so for reasons of numerical stability, $b^o$ is usually taken to be the mean of the values calculated using (8) for each nonzero $\alpha_i^o$ (Burges, 1998, Section 3.2).

Once the separating hyperplane has been determined, it can be used to classify new data points by considering which side of the hyperplane they lie on. Thus the decision or classification function for a new point $x \in G$ is

$$\mathrm{sgn}(w^o \cdot x + b^o) \ ,$$

or equivalently

$$\mathrm{sgn}(\sum_{i=1}^{l} \alpha_i^o y_i x_i \cdot x + b^o) \ .$$

## 3 The Kernel Trick

The kernel trick is a technique which first got serious attention in its application of generalizing the SVM we have considered so far to high-dimensional feature spaces (Boser, Guyon and Vapnik, 1992). The motivation for this is Cover's theorem (Haykin, 1999, Section 5.2), which essentially states that in many cases where data are not linearly separable in a low-dimensional space, a transformation of the data to a higher-dimensional feature space can yield linear separability. Thus our primary motivation for the use of the kernel trick is the case where our data are not linearly separable in the input space $G$. Since the application of the kernel trick to SVMs, it has been applied to many well known techniques applied in statistics, including principal components analysis, Fisher discriminant analysis, projection pursuit, canonical correlation analysis as well as some techniques more common in machine learning and signal processing, such as independent component analysis and self-organizing maps.

The kernel trick has its roots in the theory of Hilbert-Schmidt operators and reproducing kernel Hilbert spaces. A Hilbert space is an inner product space with the additional property of completeness. However, the essence of the kernel trick derives from Mercer's theorem. In simplified form, this states that for any bounded, non-negative function $K(x, y)$ which is symmetric in its arguments $x, y \in G$, there exists a Hilbert space $H_K$ (called the *feature space*), and a mapping $\varphi : G \to H_K$ such that $\varphi(x) \cdot \varphi(y) = K(x, y)$ for all $(x, y) \in G^2$ (Kroon, 2003). Such a $K$ is called a *kernel function.* If the mapping $\varphi$ maps the input space into a high-dimensional space, we are more likely to have linear separability of our transformed data points $\varphi(x_1), \ldots, \varphi(x_l)$ than of our original data points. Furthermore, when $G$ is not an inner product space, we cannot even define a hyperplane and hence linear separability in $G$. It thus seems reasonable to consider the construction of a separating hyperplane between the transformed data points in $H_K$.

Due to Mercer's theorem, while performing the method described in the previous section, every time an expression of the form $\varphi(x) \cdot \varphi(y)$ for $(x, y) \in G^2$ is encountered, it can be replaced by $K(x, y)$. It turns out that by means of this technique, every occurence of the map $\varphi$ can be eliminated. This allows us to find the separating hyperplane in feature space by using $K$ to calculate the inner products of the transformed data points *without ever manually performing the transformation.* Even more remarkable is that if one chooses an appropriate $K$ satisfying Mercer's theorem, the mere existence of a feature space $H_K$ and mapping $\varphi$ allows one to replace the inner products in the original procedure with kernel functions, *without even knowing the form of the mapping $\varphi$ or the feature space $H_K$!* This is all because the optimization problem (7) for finding the optimal separating hyperplane only makes use of the data points in the context of inner products. The technique of replacing these inner products by kernel functions to directly find the inner product of transformed data points is known as the *kernel trick*, and can be performed on any technique which can be formulated in such a way as to only make use of the data in inner products. This property is common to all the techniques listed at the beginning of this section.

In the SVM formulation of Section 2, a linear decision surface or hyperplane was generated by solving the quadratic programming problem generated by trying to maximize the margin, i.e. the distance between the hyperplane and the closest data point, or from a different viewpoint, enforce smoothness. Substituting an appropriate kernel function $K$ for the inner product results in a linear decision surface in a new feature space $H_K$. This corresponds to a nonlinear decision surface in the original input space. In order to generate such a nonlinear decision surface, one must maximize

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

subject to $\alpha \geq 0$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$. Note that this is identical to (7) except that $x_i \cdot x_j$ has been replaced by $K(x_i, x_j)$.

