Vladimir N. Vapnik

The Nature of Statistical Learning Theory

With 33 Illustrations

Springer

5.4 THE OPTIMAL SEPARATING HYPERPLANE

Below we consider a new type of universal learning machine that implements the second strategy: keep the value of the empirical risk fixed and minimize the confidence interval.

As in the case of neural networks, we start by considering linear decision rules (the separating hyperplanes). However, in contrast to previous considerations, we use a special type of hyperplane, the so-called Optimal separating hyperplanes (Vapnik and Chervonenkis, 1974), (Vapnik, 1979). First we consider the Optimal separating hyperplane for the case where the training data are linearly separable. Then, in Section 5.5.1 we generalize the idea of Optimal separating hyperplanes to the case of nonseparable data. Using a technique for constructing Optimal hyperplanes, we describe a new type of universal learning machine, the Support Vector machine. Finally we construct the Support Vector machine for solving regression estimation problems.

5.4.1 The Optimal Hyperplane

Suppose the training data

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

can be separated by a hyperplane:

$$(w \cdot x) + b = 0. \tag{5.7}$$

We say that this set of vectors is separated by the *Optimal hyperplane* if it is separated without error and the distance between the closest vector to the hyperplane is maximal (Fig. 5.2).

To describe the separating hyperplane let us use the following canonical form:

$$(w \cdot x_i) + b \geq 1 \quad \text{if } y_i = 1\,,$$

$$(w \cdot x_i) + b \leq -1 \quad \text{if } y_i = -1.$$

In the following we use a compact notation for these inequalities:

$$y_i[(w \cdot x_i) + b] \geq 1, \quad i = 1, \ldots, \ell. \tag{5.8}$$

It is easy to check that the Optimal hyperplane is the one that satisfies the conditions (5.8) and minimizes

$$\Phi(w) = ||w||^2. \tag{5.9}$$

(The minimization is taken with respect to both vector w and scalar b.)

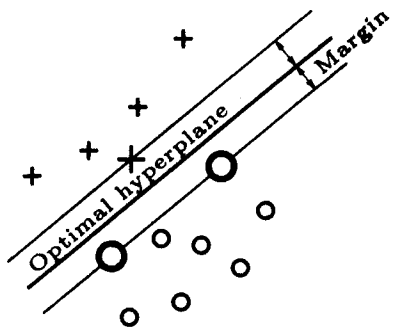

FIGURE 5.2. The Optimal separating hyperplane is the one that separates the data with maximal margin.

5.4.2 *The Structure of Canonical Hyperplanes*

Now let separating hyperplanes is defined on the set of vectors

$$X^* = x_1, \ldots, x_r,$$

bounded by a sphere of the radius R

$$|x_i - a| \leq R, \quad x_i \in X^*$$

(a is the center of the sphere). Consider a set of hyperplanes in *canonical form* (with respect to these vectors) defined by the pairs (w, b) satisfying the condition

$$\min_{x_i \in X^*} |(w \cdot x_i) + b| = 1.$$

Note that the set of canonical separating hyperplanes coincides with the set of all separating hyperplanes. It only specifies the normalization of the parameters of hyperplanes.

The idea of constructing a machine that fixes the empirical risk and minimizes the confidence interval is based on the existence of the following bound on the VC dimension of canonical hyperplanes.

Theorem 5.1. *A subset of canonical hyperplanes*

$$f(x, w, b) = sign\{(w \cdot x) + b\},$$

defined on X^* and satisfying the constraint

$$||w|| \leq A$$

has the VC dimension h bounded by the inequality

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

In Section 3.5 we stated that the VC dimension of the set of hyperplanes is equal to $n + 1$, where n is dimensionality of the space. However, the VC dimension of the *subset* of the set of hyperplanes, with canonical form satisfying $|w|^2 \leq A^2$, can be less.[2]

Below we consider hyperplanes only in canonical form, constructed on the basis of the training vectors $X^* = x_1, \ldots, x_\ell$.[3] For simplicity we call them hyperplanes.

Let us construct the structure on the set of hyperplanes by increasing the norm of the weights w. Then in order to obtain the smallest probability of error on the test set, we choose the hyperplane from the element of the structure which separates the training data and whose element of the structure gives the smallest bound on the VC dimension, that is, with the smallest norm of weights.

5.5 CONSTRUCTING THE OPTIMAL HYPERPLANE

To construct the Optimal hyperplane one has to separate the vectors x_i of the training set

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

belonging to two different classes $y \in \{-1, 1\}$ using the hyperplane with the smallest norm of coefficients.

To find this hyperplane one has to solve the following quadratic programming problem: minimize the functional

$$\Phi(w) = \frac{1}{2}(w \cdot w) \tag{5.10}$$

under the constraints of inequality type

$$y_i[(x_i \cdot w) + b] \geq 1, \quad i = 1, 2, \ldots, \ell. \tag{5.11}$$

The solution to this optimization problem is given by the saddle point of the Lagrange functional (Lagrangian):

$$L(w, b, \alpha) = \frac{1}{2}(w \cdot w) - \sum_{i=1}^{\ell} \alpha_i \{[(x \cdot w) + b] y_i - 1\}, \tag{5.12}$$

where the α_i are Lagrange multipliers. The Lagrangian has to be minimized with respect to w, b and maximized with respect to $\alpha_i > 0$.

