3 THE SUPPORT VECTOR METHOD OF FUNCTION ESTIMATION

Vladimir Vapnik*

AT&T Labs-Research, USA

vlad@research.att.com

Abstract: This chapter describes the Support Vector technique for function estimation problems such as pattern recognition, regression estimation, and solving linear operator equations. It shows that for the Support Vector method both the quality of solution and the complexity of the solution does not depend directly on the dimensionality of an input space. Therefore, on the basis of this technique one can obtain a good estimate using a given number of high-dimensional data.

3.1 INTRODUCTION

The Support Vector (SV) method is a general approach to function estimation problems, i.e. to problems of finding the function $y = f(x)$ given by its measurements y_i with noise at some (usually random) vectors x

$$(y_1, x_1), ..., (y_\ell, x_\ell). \tag{3.1}$$

The SV method can be applied for pattern recognition (to estimate indicator functions), for regression (to estimate real-valued functions), and for solving linear operator equations.

The idea behind this method is the following: one maps n-dimensional vectors x of the input space X into high-dimensional (even infinite dimensional) vectors z of the feature space Z. Using the samples

$$(y_1, z_1), ..., (y_\ell, z_\ell) \tag{3.2}$$

*This chapter has been written using materials reprinted with permission from Vapnik: STATISTICAL LEARNING THEORY (in press). Copyright © 1998, John Wiley & Sons, Inc.

J. A. K. Suykens et al. (eds.), *Nonlinear Modeling*

corresponding to (3.1) one constructs a linear function

$$y = (a * z) + b \tag{3.3}$$

of a special form. Here $(a * z)$ denotes the inner product between the two vectors a and z.

Two problems arise with this approach:

1. Which properties should the linear function (3.3) satisfy in order to have a good approximation of the desired function.

2. How to construct such a hyperplane in a high-dimensional space.

Two theoretical results made construction of SV machines possible:

1. The generalization ability of the learning machine depends on the capacity of a set of functions (in particular on the VC dimension of a set of functions) rather than on the dimensionality of a space. Therefore, according to the so called Structural Risk Minimization inductive principle the function, that describes the data well and belongs to a set of functions with low capacity, will generalize well, regardless of the dimensionality of the space [4].

2. To construct the hyperplane one only needs to evaluate the inner product between two vectors of the training data. Since in Hilbert space the general form of the inner product has a kernel representation, the evaluation of the inner product also does not depend on the dimensionality of the space.

The SV method was discovered in 1964 for the construction of separating hyperplanes in pattern recognition problems [5]. Then in 1992 – 1995 it was generalized for constructing non-linear separating functions (but linear in feature space) [2],[3]. In 1995 it was generalized for estimation of real-valued functions [3]. Lastly, in 1996 it was applied for solving linear operator equations [7].

In this Chapter we present a general description of the SV technique for pattern recognition, regression estimation, and signal processing. A detailed analysis of this method one can find in [4].

3.2 THE OPTIMAL HYPERPLANE

3.2.1 The Optimal Hyperplane for Separable Data

We say that two finite subsets of vectors x from the training set

$$(x_1, y_1), ..., (x_\ell, y_\ell), \quad x \in R^n, \quad y \in \{-1, 1\},$$

one subset I for which $y = 1$ and another subset II for which $y = -1$, are separable by the hyperplane

$$(x * \phi) = c$$

if there exist both a unit vector ϕ, ($|\phi| = 1$) and a constant c such that the inequalities

$$\begin{aligned} (x_i * \phi) > c, \quad & \text{if } x_i \in I, \\ (x_j * \phi) < c, \quad & \text{if } x_j \in II \end{aligned} \tag{3.4}$$

hold true. Here $(a * b)$ denotes the inner product between vectors a and b.

Let us determine for any unit vector ϕ two values

$$c_1(\phi) = \min_{x_i \in I}(x_i * \phi),$$

$$c_2(\phi) = \max_{x_j \in II}(x_j * \phi).$$

Consider the unit vector ϕ_0 which maximizes the function

$$\rho(\phi) = \frac{c_1(\phi) - c_2(\phi)}{2}, \quad |\phi| = 1 \tag{3.5}$$

under the condition that inequalities (3.4) are satisfied. The vector ϕ_0 and the constant

$$c_0 = \frac{c_1(\phi_0) + c_2(\phi_0)}{2}$$

determine the hyperplane that separates the vectors $x_1, ..., x_a$ of the subset I from the vectors $x_1, ..., x_b$ of the subset II, ($a + b = \ell$) and has the maximal margin (3.5). We call this hyperplane the *Maximal Margin hyperplane* or the *Optimal hyperplane*

Theorem 1 *The Optimal hyperplane is unique.*

Indeed, we need to show that the maximum point ϕ_0 of the continuous function $\rho(\phi)$, defined in the area $|\phi| \leq 1$, exists and is achieved on the boundary $|\phi| = 1$. Existence of the maximum follows from the continuity of $\rho(\phi)$ in the bounded region $|\phi| \leq 1$.

Suppose that the maximum is achieved at some interior point ϕ^*. Then the vector

$$\phi^* = \frac{\phi_0}{|\phi_0|}$$

would define a larger margin

$$\rho(\phi^*) = \frac{\rho(\phi_0)}{|\phi_0|}.$$

The maximum of the function $\rho(\phi)$ cannot be achieved on two (boundary) points. Otherwise, since the function $\rho(\phi)$ is convex, it is achieved on the line that connects these two points i.e. at an inner point, which is impossible by preceding arguments.

Our goal is to find effective methods for constructing the Optimal hyperplane. To do so we consider an equivalent statement of the problem: find a pair consisting of a vector ψ_0 and a constant (threshold) b_0 such that they satisfy the constraints

$$(x_i * \psi_0) + b_0 \geq 1, \quad \text{if } y_i = 1, \tag{3.6}$$

$$(x_j * \psi_0) + b_0 \leq -1, \quad \text{if } y_j = -1,$$

and the vector ψ_0 has the smallest norm

$$|\psi|^2 = (\psi * \psi). \tag{3.7}$$

Theorem 2. *Vector ψ_0 that minimizes (3.7) under constraints (3.6) is related to the vector that forms the optimal hyperplane by the equality*

$$S\phi_0 = \frac{\psi_0}{|\psi_0|}. \tag{3.8}$$

The margin ρ_0 between the Optimal hyperplane and separated vectors is equal to

$$\rho(\phi_0) = \sup_{\phi_0} \frac{1}{2}\left(\min_{i \in I}(x_i * \phi_0) - \max_{j \in II}(x_j * \phi_0)\right) = \frac{1}{|\psi_0|}. \tag{3.9}$$

Indeed, the vector ψ_0 that provides the minimum of the quadratic function (3.7) under the linear constraints (3.6) is unique. Let us define the unit vector

$$\phi_0 = \frac{\psi_0}{|\psi_0|}.$$

Since constraints (3.6) are valid, we find

$$\rho\left(\frac{\psi_0}{|\psi_0|}\right) = \frac{1}{2}\left(c_1\left(\frac{\psi_0}{|\psi_0|}\right) - c_2\left(\frac{\psi_0}{|\psi_0|}\right)\right) \geq \frac{1}{|\psi_0|}.$$

To prove the theorem it is sufficient to show that the inequality

$$\rho\left(\frac{\psi_0}{|\psi_0|}\right) > \frac{1}{|\psi_0|}$$

is impossible. Suppose it holds true. Then there exists a unit vector ϕ^* such that the inequality

$$\rho(\phi^*) > \frac{1}{|\psi_0|}$$

holds true. Let us construct the new vector

$$\psi^* = \frac{\phi^*}{\rho(\phi^*)},$$

which has norm smaller than $|\psi_0|$. One can check that this vector satisfies the constraints (3.6) with

$$b = -\frac{c_1(\phi) + c_2(\phi)}{2}.$$

This contradicts the assertion that ψ_0 is the smallest vector satisfying the constraints (3.6), which proves the theorem.

