UNIVERSITY OF SOUTHAMPTON

Support Vector Machines for Classification and Regression

by

Steve R. Gunn

Technical Report

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

10 May 1998

Contents

List of Figures

List of Tables

Listings

Nomenclature

$\mathbf{0}$	Column vector of zeros
$(x)_+$	The positive part of x
C	SVM misclassification tolerance parameter
$\mathcal{D}$	Dataset
$K(x, x')$	Kernel function
$R[f]$	Risk functional
$R_{emp}[f]$	Empirical Risk functional

Chapter 1

Introduction

The problem of empirical data modelling is germane to many engineering applications. In empirical data modelling a process of induction is used to build up a model of the system, from which it is hoped to deduce responses of the system that have yet to be observed. Ultimately the quantity and quality of the observations govern the performance of this empirical model. By its observational nature data obtained is finite and sampled; typically this sampling is non-uniform and due to the high dimensional nature of the problem the data will form only a sparse distribution in the input space. Consequently the problem is nearly always ill posed (Poggio et al., 1985) in the sense of Hadamard (Hadamard, 1923). Traditional neural network approaches have suffered difficulties with generalisation, producing models that can overfit the data. This is a consequence of the optimisation algorithms used for parameter selection and the statistical measures used to select the 'best' model. The foundations of Support Vector Machines (SVM) have been developed by Vapnik (1995) and are gaining popularity due to many attractive features, and promising empirical performance. The formulation embodies the Structural Risk Minimisation (SRM) principle, which has been shown to be superior, (Gunn et al., 1997), to traditional Empirical Risk Minimisation (ERM) principle, employed by conventional neural networks. SRM minimises an upper bound on the expected risk, as opposed to ERM that minimises the error on the training data. It is this difference which equips SVM with a greater ability to generalise, which is the goal in statistical learning. SVMs were developed to solve the classification problem, but recently they have been extended to the domain of regression problems (Vapnik et al., 1997). In the literature the terminology for SVMs can be slightly confusing. The term SVM is typically used to describe classification with support vector methods and support vector regression is used to describe regression with support vector methods. In this report the term SVM will refer to both classification and regression methods, and the terms Support Vector Classification (SVC) and Support Vector Regression (SVR) will be used for specification. This section continues with a brief introduction to the structural risk minimisation principle. In Chapter 2 the SVM is introduced in the setting of classification, being both historical and more accessible. This leads onto mapping the input into a higher dimensional feature space by a suitable choice of kernel function. The report then considers the problem of regression. Illustrative examples re given to show the properties of the techniques.

1.1 Statistical Learning Theory

This section is a very brief introduction to statistical learning theory. For a much more in depth look at statistical learning theory, see (Vapnik, 1998).

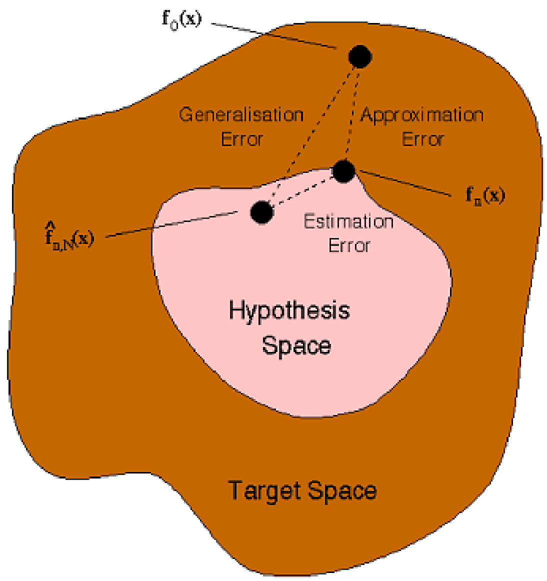


FIGURE 1.1: Modelling Errors

The goal in modelling is to choose a model from the hypothesis space, which is closest (with respect to some error measure) to the underlying function in the target space. Errors in doing this arise from two cases:

Approximation Error is a consequence of the hypothesis space being smaller than the target space, and hence the underlying function may lie outside the hypothesis space. A poor choice of the model space will result in a large approximation error, and is referred to as model mismatch.

Estimation Error is the error due to the learning procedure which results in a technique selecting the non-optimal model from the hypothesis space.

Together these errors form the generalisation error. Ultimately we would like to find the function, f, which minimises the risk,

$$R[f] = \int_{X \times Y} L\left(y, f(x)\right) P(x, y) dx dy \tag{1.1}$$

However, $P(x, y)$ is unknown. It is possible to find an approximation according to the empirical risk minimisation principle,

$$R_{emp}[f] = \frac{1}{l} \sum_{i=1}^{l} L\left(y^i, f\left(x^i\right)\right) \tag{1.2}$$

which minimises the empirical risk,

$$\hat{f_{n,l}}(x) = \arg \min_{f \in \mathcal{H}_n} R_{emp}[f] \tag{1.3}$$

Empirical risk minimisation makes sense only if,

$$\lim_{l \to \infty} R_{emp}[f] = R[f] \tag{1.4}$$

which is true from the law of large numbers. However, it must also satisfy,

$$\lim_{l \to \infty} \min_{f \in \mathcal{H}_n} R_{emp}[f] = \min_{f \in \mathcal{H}_n} R[f] \tag{1.5}$$

which is only valid when $\mathcal{H}_n$ is 'small' enough. This condition is less intuitive and requires that the minima also converge. The following bound holds with probability $1 - \delta$,

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{h \ln\left(\frac{2l}{h} + 1\right) - \ln\left(\frac{\delta}{4}\right)}{l}} \tag{1.6}$$

Remarkably, this expression for the expected risk is independent of the probability distribution.

1.1.1 VC Dimension

The VC dimension is a scalar value that measures the capacity of a set of functions.

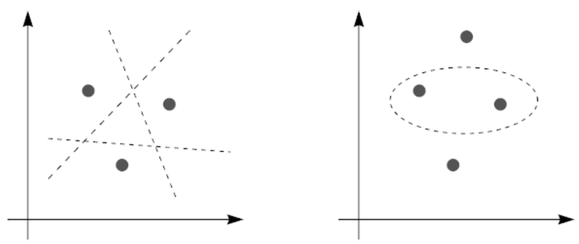


FIGURE 1.2: VC Dimension Illustration

Definition 1.1 (Vapnik–Chervonenkis). The VC dimension of a set of functions is p if and only if there exists a set of points $\{x^i\}_{i=1}^p$ such that these points can be separated in all 2^p possible configurations, and that no set $\{x^i\}_{i=1}^q$ exists where $q > p$ satisfying this property.

Figure 1.2 illustrates how three points in the plane can be shattered by the set of linear indicator functions whereas four points cannot. In this case the VC dimension is equal to the number of free parameters, but in general that is not the case; e.g. the function $A\sin(bx)$ has an infinite VC dimension (Vapnik, 1995). The set of linear indicator functions in n dimensional space has a VC dimension equal to $n+1$.

1.1.2 Structural Risk Minimisation

Create a structure such that S_h is a hypothesis space of VC dimension h then,

$$S_1 \subset S_2 \subset \ldots \subset S_\infty \tag{1.7}$$

SRM consists in solving the following problem

$$\min_{S_h} R_{emp}[f] + \sqrt{\frac{h \ln\left(\frac{2l}{h}+1\right) - \ln\left(\frac{\delta}{4}\right)}{l}} \tag{1.8}$$

If the underlying process being modelled is not deterministic the modelling problem becomes more exacting and consequently this chapter is restricted to deterministic processes. Multiple output problems can usually be reduced to a set of single output problems that may be considered independent. Hence it is appropriate to consider processes with multiple inputs from which it is desired to predict a single output.

Chapter 2

Support Vector Classification

The classification problem can be restricted to consideration of the two-class problem without loss of generality. In this problem the goal is to separate the two classes by a function which is induced from available examples. The goal is to produce a classifier that will work well on unseen examples, i.e. it generalises well. Consider the example in Figure 2.1. Here there are many possible linear classifiers that can separate the data, but there is only one that maximises the margin (maximises the distance between it and the nearest data point of each class). This linear classifier is termed the optimal separating hyperplane. Intuitively, we would expect this boundary to generalise well as opposed to the other possible boundaries.

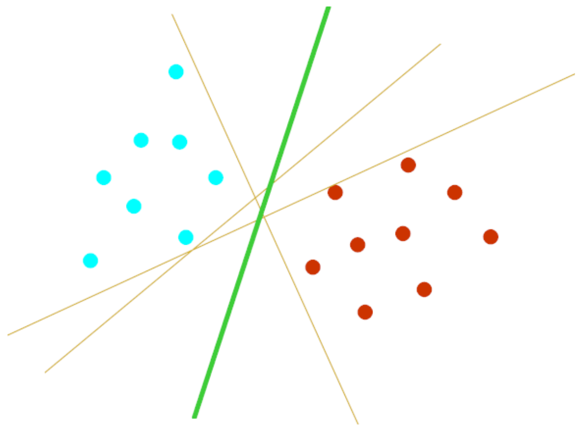


FIGURE 2.1: Optimal Separating Hyperplane

2.1 The Optimal Separating Hyperplane

Consider the problem of separating the set of training vectors belonging to two separate classes,

$$\mathcal{D} = \left\{(x^1, y^1), \ldots, (x^l, y^l)\right\}, \quad x \in \mathbb{R}^n, y \in \{-1, 1\}, \tag{2.1}$$

with a hyperplane,

$$\langle w, x \rangle + b = 0. \tag{2.2}$$

The set of vectors is said to be optimally separated by the hyperplane if it is separated without error and the distance between the closest vector to the hyperplane is maximal. There is some redundancy in Equation 2.2, and without loss of generality it is appropriate to consider a canonical hyperplane (Vapnik, 1995), where the parameters w, b are constrained by,

$$\min_i \left| \langle w, x^i \rangle + b \right| = 1. \tag{2.3}$$

This incisive constraint on the parameterisation is preferable to alternatives in simplifying the formulation of the problem. In words it states that: *the norm of the weight vector should be equal to the inverse of the distance, of the nearest point in the data set to the hyperplane*. The idea is illustrated in Figure 2.2, where the distance from the nearest point to each hyperplane is shown.

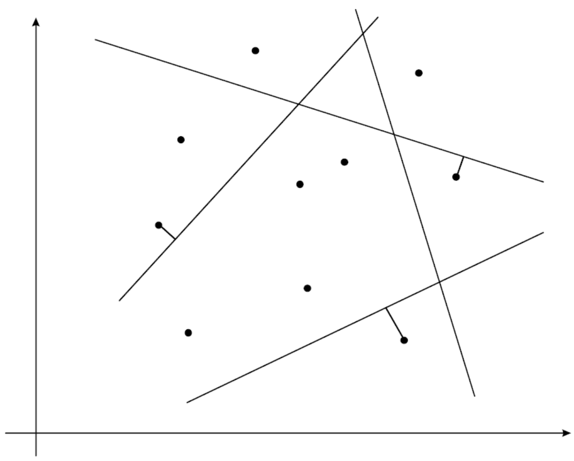


FIGURE 2.2: Canonical Hyperplanes

A separating hyperplane in canonical form must satisfy the following constraints,

$$y^i \left[\langle w, x^i \rangle + b \right] \geq 1, \quad i = 1, \ldots, l. \tag{2.4}$$

The distance $d(w, b; x)$ of a point x from the hyperplane (w, b) is,

$$d(w, b; x) = \frac{\left| \langle w, x^i \rangle + b \right|}{\|w\|}. \tag{2.5}$$

The optimal hyperplane is given by maximising the margin, ρ, subject to the constraints of Equation 2.4. The margin is given by,

$$\begin{aligned}
\rho(w,b) &= \min_{x^i:y^i=-1} d(w,b;x^i) + \min_{x^i:y^i=1} d(w,b;x^i) \\
&= \min_{x^i:y^i=-1} \frac{\left|\langle w,x^i\rangle + b\right|}{\|w\|} + \min_{x^i:y^i=1} \frac{\left|\langle w,x^i\rangle + b\right|}{\|w\|} \\
&= \frac{1}{\|w\|}\left(\min_{x^i:y^i=-1} \left|\langle w,x^i\rangle + b\right| + \min_{x^i:y^i=1} \left|\langle w,x^i\rangle + b\right|\right) \\
&= \frac{2}{\|w\|} \qquad (2.6)
\end{aligned}$$

Hence the hyperplane that optimally separates the data is the one that minimises

$$\Phi(w) = \frac{1}{2}\|w\|^2. \qquad (2.7)$$

It is independent of b because provided Equation 2.4 is satisfied (i.e. it is a separating hyperplane) changing b will move it in the normal direction to itself. Accordingly the margin remains unchanged but the hyperplane is no longer optimal in that it will be nearer to one class than the other. To consider how minimising Equation 2.7 is equivalent to implementing the SRM principle, suppose that the following bound holds,

$$\|w\| < A. \qquad (2.8)$$

Then from Equation 2.4 and 2.5,

$$d(w,b;x) \geq \frac{1}{A}. \qquad (2.9)$$

Accordingly the hyperplanes cannot be nearer than $\frac{1}{A}$ to any of the data points and intuitively it can be seen in Figure 2.3 how this reduces the possible hyperplanes, and hence the capacity.

