Theory of Support Vector Machines

M. O. Stitson, J. A. E. Weston
A. Gammerman, V. Vovk, V. Vapnik

Technical Report

CSD-TR-96-17

December 31, 1996

Royal Holloway
University of London

Department of Computer Science
Egham, Surrey TW20 0EX, England

The main text of this document is set in Computer Modern 11pt.

This document was prepared with $\LaTeX$ using a PC running Linux.

It was printed on a 600 dpi Hewlett Packard laser printer.

Contents

Abstract

Support Vector Machines perform structural risk minimisation. They create a classifier with minimised VC Dimension. If the VC Dimension is low, the expected probability of error is low as well, which means good generalisation.

Support Vector Machines use a linear separating hyperplane to create a classifier, yet some problems cannot be linearly separated in the original input space. Support Vector Machines can non-linearly transform the original input space into a higher dimensional feature space. In this feature space it is trivial [1] to find a linear optimal separating hyperplane. This hyperplane is optimal in the sense of being a maximal margin classifier with respect to the training data.

Open problems with this approach lie in two areas:

- The theoretical problem of which non-linear transformation to use.
- The practical problem of creating an efficient implementation as the basic algorithm has memory requirements which are squared with respect to the number of training examples and computational complexity which is also related to the number of training examples.

[1] Trivial only in theory, in practice this is still a computationally hard problem.

Chapter 1

Introduction

This chapter will introduce the basic idea behind support vector machines and why we use them. It starts with a general description of the problem and then gives several solution approaches.

1.1 The problem

Learning is a wide field of which we will only concentrate on a small subfield here. The problem we look at initially is the problem of finding binary classifiers.

Say we are given the weight and height of a person, we want to find a way of determining their gender. If we are given a set of examples with height, weight and gender, we can come up with a hypothesis which will enable us to determine a persons gender from their weight and height. This hypothesis might be wrong, but it hopefully is the best we can acheive from the data.

This learning from examples can also be applied to function approximation or regression estimation.

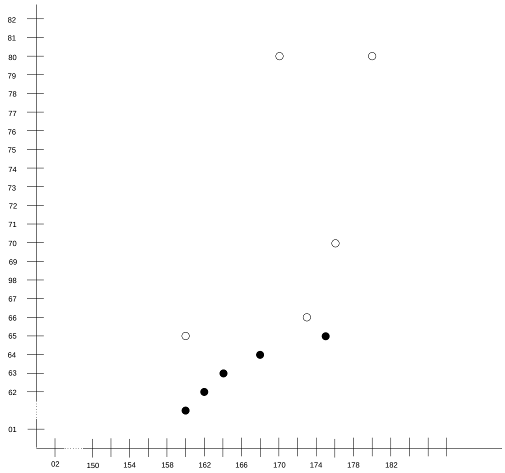
In mathematical terms this can be described by plotting the weights and heights in a two-dimensional coordinate system and then drawing a dividing line or separating hyperplane which divides our weight/height points into two regions, one female and one male.

No.	Height	Weight	Gender
1	180	80	m
2	173	66	m
3	170	80	m
4	176	70	m
5	160	65	m
6	160	61	f

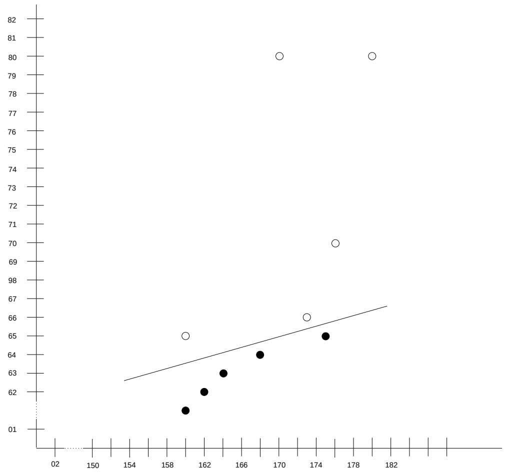
continued on next page

continued from previous page			
No.	Height	Weight	Gender
7	162	62	f
8	168	64	f
9	164	63	f
10	175	65	f