---

[2] In Section 5.7 we describe a separating hyperplane in 10^{13} dimensional space with relatively small estimate of the VC dimension ($\approx 10^3$).

[3] In Section 5.11 we will discuss this choice of the set X^*.

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

$$\frac{\partial L(w_0, b_0, \alpha^0)}{\partial b} = 0,$$

$$\frac{\partial L(w_0, b_0, \alpha^0)}{\partial w} = 0.$$

Rewriting these equations in explicit form one obtains the following properties of the Optimal hyperplane:

(i) The coefficients α_i^0 for the Optimal hyperplane should satisfy the constraints

$$\sum_{i=1}^{\ell} \alpha_i^0 y_i = 0, \quad \alpha_i^0 \geq 0, \quad i = 1, \ldots, \ell \tag{5.13}$$

(first equation).

(ii) The Optimal hyperplane (vector w_0) is a linear combination of the vectors of the training set.

$$w_0 = \sum_{i=1}^{\ell} y_i \alpha_i^0 x_i, \quad \alpha_i^0 \geq 0, \quad i = 1, \ldots, \ell \tag{5.14}$$

(second equation).

(iii) Moreover, only the so-called *support vectors* can have nonzero coefficients α_i^0 in the expansion of w_0. The support vectors are the vectors for which, in inequality (5.11), the equality is achieved. Therefore we obtain

$$w_0 = \sum_{\text{support vectors}} y_i \alpha_i^0 x_i, \quad \alpha_i^0 \geq 0. \tag{5.15}$$

This fact follows from the classical Kühn–Tucker theorem, according to which the necessary and sufficient conditions for the Optimal hyperplane are that the separating hyperplane satisfy the conditions:

$$\alpha_i^0 \{[(x_i \cdot w_0) + b_0] y_i - 1\} = 0, \quad i = 1, \ldots, \ell. \tag{5.16}$$

Putting the expression for w_0 into the Lagrangian and taking into account the Kühn–Tucker conditions, one obtains the functional

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j}^{\ell} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j). \tag{5.17}$$

It remains to maximize this functional in the non-negative quadrant

$$\alpha_i \geq 0, \quad i = 1, \ldots, \ell \tag{5.18}$$

under the constraint

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0. \tag{5.19}$$

According to Eq. (5.15) the Lagrange multipliers and support vectors determine the Optimal hyperplane. Thus, to construct the Optimal hyperplane one has to solve a simple quadratic programming problem: maximize the quadratic form (5.17) under constraints[4] (5.18) and (5.19).

Let $\alpha_0 = (\alpha_1^0, \ldots, \alpha_\ell^0)$ be a solution to this quadratic optimization problem. Then the norm of the vector w_0 corresponding to the Optimal hyperplane equals:

$$|w_0|^2 = 2W(\alpha_0) = \sum_{\text{support vectors}} \alpha_i^0 \alpha_j^0 (x_i \cdot x_j) y_i y_j.$$

The separating rule, based on the Optimal hyperplane, is the following indicator function

$$f(x) = \text{sign}\left(\sum_{\text{support vectors}} y_i \alpha_i^0 (x_i \cdot x) - b_0 \right), \tag{5.20}$$

where x_i are the support vectors, α_i^0 are the corresponding Lagrange coefficients, and b_0 is the constant (threshold)

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

where we denote by $x^*(1)$ some (any) support vector belonging to the first class and we denote by $x^*(-1)$ a support vector belonging to the second class (Vapnik and Chervonenkis, 1974), (Vapnik, 1979).

5.5.1 *Generalization for the Nonseparable Case*

To construct the Optimal type hyperplane in the case when the data are linearly nonseparable, we introduce non-negative variables $\xi_i \geq 0$ and a

[4] This quadratic programming problem is simple because it has simple constraints. For the solution of this problem, one can use special methods which are fast and applicable for the case with a large number of support vectors ($\approx 10^4$ support vectors) (More and Toraldo, 1991). Note that in the training data the support vectors constitute only a small part of the training vectors (in our experiments 3% to 5%).

function

$$F_\sigma(\xi) = \sum_{i=1}^{\ell} \xi_i^\sigma$$

with parameter $\sigma > 0$.

Let us minimize the functional $F_\sigma(\xi)$ subject to constraints

$$y_i((w \cdot x_i) + b) \geq 1 - \xi_i, \qquad i = 1, 2, \ldots, \ell, \tag{5.21}$$

and one more constraint

$$(w \cdot w) \leq c_n. \tag{5.22}$$

For sufficiently small $\sigma > 0$ the solution to this optimization problem defines a hyperplane that minimizes number of training errors under condition that the parameters of this hyperplane belong to the subset (5.22) (to the element of the structure

$$S_n = \{(w \cdot x) + b : \ (w \cdot w) \leq c_n\}$$

determined by constant c_n).

For computational reasons, however, we consider the case $\sigma = 1$. This case corresponds to the smallest $\sigma > 0$ that is still computationally simple. We call this hyperplane the Generalized Optimal hyperplane.