Thus the vector ψ_0 with smallest norm satisfying constraints (3.6) defines the Optimal hyperplane. The vector ψ_0 with the smallest norm satisfying constraints (3.6) with $b = 0$ defines the *Optimal hyperplane passing through the origin.*

To simplify our notation let us rewrite the constraint (3.6) in the equivalent form

$$y_i ((x_i * \psi_0) + b) \geq 1, \quad i = 1, ..., \ell. \tag{3.10}$$

Therefore, in order to find the Optimal hyperplane one has to solve the following quadratic optimization problem : minimize the quadratic form (3.7) subject to the linear constraints (3.10).

One can solve this problem in the *primal space* – the space of parameters ψ and b. However, deeper results can be obtained if one solve this quadratic optimization problem in the *dual space* – the space of Lagrange multipliers. In the sequel we consider this type of solution.

It is known (see for example G. Strang, *Introduction to Applied Mathematics*, Wellesley Cambridge Press, 1986) that in order to solve this quadratic optimization problem one has to find the saddle point of the Lagrange function

$$L(\psi, b, \alpha) = \frac{1}{2}(\psi * \psi) - \sum_{i=1}^{\ell} \alpha_i \left(y_i[(x_i * \psi) + b] - 1\right), \tag{3.11}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. To find the saddle point one has to minimize this function over ψ and b and to maximize it over the nonnegative Lagrange multipliers $\alpha_i \geq 0$.

According to the Fermat theorem, the minimum points of this functional have to satisfy the conditions

$$\begin{cases} \frac{\partial L(\psi,b,\alpha)}{\partial \psi} &= \psi - \sum_{i=1}^{\ell} y_i \alpha_i x_i = 0, \\ \frac{\partial L(\psi,b,\alpha)}{\partial b} &= \sum_{i=1}^{\ell} y_i \alpha_i = 0. \end{cases}$$

From these conditions it follows that for the vector ψ that defines the Optimal hyperplane, the equalities

$$\psi = \sum_{i=1}^{\ell} y_i \alpha_i x_i, \tag{3.12}$$

$$\sum_{i=1}^{\ell} y_i \alpha_i = 0 \tag{3.13}$$

hold true. Substituting (3.12) into (3.11) and taking into account (3.13) one obtains

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (x_i * x_j). \tag{3.14}$$

Note that we have changed the notation from $L(\psi, b, \alpha)$ to $W(\alpha)$ to reflect the last transformation. Now to construct the Optimal hyperplane one has to find the coefficients α_i^0 that maximize the function (3.14) in the nonnegative quadrant

$$\alpha_i \geq 0, \quad i = 1..,\ell, \tag{3.15}$$

under the constraint (3.13). Using these coefficients α_i^0 $(i = 1, ..., \ell)$ in the equation (3.12), one obtains the solution

$$\psi_0 = \sum_{i=1}^{\ell} y_i \alpha_i^0 x_i.$$

Note that the optimal solution ψ_0 and b must satisfy the Kuhn-Tucker conditions (See G. Strang (1986))

$$\alpha_i^0 \left(y_i((x_i * \psi_0) + b_0) - 1 \right) = 0, \quad i = 1, ..., \ell. \tag{3.16}$$

From conditions (3.16) one concludes that nonzero values α_i^0 correspond only to the vectors x_i that satisfy the equality

$$y_i((x_i * \psi_0) + b_0) = 1. \tag{3.17}$$

Geometrically, these vectors are the closest to the optimal hyperplane. We will call them *support vectors*. The support vectors play a crucial role in constructing a new type of learning algorithm since the vector ψ_0 that defines the Optimal hyperplane is expanded with non-zero weights on support vectors:

$$\psi_0 = \sum_{s} y_i \alpha_i^0 x_i.$$

Therefore, the Optimal hyperplane has the form

$$f(x, \alpha_0) = \sum_s y_i \alpha_i^0 (x_s * x) + b_0 \tag{3.18}$$

where $(x_s * x)$ is the inner product of the two vectors.

Note that both the separating hyperplane (3.18) and the objective function of our optimization problem (3.14) do not explicitly depend on the dimensionality of the vector x but only depend on the inner product of two vectors. This fact will allow us to construct separating hyperplanes in very high dimensional spaces, even in infinite dimensional Hilbert spaces.

At the end of this section we formulate some properties of the Optimal hyperplane, that we will use in the sequel:

1. The Optimal hyperplane is unique, i.e. the pair of vector ψ_0 and threshold b that define the Optimal hyperplane is unique. However, the expansion of the vector ψ_0 on the support vectors is not unique.

2. Let the vector ψ_0 define the Optimal hyperplane. Then the maximum of the functional $W(\alpha)$ is equal to

$$W(\alpha_0) = \frac{1}{2}(\psi_0 * \psi_0) = \frac{1}{2}\sum_i \alpha_i^0.$$

In order to show this, it is sufficient to transform functional (3.14), taking into account that for optimal pairs ψ_0 and b the equality (3.13) and equalities (3.17) hold true. This implies

$$(\psi_0 * \psi_0) = \sum_i \alpha_i^0.$$

3. The norm of the vector ψ_0 defines a margin for the Optimal separating hyperplane

$$\rho(\psi_0) = \frac{1}{|\psi_0|}.$$

4. From the properties (3.2) and (3.3) it follows that

$$W(\alpha) < W(\alpha_0) = \frac{1}{2}\left(\frac{1}{\rho(\psi_0)}\right)^2, \quad \alpha \neq q\alpha_0.$$

This expression can be chosen as a criterion of linear non-separability of two data sets.

Definition. We call two data sets *linearly δ-nonseparable* if the margin between the hyperplane and the closest vector is less than δ.

Therefore, if during the maximization procedure the value $W(\alpha)$ exceeds the value $1/2\delta^2$, one can assert that the two sets of separating data are δ-nonseparable.

Thus, in order to construct the optimal hyperplane one has either to find the maximum of the nonnegative quadratic form $W(\alpha)$ in the nonnegative quadrant under the constraint (3.13) or to establish that the maximum exceeds the value

$$W_{\max} = \frac{1}{2\delta^2}.$$

In the latter case separation with the margin δ is impossible.

5. In order to maximize the functional $W(\alpha)$ under the constraints (3.13) and (3.15), one needs to specify the support vectors and to determine the corresponding coefficients. This can be done sequentially by using every time a small amount of training data. One can start the optimization process using only n examples (maximizing $W(\alpha)$ under the condition that only n parameters differ from zero). As the conditional maximum $W(\alpha)$ is achieved, one can keep the parameters that differ from zero and add new parameters (corresponding to the vectors that were not separated correctly by the first iteration of constructing the Optimal hyperplane). One continues this process until either:

(1) all the vectors of the training set are separated, or

(2) at some step: $W(\alpha) > W_{\max}$ (the separation is impossible).

The described methods work in some sense like a sieve: at any step it maximizes the functional $W(\alpha)$ using only the elements of the training set which are the candidates for support vectors.

3.2.2 The Optimal Hyperplane for Nonseparable Data

In this section we generalize the concept of the Optimal hyperplane for the nonseparable case .

Let the set of training data

$$(x_1, y_1), ..., (x_\ell, y_\ell), \quad x \in X,\ y \in \{-1, 1\}$$

be such that they cannot be separated by a hyperplane without making an error. According to our definition of non-separability this means that there is no pair ψ, b such that

$$(\psi * \psi) \leq \frac{1}{\delta^2}$$

and the inequalities

$$y_i((x_i * \psi) + b) \geq 1 \quad i = 1, 2, ...\ell, \tag{3.19}$$

hold true.