If we plot this table we get the following picture, where the black dots are classified as female and the white dats as male:



It can be seen that there are a number of lines we can draw to divide the space into two regions. From an intuitive viewpoint the best line would be the line that has the furthest distance to any of our training points. This is because a point near a male point is probably male and a point near a female point is probably female, i.e. two people with the same weight and height are probably of the same sex. The best line or optimal separating hyperplane would then roughly be:



If we can find such a separating hyperplane automatically from the training data, as a mathematical function, it is possible using basic geometry to determine which side of the line any given point lies and so to make a classification of unseen examples.

1.2 Neural networks as a solution

Most neural networks are designed to find a separating hyperplane. This is not necessarily optimal. In fact many neural networks start with a random line and move it, until all training points are on the right side of the line. This inevatibly leaves training points very close to the line in a non-optimal way.

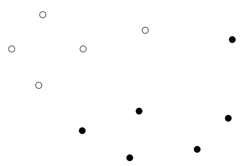
There are however now approaches in which a large margin classifier, i.e. a line approaching the optimal is sought.

1.3 Support Vector machines as a solution

Support Vector Machines use geometric properties to exactly calculate the optimal separating hyperplane directly from the training data. They also introduce methods to deal with non-linearly separable cases, i.e. where no straight line can be found, and cases in which there is noise in the training data, i.d. some of the training examples are wrong. With the ideas of chunking SV machines can also be used for certain forms of online learning.

Chapter 2

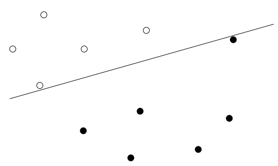
Pattern Recognition: The Optimal Seperating Hyperplane



In figure 2 it is easy to determine a separating hyperplane given the training data

$$(x_1, y_1), \ldots, (x_l, y_l), x \in R^n, y \in \{+1, -1\}$$

manually.



The hyperplane $(w \cdot x) + b = 0$ satisfies the conditions:

$$(w \cdot x_i) + b > 0 \text{ if } y_i = 1$$
$$(w \cdot x_i) + b < 0 \text{ if } y_i = -1$$

which is equivalent to[1]

$$(w \cdot x_i) + b \geq 1 \text{ if } y_i = 1 \tag{2.1}$$
$$(w \cdot x_i) + b \leq -1 \text{ if } y_i = -1 \tag{2.2}$$

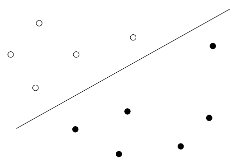
[1] given some constant factor, i.e. a different w and b

which is equivalent to

$$y_i[(w \cdot x_i) + b] \geq 1, i = 1, \ldots l$$

This however is not a good hyperplane as some points are very near the hyperplane and points near them might be classified differently although intuitively they should be the same class. Therefore it would be preferable to have a hyperplane with a large margin or even the largest possible margin. This maximum margin hyperplane is known as the optimal separating hyperplane.

In our example it would look like this:



To find the optimal separating hyperplane we have to find the hyperplane which satisfies the above condition[2] and maximises the minimum distance between the hyperplane and any sample of the training data.

$$\rho(w, b) = \min_{\{x_i | y_i = 1\}} \frac{w \cdot x_i + b}{|w|} - \max_{\{x_j | y_j = -1\}} \frac{w \cdot x_j + b}{|w|} \tag{2.3}$$

is the sum of the distances from the nearest two points to the separating hyperplane, which has to be maximised.