An important effect of using the kernel trick is then that $w^o = \sum_{i=1}^{l} \alpha_i^o y_i \varphi(x_i)$. Now, in many cases, the feature space $H_K$ associated with $K$ is infinite-dimensional, and in most other cases, $\varphi$ is unknown or difficult to determine, which makes explicit calculation of $w^o$ infeasible. This is fortunately not as much of a stumbling block as it may at first seem, since it is the decision function $\mathrm{sgn}(w^o \cdot \varphi(x) + b^o)$ for $x \in G$ that is of interest. Substituting the expression for $w^o$ into this decision function yields

$$\begin{aligned} &\mathrm{sgn}((\sum_{i=1}^{l} \alpha_i^o y_i \varphi(x_i)) \cdot \varphi(x) + b^o) \\ &= \mathrm{sgn}(\sum_{i=1}^{l} \alpha_i^o y_i \varphi(x_i) \cdot \varphi(x) + b^o) \\ &= \mathrm{sgn}(\sum_{i=1}^{l} \alpha_i^o y_i K(x_i, x) + b^o) \ , \end{aligned}$$

where we have used the kernel trick to replace the inner product in feature space. Here the value of $b^o$ used is typically the mean (over nonzero $\alpha_i^o$) of the values obtained from

$$\begin{aligned} b^o &= y_i - w^o \cdot \varphi(x_i) \\ &= y_i - \left( \sum_{j=1}^{l} \alpha_j^o y_j \varphi(x_j) \right) \cdot \varphi(x_i) \\ &= y_i - \sum_{j=1}^{l} \alpha_j^o y_j K(x_j, x_i) \ . \end{aligned}$$

Thus, by means of the kernel trick, we can solve the optimization problem in feature space, calculate $b$, and implicitly specify the hyperplane in the feature space (which then yields our decision function), without ever having to perform a direct transformation of the data points. We now consider some common kernel functions:

- An easily verifiable example of a kernel function is $K(x, y) = (x \cdot y)^2$, for $G = \mathbb{R}^2$. In this case the feature space $H_K = \mathbb{R}^3$, and
$$\varphi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} {x_1}^2 \\ {x_2}^2 \\ \sqrt{2} x_1 x_2 \end{bmatrix} \ .$$
- A generalization of the simple example above which is quite popular in practice is the polynomial kernel $K(x, y) = (x \cdot y + 1)^p$.
- The most common kernel in practice is the Gaussian kernel. This has the form
$$K(x, y) = e^{-\gamma \|x - y\|^2} \ .$$
- The linear kernel, $K(x, y) = x \cdot y$ corresponds to the case where the kernel trick is not employed. Thus the feature space is the same as the input space and $\varphi$ is an identity mapping.

The value of any kernel parameters (e.g. $\gamma, p$) are usually found by cross-validation. Although each kernel yields a linear decision surface in feature space, the exact kernel used determines the type of nonlinearity in the decision surface generated in the underlying input space. For the simple example above, the decision surface generated by the kernel is a quadratic function. More generally, a polynomial kernel of degree $p$ yields a decision surface which is a polynomial of degree $p$. The Gaussian kernel yields a decision surface which is a type of radial basis function classifier (Schölkopf, Sung, Burges, Girosi, Niyogi, Poggio and Vapnik, 1997).

The choice of the kernel function $K$ is often difficult, and the precise role of the kernel function is a thorny issue in theoretical analysis. Connections between the kernel function and regularization theory have helped in this regard (Schölkopf, 2000), and in particular have allowed the design of kernel functions incorporating prior knowledge (Smola and Schölkopf, 1997).

## 4 Handling Inseparability

A serious limitation of the original SVM was its inability to handle training data which were not linearly separable in the feature space corresponding to the chosen kernel $K$. This problem was addressed in Cortes and Vapnik (1995) with their proposal of the *soft margin* hyperplane. This has since become the standard approach used in SVMs, since although the kernel trick aims to make the data linearly separable in feature space, we still need to cater for cases where separarability is not achieved. An outline of the technique is given here.

The derivation done here is once again for a linear decision surface, the extension by using the kernel trick is identical to the linearly separable case.