1. One can show (using the technique described above) that the Generalized Optimal hyperplane is determined by the vector

$$w = \frac{1}{C^*} \sum_{i=1}^{\ell} \alpha_i y_i x_i,$$

where parameters α_i, $i = 1, ..., \ell$ and C^* are the solutions to the following convex optimization problem:

Maximize the functional

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

subject to constraints

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

$$0 \leq \alpha_i \leq 1, \quad i = 1, ..., \ell$$

$$C^* \geq 0$$

2. To simplify computations one can introduce the following (slightly modified) concept of the Generalized Optimal hyperplane (Cortes and Vapnik, 1995). The Generalized Optimal hyperplane is determined by the vector w that minimizes the functional

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

(here C is a *given* value) subject to constraint (5.21).

The technique of solution of this quadratic optimization problem is almost equivalent to the technique used in the separable case: to find the coefficients of the generalized Optimal hyperplane

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

one has to find the parameters α_i, $i = 1, ..., \ell$ one that maximize the same quadratic form as in the separable case

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

under slightly different constraints

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, \ell,$$

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

As in the separable case, only some of the coefficients α_i, $i = 1, ..., \ell$ differ from zero. They determine the support vectors.

Note that if the coefficient C in the functional $\Phi(w, \xi)$ is equal to the optimal value of parameter C^* for minimization of the functional $F_1(\xi)$

$$C = C^*,$$

then the solution to both optimization problems (defined by the functional $F_1(\xi)$ and by the functional $\Phi(w, \xi)$) coincide.

5.6 SUPPORT VECTOR (SV) MACHINES

The Support Vector (SV) machine implements the following idea: it maps the input vectors x into a high-dimensional feature space Z through some nonlinear mapping, chosen *a priori*. In this space, an Optimal separating hyperplane is constructed (Fig. 5.3).

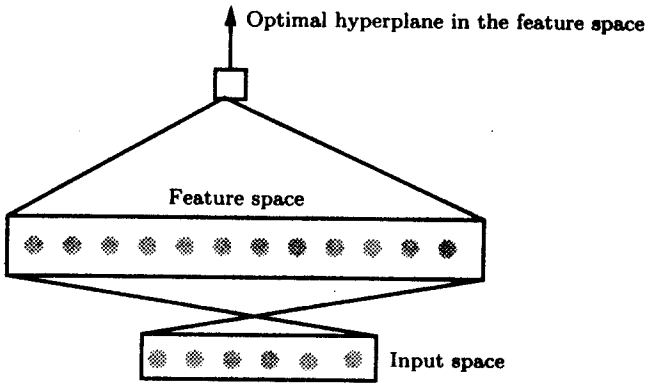

FIGURE 5.3. The SV machine maps the input space into a high-dimensional feature space and then constructs an Optimal hyperplane in the feature space.

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

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

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: one conceptual and one technical.

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

The dimensionality of the feature space will be huge, and a hyperplane that separates the training data will not necessarily generalize well.[5]

(ii) *How to treat computationally such high-dimensional spaces?* (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 this "curse of dimensionality" be overcome?

[5]Recall Fisher's concern about the small amount of data for constructing a quadratic discriminant function in classical discriminant analysis (Section 1.9).

5.6.1 Generalization in High-Dimensional Space

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

According to Theorem 5.1, if it happens that in the high-dimensional input space one can construct a separating hyperplane with a small value of $[R^2 A^2]$, the VC dimension of the corresponding element of the structure will be small, and therefore the generalization ability of the constructed hyperplane will be high.

Furthermore, the following theorem holds.

Theorem 5.2. *If the training vectors are separated by the Optimal hyperplane (or generalized Optimal hyperplane), then the expectation value of the probability of committing an error on a test example is bounded by the ratio of the expectation of the number of support vectors to the number of examples in the training set:*

$$E[P(\text{error})] \leq \frac{E[\text{number of support vectors}]}{(\text{number of training vectors}) - 1} . \qquad (5.23)$$

This bound depends neither on the dimensionality of the space, nor on the norm of the vector of coefficients, nor on the bound of the norm of the input vectors. Therefore, if the Optimal hyperplane can be constructed from a small number of support vectors relative to the training set size, the generalization ability will be high — even in an infinite-dimensional space.[6]

5.6.2 Convolution of the Inner Product

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.

In 1992 it was observed (Boser, Guyon, and Vapnik, 1992) that for constructing the Optimal separating hyperplane in the feature space Z, one

[6]One can compare the result of this theorem to result of analysis of the following compression scheme. To construct the Optimal separating hyperplane one only needs to specify among the training data the support vectors and its classification. This requires: $\approx \lceil \lg_2 m \rceil$ bits to specify the number m of support vectors, $\lceil \lg_2 C_\ell^m \rceil$ bits to specify the support vectors; and $\lceil \lg_2 C_m^{m_1} \rceil$ bits to specify representatives of the first class among the support vectors. Therefore for $m << \ell$ and $m_1 \approx m/2$ the compression coefficient is

$$K \sim \frac{m(\lg_2 \ell/m + 1)}{\ell}.$$

The expectation of this coefficient should be compared to the value $Em/(\ell - 1)$ (the right hand side of inequality (5.23)).

does not need to consider the feature space in *explicit form*. One only has to be able to calculate the inner products between support vectors and the vectors of the feature space (Eqs. (5.17) and (5.20)).