From equation 2.3 and equations 2.1 follows:

$$\begin{aligned} & \rho(w, b) &= \min_{\{x_i | y_i = 1\}} \frac{1}{|w|} - \max_{\{x_j | y_j = -1\}} \frac{-1}{|w|} && (2.4) \\ \Leftrightarrow \; & \rho(w, b) &= \frac{1}{|w|} - \frac{-1}{|w|} && (2.5) \\ \Leftrightarrow \; & \rho(w, b) &= \frac{2}{|w|} && (2.6) \\ \Leftrightarrow \; & \rho(w, b) &= \frac{2}{\sqrt{w \cdot w}} && (2.7) \end{aligned}$$

The optimal seperating hyperplane therefore minimises

$$\Phi(w) = \frac{1}{2} w \cdot w$$

[2]This is true for all separating hyperplanes.

with respect to both the vector w and the scalar b.

The solution to this problem can be found by finding the saddle point of the Lagrange functional:

$$L(w,b,\alpha) = \frac{1}{2} w \cdot w - \sum_{i=1}^{l} \alpha_i \left\{ [(x_i \cdot w) + b] y_i - 1 \right\} \tag{2.8}$$

where α_i are Lagrange multipliers. The Lagrangian has to be minimised with respect to w, b and maximised with respect to $\alpha_i \geq 0$.

In the saddle point the solutions w_0, b_0 and α^0 should satisfy the following conditions:

$$\begin{aligned} & \tfrac{\partial L(w_0,b_0,\alpha^0)}{\partial b} &= 0 \quad (2.9)\\ \Leftrightarrow \quad & \textstyle\sum_{i=1}^{l} \alpha_i^0 y_i &= 0 \quad (2.10) \end{aligned}$$

$$\begin{aligned} & \tfrac{\partial L(w_0,b_0,\alpha^0)}{\partial w} &= 0 \quad (2.11)\\ \Leftrightarrow \quad & w_0 - \textstyle\sum_{i=1}^{l} \alpha_i^0 x_i y_i &= 0 \quad (2.12) \end{aligned}$$

This results in the following properties of the optimal seperating hyperplane:

1. Constraint on the coefficients α_i^0 from equation 2.10:

$$\sum_{i=1}^{l} \alpha_i^0 y_i = 0, \qquad \alpha_i^0 \geq 0, i = 1, \ldots, l \tag{2.13}$$

2. The vector w_0 is a linear combination of the vecors in the training set (from equation 2.12):

$$w_0 = \sum_{i=1}^{l} \alpha_i^0 y_i x_i, \qquad \alpha_i^0 \geq 0, i = 1, \ldots, l \tag{2.14}$$

3. Only the so-called *support vectors* can have non-zero coefficients α_i^0 in the expansion of w_0:

$$w_0 = \sum_{\text{support vectors}} \alpha_i^0 y_i x_i, \alpha_i^0 > 0 \tag{2.15}$$

The Kühn-Tucker Theorem states that the necessary and sufficient conditions for the Optimal hyperplane are that the separating hyperplane satisfies the conditions:

$$\alpha_i^0 \{[(x_i \cdot w_0) + b_0]y_i - 1\} = 0, i = 1, \dots, l \tag{2.16}$$

as well as being a saddle point.

Substituting these results into the Lagrangian, the following functional is obtained:

$$\begin{aligned} W(\alpha) = & \tfrac{1}{2} w \cdot w - \textstyle\sum_{i=1}^{l} \alpha_i \{[(x_i \cdot w) + b]y_i - 1\} & (2.17)\\ = & \tfrac{1}{2} w \cdot w - \textstyle\sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w) - b \sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i & (2.18)\\ = & \tfrac{1}{2} w \cdot w - w \cdot w - 0 + \textstyle\sum_{i=1}^{l} \alpha_i & (2.19)\\ = & \textstyle\sum_{i=1}^{l} \alpha_i - \tfrac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) & (2.20) \end{aligned}$$

This functional has to be maximised in the non-negative quadrant,

$$\alpha_i \geq 0, i = 1, \dots, l \tag{2.21}$$

under the constraint

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{2.22}$$

Once the solution to this problem has been found in form of a vector $\alpha^0 = (\alpha_1^0, \dots \alpha_l^0)$, an indicator function can be constructed:

$$f(x) = \text{sign}\Big(\sum_{\text{support vectors}} y_i \alpha_i^0 (x_i \cdot x) - b_0\Big) \tag{2.23}$$

where x_i are the support vectors, α_i^0 are the Lagrange multipliers and b_0 is the threshold

$$b_0 = \frac{1}{2}[(w_0 \cdot x^*(1)) + (w_0 \cdot x^*(-1))] \tag{2.24}$$

where $x^*(1)$ is any support vector belonging to the first class and $x^*(-1)$ is any support vector belongign to the second class.