Our goal is to construct the hyperplane that makes the smallest number of errors. To get a formal setting of this problem we introduce the non-negative variables

$$\xi_1, ..., \xi_\ell.$$

In terms of these variables the problem of finding the hyperplane that provides the minimal number of training errors has the following formal expression: minimize the functional

$$\Phi(\xi) = C\sum_{i=1}^{\ell}\theta(\xi_i) + (\psi * \psi)$$

subject to the constraints

$$y_i((x_i * \psi) + b) \geq 1 - \xi_i, \quad i = 1, 2, ...\ell, \quad \xi_i \geq 0, \tag{3.20}$$

where $\theta(\xi) = 0$ if $\xi = 0$ and $\theta(\xi) = 1$ if $\xi > 0$. It is known that for the nonseparable case this optimization problem is NP-complete. Therefore we consider the following approximation to this problem: we would like to minimize the functional

$$\Phi(\xi) = C\sum_{i=1}^{\ell}\xi_i^{\sigma} + (\psi * \psi)$$

under the constraints (3.20), where $\sigma \geq 0$ has a small value. We will, however, choose $\sigma = 1$, the smallest σ that leads to a simple optimization problem[1].

Using the same technique with the Lagrangian, one obtains a method for solving this optimization problem which is almost equivalent to the method for solving the optimization problem in the separable case. In order to find the vector ψ of the Generalized Optimal hyperplane

$$\psi = \sum_{i=1}^{\ell}\alpha_i y_i x_i,$$

one has to maximize the same quadratic form as in the separable case

$$W(\alpha) = \sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j (x_i * x_j) \tag{3.21}$$

under slightly different constraints:

$$\begin{cases} 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, \ell, \\ \sum_{i=1}^{\ell}\alpha_i y_i = 0. \end{cases} \tag{3.22}$$

[1]The choice $\sigma = 2$ also leads to a simple optimization problem. However, for the pattern recognition problem this choice does not look attractive. It will be more attractive when we will generalize results obtained for the pattern recognition problem to the estimation of real-valued functions.

As in the separable case, only some of the coefficients $\alpha_i^0 (i = 1, ..., \ell)$ differ from zero. Together with their corresponding support vectors, they determine the Generalized Optimal separating hyperplane:

$$\sum_{i=1}^{a} \alpha_i^0 y_i (x_i * x) + b_0 = 0. \tag{3.23}$$

3.2.3 Statistical Properties of the Optimal Hyperplane

Let $X^* = (x_1, ..., x_\ell)$ be a set of ℓ vectors in R^n. For any hyperplane

$$(x * \psi^*) + b^* = 0$$

in the space R^n, consider the corresponding *canonical hyperplane* defined by the set X^* as follows:

$$\inf_{x \in X^*} |(x * \psi) + b| = 1,$$

where $\psi = c^* \psi^*$ and $b = c^* b^*$. Note that the set of canonical hyperplanes coincides with the set of separating hyperplanes. It only specifies the normalization with respect to a given set of data X^*.

Let us first establish the following important fact.

Theorem 3. *A subset of canonical hyperplanes defined on* $X^* \subset R^n$

$$\max_{x \in X^*} \min_a |x - a| \leq D, \qquad x \in X^*$$

satisfying the constraint

$$|\psi| \leq A$$

has VC dimension h bounded as follows

$$h \leq \min \left([D^2 A^2], n \right) + 1,$$

where $[a]$ *denotes the integer part of* a.

Note that the norm of the vector coefficients of the canonical hyperplane ψ defines the margin

$$\rho \left(\frac{\psi}{|\psi|} \right) = \frac{c_1 \left(\frac{\psi}{|\psi|} \right) - c_2 \left(\frac{\psi}{|\psi|} \right)}{2} = \frac{1}{|\psi|} = \frac{1}{A}.$$

To formulate the next theorems let us introduce one more concept. In Section 3.2 we mentioned that the minimal norm vector ψ, satisfying the conditions (3.6), is unique, though it can have different expansions on the support vectors.

Definition. We call the support vectors x_i that appear in all possible expansions of the vector ψ_0 the *essential support vectors*. In other words the essential support vectors is the intersection of all possible sets of support vectors. Let

$$(x_1, y_1), ..., (x_\ell, y_\ell)$$

be the training set. We denote the number of essential support vectors of this training set by

$$\mathcal{K}_\ell = \mathcal{K}((x_1, y_\ell)..., (x_\ell, y_\ell)).$$

We denote the maximal norm of essential support vectors of this training set by

$$\mathcal{D}_\ell = \mathcal{D}((x_1, y_1), ..., (x_\ell, y_\ell)) = \max_i |x_i|.$$

Let n be the dimensionality of vectors x.

The following four theorems describe the main statistical properties of the Optimal hyperplanes.

Theorem 4. *The inequality*

$$\mathcal{K}_\ell \leq n \tag{3.24}$$

holds true.

Theorem 5. *Let*

$$ER(\alpha_\ell) = ER(y, x, \alpha(y_1, x_1, ..., y_\ell, x_\ell))$$

be the expectation of the probability of the error for Optimal hyperplanes constructed on the basis of training samples of size ℓ (the expectation is taken over both the training and test data). Then the following inequality

$$ER(\alpha_\ell) \leq \frac{E\mathcal{K}_{\ell+1}}{\ell + 1} \tag{3.25}$$

holds true.

Theorem 6. *For Optimal hyperplanes passing through the origin the following inequality*

$$ER(\alpha_\ell) \leq \frac{E\left(\frac{\mathcal{D}_{\ell+1}}{\rho_{\ell+1}}\right)^2}{\ell + 1} \tag{3.26}$$

holds true, where $D_{\ell+1}$ and $\rho_{\ell+1}$ are (random) values that for a given training set of size $\ell + 1$ define the maximal norm of the essential support vectors x and the margin.

One can also prove that the following stronger bound is valid.

Theorem 7. *For the Optimal hyperplane passing through the origin the inequality*

$$ER(\alpha_\ell) \leq \frac{E \min\left(K_{\ell+1}, \left(\frac{\mathcal{D}_{\ell+1}}{\rho_{\ell+1}}\right)^2\right)}{\ell+1}$$

is valid.

The proofs of these theorems one can find in [6] and [4].

3.3 THE SUPPORT VECTOR MACHINE FOR PATTERN RECOGNITION

3.3.1 The Idea of Support Vector Machines

Now let us define the Support Vector machine. The Support Vector (SV) machine implements the following idea: it maps the input vectors x into the high-dimensional *feature space* Z through some nonlinear mapping, chosen *a priori*. In this space, an Optimal separating hyperplane is constructed.

Example. To construct a decision surface corresponding to a polynomial of degree two, one can create a feature space Z that has $N = \frac{n(n+3)}{2}$ coordinates of the form

$$z^1 = x^1, \ldots, z^n = x^n \ , \qquad (n \text{ coordinates}) \ ,$$

$$z^{n+1} = (x^1)^2, \ldots, z^{2n} = (x^n)^2 \ , \qquad (n \text{ coordinates}) \ ,$$

$$z^{2n+1} = x^1 x^2, \ldots, z^N = x^n x^{n-1} \ , \qquad (\tfrac{n(n-1)}{2} \text{ coordinates}) \ ,$$

where $x = (x^1 \ldots, x^n)$. The separating hyperplane constructed in this space is a second degree polynomial in the input space.

Two problems arise in the above approach: a conceptual and a technical one.

(i) *How to find a separating hyperplane that generalizes well?*
(The conceptual problem.)

The dimensionality of the feature space is huge, and a hyperplane that separates the training data does not necessarily generalize well.

(ii) *How to treat such high-dimensional spaces computationally?*
(The technical problem.)