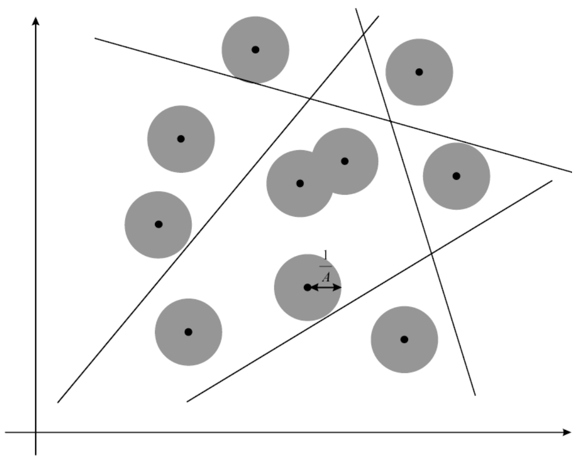


FIGURE 2.3: Constraining the Canonical Hyperplanes

The VC dimension, h, of the set of canonical hyperplanes in n dimensional space is bounded by,

$$h \leq \min[R^2 A^2, n] + 1, \tag{2.10}$$

where R is the radius of a hypersphere enclosing all the data points. Hence minimising Equation 2.7 is equivalent to minimising an upper bound on the VC dimension. The solution to the optimisation problem of Equation 2.7 under the constraints of Equation 2.4 is given by the saddle point of the Lagrange functional (Lagrangian) (Minoux, 1986),

$$\Phi(w, b, \alpha) = \tfrac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i \left(y^i \left[\langle w, x^i \rangle + b \right] - 1 \right), \tag{2.11}$$

where α are the Lagrange multipliers. The Lagrangian has to be minimised with respect to w, b and maximised with respect to $\alpha \geq \mathbf{0}$. Classical Lagrangian duality enables the primal problem, Equation 2.11, to be transformed to its dual problem, which is easier to solve. The dual problem is given by,

$$\max_{\alpha} W(\alpha) = \max_{\alpha} \left(\min_{w,b} \Phi(w, b, \alpha) \right). \tag{2.12}$$

The minimum with respect to w and b of the Lagrangian, Φ, is given by,

$$\begin{aligned} \frac{\partial \Phi}{\partial b} = 0 \quad &\Rightarrow \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \\ \frac{\partial \Phi}{\partial w} = \mathbf{0} \quad &\Rightarrow \quad w = \sum_{i=1}^{l} \alpha_i y_i x_i. \end{aligned} \tag{2.13}$$

Hence from Equations 2.11, 2.12 and 2.13, the dual problem is,

$$\max_{\alpha} W(\alpha) = \max_{\alpha} -\tfrac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_{k=1}^{l} \alpha_k, \tag{2.14}$$

and hence the solution to the problem is given by,

$$\alpha^* = \arg\min_{\alpha} \tfrac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{k=1}^{l} \alpha_k, \tag{2.15}$$

with constraints,

$$\begin{aligned} \alpha_i \geq 0 \quad i = 1, \ldots, l \\ \sum_{j=1}^{l} \alpha_j y_j = 0. \end{aligned} \tag{2.16}$$

Solving Equation 2.15 with constraints Equation 2.16 determines the Lagrange multipliers, and the optimal separating hyperplane is given by,

$$\begin{aligned} w^* &= \sum_{i=1}^{l} \alpha_i y_i x_i \\ b^* &= -\frac{1}{2}\langle w^*, x_r + x_s \rangle. \end{aligned} \tag{2.17}$$

where x_r and x_s are any support vector from each class satisfying,

$$\alpha_r, \alpha_s > 0, \quad y_r = -1, y_s = 1. \tag{2.18}$$

The hard classifier is then,

$$f(x) = \operatorname{sgn}(\langle w^*, x \rangle + b) \tag{2.19}$$

Alternatively, a soft classifier may be used which linearly interpolates the margin,

$$f(x) = h(\langle w^*, x \rangle + b) \quad \text{where} \quad h(z) = \begin{cases} -1 & : \quad z < -1 \\ z & : \quad -1 \le z \le 1 \\ +1 & : \quad z > 1 \end{cases} \tag{2.20}$$

This may be more appropriate than the hard classifier of Equation 2.19, because it produces a real valued output between -1 and 1 when the classifier is queried within the margin, where no training data resides. From the Kuhn-Tucker conditions,

$$\alpha_i \left(y^i \left[\langle w, x^i \rangle + b \right] - 1 \right) = 0, \quad i = 1, \ldots, l, \tag{2.21}$$

and hence only the points x^i which satisfy,

$$y^i \left[\langle w, x^i \rangle + b \right] = 1 \tag{2.22}$$

will have non-zero Lagrange multipliers. These points are termed Support Vectors (SV). If the data is linearly separable all the SV will lie on the margin and hence the number of SV can be very small. Consequently the hyperplane is determined by a small subset of the training set; the other points could be removed from the training set and recalculating the hyperplane would produce the same answer. Hence SVM can be used to summarise the information contained in a data set by the SV produced. If the data is linearly separable the following equality will hold,

$$\|w\|^2 = \sum_{i=1}^{l} \alpha_i = \sum_{i \in SVs} \alpha_i = \sum_{i \in SVs} \sum_{j \in SVs} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle. \tag{2.23}$$

Hence from Equation 2.10 the VC dimension of the classifier is bounded by,

$$h \le \min[R^2 \sum_{i \in SVs}, n] + 1, \tag{2.24}$$

x_1	x_2	y
1	1	-1
3	3	1
1	3	1
3	1	-1
2	2.5	1
3	2.5	-1
4	3	-1

TABLE 2.1: Linearly Separable Classification Data

and if the training data, x, is normalised to lie in the unit hypersphere,

$$h \leq 1 + \min[\sum_{i \in SVs}, n], \tag{2.25}$$

2.1.1 Linearly Separable Example

To illustrate the method consider the training set in Table 2.1.

The SVC solution is shown in Figure 2.4, where the dotted lines describe the locus of the margin and the circled data points represent the SV, which all lie on the margin.

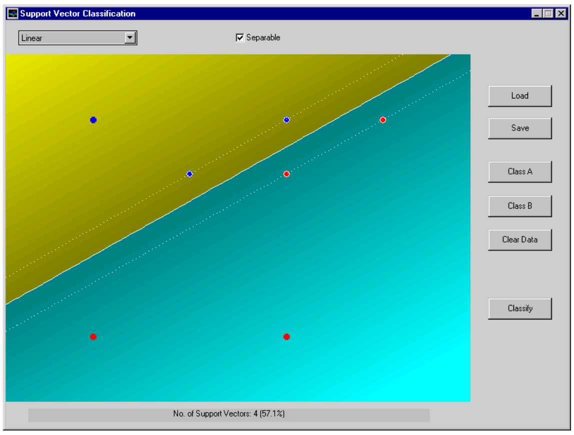


FIGURE 2.4: Optimal Separating Hyperplane

2.2 The Generalised Optimal Separating Hyperplane

So far the discussion has been restricted to the case where the training data is linearly separable. However, in general this will not be the case, Figure 2.5. There are two approaches to generalising the problem, which are dependent upon prior knowledge of the problem and an estimate of the noise on the data. In the case where it is expected (or possibly even known) that a hyperplane can correctly separate the data, a method of

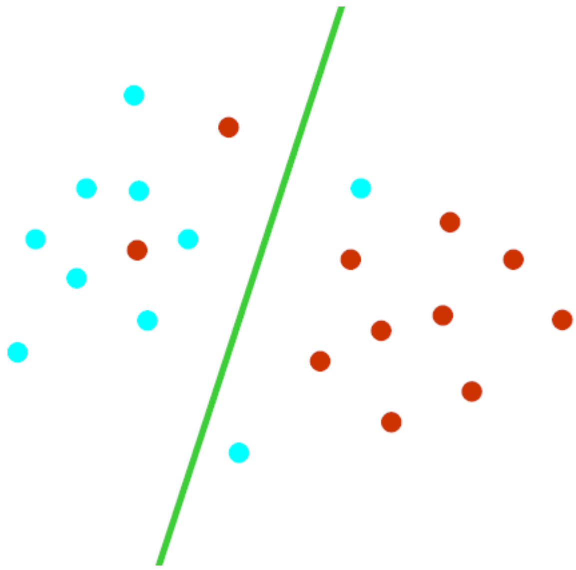


FIGURE 2.5: Generalised Optimal Separating Hyperplane

introducing an additional cost function associated with misclassification is appropriate. Alternatively a more complex function can be used to describe the boundary, as discussed in Chapter 2.1. To enable the optimal separating hyperplane method to be generalised, Cortes and Vapnik (1995) introduced non-negative variables, $\xi_i \geq 0$, and a penalty function,

$$F_\sigma(\xi) = \sum_i \xi_i^\sigma \quad \sigma > 0, \tag{2.26}$$

where the ξ_i are a measure of the misclassification errors. The optimisation problem is now posed so as to minimise the classification error as well as minimising the bound on the VC dimension of the classifier. The constraints of Equation 2.4 are modified for the non-separable case to,

$$y^i \left[\langle w, x^i \rangle + b\right] \geq 1 - \xi_i, \quad i = 1, \ldots, l. \tag{2.27}$$

where $\xi_i \geq 0$. The generalised optimal separating hyperplane is determined by the vector w, that minimises the functional,

$$\Phi(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i, \tag{2.28}$$

(where C is a given value) subject to the constraints of Equation 2.27. The solution to the optimisation problem of Equation 2.28 under the constraints of Equation 2.27 is given by the saddle point of the Lagrangian (Minoux, 1986),

$$\Phi(w, b, \alpha, \xi, \beta) = \tfrac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_{i=1}^{l} \alpha_i \left(y^i \left[w^T x^i + b\right] - 1 + \xi_i\right) - \sum_{j=1}^{l} \beta_i \xi_i, \tag{2.29}$$

where α, β are the Lagrange multipliers. The Lagrangian has to be minimised with respect to w, b, x and maximised with respect to α, β. As before, classical Lagrangian duality enables the primal problem, Equation 2.29, to be transformed to its dual problem. The dual problem is given by,

$$\max_{\alpha} W(\alpha, \beta) = \max_{\alpha,\beta} \left(\min_{w,b,\xi} \Phi(w, b, \alpha, \xi, \beta) \right). \tag{2.30}$$

The minimum with respect to w, b and ξ of the Lagrangian, Φ, is given by,

$$\begin{aligned} \frac{\partial \Phi}{\partial b} = 0 \quad &\Rightarrow \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \\ \frac{\partial \Phi}{\partial w} = \mathbf{0} \quad &\Rightarrow \quad w = \sum_{i=1}^{l} \alpha_i y_i x_i \\ \frac{\partial \Phi}{\partial \xi} = \mathbf{0} \quad &\Rightarrow \quad \alpha_i + \beta_i = C. \end{aligned} \tag{2.31}$$

Hence from Equations 2.29, 2.30 and 2.31, the dual problem is,

$$\max_{\alpha} W(\alpha) = \max_{\alpha} -\tfrac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_{k=1}^{l} \alpha_k, \tag{2.32}$$

and hence the solution to the problem is given by,

$$\alpha^* = \arg\min_{\alpha} \tfrac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{k=1}^{l} \alpha_k, \tag{2.33}$$

with constraints,

$$\begin{aligned} 0 \leq \alpha_i \leq C \quad i = 1, \ldots, l \\ \sum_{j=1}^{l} \alpha_j y_j = 0. \end{aligned} \tag{2.34}$$

The solution to this minimisation problem is identical to the separable case except for a modification of the bounds of the Lagrange multipliers. The uncertain part of Cortes's approach is that the coefficient C has to be determined. This parameter introduces additional capacity control within the classifier. C can be directly related to a regularisation parameter (Girosi, 1997; Smola and Schölkopf, 1998). Blanz et al. (1996) uses a value of $C = 5$, but ultimately C must be chosen to reflect the knowledge of the noise on the data. This warrants further work, but a more practical discussion is given in Chapter 4.

x_1	x_2	y
1	1	-1
3	3	1
1	3	1
3	1	-1
2	2.5	1
3	2.5	-1
4	3	-1
1.5	1.5	1
1	2	-1

TABLE 2.2: Non-Linearly Separable Classification Data

2.2.1 Linearly Non-Separable Example

Two additional data points are added to the separable data of Table 2.1 to produce a linearly non-separable data set, Table 2.2.