This solution only holds for separable data, but has to be slightly modified for non-separable data.

For non-separable data the α_i^0's have to be bounded:

$$0 \leq \alpha_i^0 \leq C \tag{2.25}$$

where C is a constant choosen a priori.

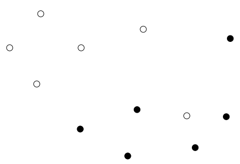
For more information on this see [Vap95] pages 131-133.

Chapter 3

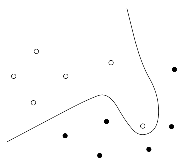
Pattern Recognition: The Support Vector Machine

Support Vector machines form an extension to the optimal separating hyperplane method. They map the input space into a high-dimensional feature space through some non-linear mapping choosen a priori and then construct the optimal separating hyperplane in the feature space. This makes it possible to construct linear decision surfaces in feature space which correspond to non-linear decision surfaces in input space.

Taking the example from the previous section with one extra point



it is obvious that no linear separating hyperplane can be found, yet if we could use a non-linear separating hyperplane, which is equivalent to mapping into a high dimensional space, a separation is possible:



The main problem that arises here is how to handle the problem of mapping all vectors into the feature space. This mapping would generate vectors with high dimensionality entailing large storage and computational expense. This curse of dimensionality however turns out not to be a problem as all we ever need is the dot product of the vectors in feature space.

If we choose a mapping from the original space X to the higher dimensional space Z $(k : X \to Z)$:

$$k(x) = z$$

where $x \in X$ and $z \in Z$ then we would have to maximise[1]

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (k(x_i) \cdot k(x_j)) \tag{3.1}$$

to find the Lagrangian multipliers under the constraints from the previous chapter and calculate the value of the classifier

$$f(x) = \text{sign}(\sum_{\text{support vectors}} y_i \alpha_i^0 (k(x_i) \cdot k(x) - b_0) \tag{3.2}$$

as before.

We do not have to know $k(x)$ as long as we know $K(x_i, x_j) = k(x_i) \cdot k(x_j)$.

According to Hilbert-Schmidt theory $K(x_i, x_j)$ can be any symetric function satisfying the following conditions [CH53]:

Theorem 3.1 *(Mercer) To garantuee that the symetric function $K(x_i, x_j)$ has the expansion*

$$K(x_i, x_j) = \sum_{k=1}^{\infty} \alpha_k \psi_k(x_i) \psi_k(x_j) \tag{3.3}$$

with positive coefficients $\alpha_k > 0$ (i.e. $K(x_i, x_j)$ describes an inner product in some feature space), it is neccessary and sufficient that the condition

$$\int \int K(x_i, x_j) g(x_i) g(x_j) dx_i dx_j > 0 \tag{3.4}$$

be valid for all $g \neq 0$ for which

$$\int g^2(x_i) dx_i < \infty \tag{3.5}$$

A simple example of this is a mapping from a two dimensional space into a six dimensional space:

[1] l is the number of training examples.

Say there exists a transformation $k : x \to z$ such that the inner product in the six dimensional space can be described by $((x_i \cdot x_j) + 1)^2$.