To construct a polynomial of degree 4 or 5 in a 200-dimensional space it is necessary to construct hyperplanes in a billion-dimensional feature space. How can one overcome this "curse of dimensionality"?

3.3.2 Generalization in High-Dimensional Space

The conceptual part of this problem can be solved by constructing the Optimal hyperplane.

According to the theorems described above, if one can construct a separating hyperplane with a small expectation of $(\mathcal{D}/\rho)^2$ or with expectation of a small number of support vectors, the generalization ability of the constructed hyperplane is high, even if the feature space is high dimensional.

3.3.3 Hilbert-Schmidt theory and Mercer's theorem

However, even if the Optimal hyperplane generalizes well and can theoretically be found, the technical problem of how to treat the high-dimensional feature space remains.

Note, however, that for constructing the Optimal separating hyperplane in the feature space Z, one does not need to consider the feature space in *explicit form.* One only has to calculate the inner products between support vectors and the vectors of the feature space (see, for example, (3.21) and (3.23)).

Consider a general property of the inner product in a Hilbert space. Suppose one maps the vector $x \in R^n$ into a Hilbert space with coordinates

$$z_1(x), ..., z_n(x), ...$$

According to the Hilbert-Schmidt theory the inner product in a Hilbert space has an equivalent representation

$$(z_1 * z_2) = \sum_{r=1}^{\infty} a_r z_r(x_1) z(x_2) \Longleftrightarrow K(x_1, x_2), \qquad a_r \geq 0$$

where $K(x, x_i)$ is a symmetric function satisfying the following conditions.

Theorem 8. (Mercer) *To guarantee that a continuous symmetric function $K(u,v)$ from $L_2(C)$ has an expansion*[2]

$$K(u,v) = \sum_{k=1}^{\infty} a_k z_k(u) z_k(v) \tag{3.27}$$

with positive coefficients $a_k > 0$ (i.e., $K(u,v)$ describes an inner product in some feature space), it is necessary and sufficient that the condition

$$\int\int K(u,v) g(u) g(v)\, du\, dv \geq 0$$

be valid for all $g \in L_2(C)$ (C being a compact subspace of R^n).

[2]This means that the right-hand side of (3.27) converges to function $K(u,v)$ uniformly.

The remarkable property of the structure of the inner product in Hilbert space that leads to construction of the SV machine is that for any kernel function $K(u,v)$ satisfying Mercer's condition, there exists a feature space where this function generates the inner product.

3.3.4 *Constructing SV Machines*

Generating the inner product in a (high-dimensional) feature space allows to construct separating functions that are nonlinear in input space

$$f(x,\alpha) = \text{sign}\left(\sum_{\text{support vectors}} y_i\alpha_i^0 K(x_i,x) - b\right) \tag{3.28}$$

and are equivalent to linear decision functions in the feature space $z_1(x),\ldots,z_N(x),\ldots$

$$f(x,\alpha) = \text{sign}\left(\sum_{\text{support vectors}} y_i\alpha_i^0 \sum_{r=1}^{\infty} z_r(x_i)z_r(x) - b\right).$$

$K(x_i,x)$ is the kernel that generates the inner product for this feature space. Therefore, in order to construct function (3.28) one can use methods developed for constructing linear separating hyperplanes. Instead of the inner products defined as $(x * x_i)$, one uses the inner product defined by kernel $K(x,x_i)$:

1. To find the coefficients α_i in the separable case it is sufficient to find the maximum of the functional

$$W(\alpha) = \sum_{i=1}^{\ell}\alpha_i - \frac{1}{2}\sum_{i,j}^{\ell}\alpha_i\alpha_j y_i y_j K(x_i,x_j) \tag{3.29}$$

subject to the constraints

$$\begin{cases} \sum_{i=1}^{\ell}\alpha_i y_i = 0, \\ \alpha_i \geq 0, \quad i = 1,2,\ldots,\ell. \end{cases} \tag{3.30}$$

2. To find the optimal soft margin solution for the non-separable case it is sufficient to maximize (3.29) under constraints

$$\begin{cases} \sum_{i=1}^{\ell}\alpha_i y_i = 0, \\ 0 \leq \alpha_i \leq C. \end{cases}$$

The learning machines that construct decision functions of this type are called *Support Vector (SV) Machines*[3].

[3]With this name we stress the idea of expanding the solution on support vectors. In SV machines the complexity of the construction depends on the number of support vectors rather than on the dimensionality of the feature space.

3.3.5 *Selection of a SV Machine Using Bounds*

The bounds on the generalization ability described in Section 3.2 allows us to choose a specific model of SV machine from a given set of models.

Indeed, let

$$z(x) = (z_1(x), \ldots, z_N(x))$$

be the feature space and $w = (w_1, \ldots, w_N)$ be a vector of weights determining a hyperplane in this space. Consider a set of hyperplanes containing the functions that satisfy the conditions

$$[D^2|w|^2] \leq k,$$

where D is the radius of the smallest sphere that contains the vectors $z(x)$, $|w|$ is the norm of the weights (we use canonical hyperplanes in the feature space with respect to the vectors $z_i = z(x_i)$ where x_i are the elements of the training data). According to Theorem 3 (now applied in the feature space), k gives an estimate of the VC dimension of the set of functions S_k defined on the training data.

The SV machine separates the training data

$$y_i\,[(z(x_i) * w) + b] \geq 1, \quad y_i = \{+1, -1\}, \quad i = 1, 2, \ldots, \ell,$$

without errors and has minimal norm $|w|$. In other words, the SV machine separates the training data by using functions which have the smallest estimate for the VC dimension.

Also, according to Theorem 6, the expectation of the error's probability of of the support vector machine is defined by the expectation of the value $\mathcal{D}_\ell{}^2/\rho_\ell^2$. It means that to minimize the expectation one has to use the algorithm that minimizes the value $D_\ell^2|w_\ell|^2$ for a given training set.

Therefore, one can use the following criteria for choosing the best model:

$$R(D_\ell, w_\ell) = D_\ell^2|w_\ell|^2, \tag{3.31}$$

where both the value of w_ℓ and D_ℓ are calculated from the training data.

Recall that in feature space the equality

$$\frac{1}{\rho_\ell^2} = |w_\ell|^2 = \sum_i^\ell \alpha_i^0 \alpha_j^0 y_i y_j (z(x_i) * z(x_j)) = \sum_i^\ell \alpha_i^0 \alpha_j^0 y_i y_j K(x_i, x_j) \tag{3.32}$$

holds true.

To define the radius of the smallest sphere D_ℓ that includes the training vectors, one has to solve the following simple optimization problem: minimize functional (D_ℓ, D_ℓ) subject to constraints

$$||z(x_i) - a||^2 \leq D_\ell^2, \quad i = 1, .., n$$

where $x_i (i = 1, ..., n)$ are Support vectors and a is the center of the sphere.

Using the same technique with Lagrange multipliers one can find that

$$D_\ell^2 = \sum_{i,j=1}^{n} \beta_i \beta_j (z(x_i) * z(x_j)),$$

where the coefficients β_i $(i = 1, ..., n)$ are a solution to the following quadratic optimization problem: maximize the functional

$$W^*(\beta) = \sum_{i=1}^{n} \beta_i (z(x_i) * z(x_i)) - \sum_{i,j=1}^{n} \beta_i \beta_j (z(x_i) * z(x_j)) \tag{3.33}$$

subject to constraints

$$\sum_{i=1}^{n} \beta_i = 1, \quad \beta_i \geq 0. \tag{3.34}$$

Using a kernel representation we can rewrite the radius of the smallest sphere in the form

$$D_\ell^2 = \sum_{i,j=1}^{n} \beta_i \beta_j K(x_i, x_j) \tag{3.35}$$

and functional (3.33) in the form

$$W^*(\beta) = \sum_{i=1}^{n} \beta_i K(x_i, x_i) - \sum_{i,j=1}^{n} \beta_i \beta_j K(x_i, x_j). \tag{3.36}$$

Therefore, to control the generalization ability of the machine (to minimize the expectation of the test error) one has to construct the separating hyperplane that minimizes the functional

$$\Phi(D_\ell, w_\ell) = D_\ell{}^2 |w_\ell|^2. \tag{3.37}$$

where $|w_\ell|^2$ and D_ℓ^2 are defined by (3.32) and (3.35).