Consider a general expression for the inner product in Hilbert space[7]

$$(z_i \cdot z) = K(x, x_i),$$

where z is the image in feature space of the vector x in input space.

According to Hilbert–Schmidt theory, $K(x, x_i)$ can be any symmetric function satisfying the following general conditions (Courant and Hilbert, 1953):

Theorem 5.3. (Mercer) *To guarantee that the symmetric function $K(u, v)$ from L_2 has an expansion*

$$K(u, v) = \sum_{k=1}^{\infty} a_k \psi_k(u) \psi_k(v) \tag{5.24}$$

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

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

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

$$\int g^2(u) du < \infty.$$

5.6.3 Constructing SV Machines

The convolution of the inner product allows the construction of decision functions that are nonlinear in the input space

$$f(x) = \text{sign}\left(\sum_{\text{support vectors}} y_i \alpha_i K(x_i, x) - b \right), \tag{5.25}$$

and that are equivalent to linear decision functions in the high-dimensional feature space $\psi_1(x), \ldots, \psi_N(x)$ ($K(x_i, x)$ is a convolution of the inner product for this feature space).

[7]This idea was used in 1964 by Aizerman, Braverman, and Rozonoer in their analysis of the convergence properties of the method of Potential functions (Aizerman, Braverman, and Rozonoer, 1964, 1970). It happened at the same time (1965) when the method of the Optimal hyperplane was developed (Vapnik and Chervonenkis 1965). However, combining these two ideas, which lead to the SV machines, was only done in 1992.

To find the coefficients α_i in the separable case (analogously in the non-separable case) it is sufficient to find the maximum of the functional

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j}^{\ell} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{5.26}$$

subject to the constraints

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

$$\alpha_i \geq 0, \quad i = 1, 2, \ldots, \ell. \tag{5.27}$$

This functional coincides with the functional for finding the Optimal hyperplane, except for the form of the inner products: instead of inner products $(x_i \cdot x_j)$ in Eq. (5.17), we now use the convolution of the inner products $K(x_i, x_j)$.

The learning machines which construct decision functions of the type (5.25) are called *Support Vector (SV) Machines*. (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.) The scheme of SV machines is shown in Fig. 5.4.

5.6.4 Examples of SV Machines

Using different functions for convolution of the inner products $K(x, x_i)$, one can construct learning machines with different types of nonlinear decision surfaces in input space. Below, we consider three types of learning machines:

(i) Polynomial Learning Machines,

(ii) Radial Basis Functions Machines, and

(iii) Two Layer Neural Networks.

For simplicity we consider here the regime where the training vectors are separated without error.

Note that the support vector machines implement the SRM principle. Indeed, let

$$\Psi(x) = (\psi_1(x), \ldots, \psi_N(x))$$

be a feature space and $w = (w_1, \ldots, w_N)$ be a vector of weights determining a hyperplane in this space. Consider a structure on the set of hyperplanes with elements S_k containing the functions satisfying the conditions

$$R^2 |w|^2 \leq k,$$

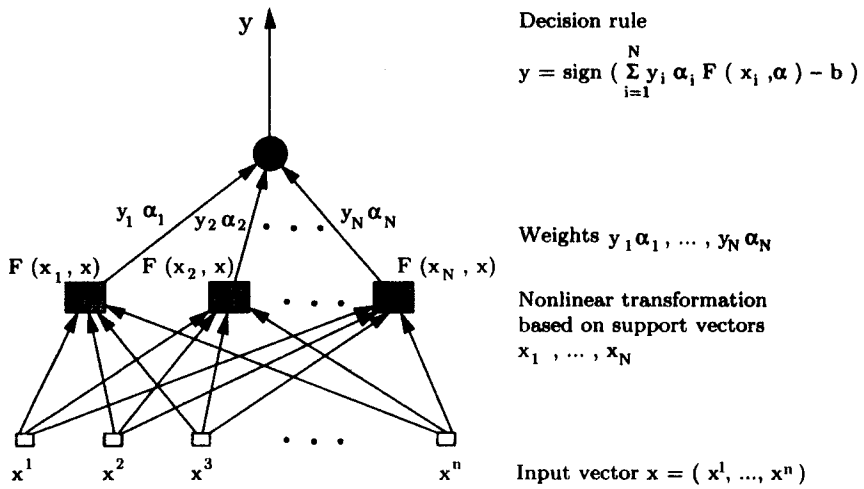

FIGURE 5.4. The two-layer SV machine is a compact realization of an Optimal hyperplane in the high-dimensional feature space Z.

where R is the radius of the smallest sphere that contains the vectors $\Psi(x)$, $|w|$ is the norm of the weights (we use canonical hyperplanes in feature space with respect to the vectors $z = \Psi(x_i)$ where x_i are the elements of the training data).

According to Theorem 5.1 (now applied in the feature space), k gives an estimate of the VC dimension of the set of functions S_k.

The SV machine separates without error the training data

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

and has a minimal norm $|w|$.

In other words, the SV machine separates the training data using functions from element S_k with the smallest estimate of the VC dimension.