This can be simply proven:

$$
\begin{aligned}
((x_i \cdot x_j) - 1)^2 = \quad & (x_{i_1} x_{j_1} + x_{i_2} x_{j_2})^2 + 2(x_{i_1} x_{j_1} + x_{i_2} x_{j_2}) + 1 & (3.6) \\
\Leftrightarrow \qquad = \quad & (x_{i_1} x_{j_1})^2 + 2(x_{i_1} x_{j_1})(x_{i_2} x_{j_2}) + (x_{i_2} x_{j_2})^2 + & (3.7) \\
& 2(x_{i_1} x_{j_1}) + 2(x_{i_2} x_{j_2}) + 1 & (3.8)
\end{aligned}
$$

$$
\Rightarrow k \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1 x_2 \\ 1 \end{pmatrix}
$$

3.1 Kernel Functions

Depending on the kernel function used different type of SV machine are constructed:

- A polynomial machine is constructed using:

$$K(x_i, x_j) = (x_i \cdot x_j)^d, d = 1, \ldots \tag{3.9}$$

- An alternative polynomial machine, which avoids some problems encountered with the above[2], is constructed using:

$$K(x_i, x_j) = ((x_i \cdot x_j) + 1)^d, d = 1, \ldots \tag{3.10}$$

- A radial basis function machine with convolution function:

$$K(x_i, x_j) = \exp\{-\frac{(x_i - x_j)^2}{\sigma^2}\} \tag{3.11}$$

- A two layer neural network machine with convolution function:

$$K(x_i, x_j) = \tanh(b(x_i \cdot x_j) - c) \tag{3.12}$$

[2]The determinant of the hessian can become zero, which make solving the Lagrangian impossible.

- ⋄ This is an interesting experimental kernel function, which might work

$$K(x_1, x_2) = \sum_{k=1}^{l} (x_1 - t_k)(x_2 - t_k) \tag{3.13}$$

  where t_ks are from the samples. This however is not a function determined a priori and thus is not strictly a proper dot product.

Finding a suitable kernel function is the most ill-defined area of using SV machines.

3.2 Non-Separable Cases

Again non-separable case can be treated very much like the separable case, except that there are further constraints on the α_i:

$$0 \leq \alpha_i \leq C \tag{3.14}$$

For erroneously classified vectors the following holds true:

$$\alpha_i \approx C \tag{3.15}$$

For more information on this see [Vap95].

Chapter 4

Regression Estimation

Huber defined the least modulus method for regression estimation (1964). This is the so called robust regression function. It is the best function if nothing specific is known about the model of noise. This provides the loss-function:

$$L(y, f(x, \alpha)) = |y - f(x, \alpha)| \tag{4.1}$$

We will consider a slightly more general type of loss function than 4.1, the so-called linear loss-function with ϵ-insensitive zone:

$$|y - f(x, \alpha)|_\epsilon = \{ \begin{matrix} \epsilon, & \text{if } |y - f(x, \alpha)| \leq \epsilon \\ |y - f(x, \alpha)|, & \text{otherwise} \end{matrix} \tag{4.2}$$

Huber's loss-function 4.1 is a particular case of the loss-function 4.2 with $\epsilon = 0$.

Again it is possible to use the SRM principle, where

$$(w \cdot w) \leq c_n \tag{4.3}$$

Given the training data $(x_1, y_1), \dots, (x_l, y_l)$ the problem is to minimise the empirical risk

$$R_{\text{emp}}(w, b) = \frac{1}{l} \sum_{i=1}^{l} |y_i - (w \cdot x_i) - b|_\epsilon \tag{4.4}$$

under the constraint 4.3 which is equivalent to finding the pair w, b that minimises the quantity defined by the slack variables $\xi_i, \xi_i^*, i = 1, \dots, l$

$$F(\xi, \xi^*) = (\sum_{i=1}^{l} \xi_i^* + \sum_{i=1}^{l} \xi_i) \tag{4.5}$$

under the constraints

$$\begin{array}{rcll} (w \cdot x_i) + b & \geq & y - \epsilon - \xi_i^* & i = 1, \dots, l \\ (w \cdot x_i) + b & \leq & y + \epsilon + \xi_i & i = 1, \dots, l \\ \xi_i^* & \geq & 0, & i = 1, \dots, l \\ \xi_i & \geq & 0, & i = 1, \dots, l \end{array} \tag{4.6}$$

and constraint 4.3.