Choosing among different models (different kernels), the model that minimizes the estimate (3.37) is selecting the best SV machine.

Thus the model selection is based on the following idea: among different SV machines that separate the training data the one with smallest VC dimension is the best. In this section we used the upper bound (3.37) to estimate the VC dimension.

3.4 EXAMPLES OF SV MACHINES FOR PATTERN RECOGNITION

In this section we use different kernels for generating the inner products $K(x, x_i)$, in order to construct learning machines with different types of nonlinear decision surfaces in the input space. We consider three types of learning machines [1], [3], [4]:

(i) Polynomial SV Machines,

(ii) Radial Basis Function SV Machines,

(iii) Two Layer Neural Network SV Machines.

For simplicity we consider here the case where the training vectors are separated without errors.

3.4.1 *Polynomial Support Vector Machines*

To construct polynomial of degree d decision rules, one can use the following generating kernel

$$K(x, x_i) = [(x * x_i) + 1]^d. \tag{3.38}$$

This symmetric function, rewritten in coordinates space, describes the inner product in the feature space that contains all the products $x_i \cdot x_j \cdot x_k$ up to degree d. Using this generating kernel, one constructs a decision function of the form

$$f(x, \alpha) = \text{sign}\left(\sum_{\text{support vectors}} y_i \alpha_i [(x_i * x) + 1]^d - b\right),$$

which is a factorization of d-dimensional polynomials in the n-dimensional input space. It is easy to see that kernels

$$K(u, v) = [(u * v) + 1]^p$$

satisfy the Mercer conditions.

Despite of the very high dimension of the feature space (polynomials of degree d in n-dimensional input space have $O(n^d)$ free parameters), the estimate of the VC dimension of the subset of polynomials, that solve real life problems on the given training data, can be low (Theorem 3). Also, if the expectation $\mathcal{D}_\ell/\rho_\ell$ has a small value then the expectation of the probability of the error is small (Theorem 6).

Note that both the value D_ℓ and the norm of the weights $|w_\ell|$ in the feature space depend on the degree of the polynomial. This offers the possibility to choose the best degree polynomial for the given data by minimizing the functional (3.37).

3.4.2 *Radial Basis Function SV Machines*

Classical Radial Basis Function (RBF) Machines use the following set of decision rules:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} a_i K_\gamma(|x - x_i|) - b\right), \tag{3.39}$$

where $K_\gamma(|x - x_i|)$ depends on the distance $|x - x_i|$ between two vectors.

The function $K_\gamma(|z|)$ is a non-negative monotonic function for any fixed $\gamma > 0$; it tends to zero as $|z|$ goes to infinity. The most popular function of this type is

$$K_\gamma(|x - x_i|) = \exp\{-\gamma|x - x_i|^2\}. \tag{3.40}$$

To construct the decision rule from the set (3.39) one has to estimate

(i) the value of the parameter γ,

(ii) the number N of the centers x_i,

(iii) the vectors x_i, describing the centers,

(iv) the values of the parameters a_i.

In the classical RBF method the first three steps (determining the parameters γ, N, and vectors (centers) x_i, $i = 1, \ldots, N$) are based on heuristics and only the fourth step (after finding these parameters) is determined by minimizing the empirical risk functional.

One can show that kernel (3.40) satisfies the condition of Mercer's Theorem. Therefore, one can choose the function $K_\gamma(|x - x_i|)$ as a kernel generating the inner product in some feature space. Using this kernel one can construct a SV radial basis function machine.

In contrast to classical RBF methods, in the SV technique all four types of parameters are chosen automatically:

(i) N, the number of support vectors,

(ii) x_i, $(i = 1, ..., N)$ the support vectors;

(iii) $a_i = \alpha_i y_i$, the coefficients of expansion, and

(iv) γ, the width parameter of the kernel function chosen to minimize the functional (3.37).

The SV machine defines the first three parameters automatically, the width parameter γ is chosen to minimize (3.37).

3.4.3 Two-Layer Neural SV Machines

One can define two-layer neural networks by choosing kernels:

$$K(x, x_i) = S[(x * x_i)],$$

where $S(u)$ is a sigmoid function. In contrast to kernels for polynomial machines or for radial basis function machines that always satisfy Mercer's condition, the sigmoid kernel

$$S[(x * x_i)] = \frac{1}{1 + \exp\{v(x * x_i) - c\}}$$

satisfies Mercer's condition only for some parameters. For example if $|x| = 1$, $|x_i| = 1$ the parameters c and v of the sigmoid function has to satisfy the inequality[4] $c \geq v$. For these parameters values one can construct SV machines that implement the rules

$$f(x, \alpha) = \text{sign}\left\{\sum_{i=1}^{N} \alpha_i S(v(x * x_i) - c) + b\right\}.$$

Using the technique described above, one can automatically find the following:

(i) The architecture of the two layer machine, determining the number N of hidden units (the number of support vectors),

(ii) the vectors of the weights $w_i = x_i$ in the neurons of the first (hidden) layer (the support vectors), and

(iii) the vector of weights for the second layer (values of α).

3.5 THE SVM FOR REGRESSION ESTIMATION

3.5.1 ε-Insensitive Loss-Functions

To estimate a regression function $f(x)$ in a class of linear real valued functions

$$f(x, w, b) = (w * x) + b$$

from a given collection of data

$$(y_1, x_1), ..., (y_\ell, x_\ell)$$

one usually employs the empirical risk minimization method

$$R(w, b) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(y_i - (w * x_i) - b)$$

with quadratic loss-function

$$L(y, x, w, b) = (y - (w * x_i) - b)^2 \tag{3.41}$$

(the least square method).

Under conditions where y_i is the result of measuring a regression-function at point x_i with normal additive noise ξ_i, the ERM principle provides for this loss-function an efficient (best unbiased) estimator of the regression $f(x, \alpha_0)$.

[4]In this case one can introduce two $n + 1$-dimensional vectors: vector x^* for which the first n coordinates coincide with vector x and the last coordinate is equal to zero and vector vector x_i^* of which the first n coordinates coincide with vector x_i and the last coordinate is equal to $a = \sqrt{2(c - v)}$. If $c \geq v$ then a is real and one can rewrite the sigmoid function in n-dimensional input space as a radial basis function in $n + 1$ dimensional space $S[(x * x_i)] = \left(1 + \exp\{-0.5v||x^* - x_i^*||^2\}\right)^{-1}$.

It is known, however, that if additive noise is generated by other laws, better approximations to the regression (for the ERM principle) give estimators based on other loss-functions (associated with these laws)

$$L(y, x, w, b) = L(y - (w * x_i) - b)).$$

In 1964, Huber developed a theory that allows finding the best strategy for choosing the loss-function by using only general information about the model of the noise. In particular, he showed that if one only knows that the density describing the noise is a symmetric function, then the best minimax strategy for regression approximation (the best approximation for the worst possible model of noise) is provided by the loss-function

$$L(y, x, w, b) = |y - (w * x_i) - b)|. \tag{3.42}$$

Minimizing the empirical risk with respect to this loss-function is called the *least modulus* method. It belongs to the so-called *robust regression* family. However, this is an extreme case where one has the minimal information about the unknown density. Huber also considered the following loss-function that is robust if the noise is defined by a mixture (with a fixed proportion) of the normal density and some unknown symmetric density function:

$$H(y, x, w, b) = \begin{cases} c|y - (w * x_i) - b)| - \frac{c^2}{2} & \text{for } |y - (w * x_i) - b)| > c \\ \frac{1}{2}|y - (w * x_i) - b)|^2 & \text{for } |y - (w * x_i) - b)| \le c \end{cases} \tag{3.43}$$

where the value of c is defined by this proportion. This function smoothly combines quadratic and linear functions.