Recall that in the feature space the equality

$$|w_0|^2 = \sum_{i}^{\ell} \alpha_i^0 \alpha_j^0 K(x_i, x_j) y_i y_j \tag{5.28}$$

holds true. To control the generalization ability of the machine (to minimize the probability of test errors) one has to construct the separating

hyperplane that minimizes the functional

$$\Phi(R, w_0, \ell) = \frac{R^2|w_0|^2}{\ell}. \tag{5.29}$$

Indeed, for separating hyperplanes the probability of test errors with probability $1 - \eta$ is bounded by the expression

$$\mathcal{E} = 4\frac{h(\ln\frac{2\ell}{h} + 1) - \ln\eta/4}{\ell}.$$

The right-hand side of $\mathcal{E}$ attains its minimum when h/ℓ is minimal. We estimate the minimum of h/ℓ by estimating h by $h_{est} = R^2|w_0|^2$. To estimate this functional it is sufficient to estimate $|w_0|^2$ (say by expression (5.28)) and estimate R^2 by finding

$$R^2 = R^2(K) = \min_{a} \max_{x_i} [K(x_i, x_i) + K(a, a) - 2K(x_i, a)]\,. \tag{5.30}$$

Polynomial Learning Machine

To construct polynomial decision rules of degree d, one can use the following function for convolution of the inner product:

$$K(x, x_i) = [(x \cdot x_i) + 1]^d. \tag{5.31}$$

This symmetric function satisfies the conditions of Theorem 5.3, therefore it describes a convolution of the inner product in the feature space that contains all products $x_i \cdot x_j \cdot x_k$ up to degree d. Using the described technique, one constructs a decision function of the form

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

which is a factorization of d-dimension polynomials in n-dimensional input space.

In spite of the very high dimensionality 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 can be low.

As described above to estimate the VC dimension of the element of the structure from which the decision function is chosen, one has only to estimate the radius R of the smallest sphere that contains the training data, and the norm of weights in feature space (Theorem 5.1).

Note that both the radius $R = R(d)$ and the norm of weights in the feature space depends on the degree of the polynomial.

This gives the opportunity to choose the best degree of polynomial for the given data.

To make a *local polynomial* approximation in the neighborhood of a point of interest x_0, let us consider the hard threshold neighborhood function (4.16). According to the theory of local algorithms, one chooses a ball with radius R_β around point x_0 in which ℓ_β elements of the training set fall, and then using only these training data, constructs the decision function that minimizes the probability of errors in the chosen neighborhood. The solution to this problem is a radius R_β that minimizes the functional

$$\Phi(R_\beta, w_0, \ell_\beta) = \frac{R_\beta^2 |w_0|^2}{\ell_\beta} \tag{5.32}$$

(the parameter $|w_0|$ depends on the chosen radius as well). This functional describes a trade-off between the chosen radius R_β, the value of the minimum of the norm $|w_0|$, and the number of training vectors ℓ_β that fall into radius R_β.

Radial Basis Function Machines

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

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} a_i K_\gamma(|x - x_i|) - b\right), \tag{5.33}$$

where $K_\gamma(|x - x_i|)$ depends on the distance $|x - x_i|$ between two vectors. For the theory of RBF machines see (Micchelli, 1986), (Powell, 1992).

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

$$K_\gamma(|x - x_i|) = \exp\{-\gamma |x - x_i|^2\}. \tag{5.34}$$

To construct the decision rule (5.33) 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 value 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.

The radial function can be chosen as a function for the convolution of the inner product for a SV machine. In this case, the SV machine will construct a function from the set (5.33). One can show (Aizerman, Braverman, and Rozonoer, 1964, 1970) that radial functions (5.34) satisfy the condition of Theorem 5.3.

In contrast to classical RBF methods, in the SV technique all four types of parameters are chosen to minimize the bound on probability of test error by controlling the parameters R, w_0 in the functional (5.29). By minimizing the functional (5.29) one determines

(i) N, the number of support vectors,

(ii) x_i, (the pre-images of) support vectors;

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

(iv) γ, the width parameter of the kernel-function.

Two-Layer Neural Networks

Finally, one can define two-layer neural networks by choosing kernels:

$$K(x, x_i) = S[v(x \cdot x_i) + c],$$

where $S(u)$ is a sigmoid function. In contrast to kernels for polynomial machines or for radial basis function machines that alway satisfy Mercer conditions, the sigmoid kernel $\tanh(vu + c)$, $|u| \leq 1$, satisfies Mercer conditions only for some values of parameters v, c. For these values of parameters one can construct SV machines implementing the rules:

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

Using the technique described above, the following are found automatically:

(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).

5.7 EXPERIMENTS WITH SV MACHINES

In the following we will present two types of experiments constructing the decision rules in the pattern recognition problem[8]:

[8]The experiments were conducted in the Adaptive System Research Department, AT&T Bell Laboratories.

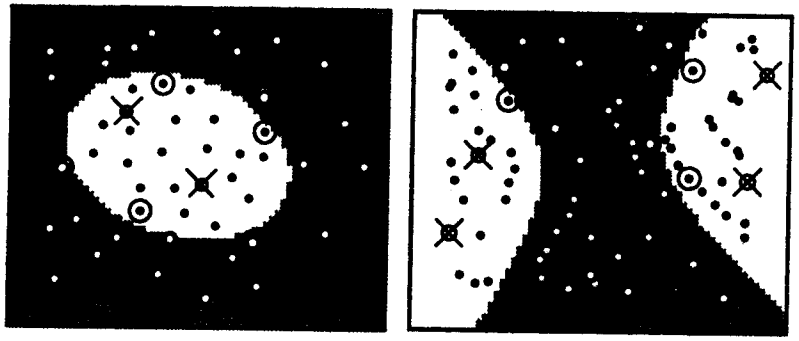

FIGURE 5.5. Two classes of vectors are represented in the picture by black and white balls. The decision boundaries were constructed using an inner product of polynomial type with $d = 2$. In the pictures the examples cannot be separated without errors; the errors are indicated by crosses and the support vectors by double circles.