As in the pattern recognition case the solution can be found by finding the saddle point of the Lagrange functional

$$\begin{aligned} L(w, \xi^*, \xi, \alpha^*, \alpha, C^*, \gamma, \gamma^*) = & \sum_{i=1}^{l} (\xi_i^* + \xi_i) - \\ & \sum_{i=1}^{l} \alpha_i [y_i - (w \cdot x_i) - b + \epsilon + \xi_i] - \\ & \sum_{i=1}^{l} \alpha_i^* [(w \cdot x_i) + -y_i b + \epsilon + \xi_i^*] - \\ & \frac{C^*}{2} (c_n - (w \cdot w)) - \\ & \sum_{i=1}^{l} (\gamma_i^* \xi_i^* + \gamma_i \xi_i) \end{aligned} \tag{4.7}$$

This has to be minimised with respect to w, b, ξ_i^* and ξ_i and maximised with respect to the Lagrange multipliers $C^* \geq 0$, $\alpha_i^* \geq 0$, $\alpha_i \geq 0$,$\gamma_i^* \geq 0$ and $\gamma_i \geq 0$ for $i = 1, \dots, l$.

As this leads to a complicated optimisation procedure, a simpler approach can be used, if instead of minimising the functional 4.5 under the appropriate constraints, one minimises

$$\Phi(x, \xi^*, \xi) = \frac{1}{2}(w \cdot w) + C(\sum_{i=1}^{l} \xi_i^* + \sum_{i=1}^{l} \xi_i) \tag{4.8}$$

for a given value of C and subject to the constraints 4.6. To find the desired vector

$$w = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) x_i \tag{4.9}$$

one has to find the coefficients α_i^* and α_i, such that the quadratic form

$$W(\alpha^*, \alpha) = -\epsilon \sum_{i=1}^{l} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{l} y_i (\alpha_i^* - \alpha_i) - \frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(x_i \cdot x_j) \tag{4.10}$$

is maximised subject to the constraints

$$\begin{aligned} \sum_{i=1}^{l} \alpha_i^* &= \sum_{i=1}^{l} \alpha_i \\ 0 \leq \alpha_i^* &\leq C, i = 1, \ldots, l \\ 0 \leq \alpha_i &\leq C, i = 1, \ldots, l \end{aligned} \tag{4.11}$$

The simplified approach coincides with the full approach when $C = C^*$.

The solution of this problem has some nice properties:

- Only a small number of α_i^*s and α_is will be non-zero.
- $\alpha_i^* \neq 0 \Rightarrow \alpha_i = 0$
- $\alpha_i \neq 0 \Rightarrow \alpha_i^* = 0$

4.1 Using a convolved inner product

As with the pattern recognition case we can use a convolved inner product rather than the standard inner product with the same type of kernel functions as well as splines which are very good at handling low dimensional cases.

4.2 Controlling the generalisation ability

By controlling the two parameters c_n and ϵ (or C and ϵ in the quadratic optimisation approach) it is possible to control the generalisation ability of the SV machine.

4.3 Calculating the value of an unseen example

When all α_i^*s and α_is have been found the function is approximated by

$$f(x, \alpha^*, \alpha) = (\sum_{i=1}^{l} (\alpha_i^* - \alpha_i) K(x_i, x)) + b \tag{4.12}$$

where b is a unique constant which minimises the error over the training set:

$$\min_b \sum_{i=1}^{l} \Big(y_i - \Big(\sum_{i=1}^{l} (\alpha_i^* - \alpha_i) K(x_i, x) \Big) - b \Big)^2 \tag{4.13}$$

or

$$\frac{1}{2}(\min(-y_i + f(x_i, \alpha^*, \alpha) - b) + \max(-y_j + f(x_j, \alpha^*, \alpha) - b) \tag{4.14}$$

or

$$b_0 = \frac{1}{2} \min_{x_j | \alpha_j > 0} (-y_j + \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) K(x_j, x_i)) + \min_{x_k | \alpha_k^* > 0} (-y_k + \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) K(x_k, x_i)) \tag{4.15}$$

where (y_i, x_i) and (y_j, x_j) are support vectors from the training set.