To construct the SV machine for real-valued functions we use a new type of loss-functions, the so-called ε-insensitive loss-functions [3], [4]:

$$L(y, x, w, b) = L(|y - (w * x_i) - b)|_\varepsilon)$$

where we denote

$$|y - (w * x_i) - b)|_\varepsilon = \begin{cases} 0, & \text{if } |y - (w * x_i) - b)| \le \varepsilon \\ |y - (w * x_i) - b)| - \varepsilon, & \text{otherwise.} \end{cases} \tag{3.44}$$

These loss-functions describe the ε-insensitive model: the loss is equal to 0 if the discrepancy between the predicted and the observed values is less than ε. In the sequel we consider three loss-functions: linear-insensitive, quadratic-insensitive, and Huber.

3.5.2 Minimizing the Risk with ε-insensitive Loss-function

The problem of minimizing the empirical risk functional with ε-insensitive losses is equivalent to the problem of finding the pair w, b that minimizes the quantity

defined by slack variables $\xi_i,\ \xi_i^*,\ i = 1, ..., \ell$

$$R(\xi, \xi^*) = \left(\sum_{i=1}^{\ell} F(\xi_i^*) + \sum_{i=1}^{\ell} F(\xi_i) \right)$$

under the constraints

$$\begin{cases} y_i - (w * x_i) - b \leq \varepsilon + \xi_i^*, \quad i = 1, ..., \ell, \\ (w * x_i) + b - y_i \leq \varepsilon + \xi_i, \quad i = 1, ..., \ell, \\ \xi_i^* \geq 0, \quad i = 1, ..., \ell, \\ \xi_i \geq 0, \quad i = 1, ..., \ell. \end{cases} \tag{3.45}$$

We however consider the following problem: minimize the functional

$$R(\xi, \xi^*) = C \left(\sum_{i=1}^{\ell} F(\xi_i^*) + \sum_{i=1}^{\ell} F(\xi_i) \right) + \frac{1}{2}(w * w) \tag{3.46}$$

subject to constraints (3.45). As before, to solve the optimization problem with constraints of inequality type one has to find the saddle point of the Lagrange functional

$$L(w, \xi^*, \xi; \alpha^*, \alpha, , \gamma, \beta, \beta^*) =$$

$$\tfrac{1}{2}(w * w) + \textstyle\sum_{i=1}^{\ell} [F(\xi_i^*) + F(\xi_i)] - \sum_{i=1}^{\ell} \alpha_i [y_i - (w * x_i) - b + \varepsilon_i + \xi_i] -$$

$$\textstyle\sum_{i=1}^{\ell} \alpha_i^* [(w * x_i) + b - y_i + \varepsilon_i + \xi_i^*] - \sum_{i=1}^{\ell} (\beta_i^* \xi_i^* + \beta_i \xi_i). \tag{3.47}$$

One takes the minimum with respect to elements w, b, ξ_i^*, and ξ_i^* and the maximum with respect to Lagrange multipliers $\alpha_i^* \geq 0$, $\alpha_i \geq 0$, $\beta_i^* \geq 0$, and $\beta_i \geq 0$ $(i = 1, ..., \ell)$. Minimization with respect to w, b and ξ_i^*, ξ_i implies the following three conditions:

$$w = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) x_i, \tag{3.48}$$

$$\sum_{i=1}^{\ell} \alpha_i^* = \sum_{i=1}^{\ell} \alpha_i, \tag{3.49}$$

$$\gamma_i + \alpha_i^* \leq F'(\xi_i^*), \quad i = 1, ..., \ell, \tag{3.50}$$

$$\gamma_i + \alpha_i \leq F^{'}(\xi_i), \quad i = 1, ..., \ell.$$

Condition (3.48) means that the desired vector w has an expansion on some elements of the training data. To find the parameters α_i^*, α_i of this expansion (to find elements of the saddle point) we put (3.48) into the Lagrangian (3.47). Then taking into account (3.49) and (3.50) we obtain that to find parameters α_i^*, α_i of the saddle point we have to solve the following optimization problems.

Case $F(\xi) = \xi$. In order to find coefficients α^*, α that specifies w one has to maximize the functional

$$\begin{aligned} W(\alpha, \alpha^*) &= -\sum_{i=1}^{\ell} \varepsilon_i(\alpha_i^* + \alpha_i) + \sum_{i=1}^{\ell} y_i(\alpha_i^* - \alpha_i) \\ &\quad -\tfrac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(x_i * x_j) \end{aligned} \tag{3.51}$$

subject to constraints

$$\begin{cases} \sum_{i=1}^{\ell} \alpha_i^* = \sum_{i=1}^{\ell} \alpha_i \\ 0 \leq \alpha_i^*, \leq C \\ 0 \leq \alpha_i, \leq C. \end{cases} \tag{3.52}$$

The parameters α_i, α_i^* specify the desired approximation

$$f(x) = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i)(x_i * x) + b.$$

From (3.51) it is easy to see that for any i the equality

$$\alpha_i^* \times \alpha_i = 0$$

holds true. Hence for the particular case $\varepsilon = 0$ and $y_i \in \{-1, 1\}$ the considered optimization problem coincides with the one described for pattern recognition.

Case $F(\xi) = \xi^2$. In order to find coefficients α^*, α that specify w one has to maximize the functional

$$\begin{aligned} W(\alpha, \alpha^*) &= -\sum_{i=1}^{\ell} \varepsilon_i(\alpha_i^* + \alpha_i) + \sum_{i=1}^{\ell} y_i(\alpha_i^* - \alpha_i) \\ &\quad -\tfrac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(x_i * x_j) - \tfrac{1}{2C}\sum_{i=1}^{\ell}\left((\alpha_i^*)^2 + \alpha_i^2\right) \end{aligned} \tag{3.53}$$

subject to the constraints

$$\begin{cases} \sum_{i=1}^{\ell} \alpha_i^* = \sum_{i=1}^{\ell} \alpha_i \\ \alpha_i^* \geq 0 \\ \alpha_i \geq 0. \end{cases} \tag{3.54}$$

Again the parameters α_i, α_i^* specify the desired approximation

$$f(x) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i)(x_i * x) + b.$$

Huber loss-function. Consider the Huber loss-function

$$H(\xi) = \begin{cases} c|\xi| - \frac{c^2}{2} & \text{for } |\xi| > c \\ \frac{1}{2}|\xi|^2 & \text{for } |\xi| \le c. \end{cases}$$

Let us minimize the functional

$$\Phi(w, \xi^*, \xi) = \frac{1}{2}(w * w) + C\left(\sum_{i=1}^{\ell} H(\xi_i^*) + \sum_{i=1}^{\ell} H(\xi_i)\right)$$

subject to constraints

$$\begin{cases} y_i - (w * x_i) - b \le \xi_i^*, \quad i = 1, ..., \ell, \\ (w * x_i) + b - y_i \le \xi_i, \quad i = 1, ..., \ell, \\ \xi_i^* \ge 0, \quad i = 1, ..., \ell, \\ \xi_i \ge 0, \quad i = 1, ..., \ell. \end{cases}$$

For this loss function, in order to find the desired linear function

$$(w_0 * x) + b = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i)(x_i * x) + b$$

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

$$W(\alpha, \alpha^*) = \textstyle\sum_{i=1}^{\ell} y_i(\alpha_i^* - \alpha_i) -$$

$$\frac{1}{2}\left(\textstyle\sum_{i,j=1}^{\ell} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(x_i * x_j) + \frac{c}{C}\sum_{i=1}^{\ell} (\alpha_i^*)^2 + \frac{c}{C}\sum_{i=1}^{\ell} (\alpha_i)^2\right) \tag{3.55}$$

subject to constraints

$$\begin{cases} \sum_{i=1}^{\ell} \alpha_i^* = \sum_{i=1}^{\ell} \alpha_i \\ 0 \le \alpha_i^* \le C, \quad i = 1, ..., \ell, \\ 0 \le \alpha_i \le C, \quad i = 1, ..., \ell, \end{cases}$$

When $c = \varepsilon < 1$ the solution obtained for the Huber loss-function is close to the solution obtained for the ε-insensitive loss-function. However, the expansion of the solution for the ε-insensitive loss-function uses less support vectors.