The approach of the soft margin SVM is to allow the margin constraints (1) to be violated, but in such a way that any such violation will increase the objective function. Thus we change our interpretation of the margin $\frac{1}{\|w\|}$: now, the margin is the distance which must be attained between the separating hyperplane and a point in order to avoid penalization of the objective. We proceed by introducing *slack variables* $\xi = (\xi_1, \ldots, \xi_l)$. The constraints are modified to

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, i \in \{1, \ldots, l\} \tag{9}$$

for $\xi \geq 0$. In this case, we would want the $\xi_i$ to be as small as possible. Hence, we propose to penalize the hard-margin objective function by changing it to

$$\frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \xi_i \ , \tag{10}$$

where $C \in \mathbb{R}$ is a parameter determining the extent to which violations of the original constraints (1) should be penalized. This once again yields a quadratic program (Burges, 1998) which can easily be solved with various computer packages. Furthermore, we shall see that the slack variables vanish in the dual formulation (Burges, 1998). The penalty term can be replaced by more complex cost functions if desired. Note that for more complex cost functions, the cost function must typically be convex to yield a feasible optimization problem.

The derivation of the quadratic programming problem is similar to that in Section 2. First, Lagrange multipliers are introduced for the constraints: $\alpha \geq 0$ is once again used for the constraints (9), and the multipliers $\eta = (\eta_1, \cdots \eta_l) \geq 0$ to enforce the constraint $\xi \geq 0$.

So our primal Lagrangian is

$$\begin{aligned} L_P(w, b, \xi, \alpha, \eta) = \frac{1}{2} w \cdot w + C \sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i \{ y_i (w \cdot x_i + b) - 1 + \xi_i \} \\ - \sum_{i=1}^{l} \eta_i \xi_i \ . \end{aligned} \tag{11}$$

Taking partial derivatives with respect to $w$, $b$ and $\xi_i$, and setting to zero we obtain the two constraints encountered with the original *hard margin* SVM,

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{12}$$

and

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \ , \tag{13}$$

together with a new constraint

$$C = \alpha_i + \eta_i \ . \tag{14}$$

Using these constraints, we obtain the Wolfe dual objective function

$$L_D(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \ .$$

So the modification to the objective function has no effect on the dual objective function. However, there is a change in the constraints. For the separable case, we required $\alpha \geq 0$ and $\sum_{i=1}^{l} \alpha_i y_i = 0$. In the current problem, we also require that $\alpha_i \leq C$. This follows from (14), together with $\eta \geq 0$. Thus the modification for linear inseparability has the effect of limiting the search space for $\alpha$.

Once an optimal $\alpha^o$ has been found for the problem above, the corresponding $w^o$ and $b^o$ can be found precisely as in Section 2. Note that the restriction $\alpha_i \leq C$ limits the influence any one point can have on the determination of the soft-margin separating hyperplane.

The value $C$ is called a regularization parameter because the SVM optimization problem has an equivalent formulation as a regularization problem. Regularization is a common technique in statistics for reducing the variance of an estimator at the cost of a possible increase in bias, e.g. ridge regression (Hastie, Tibshirani and Friedman, 2001). In the case of soft-margin SVMs, one is trying to optimize

$$\sum_{i=1}^{l} \xi_i + \lambda \|w\|^2$$

for an appropriate $\lambda$ (see (10)). In this case, we are dealing with a regularization problem with the *hinge loss* function

$$L(y_i, f_{w,b}(x_i)) = \xi_i = (1 - y_i f_{w,b}(x_i))_+ \ , \tag{15}$$

(where $x_+$ denotes the positive part of $x$), and the regularizer $Q(w) = \|w\|^2$. It is clear that there is a one-to-one correspondence between $C$ and $\lambda$ in these problems, and this is why $C$ is usually called the regularization parameter. The view of SVMs as solutions to a regularization problem is useful to many researchers, as it further helps them to place the SVM in context among many similar techniques, such as kernel ridge regression, regularization networks and others. A good reference on SVMs and regularization is Evgeniou, Pontil and Poggio (2000).