The resulting SVC is shown in Figure 2.6, for $C = 1$. The SV are no longer required to lie on the margin, as in Figure 2.4, and the orientation of the hyperplane and the width of the margin are different.

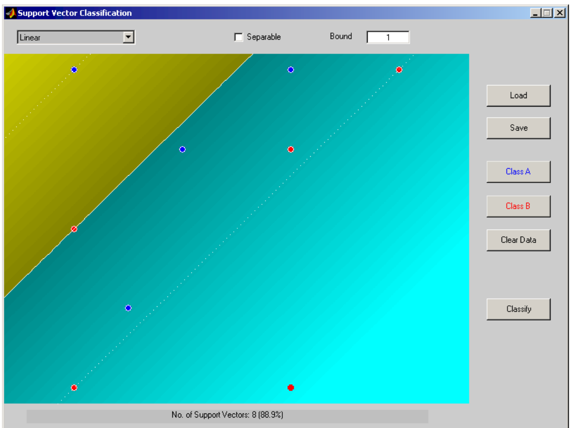


FIGURE 2.6: Generalised Optimal Separating Hyperplane Example ($C = 1$)

In the limit, $\lim_{C\to\infty}$ the solution converges towards the solution obtained by the optimal separating hyperplane (on this non-separable data), Figure 2.7.

In the limit, $\lim_{C\to 0}$ the solution converges to one where the margin maximisation term dominates, Figure 2.8. Beyond a certain point the Lagrange multipliers will all take on the value of C. There is now less emphasis on minimising the misclassification error, but purely on maximising the margin, producing a large width margin. Consequently as C decreases the width of the margin increases.

The useful range of C lies between the point where all the Lagrange Multipliers are equal to C and when only one of them is just bounded by C.

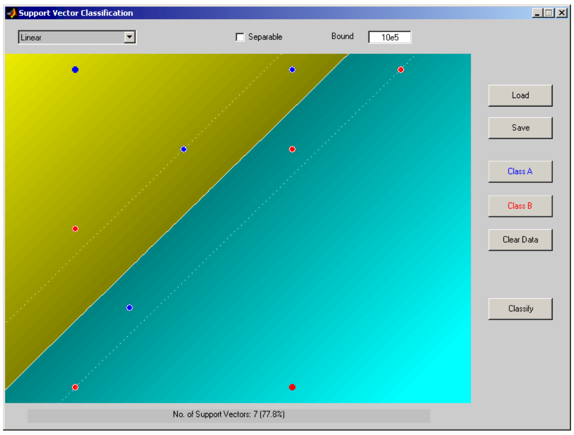


FIGURE 2.7: Generalised Optimal Separating Hyperplane Example ($C = 10^5$)

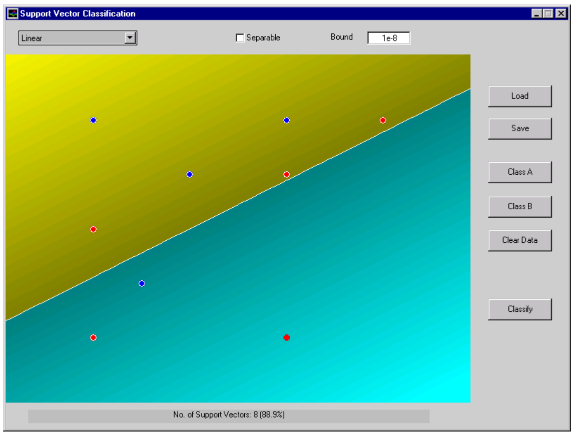


FIGURE 2.8: Generalised Optimal Separating Hyperplane Example ($C = 10^{-8}$)

2.3 Generalisation in High Dimensional Feature Space

In the case where a linear boundary is inappropriate the SVM can map the input vector, x, into a high dimensional feature space, z. By choosing a non-linear mapping a priori, the SVM constructs an optimal separating hyperplane in this higher dimensional space, Figure 2.9. The idea exploits the method of Aizerman et al. (1964) which, enables the curse of dimensionality (Bellman, 1961) to be addressed.

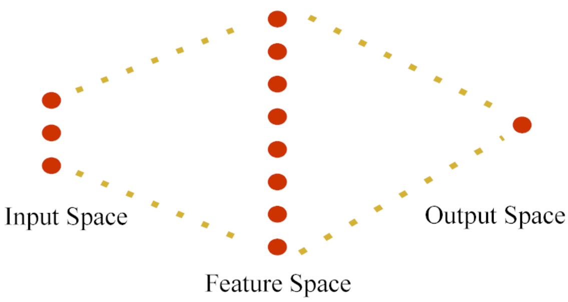


FIGURE 2.9: Mapping the Input Space into a High Dimensional Feature Space

There are some restrictions on the non-linear mapping that can be employed, see Chapter 3, but it turns out, surprisingly, that most commonly employed functions are acceptable. Among acceptable mappings are polynomials, radial basis functions and certain sigmoid functions. The optimisation problem of Equation 2.33 becomes,

$$\alpha^* = \arg\min_{\alpha} \tfrac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{k=1}^{l} \alpha_k, \tag{2.35}$$

where $K(x, x')$ is the kernel function performing the non-linear mapping into feature space, and the constraints are unchanged,

$$\begin{aligned} 0 \leq \alpha_i \leq C \quad i = 1, \ldots, l \\ \sum_{j=1}^{l} \alpha_j y_j = 0. \end{aligned} \tag{2.36}$$

Solving Equation 2.35 with constraints Equation 2.36 determines the Lagrange multipliers, and a hard classifier implementing the optimal separating hyperplane in the feature space is given by,

$$f(x) = \operatorname{sgn}\Big(\sum_{i \in SVs} \alpha_i y_i K(x_i, x) + b\Big) \tag{2.37}$$

where

$$\begin{aligned} \langle w^*, x \rangle &= \sum_{i=1}^{l} \alpha_i y_i K(x_i, x) \\ b^* &= -\frac{1}{2} \sum_{i=1}^{l} \alpha_i y_i \left[K(x_i, x_r) + K(x_i, x_r) \right]. \end{aligned} \tag{2.38}$$

The bias is computed here using two support vectors, but can be computed using all the SV on the margin for stability (Vapnik et al., 1997). If the Kernel contains a bias term, the bias can be accommodated within the Kernel, and hence the classifier is simply,

$$f(x) = \operatorname{sgn}\Big(\sum_{i \in SVs} \alpha_i K(x_i, x)\Big) \tag{2.39}$$

Many employed kernels have a bias term and any finite Kernel can be made to have one (Girosi, 1997). This simplifies the optimisation problem by removing the equality constraint of Equation 2.36. Chapter 3 discusses the necessary conditions that must be satisfied by valid kernel functions.

2.3.1 Polynomial Mapping Example

Consider a polynomial kernel of the form,

$$K(x, x') = (\langle x, x' \rangle + 1)^2, \tag{2.40}$$

which maps a two dimensional input vector into a six dimensional feature space. Applying the non-linear SVC to the linearly non-separable training data of Table 2.2, produces the classification illustrated in Figure 2.10 ($C = \infty$). The margin is no longer of constant width due to the non-linear projection into the input space. The solution is in contrast to Figure 2.7, in that the training data is now classified correctly. However, even though SVMs implement the SRM principle and hence can generalise well, a careful choice of the kernel function is necessary to produce a classification boundary that is topologically appropriate. It is always possible to map the input space into a dimension greater than the number of training points and produce a classifier with no classification errors on the training set. However, this will generalise badly.

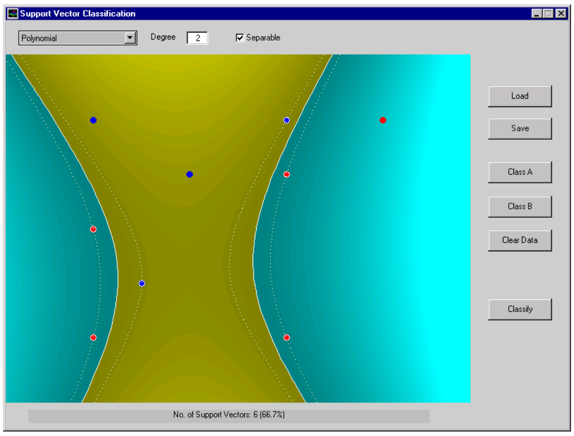


FIGURE 2.10: Mapping input space into Polynomial Feature Space

2.4 Discussion

Typically the data will only be linearly separable in some, possibly very high dimensional feature space. It may not make sense to try and separate the data exactly, particularly when only a finite amount of training data is available which is potentially corrupted by noise. Hence in practice it will be necessary to employ the non-separable approach which places an upper bound on the Lagrange multipliers. This raises the question of how to determine the parameter C. It is similar to the problem in regularisation where the regularisation coefficient has to be determined, and it has been shown that the parameter C can be directly related to a regularisation parameter for certain kernels (Smola and Schölkopf, 1998). A process of cross-validation can be used to determine this parameter, although more efficient and potentially better methods are sought after. In removing the training patterns that are not support vectors, the solution is unchanged and hence a fast method for validation may be available when the support vectors are sparse.

Chapter 3

Feature Space

This chapter discusses the method that can be used to construct a mapping into a high dimensional feature space by the use of reproducing kernels. The idea of the kernel function is to enable operations to be performed in the input space rather than the potentially high dimensional feature space. Hence the inner product does not need to be evaluated in the feature space. This provides a way of addressing the curse of dimensionality. However, the computation is still critically dependent upon the number of training patterns and to provide a good data distribution for a high dimensional problem will generally require a large training set.

3.1 Kernel Functions

The following theory is based upon Reproducing Kernel Hilbert Spaces (RKHS) (Aronszajn, 1950; Girosi, 1997; Heckman, 1997; Wahba, 1990). An inner product in feature space has an equivalent kernel in input space,

$$K(x, x') = \langle \phi(x), \phi(x') \rangle, \tag{3.1}$$

provided certain conditions hold. If K is a symmetric positive definite function, which satisfies Mercer's Conditions,

$$K(x, x') = \sum_{m}^{\infty} a_m \phi_m(x) \phi_m(x'), \quad a_m \geq 0, \tag{3.2}$$

$$\iint K(x, x') g(x) g(x') dx dx' > 0, \quad g \in L_2, \tag{3.3}$$

then the kernel represents a legitimate inner product in feature space. Valid functions that satisfy Mercer's conditions are now given, which unless stated are valid for all real x and x'.