3.5.3 SV Machines for Regression Estimation

Now we are ready to construct the support vector machine for real-valued function estimation problems. As in the pattern recognition case we map the input vectors x into a high-dimensional feature space Z. We consider linear functions

$$f(x, \alpha, \alpha^*) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i)(z * z_i) + b. \tag{3.56}$$

As in the pattern recognition case we will not perform the mapping explicitly. We will perform it implicitly by using kernels for estimating the inner product in feature space. To construct the linear function (3.56) in feature space Z we use results obtained in the previous sections with only one correction: in all obtained formulas we replace the inner product in input space $(x_i * x_j)$ with the inner product in feature space described by the corresponding kernel $K(x_i, x_j)$ that satisfies Mercer's condition.

Hence, our linear function in feature space (3.56) has the following equivalent representation in input space

$$f(x, \alpha, \alpha^*) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) K(x, x_i) + b, \tag{3.57}$$

where α, α^*, $i = 1, .., \ell$ are scalars, obtained by minimizing (3.51), (3.53) or (3.55) under corresponding constraints, where instead of inner product $(x * x_i)$ one uses kernels $K(x, x_i)$ that satisfy Mercer's condition.

3.6 KERNELS FOR ESTIMATING REAL-VALUED FUNCTIONS

To construct different types of SV machines for estimating real-valued functions one has to choose different kernels $K(x, x_i)$ that satisfy Mercer's condition.

In particular one can use the same kernels as were used for the approximation of indicator functions:

(i) kernels generating polynomials

$$K(x, x_i) = [(x * x^*) + 1]^d,$$

(ii) kernels generating radial basis functions

$$K(x, x_i) = K(|x - x_i|),$$

for example

$$K(|x - x_i|) = \exp\left\{-\gamma |x - x_i|^2\right\},$$

(iii) kernels generating two-layer neural networks

$$K(x, x_i) = S(v(x * x_i) + c).$$

On the basis of these kernels one can obtain the approximation

$$f(x, \alpha^*, \alpha) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) K(x, x_i) + b \tag{3.58}$$

by using the optimization techniques described above.

These kernels imply approximating functions $f(x, \alpha)$ that were used in the pattern recognition problem under discrimination sign, i.e. the functions sign $[f(x, \alpha)]$.

However, the problem of approximating real-valued functions is more delicate than the approximation of indicator functions. Various real-valued function estimation problems need various sets of approximating functions. Therefore it is important to construct special kernels that reflect special properties of approximating functions.

To construct such kernels we will use two main techniques:

(i) constructing kernels for approximating one-dimensional functions, and

(ii) composition of multidimensional kernels using one-dimensional kernels.

3.6.1 Kernels Generating Splines

Here we introduce kernels that can be used to construct a spline-approximation of high dimensional functions . We construct splines with both a fixed number of knots and with an infinite number of knots. In all cases, the computational complexity of the solution depends on the number of support vectors that one needs in order to approximate the desired function with ε-accuracy, rather than on the dimensionality of the space or on the number of knots.

Let us approximate functions on the interval $[0, a]$ by using splines of order $d \geq 0$ with m knots

$$(t_1, ..., t_m), \quad t_i = \frac{ia}{m}, \quad i = 1, ..., m.$$

According to their definition, spline approximations have the form

$$f(x) = \sum_{r=0}^{d} a_r^* x^r + \sum_{i=1}^{m} a_i (x - t_i)_+^d, \tag{3.59}$$

where we denote

$$(x - t_k)_+^d = \begin{cases} 0 & \text{if } x \leq t_k \\ (x - t_k)^d & \text{if } x > t_k. \end{cases}$$

Consider the following mapping of the one dimensional variable x into a $m + d + 1$-dimensional vector u:

$$x \longrightarrow u = (1, x, ..., x^d, (x - t_1)_+^d, ...(x - t_m)_+^d).$$

Since spline approximation (3.59) can be considered as the inner product of two vectors

$$f(x) = (a * u),$$

where $a = (a_0, ..., a_{m+d+1})$, one can define the kernel that generates the inner product in feature space as follows [7], [4]:

$$K(x, x_t) = (u * u_t) = \sum_{r=0}^{d} x^r x_i^r + \sum_{i=1}^{m} (x - t_i)_+^d (x_t - t_i)_+^d. \tag{3.60}$$

Using the generating kernel (3.60) the SV machine constructs the function

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

which is a spline of order d defined on m knots.

Note that n-dimensional splines are defined as an expansion on the basis functions that are tensor products of one dimensional basis functions. Hence, kernels generating n-dimensional splines are the product of n one dimensional kernels:

$$K(x, x_i) = \prod_{k=1}^{n} K(x^k, x_i^k),$$

where $x = (x^1, ..., x^k)$.

3.6.2 *Kernels Generating Splines with an Infinite Number of Knots*

In applications of SV machines the number of knots does not play an important role (the values of ε_i are more important). Therefore to simplify the calculation, we use splines with an infinite number of knots defined on the interval $(0, a)$ with $0 < a < \infty$ as the expansion

$$f(x) = \sum_{i=0}^{d} a_i x^i + \int_0^a a(t)(x - t)_+ dt$$

where a_i $(i = 0, .., d)$ are unknown values and $a(t)$ is an unknown function which defines the expansion. One can consider this expansion as an inner product. Therefore one can construct the following kernel for generating splines of order d with an infinite number of knots [7],[4]:

$$K(x_j, x_i) = \int_0^a (x_j - t)_+^d (x_i - t)_+^d dt + \sum_{r=0}^{d} x^r x_i^r \tag{3.61}$$

$$= \int_0^{(x_j \wedge x_i)} (x_j - t)^d (x_i - t)^d dt + \sum_{r=0}^{d} x_j^r x_i^r$$

$$= \int_0^{(x_j \wedge x_i)} u^d (u + |x_j - x_i|)^d du + \sum_{r=1}^{d} x_j^r x_i^r$$

$$= \sum_{r=0}^{d} \frac{C_d^r}{2d - r + 1} (x_j \wedge x_i)^{2d-r+1} |x_j - x_j|^r + \sum_{r=0}^{d} x_j^r x_i^r,$$

where we denote $\min(x, x_i) = (x \wedge x_i)$. In particular, for the linear spline ($d = 1$) we have

$$K_1(x_j, x_i) = 1 + x_j x_i + \frac{1}{2}|x_j - x_i|(x_j \wedge x_i)^2 + \frac{(x_j \wedge x_i)^3}{3}.$$

Again, the kernel for n-dimensional splines with an infinite number of knots is the product of the n kernels for one dimensional splines.