(i) Experiments in the plane with artificial data that can be visualized, and

(ii) experiments with real-life data.

5.7.1 *Example in the Plane*

To demonstrate the SV technique we first give an artificial example (Fig. 5.5).

The two classes of vectors are represented in the picture by black and white balls. The decision boundaries were constructed using a inner product of polynomial type with $d = 2$. In the pictures the examples cannot be separated without errors; the errors are indicated by crosses and the support vectors by double circles.

Notice that in both examples the number of support vectors is small relative to the number of training data and that the number of training errors is minimal for polynomials of degree two.

5.7.2 *Handwritten Digit Recognition*

Since the first experiments of Rosenblatt, the interest in the problem of learning to recognize handwritten digits has remained strong. In the following we describe results of experiments on learning the recognition of handwritten digits using different SV machines. We also compare these results to results obtained by other classifiers. In these experiments, the U.S. Postal Service database (LeCun et *al.*, 1990) was used. It contains 7,300 training patterns and 2,000 test patterns collected from real-life zip–codes. The resolution of the database is 16×16 pixels, therefore the dimensionality

Classifier	Raw error%
Human performance	2.5
Decision tree, C4.5	16.2
Best two-layer neural network	5.9
Five-layer network (LeNet 1)	5.1

TABLE 5.1. Human performance and performance of the various learning machines, solving the problem of digit recognition on U.S. Postal Service data.

of the input space is 256. Figure 5.6 gives examples from this data-base.

Table 5.1 describes the performance of various classifiers, solving this problem.[9]

For constructing the decision rules three types of SV machines were used[10]:

(i) A polynomial machine with convolution function:

$$K(x, x_i) = \left(\frac{(x \cdot x_i)}{256}\right)^d, \quad d = 1, .., 7.$$

(ii) A radial basis function machine with convolution function:

$$K(x, x_i) = \exp\left\{-\frac{(x - x_i)^2}{256\sigma^2}\right\}.$$

(iii) A two layers neural network machine with convolution function:

$$K(x, x_i) = \tanh\left(\frac{b(x \cdot x_i)}{256} - c\right).$$

All machines constructed ten classifiers, each one separating one class from the rest. The ten class classification was done by choosing the class with the largest classifier output value.

The results of these experiments are given in Table 5.2. For different types of SV machines, Table 5.2 shows: the best parameters for the machines (column 2), the average (over one classifier) of the number of support vectors, and the performance of machine.

[9]The result of human performance was reported by J. Bromley and E. Säckinger; the result of C4.5 was obtained by C. Cortes; the result for the two layer neural net was obtained by B. Schölkopf; the results for the special purpose neural network architecture with five layers (LeNet 1), was obtained by Y. LeCun *et al.*

[10]The results were obtained by C. Burges, C. Cortes, and B. Schölkopf.