3.1.1 Polynomial

A polynomial mapping is a popular method for non-linear modelling,

$$K(x, x') = \langle x, x' \rangle^d. \tag{3.4}$$

$$K(x, x') = \left(\langle x, x' \rangle + 1\right)^d. \tag{3.5}$$

The second kernel is usually preferable as it avoids problems with the hessian becoming zero.

3.1.2 Gaussian Radial Basis Function

Radial basis functions have received significant attention, most commonly with a Gaussian of the form,

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right). \tag{3.6}$$

Classical techniques utilising radial basis functions employ some method of determining a subset of centres. Typically a method of clustering is first employed to select a subset of centres. An attractive feature of the SVM is that this selection is implicit, with each support vectors contributing one local Gaussian function, centred at that data point. By further considerations it is possible to select the global basis function width, s, using the SRM principle (Vapnik, 1995).

3.1.3 Exponential Radial Basis Function

A radial basis function of the form,

$$K(x, x') = \exp\left(-\frac{\|x - x'\|}{2\sigma^2}\right). \tag{3.7}$$

produces a piecewise linear solution which can be attractive when discontinuities are acceptable.

3.1.4 Multi-Layer Perceptron

The long established MLP, with a single hidden layer, also has a valid kernel representation,

$$K(x, x') = \tanh\left(\rho\langle x, x' \rangle + \varrho\right) \tag{3.8}$$

for certain values of the scale, ρ, and offset, ϱ, parameters. Here the SV correspond to the first layer and the Lagrange multipliers to the weights.

3.1.5 Fourier Series

A Fourier series can be considered an expansion in the following $2N+1$ dimensional feature space. The kernel is defined on the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$,

$$K(x, x') = \frac{\sin(N+\frac{1}{2})(x-x')}{\sin(\frac{1}{2}(x-x'))}. \tag{3.9}$$

However, this kernel is probably not a good choice because its regularisation capability is poor, which is evident by consideration of its Fourier transform (Smola and Schölkopf, 1998).

3.1.6 Splines

Splines are a popular choice for modelling due to their flexibility. A finite spline, of order κ, with N knots located at τ_s is given by,

$$K(x, x') = \sum_{r=0}^{\kappa} x^r x'^r + \sum_{s=1}^{N} (x-\tau_s)_+^{\kappa} (x'-\tau_s)_+^{\kappa}. \tag{3.10}$$

An infinite spline is defined on the interval $[0, 1)$ by,

$$K(x, x') = \sum_{r=0}^{\kappa} x^r x'^r + \int_0^1 (x-\tau_s)_+^{\kappa} (x'-\tau_s)_+^{\kappa} d\tau. \tag{3.11}$$

In the case when $\kappa = 1$, (S_1^{∞}), the kernel is given by,

$$K(x, x') = 1 + \langle x, x' \rangle + \frac{1}{2} \langle x, x' \rangle \min(x, x') - \frac{1}{6} \min(x, x')^3, \tag{3.12}$$

where the solution is a piece-wise cubic.

3.1.7 B splines

Bsplines are another popular spline formulation. The kernel is defined on the interval $[-1, 1]$, and has an attractive closed form,

$$K(x, x') = B_{2N+1}(x - x'). \tag{3.13}$$

3.1.8 Additive Kernels

More complicated kernels can be obtained by forming summing kernels, since the sum of two positive definite functions is positive definite.

$$K(x, x') = \sum_i K_i(x, x') \tag{3.14}$$

3.1.9 Tensor Product

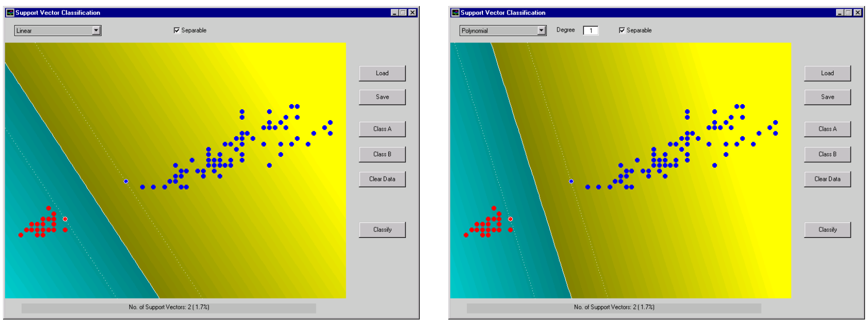
Kernels Multidimensional kernels can be obtained by forming tensor products of kernels (Aronszajn, 1950),

$$K(x, x') = \prod_i K_i(x_i, x'_i) \tag{3.15}$$

This is particularly useful in the construction of multidimensional spline kernels, which are simply obtained from the product of the univariate kernels.

3.2 Implicit vs. Explicit Bias

It was remarked in the previous chapter that kernels may or may not contain an implicit bias. The inclusion of a bias within the kernel function can lead to a slightly more efficient method of implementation. However, the solutions obtained with an implicit and explicit bias are not the same, which may initially come as a surprise. This difference helps to highlight the difficulties with the interpretation of generalisation in high dimensional feature spaces. Figure 3.1 compares a linear kernel with explicit bias against polynomial of degree 1 with implicit bias. It is evident that the solutions are different, although both solutions would seem to offer good generalisation.



(a) Explicit (linear) (b) Implicit (polynomial degree 1)

FIGURE 3.1: Comparison between Implicit and Explicit bias for a linear kernel

3.3 Data Normalisation

Data normalisation is required for particular kernels due to their restricted domain, and may also be advantageous for unrestricted kernels. To determine if normalisation (isotropic or non-isotropic) of the data is necessary requires the consideration of the input features. Additionally, normalisation will improve the condition number of the hessian in the optimisation problem.

3.4 Kernel Selection

The obvious question that arises is that with so many different mappings to choose from, which is the best for a particular problem? This is not a new question, but with the inclusion of many mappings within one framework it is easier to make a comparison. The upper bound on the VC dimension, Equation 2.10, is a potential avenue to provide a means of comparing the kernels. However, it requires the estimation of the radius of the hypersphere enclosing the data in the non-linear feature space. As a final caution, even if a strong theoretical method for selecting a kernel is developed, unless this can be validated using independent test sets on a large number of problems, methods such as bootstrapping and cross-validation will remain the preferred method for kernel selection.

Chapter 4

Classification Example: IRIS data

The iris data set is an established data set used for demonstrating the performance of classification algorithms. The data set contains four attributes of an iris, and the goal is to classify the class of iris based on these four attributes. To visualise the problem we restrict ourselves to the two features that contain the most information about the class, namely the petal length and the petal width. The distribution of the data is illustrated in Figure 4.1.

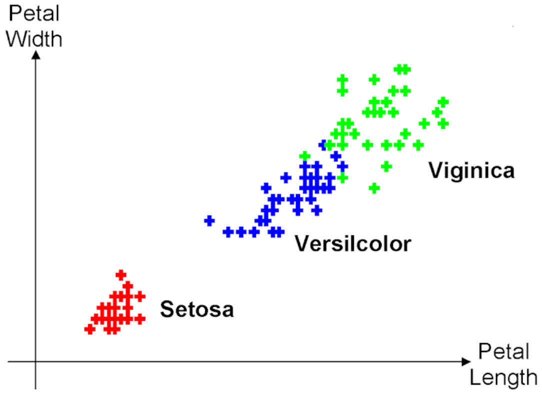


FIGURE 4.1: Iris data set

The Setosa and Versilcolor classes are easily separated with a linear boundary and the SVC solution using an inner product kernel is illustrated in Figure 4.2, with the two support vectors circled.

The two support vectors contain the important information about the classification boundary and hence illustrate the potential of SVC for data selection. The separation of the class Viginica from the other two classes is not so trivial. In fact, two of the examples are identical in petal length and width, but correspond to different classes. Figure 4.3 illustrates the SVC solution obtained using a degree 2 polynomial and it is clear that the area of input space where there is little data is classified as Viginica.

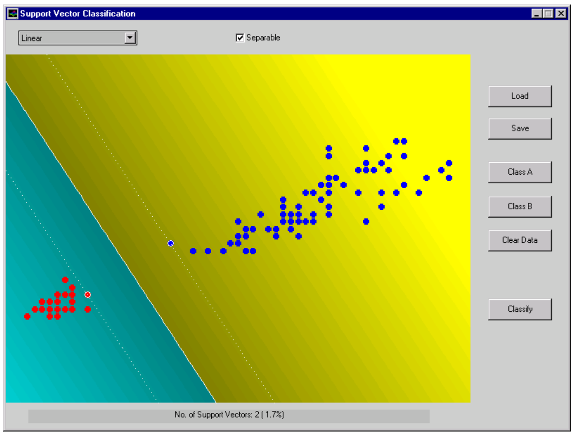


FIGURE 4.2: Separating Setosa with a linear SVC ($C = \infty$)

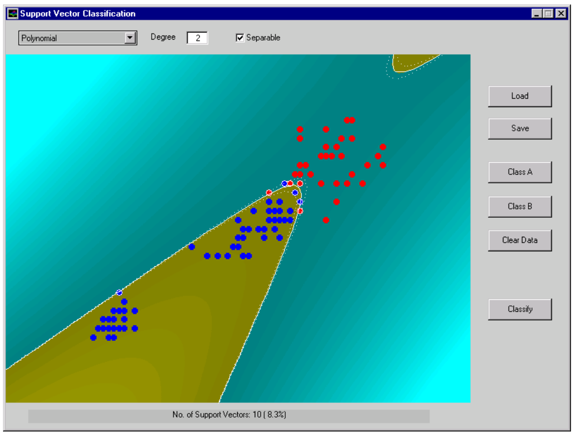


FIGURE 4.3: Separating Viginica with a polynomial SVM (degree 2,$C = \infty$)

Figure 4.4 illustrates the use of a higher order polynomial to separate the Viginica, with no additional capacity control. This SVC determines a hyperplane in a 55 dimensional feature space. There is evidence of overfitting due to the high dimensional nature of the kernel function, which is emphasised by the disjoint region in the top of the illustration.

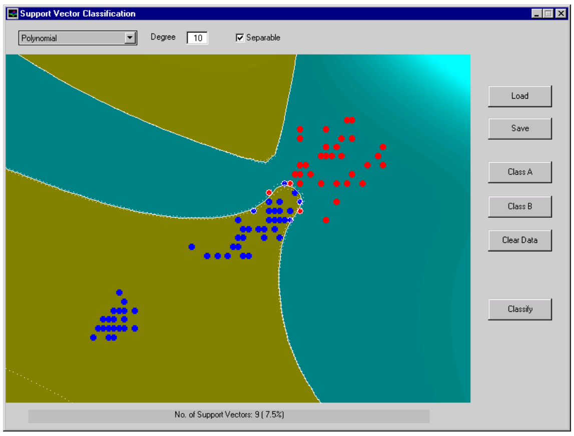


FIGURE 4.4: Separating Viginica with a polynomial SVM (degree 10, $C = \infty$)

Figure 4.5 illustrates a Gaussian radial basis function SVC using a pre-specified variance. The result is similar to that of the degree 2 polynomial.

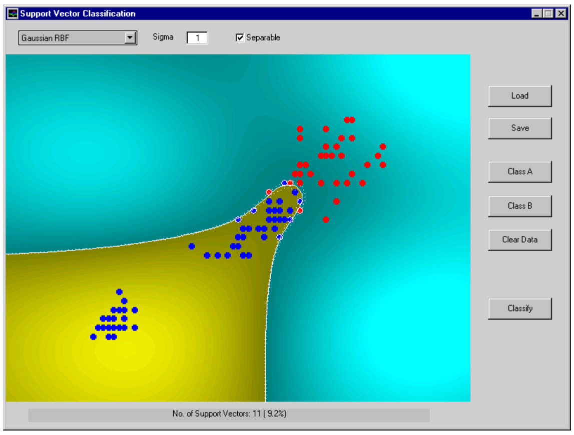


FIGURE 4.5: Separating Viginica with a Radial Basis Function SVM ($\sigma = 1.0$, $C = \infty$)

Figure 4.6 illustrates the SVC solution obtained using the degree 2 polynomial with some tolerance to misclassification errors ($C = 10$). This can be seen to produce a solution with good expected generalisation, emphasising the importance of tolerating misclassification errors in this example. This is necessary due to the non-separable nature of the data using just two input features.