On the basis of this kernel one can construct a spline approximation (by means of the techniques described in the previous section) that has the form

$$f(x, \alpha^*, \alpha) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) K(x, x_i).$$

3.6.3 *Kernels Generating Fourier Expansions*

An important role in signal processing is played by Fourier expansions. In this section we construct kernels for SV Fourier expansion. As before we start with the one dimensional case.

Suppose we would like to analyze a one-dimensional signal in terms of a Fourier series expansion.

Let us map the input variable x into the $(2N + 1)$ dimensional vector

$$u = (1/\sqrt{2}, \sin x, ..., \sin Nx, \ \cos x, ..., \cos Nx).$$

Then for any fixed x the Fourier expansion can be considered as the inner product in this $2N + 1$ dimensional feature space

$$f(x) = (a * u) = \frac{a_0}{\sqrt{2}} + \sum_{k=1}^{N} (a_k \sin kx + b_k \cos kx). \tag{3.62}$$

Hence, the inner product of two vectors in this space has the form

$$K_N(x, x_i) = \frac{1}{2} + \sum_{k=1}^{N} (\sin kx \sin kx_i + \cos kx \cos kx_i).$$

After obvious transformations and taking into account Dirichlet function we obtain [7], [4]

$$K_N(x, x_i) = \frac{1}{2} + \sum_{k=1}^{N} \cos k(x - x_i) = \frac{\sin \frac{(2N+1)}{2}(x - x_i)}{\sin \frac{(x-x_i)}{2}}.$$

To construct the following regularized Fourier expansion

$$f(x) = (a * u) = \frac{a_0}{\sqrt{2}} + \sum_{k=1}^{\infty} q^k \left(a_k \sin kx + b_k \cos kx\right), \quad 0 < q < 1$$

one employs the following kernel

$$K(x_i, x_j) = \frac{1 - q^2}{1 - 2q\cos(x_i - x_j) + q^2}.$$

To define the signal in terms of the Fourier expansion, the SV machine uses the representation

$$f(x, \alpha^*, \alpha) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) K_N(x, x_i).$$

Again, to construct the SV machine for the d-dimensional vector space $x = (x^1, ..., x^n)$, it is sufficient to use the generating kernel that is a product of one dimensional kernels

$$K(x, x_i) = \prod_{k=1}^{n} K_N(x^k, x_i^k).$$

3.7 SOLVING OPERATOR EQUATIONS

One can also use the SV technique for solving linear operator equation [6], [4]

$$Af(t) = F(x)$$

on the basis of measurements F_i (with accuracy ε_i) of the right hand side of equation at the points x_i. In other words, one has to solve the equation on the basis of triplets

$$(x_1, F_1, \varepsilon_1), ..., (x_\ell, F_\ell, \varepsilon_\ell).$$

Consider the solution of the equation in the set of functions

$$f(t, g) = \int_{\infty}^{\infty} g(\tau)\psi(t, \tau)d\tau \tag{3.63}$$

where $\psi(\tau, t)$ is some known function and $g(\tau)$ is an unknown function from L_2. Finding the solution means specifying the function $g(u)$ in (3.63).

Suppose that operator A maps this set of functions into another set

$$F(x,g) = \int_{-\infty}^{\infty} g(\tau)\phi(x,\tau)d\tau \tag{3.64}$$

in a one-to-one manner where $\phi(x,\tau) = A\psi(t,\tau)$. For every fixed x one can consider the function (3.64) as an inner product in Hilbert space. Therefore one can construct the kernel

$$K(x_i,x_j) = \int_{-\infty}^{\infty} \phi(x_i,\tau)\phi(x_j,\tau)d\tau. \tag{3.65}$$

By using this kernel one estimates the function $g(\tau)$ on the basis of the technique described in Section 3.5:

$$g(\tau) = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i)\phi(x_i,\tau).$$

Putting the expression for $g(\tau)$ back into (3.63), one obtains

$$f(t) = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i)\mathcal{K}(x_i,t) \tag{3.66}$$

where $\mathcal{K}(x_i,t)$ denotes the cross-kernel function

$$\mathcal{K}(x_i,t) = \int_{-\infty}^{\infty} \phi(x_i,\tau)\psi(t,\tau)d\tau. \tag{3.67}$$

In order to solve the linear operator equation by means of the SV method, one proceeds then as follows:

1. Define the corresponding regression problem in image space.
2. Construct the kernel function $K(x_i,x_j)$ for solving the regression problem by using the SV method.
3. Construct the corresponding cross-kernel function $\mathcal{K}(x_i,t)$.
4. Using the kernel function $K(x_i,x_j)$ solve the regression problem by the SV method (i.e. find the support vectors x_1^*, $i = 1,...,N$ and the corresponding coefficients α_i^*, α_i, $i = 1,...,N$).
5. Using these support vectors and the corresponding coefficients, define the solution

$$f(t) = \sum_{r=1}^{N}(\alpha_r^* - \alpha_r)\mathcal{K}(x_r,t). \tag{3.68}$$

In these five steps the first three steps (constructing regression problem, constructing kernel in image space, and constructing corresponding cross-kernel function) reflect the singularity of the problem at hand (they depend on operator A). The last two steps (solving the regression problem by SV machine and constructing the solution to the desired problem) are routine.

The main problem in solving the operator equation by means of the SV technique is, for a given operator equation, to obtain both the explicit expression for the kernel function in image space and an explicit expression for the corresponding cross-kernel function.

3.8 CONCLUSION

In this chapter we described the SVM technique for function estimation problems. This technique is based on the so-called Structural Risk Minimization induction principle [4]. According to this principle, the quality of the estimated function depends on the accuracy of the approximation of the data and on the capacity of the set of functions from which the approximating function is chosen. The capacity concepts have pure combinatorial definitions and do not depend directly on the dimensionality of the space.

Therefore, by means of the SVM technique one can obtain a good solution in high dimensional spaces by controlling the capacity factors. For example, in the problem of text categorization (sorting texts into some number of categories) the obtained SV rules, which outperform existing state-of-the-art rules, are polynomials of degree 5 in $\approx 10,000$ dimensional space[5]. In order to construct text categorization rules on the basis of 10,000 given examples, all combinations up to 5 words (the order of polynomials) were taken into account from a dictionary containing 10,000 words (dimensionality of the input space).

One can consider the SVM method as a tool to overcome the curse of dimensionality in function estimation problems. It combines the principle of generalization in high dimensional spaces by controlling capacity factors (the margin and the ε-insensitivity value) with the constructive approach to function estimation in high dimensional spaces (kernel representations of inner products).

References

[1] Boser, B. E., Gayon, I. M. and Vapnik, V. N. (1992). A training algorithm for optimal margin classifier. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory.* pp.144-152, Pittsburgh, PA.

[2] Cortes, C. and Vapnik, V. (1995). Support Vector Network. *Machine Learning,* 20: 273-297.

[3] Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* Springer Verlag, New-York.

[4] Vapnik, V. (1998). *Statistical Learning Theory.* J. Wiley, New-York.

[5] T. Joachims "Text categorization with Support Vector Machines" LS VIII Technical Report No.23, University of Dortmund.

[5] Vapnik, V. N. and Chervonenkis, A. Ya. (1964). A note on one class of perceptrons. *Automation and Remote Control* (25)1.

[6] Vapnik, V. N. and Chervonenkis, A. Ya. (1974)(1979). *Theory of Pattern Recognition* (in Russian) Nauka, Moscow, 1974. (German translation: W.N. Wapnik, A Ja. Tscherwonenkis *Teorie der Zeichenerkennung* Akademia-Verlag, Berlin, 1979.)

[7] Vapnik, V., Golowich, S. and Smola, A. (1997). Support vector method for function approximation, regression estimation and signal processing. *In Advances in Neural Information Processing Systems, 9.*, MIT Press.