It is important to note now that this technique allows a soft-margin hyperplane to be constructed even when linear separability is achieved. In some cases a smoother (lower variance) function is preferable, even if the hyperplane does not separate the data perfectly: a tradeoff exists between increasing the margin ($\frac{1}{\|w\|}$) and decreasing the violation of the original constraints ($\sum_{i=1}^{l} \xi_i$). This tradeoff is controlled by the *regularization parameter* $C$. For example, by letting C tend to infinity, we effectively penalize all non-zero slack variables infinitely. In this case, the SVM using a Gaussian kernel would find a hyperplane separating the data perfectly with a very small margin, or, equivalently, a very rough, high-variance function. How best to choose this parameter for SVMs, and indeed for regularization techniques in general, is a contentious issue, and a number of techniques exist for this purpose, the most popular being cross-validation.

# 5 Multi-class Support Vector Machines

One shortcoming of the SVM classifier we have introduced is that it is restricted to dichotomous classification. Extending the technique to multi-class problems has received much attention in the last six years, with two major avenues followed.

Firstly, there are a number of techniques which can generally be utilized for extending dichotomous pattern recognition tools to the polychotomous case. Secondly, there have been attempts to modify the underlying architecture of the SVM.

Among the general multi-class techniques are approaches such as one-per-class, pairwise coupling, and other voting methods. Extensions of the standard SVM directly have mainly focused on extending either the objective function (Weston, 1999, Weston and Watkins, 1998) or the concept of the margin (Crammer and Singer, 2001, Guermeur, Elisseeff and Paugam-Moisy, 2000). Platt, Cristianini and Shawe-Taylor (2000) asserts that the problem of construction of multi-class SVMs is currently unsolved. In practice, pairwise coupling (one-versus-one) is typically used for multi-class SVM classification.

## 6 SV Regression

The SRM principle concerns itself with minimizing the sum of a term reflecting the loss suffered due to errors on the training set, and a term reflecting the complexity of the function class from which an estimator is selected.

In the separable case, it is possible to obtain a zero-loss performance on the training set. Once the data become inseparable, however, the loss suffered by an incorrect classification must be specified. Referring to the constraints (9), and the objective function (10), we see that for each data point in the training set not achieving the margin, the objective function is increased. This increase in the objective function corresponds to the loss function employed in training the SVM (which is the loss function of the regularization problem).

By modifying the loss function, one can create SVMs for regression (Smola and Schölkopf, 1997, 1998, Vapnik, Golowich and Smola, 1997), and through the kernel trick, nonlinear regression. Here we view the hyperplane in feature space as the estimator of the function, and penalize deviations of data points from the hyperplane. Once again, $\|w\|$ reflects the smoothness of the hyperplane, so that the objective function is

$$\frac{1}{2} w \cdot w + C \sum_{i=1}^{l} L(y_i, w \cdot x_i + b) \ .$$

Let us begin by considering the loss function

$$L_\varepsilon(y_i, y) = (y_i - y)^2 \ .$$

If we use this loss function, we obtain the familiar ridge regression, a more stable modification of linear regression in many cases. Applying the kernel trick yields a nonlinear ridge regression.

The standard loss function employed for SV regression is the $\varepsilon$-insensitive loss function

$$L_\varepsilon(y_i, y) = \max(0, |y_i - y| - \varepsilon) \ .$$

This penalises only training points which differ from the estimator by more than $\varepsilon$, and then in a linear fashion. This yields a robust alternative to standard regression, in the form of a sparse representation — the regression function depends only on a subset of the training data. The derivation of the resulting optimization differs slightly from what we have seen earlier (due to the presence of additional Lagrange multipliers resulting from the absolute value in the loss function). However, we finally obtain the Wolfe dual problem of maximizing

$$\begin{aligned} L_D(\alpha_i, \alpha_i^*) = &-\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)x_i \cdot x_j \\ &-\varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \\ \text{subject to} \quad &\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \ , \end{aligned}$$

with $\alpha_i, \alpha_i^* \in [0, C]$. Again, $w^o$ can be expanded as a linear combination of the $x_i$'s, and $b^o$ can be computed from the data points with Lagrange multipliers in the open interval $(0, C)$. Further details are available in Kroon (2003). This SV regression technique can also be used for time series prediction as detailed in Fernández (1999), Mukherjee, Osuna and Girosi (1997), Müller, Smola, Rätsch, Schölkopf, Kohlmorgen and Vapnik (1997).