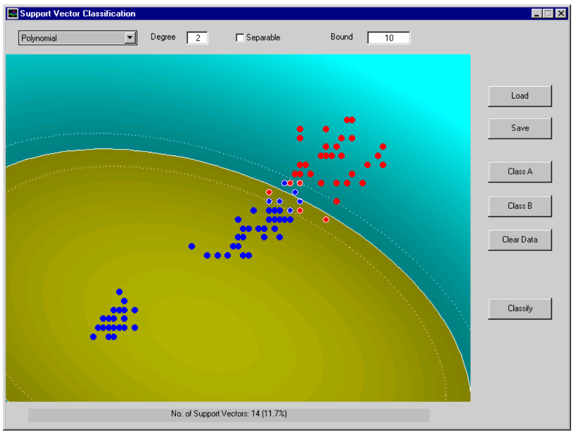
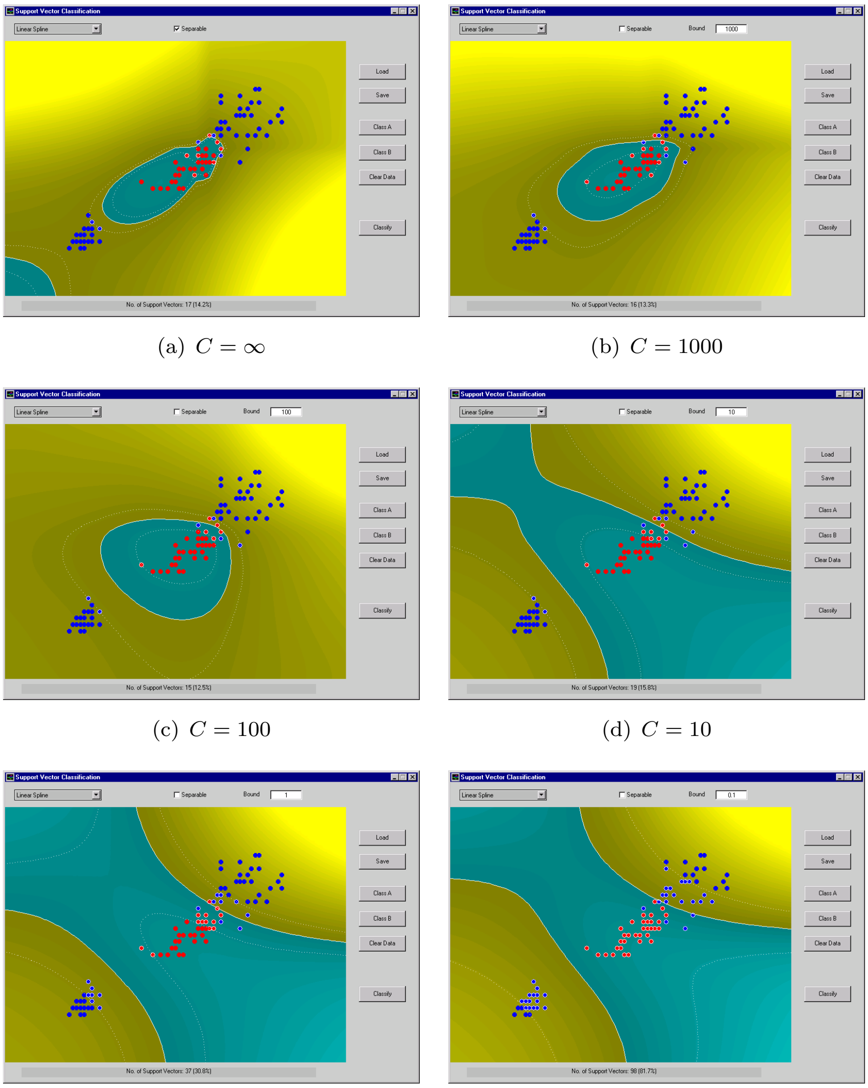


FIGURE 4.6: Separating Viginica with a polynomial SVM (degree 2, $C = 10$)

To visualise the effect of the tolerance to misclassification errors on the topology of the classifier boundary, Figure 4.7 shows the results of a linear spline SVC for various degrees of misclassification tolerance.

Interestingly, the values of $C = 1$ and $C = 100$ seem to offer good solutions, depending upon whether an open boundary, Figure 4.7(a) or a closed boundary, Figure 4.7(c) is more appropriate. This demonstrates that the parameter C may have more than one optimal value and prior knowledge about the problem under consideration may be required to select the final solution.



(a) $C = \infty$ (b) $C = 1000$

(c) $C = 100$ (d) $C = 10$

(e) $C = 1$ (f) $C = 0.1$

FIGURE 4.7: The effect of C on the separation of Versilcolor with a linear spline SVM

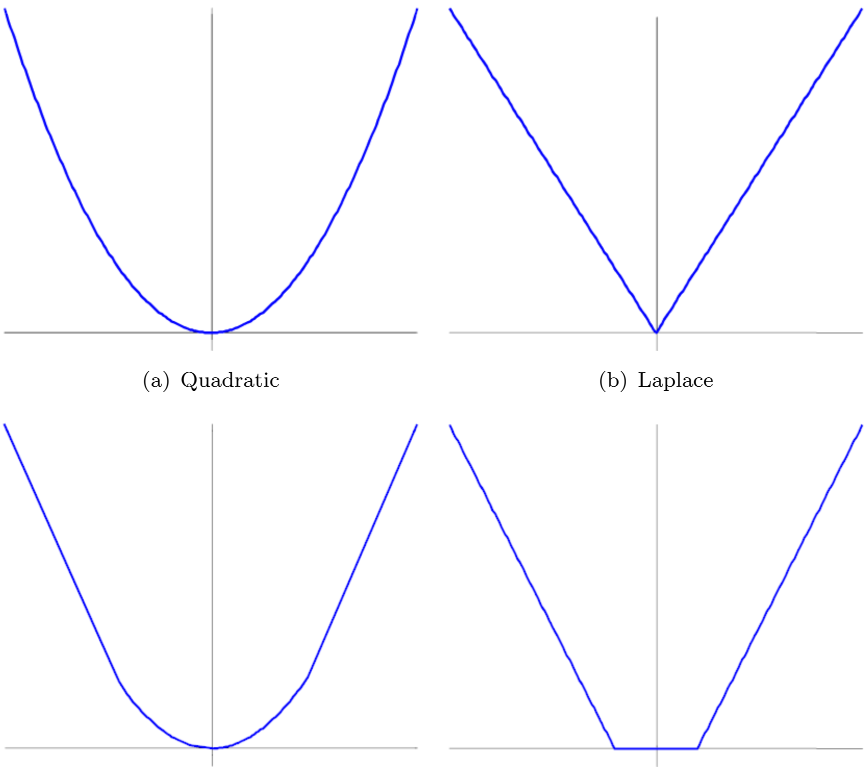
4.1 Applications

Larger and more complex classification problems have been attacked with SVC. Notably, Osuna et al. (1997) has applied SVC to the exacting problem of face recognition, with encouraging results. In conclusion, SVC provides a robust method for pattern classification by minimising overfitting problems by adopting the SRM principle. Use of a kernel function enables the curse of dimensionality to be addressed, and the solution implicitly contains support vectors that provide a description of the significant data for classification.

Chapter 5

Support Vector Regression

SVMs can also be applied to regression problems by the introduction of an alternative loss function, (Smola, 1996). The loss function must be modified to include a distance measure. Figure 5.1 illustrates four possible loss functions.



(a) Quadratic

(b) Laplace

(c) Huber

(d) ϵ-insensitive

FIGURE 5.1: Loss Functions

The loss function in Figure 5.1(a) corresponds to the conventional least squares error criterion. The loss function in Figure 5.1(b) is a Laplacian loss function that is less sensitive to outliers than the quadratic loss function. Huber proposed the loss function in Figure 5.1(c) as a robust loss function that has optimal properties when the underlying distribution of the data is unknown. These three loss functions will produce no sparseness in the support vectors. To address this issue Vapnik proposed the loss function in Figure 5.1(d) as an approximation to Huber's loss function that enables a sparse set of support vectors to be obtained.

5.1 Linear Regression

Consider the problem of approximating the set of data,

$$\mathcal{D} = \left\{ (x^1, y^1), \ldots, (x^l, y^l) \right\}, \quad x \in \mathbb{R}^n, y \in \mathbb{R}, \tag{5.1}$$

with a linear function,

$$f(x) = \langle w, x \rangle + b. \tag{5.2}$$

the optimal regression function is given by the minimum of the functional,

$$\Phi(w, \xi) = \frac{1}{2}\|w\|^2 + C \sum_i (\xi_i^- + \xi_i^+), \tag{5.3}$$

where C is a pre-specified value, and ξ^-, ξ^+ are slack variables representing upper and lower constraints on the outputs of the system.

5.1.1 ϵ-insensitive Loss Function

Using an ϵ-insensitive loss function, Figure 5.1(d),

$$L_\epsilon(y) = \begin{cases} 0 & \text{for} & |f(\mathbf{x}) - y| < \varepsilon \\ |f(\mathbf{x}) - y| - \varepsilon & \text{otherwise} & \end{cases}. \tag{5.4}$$

the solution is given by,

$$\max_{\alpha, \alpha^*} W(\alpha, \alpha^*) = \max_{\alpha, \alpha^*} -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i - \alpha_i^*) \left(\alpha_j - \alpha_j^*\right) \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \sum_{i=1}^{l} \alpha_i (y_i - \epsilon) - \alpha_i^* (y_i + \epsilon) \tag{5.5}$$

or alternatively,

$$\bar{\alpha}, \bar{\alpha}^* = \arg\min_{\alpha, \alpha^*} \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i - \alpha_i^*) \left(\alpha_j - \alpha_j^*\right) \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) y_i + \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) \epsilon \tag{5.6}$$

with constraints,

$$0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1, \ldots, l \tag{5.7}$$
$$\sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0.$$

Solving Equation 5.5 with constraints Equation 5.7 determines the Lagrange multipliers, α, α^* , and the regression function is given by Equation 5.2, where

$$\bar{\boldsymbol{w}} = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) \boldsymbol{x}_i \tag{5.8}$$
$$\bar{b} = -\frac{1}{2}\langle \bar{\boldsymbol{w}}, (\boldsymbol{x}_r + \boldsymbol{x}_s)\rangle.$$

The Karush-Kuhn-Tucker (KKT) conditions that are satisfied by the solution are,

$$\bar{\alpha}_i \bar{\alpha}_i^* = 0, \quad i = 1, \ldots, l. \tag{5.9}$$

Therefore the support vectors are points where exactly one of the Lagrange multipliers is greater than zero. When $\epsilon = 0$, we get the L_1 loss function and the optimisation problem is simplified,

$$\min_{\beta} \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \beta_i \beta_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \sum_{i=1}^{l} \beta_i y_i \tag{5.10}$$

with constraints,

$$-C \leq \beta_i \leq C, \quad i = 1, \ldots, l \tag{5.11}$$
$$\sum_{i=1}^{l} \beta_i = 0,$$

and the regression function is given by Equation 5.2, where

$$\bar{\boldsymbol{w}} = \sum_{i=1}^{l} \beta_i \boldsymbol{x}_i \tag{5.12}$$
$$\bar{b} = -\frac{1}{2}\langle \bar{\boldsymbol{w}}, (\boldsymbol{x}_r + \boldsymbol{x}_s)\rangle.$$

5.1.2 Quadratic Loss Function

Using a quadratic loss function, Figure 5.1(a),

$$L_{quad}(f(\mathbf{x}) - y) = (f(\mathbf{x}) - y)^2. \tag{5.13}$$

the solution is given by,

$$\begin{aligned}\max_{\alpha,\alpha^*} W(\alpha,\alpha^*) &= \max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i-\alpha_i^*)\left(\alpha_j-\alpha_j^*\right)\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle \\ &\quad +\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)\,y_i-\frac{1}{2C}\sum_{i=1}^{l}\left(\alpha_i^2+(\alpha_i^*)^2\right). \end{aligned} \tag{5.14}$$

The corresponding optimisation can be simplified by exploiting the KKT conditions, Equation 5.9 and noting that these imply $\beta_i^* = |\beta_i|$. The resultant optimisation problems is,

$$\min_{\beta}\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\beta_i\beta_j\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle-\sum_{i=1}^{l}\beta_i y_i+\frac{1}{2C}\sum_{i=1}^{l}\beta_i^2 \tag{5.15}$$

with constraints,

$$\sum_{i=1}^{l}\beta_i=0. \tag{5.16}$$

and the regression function is given by Equations 5.2 and 5.12.