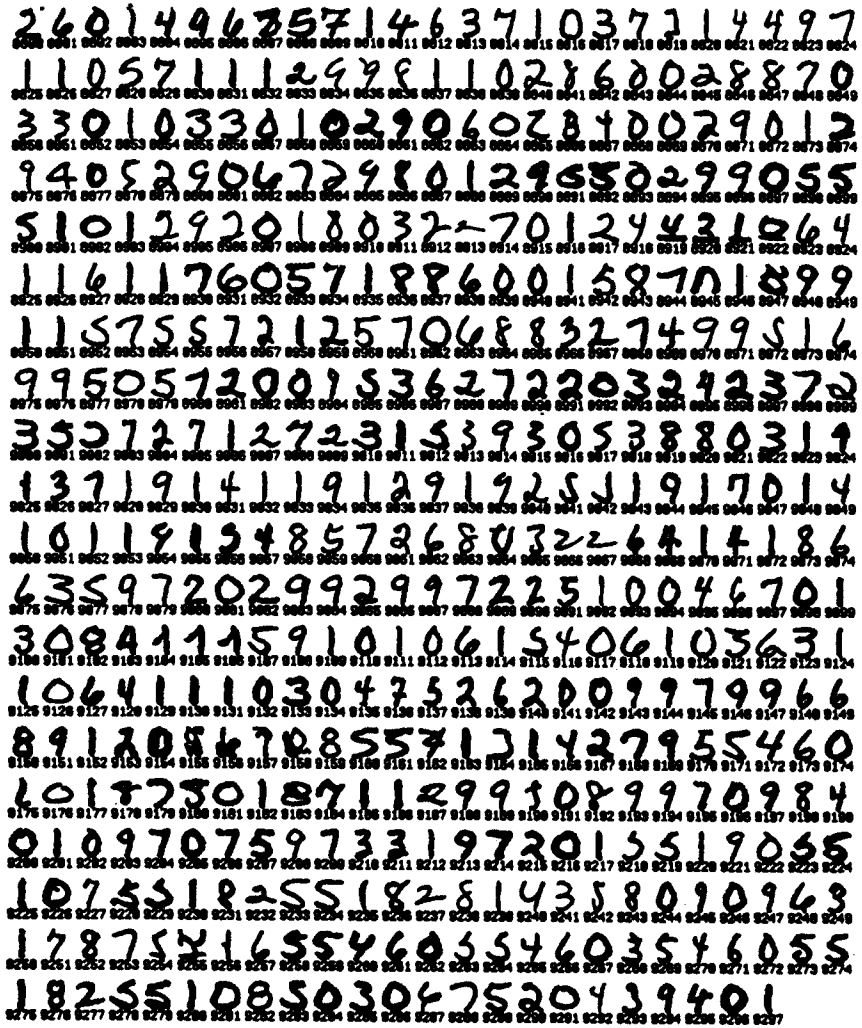

FIGURE 5.6. Examples of patterns (with labels) from the U.S. Postal Service database.

Type of SV classifier	Parameters of classifier	Number of support vectors	Raw error
Polynomials	d=3	274	4.0
RBF classifiers	$\sigma^2 = 0.3$	291	4.1
Neural network	$b = 2,\ c = 1$	254	4.2

TABLE 5.2. Results of digit recognition experiments with various SV machines using the U.S. Postal Service database. The number of support vectors means the average per classifier.

	Poly	RBF	NN	Common
total# of sup.vect.	1677	1727	1611	1377
% of common sup. vect.	82	80	85	100

TABLE 5.3. Total number (in ten classifiers) of support vectors for various SV machines and percentage of common support vectors.

Note that for this problem, all types of SV machines demonstrate approximately the same performance. This performance is better than the performance of any other type of learning machine solving the digit recognition problem by constructing the entire decision rules on the basis of the U.S. Postal Service database.[11]

In these experiments one important singularity was observed: different types of SV machines use approximately the same set of support vectors. The percentage of common support vectors for three different classifiers exceeded 80%.

Table 5.3 describes the total number of different support vectors for ten classifiers of different machines: Polynomial machine (Poly), Radial Basis Function machine (RBF), and Neural Network machine (NN). It shows also the number of common support vectors for all machines.

[11]Note that using a local approximation approach described in Section 5.7 (that does not construct entire decision rule but approximates the decision rule in any point of interest) one can obtain a better result: 3.3% error rate (L. Bottou and V Vapnik, 1992).

The best results for this database, 2.7% was obtained by P. Simard, Y. LeCun, and J. Denker without using any learning methods. They suggested a special method of elastic matching with 7200 templates using a smart concept of distance (so-called Tangent distance) that takes into account invariance with respect to small translations, rotations, distortions, and so on (P. Simard, Y. LeCun, and J. Denker, 1993).

	Poly	RBF	NN
Poly	100	84	94
RBF	87	100	88
NN	91	82	100

TABLE 5.4. Percentage of common (total) support vectors for two SV machines.

Table 5.4 describes the percentage of support vectors of the classifier given in the columns contained in the support vectors of the classifier given in the rows.

This fact, if it holds true for a wide class of real-life problems, is very important.

5.7.3 Some Important Details

In this subsection we give some important details on solving the digit recognition problem using a polynomial SV machine.

The training data are not linearly separable. The total number of misclassifications on the training set for linear rules is equal to 340 ($\approx$ 5% errors). For second degree polynomial classifiers the total number of misclassifications on the training set is down to four. These four mis-classified examples (with desired labels) are shown in Fig. 5.7.Starting with polynomials of degree three, the training data are separable.

Table 5.5 describes the results of experiments using decision polynomials (ten polynomials, one per classifier in one experiment) of various degrees. The number of support vectors shown in the table is a mean value per classifier.

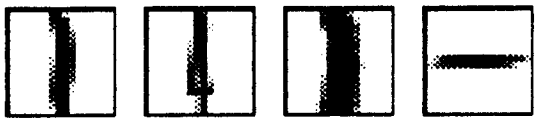

FIGURE 5.7. Labeled examples of training errors for the second degree polynomials.

degree of polynomial	dimensionality of feature space	support vectors	raw error
1	256	282	8.9
2	~ 33000	227	4.7
3	$\sim 1 \times 10^6$	274	4.0
4	$\sim 1 \times 10^9$	321	4.2
5	$\sim 1 \times 10^{12}$	374	4.3
6	$\sim 1 \times 10^{14}$	377	4.5
7	$\sim 1 \times 10^{16}$	422	4.5

TABLE 5.5. Results of experiments with polynomials of the different degrees.

Note that the number of support vectors increases slowly with the degree of the polynomials. The seventh degree polynomial has only 50% more support vectors than the third degree polynomial.[12]

The dimensionality of the feature space for a seventh degree polynomial is however 10^{10} times larger than the dimensionality of the feature space for a third degree polynomial classifier. Note that the performance does not change significantly with increasing dimensionality of the space — indicating no overfitting problems.