Unfortunately, due to the optimization required, it is generally not possible to obtain good confidence bands on SVM regression functions without resorting to resampling plans such as the bootstrape (Efron, 1982).

Besides classification and regression, the support vector methodology has been extended to other problems, including density estimation (Weston, 1999) and operator inversion (Vapnik, Golowich and Smola, 1997).

## 7 Summary

This paper presented the SVM with a focus on the classification problem. As can be seen from its formulation as a regularization problem, the algorithm is designed with inherent complexity control.

Despite its stronger theoretical basis than many other common machine learning techniques, the SVM is still far from well understood in the statistical arena — many possibilities for further study exist here.

SVMs have exhibited excellent performance on many benchmark and practical problems, and are fast becoming a popular method in the machine learning toolbox. A followup paper will discuss common SVM packages available, and discuss issues involved in applying SVMs with reference to an example.

## Acknowledgements

The authors would like to thank Prof. Sarel Steel, as well as the two anonymous referees, for useful comments and suggestions.

## References

Bertsekas, D. P. (1982). *Constrained Optimization and Lagrange Multiplier Methods*. Computer Science and Applied Mathematics. A Series of Monographs and Textbooks. Academic Press, Inc., New York.

Boser, B. E., Guyon, I. M. and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. D. Haussler (editor) *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152. ACM Press, Pittsburg, PA.

Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C., Manuel Ares, Jr. and Haussler, D. (1999). Support vector machine classification of microarray gene expression data. UCSC-CRL 99-09, University of California, Santa Cruz, Santa Cruz, CA.

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, **2** (2), 121–167.

Campbell, C. (2000). Kernel methods: A survey of current techniques. Draft of a tutorial submission appearing in Neurocomputing, volume 48, pages 63–84.

CAMPBELL, C. and CRISTIANINI, N. (1998). Simple learning algorithms for training support vector machines. Tech. rep., Department of Engineering Mathematics, University of Bristol, Bristol.

CORTES, C. and VAPNIK, V. (1995). Support-Vector networks. *Machine Learning*, **20** (3), 273–297.

CRAMMER, K. and SINGER, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, pp. 265–292.

EFRON, B. (1982). *The Jackknife, the Bootstrap and other Resampling Plans*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania.

EVGENIOU, T., PONTIL, M. and POGGIO, T. (2000). A unified framework for regularization networks and support vector machines. Tech. Rep. A.I. Memo No. 1654/C.B.C.L. Paper No. 171, Massachusetts Institute of Technology: Artificial Intelligence Laboratory/Center for Biological and Computational Learning (Department of Brain and Cognitive Sciences), Cambridge, Mass. Also appeared as 'Regularization Networks and Support Vector Machines' in 'Advances in Large-Margin Classifiers (Neural Information Processing) (Alexander Johannes Smola, Peter L. Bartlett, Bernhard Schölkopf, and D. Schuurmans, eds.), ch. 10, MIT Press, 2000'.

FERNÁNDEZ, R. (1999). Predicting time series with a local support vector regression machine. *Proceedings of ACAI'99*.

GIROSI, F., JONES, M. and POGGIO, T. (1995). Regularization theory and neural networks architectures. *Neural Computation*, **7** (2), 219–269.

GUERMEUR, Y., ELISSEEFF, A. and PAUGAM-MOISY, H. (2000). A new multi-class SVM based on a uniform convergence result. S.-I. Amari, C. L. Giles, M. Gori and V. Piuri (editors) *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000)*.

GUYON, I., BOSER, B. E. and VAPNIK, V. N. (1993). Automatic capacity tuning of very large VC-dimension classifiers. *Advances in Neural Information Processing Systems 5 (NIPS'92)*, pp. 147–155. Morgan Kaufmann, San Mateo, CA.

Hastie, T., Tibshirani, R. and Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Series in Statistics. Springer.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, second ed. ISBN: 0-13-273350-1.

Koo, D. (1977). *Elements of Optimization*. Heidelberg Science Library. Springer-Verlag.

Kroon, R. S. (2003). *Support Vector Machines, Generalization Bounds and Transduction*. Master's thesis, Department of Computer Science, University of Stellenbosch.

Lee, Y., Lin, Y. and Wahba, G. (2001). Multicategory support vector machines. Tech. Rep. TR 1043, Department of Statistics, University of Wisconsin-Madison. To appear, Proceedings of the 33rd Symposium on the Interface.

Mukherjee, S., Osuna, E. and Girosi, F. (1997). Nonlinear prediction of chaotic time series using support vector machines. J. Principe, L. Gile, N. Morgan and E. Wilson (editors) *Neural Networks for Signal Processing VII - Proceedings of the 1997 IEEE Workshop*, pp. 511–520. IEEE, New York.

Müller, K.-R., Smola, A. J., Rätsch, G., Schölkopf, B., Kohlmorgen, J. and Vapnik, V. N. (1997). Predicting time series with support vector machines. W. Gerstner, A. Germond, M. Hasler and J.-D. Nicoud (editors) *Artificial Neural Networks - ICANN '97*, vol. 1327 of *Lecture Notes in Computer Science*, pp. 999–1004. ICANN, Springer, Berlin. ISBN 3-540-63631-5. Also in 'Advances in Kernel Methods — Support Vector Learning, pages 243 – 253. MIT Press, Cambridge, MA, 1999' as 'Using Support Vector Machines for Time Series Prediction'.

Panik, M. J. (1976). *Classical Optimization: Foundations and Extensions*, vol. 16 of *Studies in Mathematical and Managerial Economics*. North-Holland/American Elsevier.

Platt, J. C., Cristianini, N. and Shawe-Taylor, J. (2000). Large margin DAGs for multiclass classification. S. Solla, T. Leen and K. Müller (editors) *Advances in Neural Information Processing Systems 12*. MIT Press.

Schölkopf, B. (1999). Support vector learning. Available on the web. Lecture notes and slides. http://www.first.gmd.de/~bs.

Schölkopf, B. (2000). Statistical learning and kernel methods. Tech. Rep. MSR-TR-2000-23, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052. Course notes of the Interdisciplinary College, Günne, Germany, in March 2000. http://research.microsoft.com/~bsc.

Schölkopf, B., Smola, A. J., Williamson, R. C. and Bartlett, P. L. (1998). New support vector algorithms. Technical Report NC-TR-98-031, NeuroCOLT, UK. Also appeared in Neural Computation, vol. 12, pp. 1207–1245, 2000.

Schölkopf, B., Sung, K.-K., Burges, C. J., Girosi, F., Niyogi, P., Poggio, T. and Vapnik, V. N. (1997). Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, **45** (11), 2758–2765.

Smola, A. J. and Schölkopf, B. (1997). On a kernel-based method for pattern recognition, regression, approximation and operator inversion. Technical Report 1064, GMD FIRST. Also appeared in Algorithmica, 22:211–231, 1998.

Smola, A. J. and Schölkopf, B. (1998). A tutorial on support vector regression. Tech. Rep. NC2-TR-1998-030, NeuroCOLT, Royal Holloway College, University of London. Also published in Statistics and Computing 2001. http://www.neurocolt.com.

Vapnik, V. N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer Series in Statistics. Springer-Verlag. ISBN 0-387-90733-5. Translated by Samuel Kotz.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Vapnik, V. N., Golowich, S. E. and Smola, A. J. (1997). Support vector method for function approximation, regression estimation, and signal processing. M. Mozer, M. I. Jordan and T. Petsche (editors) *Advances in Neural Information Processing Systems 9*, pp. 281–287. MIT Press, Cambridge, MA.

Weston, J. A. E. (1999). *Extensions to the Support Vector Method*. Ph.D. thesis, Department of Computer Science, Royal Holloway, University of London, Egham, Surrey.

Weston, J. A. E. and Watkins, C. (1998). Multi-class support vector machines. Tech. Rep. CSD-TR-98-04, Department of

Computer Science, Royal Holloway, University of London, Egham,
Surrey TW20 0EX, England.