5.1.3 Huber Loss Function

Using a Huber loss function, Figure 5.1(c),

$$L_{huber}\left(f(\mathbf{x})-y\right)=\begin{cases}\frac{1}{2}\left(f(\mathbf{x})-y\right)^2 & \text{for} \quad |f(\mathbf{x})-y|<\mu \\ \mu|f(\mathbf{x})-y|-\frac{\mu^2}{2} & \text{otherwise}\end{cases} \tag{5.17}$$

, the solution is given by,

$$\begin{aligned}\max_{\alpha,\alpha^*} W(\alpha,\alpha^*) &= \max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i-\alpha_i^*)\left(\alpha_j-\alpha_j^*\right)\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle \\ &\quad +\sum_{i=1}^{l}(\alpha_i-\alpha_i^*)\,y_i-\frac{1}{2C}\sum_{i=1}^{l}\left(\alpha_i^2+(\alpha_i^*)^2\right)\mu, \end{aligned} \tag{5.18}$$

The resultant optimisation problems is,

$$\min_{\beta}\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\beta_i\beta_j\langle \boldsymbol{x}_i,\boldsymbol{x}_j\rangle-\sum_{i=1}^{l}\beta_i y_i+\frac{1}{2C}\sum_{i=1}^{l}\beta_i^2\mu \tag{5.19}$$

with constraints,

$$\begin{aligned} &-C\le\beta_i\le C, \quad i=1,\ldots,l \\ &\sum_{i=1}^{l}\beta_i=0, \end{aligned} \tag{5.20}$$

x	y
1.0	-1.6
3.0	-1.8
4.0	-1.0
5.6	1.2
7.8	2.2
10.2	6.8
11.0	10.0
11.5	10.0
12.7	10.0

TABLE 5.1: Regression Data

and the regression function is given by Equations (56) and (66).

5.1.4 Example

Consider the example data set in Table 5.1. The SVR solution for a laplace loss function (Figure 5.1(b)) with no additional capacity control is shown in Figure 5.2.

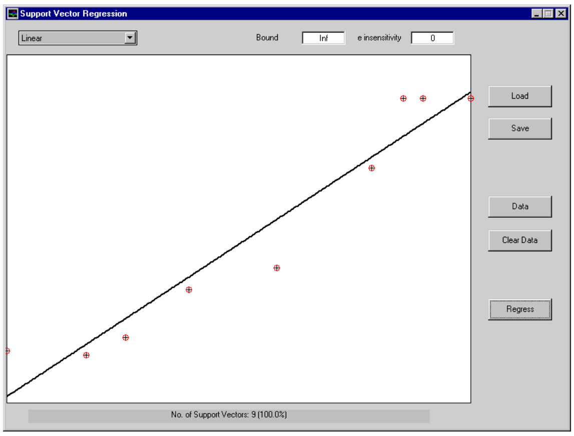


FIGURE 5.2: Linear regression

5.2 Non Linear Regression

Similarly to classification problems, a non-linear model is usually required to adequately model data. In the same manner as the non-linear SVC approach, a non-linear mapping can be used to map the data into a high dimensional feature space where linear regression is performed. The kernel approach is again employed to address the curse of dimensionality. The non-linear SVR solution, using an ϵ-insensitive loss function,

Figure 5.1(d), is given by,

$$\max_{\alpha,\alpha^*} W(\alpha, \alpha^*) = \max_{\alpha,\alpha^*} \sum_{i=1}^{l} \alpha_i^* (y_i - \epsilon) - \alpha_i (y_i + \epsilon) - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\alpha_i^* - \alpha_i) \left(\alpha_j^* - \alpha_j\right) K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{5.21}$$

with constraints,

$$\begin{aligned} &0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1, \ldots, l \\ &\sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0. \end{aligned} \tag{5.22}$$

Solving Equation 5.21 with constraints Equation 5.22 determines the Lagrange multipliers, α_i, α_i^*, and the regression function is given by,

$$f(\mathbf{x}) = \sum_{\mathrm{SVs}} (\bar{\alpha}_i - \bar{\alpha}_i^*) K(\mathbf{x}_i, \mathbf{x}) + \bar{b} \tag{5.23}$$

where

$$\begin{aligned} \langle \bar{\boldsymbol{w}}, \boldsymbol{x} \rangle &= \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \bar{b} &= -\frac{1}{2} \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) (K(\boldsymbol{x}_i, \boldsymbol{x}_r) + K(\boldsymbol{x}_i, \boldsymbol{x}_s)). \end{aligned} \tag{5.24}$$

As with the SVC the equality constraint may be dropped if the Kernel contains a bias term, b being accommodated within the Kernel function, and the regression function is given by,

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} (\bar{\alpha}_i - \bar{\alpha}_i^*) K(\boldsymbol{x}_i, \boldsymbol{x}). \tag{5.25}$$

The optimisation criteria for the other loss functions of Chapter 5.1 are similarly obtained by replacing the dot product with a kernel function. The ϵ-insensitive loss function is attractive because unlike the quadratic and Huber cost functions, where all the data points will be support vectors, the SV solution can be sparse. The quadratic loss function produces a solution which is equivalent to ridge regression, or zeroth order regularisation, where the regularisation parameter $\lambda = \frac{1}{2C}$.

5.2.1 Examples

To illustrate some of the non-linear SVR solutions, various kernel functions were used to model the regression data in Table 5.1, with an ϵ-insensitive loss function ($\epsilon = 0.5$) and no additional capacity control. Figure 5.3 shows the SVR solution for a degree 2 polynomial, with the SV circled as before. The dotted line describes the ϵ-insensitive region around the solution (N.B. if all the data points lie within this region there will be zero error associated with the loss function). The result demonstrates that there are no support vectors within the ϵ-insensitive region. Figure 5.4 illustrates the SVR solution

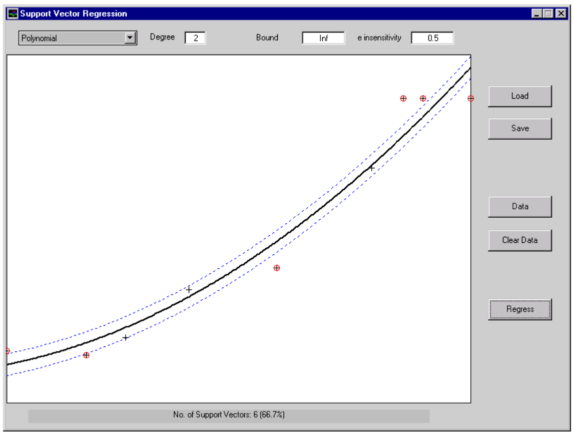


FIGURE 5.3: Polynomial Regression

for a radial basis function with $\sigma = 1.0$. In this example the model is flexible enough to model the function with zero error associated with the loss function, as is verified by the fact that all the data points lie on, or within, the ϵ-insensitive zone. Figure 5.5 shows the

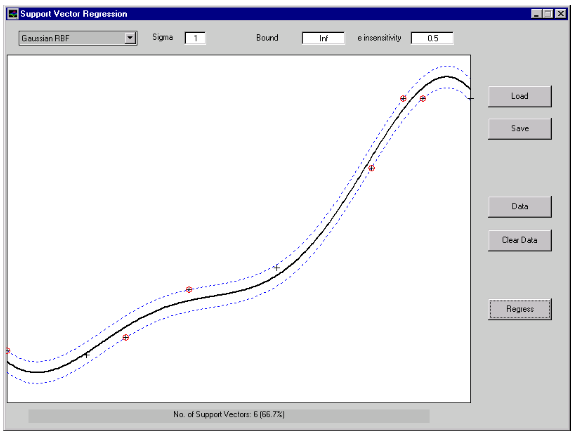


FIGURE 5.4: Radial Basis Function Regression

SVR solution for a linear spline kernel. The resulting model is a piecewise cubic spline, and again due to the high capacity of this function it is able to model the data with zero loss function error, but notice that overfitting is controlled. Figure 5.6 shows the SVR solution for an infinite B-spline kernel, which has a similar solution to the spline kernel except for the endpoints. Figure 5.7 shows the solution for an exponential RBF kernel, which is a piecewise linear spline. Although this model has a high capacity it shows sensible behaviour in the extremity regions.

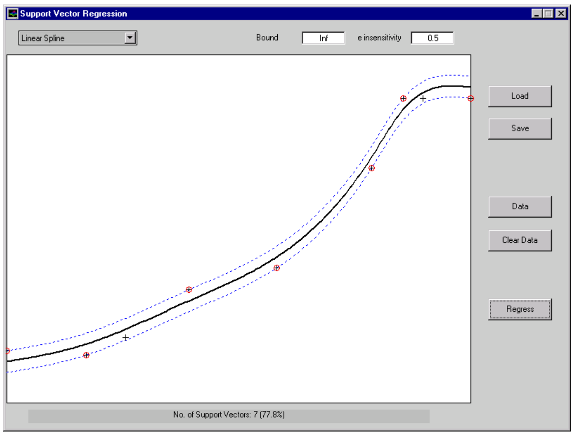


FIGURE 5.5: Spline Regression

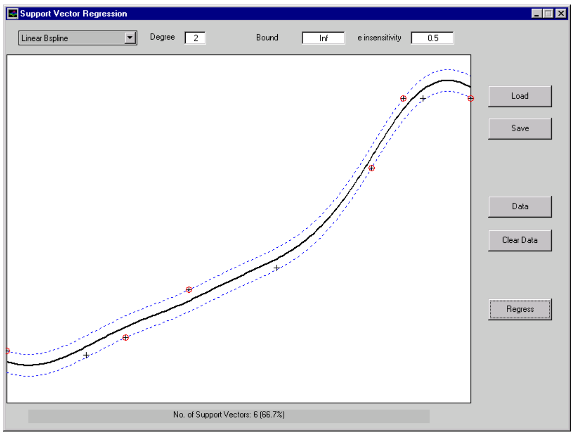


FIGURE 5.6: B-spline Regression

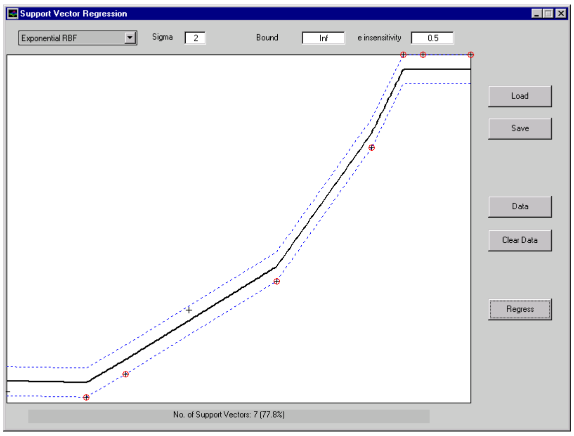


FIGURE 5.7: Exponential RBF Regression

5.2.2 Comments

In the regression method it is necessary to select both a representative loss function and any additional capacity control that may be required. These considerations must be based on prior knowledge of the problem and the distribution of the noise. In the absence of such information Huber's robust loss function, Figure 5.1(c), has been shown to be a good alternative (Vapnik, 1995). Vapnik developed the ϵ-insensitive loss function as a trade-off between the robust loss function of Huber and one that enables sparsity within the SVs. However, its implementation is more computationally expensive and the ϵ-insensitive region can have drawbacks, as will be demonstrated in the next section.

Chapter 6

Regression Example: Titanium Data

The example given here considers the titanium data (Dierckx, 1993) as an illustrative example of a one dimensional non-linear regression problem. There are three methods for controlling the regression model, the loss function, the kernel, and additional capacity control, C. The results shown in this chapter were obtained using an ϵ-insensitive loss function (e=0.05), with different kernels and different degrees of capacity control. Figure 6.1 illustrates the solution for a linear spline kernel and no additional capacity control. It is evident that the solution lies within the ϵ-insensitive region. Figure 6.2 illustrates

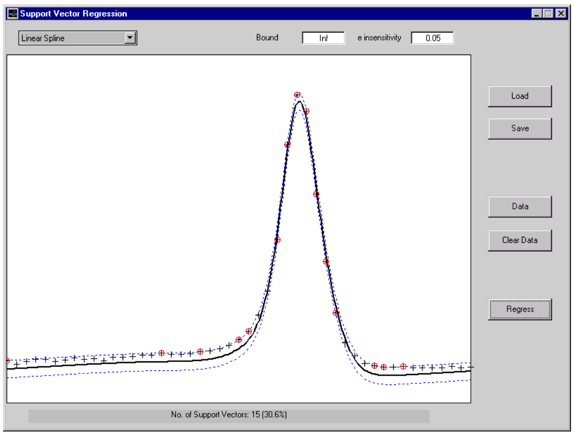


FIGURE 6.1: Titanium Linear Spline Regression ($\epsilon = 0.05$, $C = \infty$)

the solution for a B-spline kernel with no additional capacity control. This particular B-spline kernel would appear to be prone to oscillation when an ϵ-insensitive region is used, and hence the linear spline kernel, or an alternative loss function is to be preferred. Figure 6.3 illustrates the solution for a Gaussian RBF kernel ($\sigma = 1.0$) with no additional capacity control. It can be seen that the RBF is too wide to accurately model the data. Figure 6.4 illustrates the solution for a Gaussian RBF kernel ($\sigma = 0.3$) with no additional

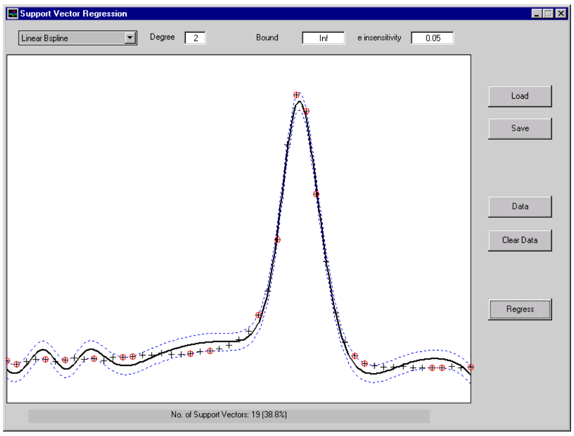


FIGURE 6.2: Titanium B-Spline Regression ($\epsilon = 0.05$, $C = \infty$)

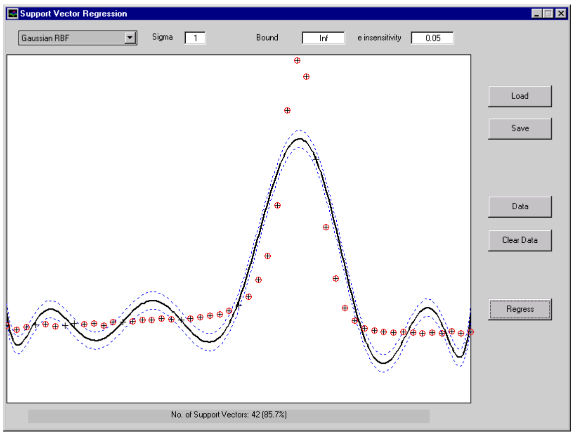


FIGURE 6.3: Titanium Gaussian RBF Regression ($\epsilon = 0.05$, $\sigma = 1.0$, $C = \infty$)

capacity control. It can be seen that the RBF is now able to accurately model the data. However, this is at the expense of the introduction of oscillation, which is not penalised in the ϵ-insensitive region. Figure 6.5 illustrates the solution for an exponential RBF

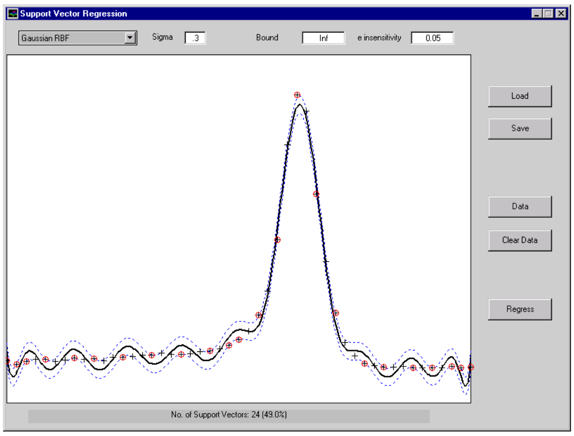


FIGURE 6.4: Titanium Gaussian RBF Regression ($\epsilon = 0.05$, $\sigma = 0.3$, $C = \infty$)

kernel ($\sigma = 0.3$) with no additional capacity control. The corresponding solution is a piece-wise linear function and consequently oscillation is avoided. Figure 6.6 illustrates

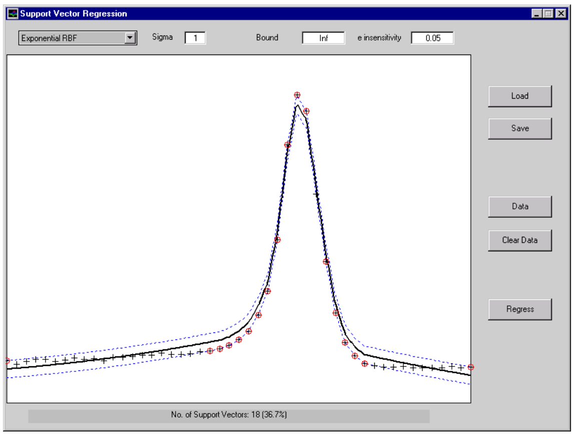


FIGURE 6.5: Titanium Exponential RBF Regression ($\epsilon = 0.05$, $\sigma = 1.0$, $C = \infty$)

the solution for a degree 3 Fourier kernel with no additional capacity control. The solution suffers similar problems to the wide Gaussian RBF kernel in that the kernel cannot accurately model the data. Figure 6.7 illustrates the solution for a linear spline

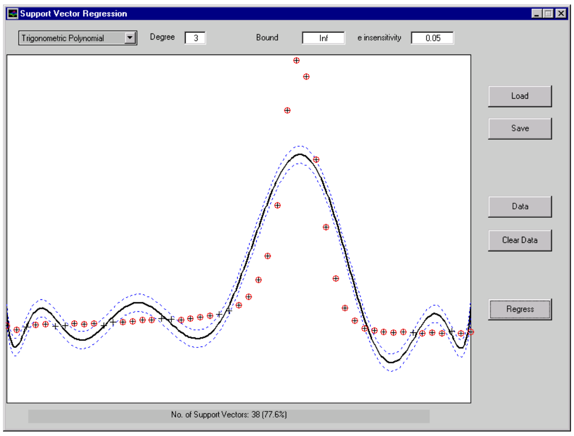


FIGURE 6.6: Titanium Fourier Regression ($\epsilon = 0.05$, degree 3, $C = \infty$)

kernel, with additional capacity control, ($C = 10$). The extra capacity control renders the solution incapable of accurately modelling the peak in the data, in contrast to Figure 6.1. Figure 6.8 illustrates the solution for a B-spline kernel, with additional capacity control, ($C = 10$). The extra capacity control renders the solution incapable of accurately modelling the peak in the data, in contrast to Figure 6.2. The examples that have been shown here are not a representative set. The ϵ-insensitive region has been exaggerated for the purposes of illustration, and typically careful selection of additional capacity control with methods such as cross validation will be required. The ϵ-insensitive loss function may be an inappropriate choice for particular kernels causing the solution to oscillate within the ϵ-insensitive region.

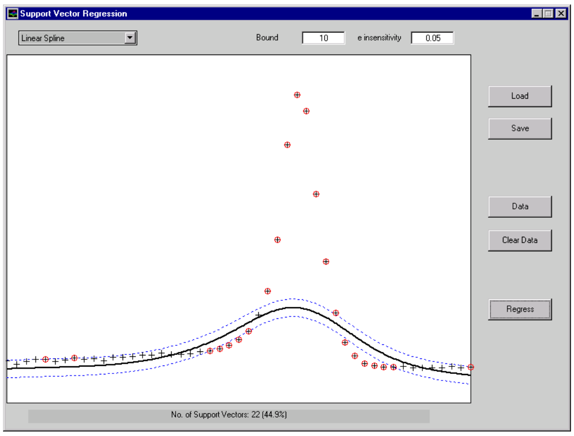


FIGURE 6.7: Titanium Linear Spline Regression ($\epsilon = 0.05$, $C = 10$)

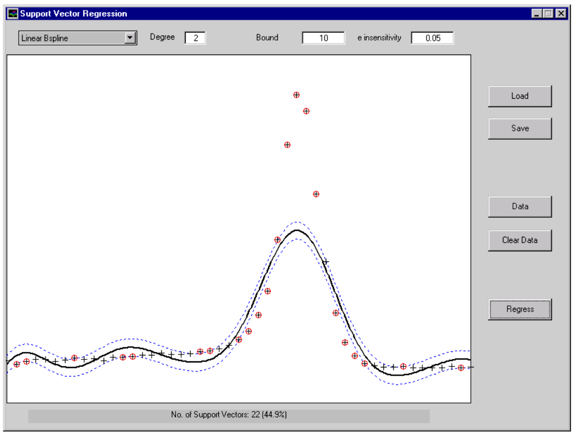


FIGURE 6.8: Titanium B-Spline Regression ($\epsilon = 0.05$, $C = 10$)

6.1 Applications

SVR has been applied to some time series modelling problems (Mukherjee et al., 1997). Notably, (Müller et al., 1999) has achieved excellent results in applying SVR to one of the data sets from the Santa Fe time series competition.

Chapter 7

Conclusions

Support Vector Machines are an attractive approach to data modelling. They combine generalisation control with a technique to address the curse of dimensionality. The formulation results in a global quadratic optimisation problem with box constraints, which is readily solved by interior point methods. The kernel mapping provides a unifying framework for most of the commonly employed model architectures, enabling comparisons to be performed. In classification problems generalisation control is obtained by maximising the margin, which corresponds to minimisation of the weight vector in a canonical framework. The solution is obtained as a set of support vectors that can be sparse. These lie on the boundary and as such summarise the information required to separate the data. The minimisation of the weight vector can be used as a criterion in regression problems, with a modified loss function. Future directions include: A technique for choosing the kernel function and additional capacity control; Development of kernels with invariances.

Appendix A

Implementation Issues

The resulting optimisation problems are dependent upon the number of training examples. As such, when this data is large methods have been proposed for speeding up the algorithm by decomposing the problem into smaller ones. Approximate (Stitson and Weston, 1996) and exact (Osuna et al., 1997) methods have been proposed. MATLAB implementations of the main support vector routines are shown below. Note that these routines are not optimised in any sense! Typically the quadratic matrix, H, is badly conditioned which can render quadratic program optimisers incapable of producing an accurate solution. To address this, a quick fix of using zero order regularisation can be used, but too large a value will perturb the solution significantly. (Note that this is the capacity control used in some of the SVR routines.)

A.1 Support Vector Classification

The optimisation problem can be expressed in matrix notation as,

$$\min_x \frac{1}{2}\alpha^T H\alpha + c^T\alpha \tag{A.1}$$

where

$$H = ZZ^T, \quad c^T = (-1, \ldots, -1) \tag{A.2}$$

with constraints

$$\alpha^T Y = 0, \quad \alpha_i \geq 0, i = 1, \ldots, l. \tag{A.3}$$

where

$$Z = \begin{bmatrix} y_1\mathbf{x}_1 \\ \vdots \\ y_l\mathbf{x}_l \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_l \end{bmatrix} \tag{A.4}$$

The MATLAB implementation is given below:

```
function [nsv, alpha, b0] = svc(X,Y,ker,C)
%SVC Support Vector Classification
%
%  Usage: [nsv alpha bias] = svc(X,Y,ker,C)
%
%  Parameters: X      - Training inputs
%              Y      - Training targets
%              ker    - kernel function
%              C      - upper bound (non-separable case)
%              nsv    - number of support vectors
%              alpha  - Lagrange Multipliers
%              b0     - bias term
%
%  Author: Steve Gunn (srg@ecs.soton.ac.uk)

  if (nargin <2 | nargin>4) % check correct number of arguments
    help svc
  else

    fprintf('Support Vector Classification\n')
    fprintf('_____________________________\n')
    n = size(X,1);
    if (nargin<4) C=Inf;, end
    if (nargin<3) ker='linear';, end

    % Construct the Kernel matrix
    fprintf('Constructing ...\n');
    H = zeros(n,n);
    for i=1:n
       for j=1:n
          H(i,j) = Y(i)*Y(j)*svkernel(ker,X(i,:),X(j,:));
       end
    end
    c = -ones(n,1);

    % Add small amount of zero order regularisation to
    % avoid problems when Hessian is badly conditioned.
    H = H+1e-10*eye(size(H));

    % Set up the parameters for the Optimisation problem

    vlb = zeros(n,1);      % Set the bounds: alphas >= 0
    vub = C*ones(n,1);     %                 alphas <= C
    x0 = zeros(n,1);       % The starting point is [0 0 0   0]
    neqcstr = nobias(ker); % Set the number of equality constraints (1 or 0)
    if neqcstr
       A = Y';, b = 0;     % Set the constraint Ax = b
    else
       A = [];, b = [];
    end

    % Solve the Optimisation Problem

    fprintf('Optimising ...\n');
    st = cputime;

    [alpha lambda how] = qp(H, c, A, b, vlb, vub, x0, neqcstr);
```

```
  fprintf('Execution time: %4.1f seconds\n',cputime - st);
  fprintf('Status : %s\n',how);
  w2 = alpha'*H*alpha;
  fprintf('|w0|^2    : %f\n',w2);
  fprintf('Margin    : %f\n',2/sqrt(w2));
  fprintf('Sum alpha : %f\n',sum(alpha));


  % Compute the number of Support Vectors
  epsilon = svtol(alpha);
  svi = find( alpha > epsilon);
  nsv = length(svi);
  fprintf('Support Vectors : %d (%3.1f%%)\n',nsv,100*nsv/n);

  % Implicit bias, b0
  b0 = 0;

  % Explicit bias, b0
  if nobias(ker) ~= 0
    % find b0 from average of support vectors on margin
    % SVs on margin have alphas: 0 < alpha < C
    svii = find( alpha > epsilon & alpha < (C - epsilon));
    if length(svii) > 0
      b0 =  (1/length(svii))*sum(Y(svii) - H(svii,svi)*alpha(svi).*Y(svii));
    else
      fprintf('No support vectors on margin - cannot compute bias.\n');
    end
  end

end
```

LISTING A.1: Support Vector Classification MATLAB Code

A.2 Support Vector Regression

The optimisation problem for an ϵ-insensitive loss function can be expressed in matrix notation as,

$$\min_x \frac{1}{2} x^T H x + c^T x \tag{A.5}$$

where

$$H = \begin{bmatrix} XX^T & -XX^T \\ -XX^T & XX^T \end{bmatrix}, \quad c = \begin{bmatrix} \varepsilon + Y \\ \varepsilon - Y \end{bmatrix}, \quad x = \begin{bmatrix} \alpha \\ \alpha^* \end{bmatrix} \tag{A.6}$$

with constraints

$$x \cdot (1, \ldots, 1, -1, \ldots, -1) = 0, \quad \alpha_i, \alpha_i^* \geq 0, i = 1, \ldots, l. \tag{A.7}$$

where

$$X = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_l \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_l \end{bmatrix} \tag{A.8}$$

The MATLAB implementation is given below:

```
function [nsv, beta, bias] = svr(X,Y,ker,C,loss,e)
%SVR Support Vector Regression
%
%  Usage: [nsv beta bias] = svr(X,Y,ker,C,loss,e)
%
%  Parameters: X      - Training inputs
%              Y      - Training targets
%              ker    - kernel function
%              C      - upper bound (non-separable case)
%              loss   - loss function
%              e      - insensitivity
%              nsv    - number of support vectors
%              beta   - Difference of Lagrange Multipliers
%              bias   - bias term
%
%  Author: Steve Gunn (srg@ecs.soton.ac.uk)


  if (nargin < 3 | nargin > 6) % check correct number of arguments
    help svr
  else

    fprintf('Support Vector Regressing ....\n')
    fprintf('______________________________\n')
    n = size(X,1);
    if (nargin<6) e=0.0;, end
    if (nargin<5) loss='eInsensitive';, end
    if (nargin<4) C=Inf;, end
    if (nargin<3) ker='linear';, end

    % Construct the Kernel matrix

    fprintf('Constructing ...\n');
    H = zeros(n,n);
    for i=1:n
       for j=1:n
          H(i,j) = svkernel(ker,X(i,:),X(j,:));
       end
    end

    % Set up the parameters for the Optimisation problem
    switch lower(loss)
      case 'einsensitive',
        Hb = [H -H; -H H];
        c = [(e*ones(n,1) - Y); (e*ones(n,1) + Y)];
        vlb = zeros(2*n,1);     % Set the bounds: alphas >= 0
        vub = C*ones(2*n,1);    %                 alphas <= C
        x0 = zeros(2*n,1);      % The starting point is [0 0 0   0]
        neqcstr = nobias(ker); % Set the number of equality constraints (1 or 0)
        if neqcstr
          A = [ones(1,n) -ones(1,n)];, b = 0;      % Set the constraint Ax = b
        else
          A = [];, b = [];
        end
      case 'quadratic',
        Hb = H + eye(n)/(2*C);
        c = -Y;
        vlb = -1e30*ones(n,1);
```

```
    vub = 1e30*ones(n,1);
    x0 = zeros(n,1);                 % The starting point is [0 0 0   0]
    neqcstr = nobias(ker);           % Set the number of equality constraints (1 or 0)
    if neqcstr
      A = ones(1,n);, b = 0;         % Set the constraint Ax = b
    else
      A = [];, b = [];
    end
  otherwise, disp('Error: Unknown Loss Function\n');
end

% Add small amount of zero order regularisation to
% avoid problems when Hessian is badly conditioned.
% Rank is always less than or equal to n.
% Note that adding to much reg will peturb solution

Hb = Hb+1e-10*eye(size(Hb));

% Solve the Optimisation Problem

fprintf('Optimising ...\n');
st = cputime;

[alpha lambda how] = qp(Hb, c, A, b, vlb, vub, x0, neqcstr);

fprintf('Execution time : %4.1f seconds\n',cputime - st);
fprintf('Status : %s\n',how);

switch lower(loss)
  case 'einsensitive',
    beta =  alpha(1:n) - alpha(n+1:2*n);
  case 'quadratic',
    beta = alpha;
end
fprintf('|w0|^2    : %f\n',beta'*H*beta);
fprintf('Sum beta : %f\n',sum(beta));

% Compute the number of Support Vectors
epsilon = svtol(abs(beta));
svi = find( abs(beta) > epsilon );
nsv = length( svi );
fprintf('Support Vectors : %d (%3.1f%%)\n',nsv,100*nsv/n);

% Implicit bias, b0
bias = 0;

% Explicit bias, b0
if nobias(ker) ~= 0
  switch lower(loss)
    case 'einsensitive',
      % find bias from average of support vectors with interpolation error e
      % SVs with interpolation error e have alphas: 0 < alpha < C
      svii = find( abs(beta) > epsilon & abs(beta) < (C - epsilon));
      if length(svii) > 0
        bias = (1/length(svii))*sum(Y(svii) - e*sign(beta(svii)) - H(svii,svi)*beta(svi));
      else
        fprintf('No support vectors with interpolation error e - cannot compute bias.\n');
        bias = (max(Y)+min(Y))/2;
      end
```

```
        case 'quadratic',
            bias = mean(Y - H*beta);
      end
    end

  end
```

Listing A.2: Support Vector Regression MATLAB Code

Appendix B

MATLAB SVM Toolbox

A MATLAB toolbox implementing SVM is freely available for academic purposes:

1. Download it from: http://www.isis.ecs.soton.ac.uk/resources/svminfo/
2. Extract the tar file **svm.tar** under the matlab toolbox directory.
3. Add **.../matlab/toolbox/svm** to your MATLAB path.
4. Type **help svm** at the MATLAB prompt for help.

The two main user interfaces are for 2D classification and 1D regression (**uiclass** and **uiregress** respectively).

Bibliography

M. A. Aizerman, E. M. Braverman, and L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 686:337–404, 1950.

R.E. Bellman. *Adaptive Control Processes.* Princeton University Press, Princeton, NJ, 1961.

V. Blanz, B. Schölkopf, H. Bülthoff, C. Burges, V. Vapnik, and T. Vetter. Comparison of view–based object recognition algorithms using realistic 3D models. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks — ICANN'96*, pages 251 – 256, Berlin, 1996. Springer Lecture Notes in Computer Science, Vol. 1112.

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

P. Dierckx. *Curve and Surface Fitting with Splines.* Monographs on Numerical Analysis. Clarendon Press, Oxford, 1993.

F. Girosi. An equivalence between sparse approximation and Support Vector Machines. A.I. Memo 1606, MIT Artificial Intelligence Laboratory, 1997.

S.R. Gunn, M. Brown, and K.M. Bossley. Network performance assessment for neurofuzzy data modelling. In X. Liu, P. Cohen, and M. Berthold, editors, *Intelligent Data Analysis*, volume 1208 of *Lecture Notes in Computer Science*, pages 313–323, 1997.

J. Hadamard. *Lectures on the Cauchy Problem in Linear Partial Differential Equations.* Yale University Press, 1923.

N. Heckman. The theory and application of penalized least squares methods or reproducing kernel hilbert spaces made easy, 1997.

M. Minoux. *Mathematical Programming: Theory and Algorithms.* John Wiley and Sons, 1986.

S. Mukherjee, E. Osuna, and F. Girosi. Nonlinear prediction of chaotic time series using a support vector machine. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, New York, 1997. IEEE.

K.-R. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 243–254, Cambridge, MA, 1999. MIT Press. Short version appeared in ICANN'97, Springer Lecture Notes in Computer Science.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, *Neural Networks for Signal Processing VII — Proceedings of the 1997 IEEE Workshop*, pages 276 – 285, New York, 1997. IEEE.

T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317:314–319, 1985.

A. J. Smola. Regression estimation with support vector learning machines. Master's thesis, Technische Universität München, 1996.

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

M. O. Stitson and J. A. E. Weston. Implementational issues of support vector machines. Technical Report CSD-TR-96-18, Computational Intelligence Group, Royal Holloway, University of London, 1996.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, N.Y., 1995. ISBN 0-387-94559-8.

V. Vapnik. *Statistical Learning Theory*. Springer, N.Y., 1998.

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

G. Wahba. *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.