Appendix A

Useful formulae

A.1 SV Machines for Pattern Recognition

A.1.1 Equation 2.17

For many quadratic optimisation packages the first and second derivative of the objective function 2.17 are needed:

$$\frac{\partial W}{\partial \alpha_i} = 1 - \sum_{j=1}^{l} \alpha_j y_i y_j K(x_i, x_j) \tag{A.1}$$

$$\frac{\partial^2 W}{\partial \alpha_i \partial \alpha_j} = -y_i y_j K(x_i, x_j) \tag{A.2}$$

A.1.2 Equation 2.24

Equation 2.24 can also be written as:

$$b_0 = \sum_{\text{support vectors}} a_i y_i \left(\frac{K(x_i, x^*(1)) + K(x_i, x^*(-1))}{2}\right) \tag{A.3}$$

A.2 SV Machines for Regression Estimation

A.2.1 Equation 4.10

For many quadratic optimisation packages the first and second derivative of the objective function 4.10 are needed:

The first derivative with respect to the α_i's:

$$\frac{\partial W}{\partial \alpha_i} = -\epsilon - y_i + \sum_{j=1}^{l} (\alpha_j^* - \alpha_j) K(x_i, x_j) \tag{A.4}$$

The first derivative with respect to the α_i^*'s:

$$\frac{\partial W}{\partial \alpha_i} = -\epsilon + y_i - \sum_{j=1}^{l} (\alpha_j^* - \alpha_j) K(x_i, x_j) \tag{A.5}$$

The second derivative with respect to α_i and α_j

$$\frac{\partial^2 W}{\partial \alpha_i \partial \alpha_j} = -K(x_i, x_j) \tag{A.6}$$

The second derivative with respect to α_i^* and α_j^*

$$\frac{\partial^2 W}{\partial \alpha_i^* \partial \alpha_j^*} = -K(x_i, x_j) \tag{A.7}$$

The second derivative with respect to α_i and α_j^*

$$\frac{\partial^2 W}{\partial \alpha_i \partial \alpha_j^*} = K(x_i, x_j) \tag{A.8}$$

The second derivative with respect to α_i^* and α_j

$$\frac{\partial^2 W}{\partial \alpha_i^* \partial \alpha_j} = K(x_i, x_j) \tag{A.9}$$

Bibliography

[CH53] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume Volume I. Springer Verlag, Berlin, first English edition, 1953.

[LS92] George F. Luger and William A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Benjamin / Cummings, Redwood City, California, second edition, 1992.

[Sch91] Detlef Lind / Harald Scheid. *Abiturwissen Stochastik*. Ernst Klett Verlag für Wissen und Bildung, Stuttgart, fourth edition, 1991.

[SW96] M. O. Stitson and J. A. E. Weston. Implementational issues of support vector machines. Technical Report CSD-TR-96-18, Royal Holloway, University of London, December 1996.

[Vap82] Vladimir N. Vapnik. *Estimation of Dependences Baed on Empirical Data*. Springer, New York, 1982.

[Vap95] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[VC79] Vladimir N. Vapnik and A. J. Cervonenkis. *Theorie der Zeichenerkennung*. Akademie-Verlag, Berlin, 1979. Translated from the Russian original (1974).

[Wal86] Ing. Rudolf Walther, editor. *Wörterbuch der Technik Englisch-Deutsch*. Cornelsen Verlag Schwann-Giradet, Düsseldorf, fifth edition, 1986.

[Wal88] Ing. Rudolf Walther, editor. *Wörterbuch der Technik Deutsch-Englisch*. Cornelsen Verlag Schwann-Giradet, Düsseldorf, fourth edition, 1988.

[WSG+96] J. A. E. Weston, M. O. Stitson, A. Gammerman, V. Vovk, and V. Vapnik. Implementational issues of support vector machines.

Technical Report CSD-TR-96-18, Royal Holloway, University of London, December 1996.