To choose the degree of the best polynomials for one specific classifier we estimate the VC dimension (using the estimate $[R^2A^2]$) for all constructed polynomials (from degree two up to degree seven) and choose the one with the smallest estimate of the VC dimension. In this way we found the ten best classifiers (with different degrees of polynomials) for the ten two-class problems. These estimates are shown on Fig. 5.8 where for all ten two-class decision rules, the estimated VC dimension, is plotted versus the degree of the polynomials.The question is:

Do the polynomials with the smallest estimate of the VC dimension provide the best classifier?

To answer this question we constructed Table 5.6 which describes the performance of the classifiers for each degree of polynomial.

Each row describes one two–class classifier separating one *digit* (stated in the first column) from the all ather digits.

The remaining columns contain:

deg.: the degree of the polynomial as chosen (from two up to seven) by the described procedure,

[12]The relatively high number of support vectors for the linear separator is due to nonseparability: the number 282 includes both support vectors and missclassified data.

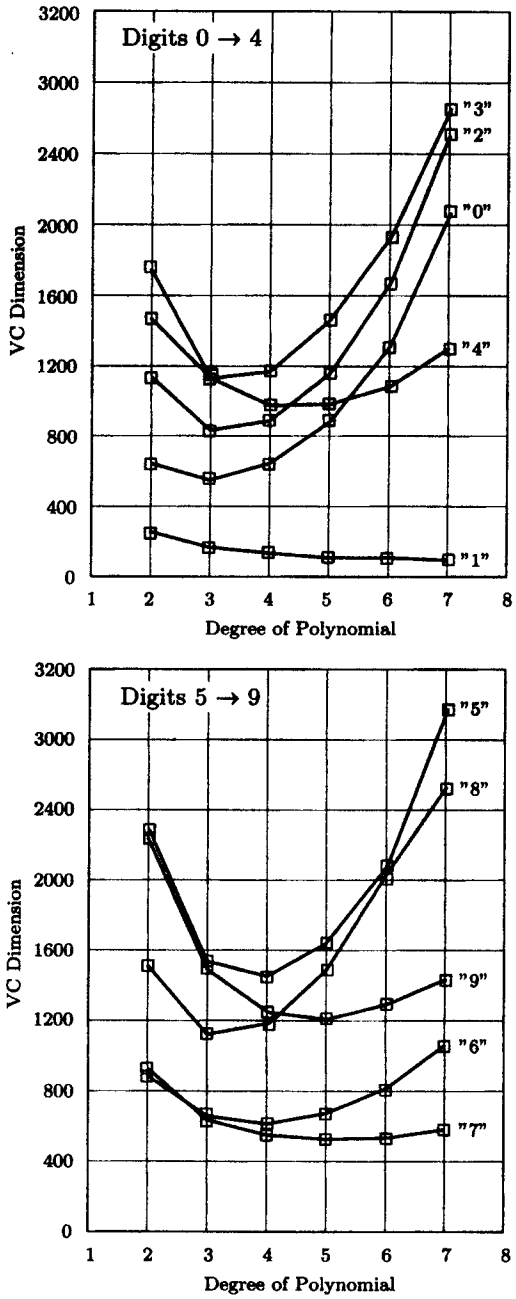

FIGURE 5.8. The estimate of the VC dimension of the best element of the structure (defined on the set of canonical hyperplanes in the corresponding feature space) versus the degree of polynomial for various two-class digit recognition problems (denoted digit versus the rest).

	Chosen classifier			Number of test errors						
Digit	deg.	dim.	$h_{est.}$	1	2	3	4	5	6	7
0	3	$\sim 10^6$	530	36	14	**11**	11	11	12	17
1	7	$\sim 10^{16}$	101	17	15	14	11	10	10	**10**
2	3	$\sim 10^6$	842	53	32	**28**	26	28	27	32
3	3	$\sim 10^6$	1157	57	25	**22**	22	22	22	23
4	4	$\sim 10^9$	962	50	32	32	**30**	30	29	33
5	3	$\sim 10^6$	1090	37	20	**22**	24	24	26	28
6	4	$\sim 10^9$	626	23	12	12	**15**	17	17	19
7	5	$\sim 10^{12}$	530	25	15	12	10	**11**	13	14
8	4	$\sim 10^9$	1445	71	33	28	**24**	28	32	34
9	5	$\sim 10^{12}$	1226	51	18	15	11	**11**	12	15

TABLE 5.6. Experiments on choosing the best degree of polynomial.

dim.: the dimensionality of the corresponding feature space, which is also the maximum possible VC dimension for linear classifiers in that space,

$h_{est.}$: the VC dimension estimate for the chosen polynomial, (which is much smaller than the number of free parameters),

Number of test errors: the number of test errors, using the constructed polynomial of corresponding degree; the boxes show the number of errors for the chosen polynomial.

Thus, Table 5.5 shows that for the SV polynomial machine there are no overfitting problems with increasing degree of polynomials, while Table 5.6 shows that even in situations where the difference between the best and the worst solutions is small (for polynomials starting from degree two up to degree seven), the theory gives a method for approximating the best solutions (finding the best degree of the polynomial).

Note also that Table 5.6 demonstrates that the problem is essentially nonlinear. The difference in the number of errors between the best polynomial classifier and the linear classifier can be as much as a factor of four (for digit 9).

5.8 REMARKS ON SV MACHINES

The quality of any learning machine is characterized by three main components: