CHAPTER 6

Support Vector Machines

6.1 INTRODUCTION

In Chapter 4 we studied multilayer perceptrons trained with the back-propagation algorithm. In Chapter 5 we studied another class of layered feedforward networks, radial-basis function networks. Both of these neural networks are universal approximators in their own ways. In this chapter we discuss another category of universal feedforward networks, known as *support vector machines (SVM),* pioneered by Vapnik (Boser, Guyon, and Vapnik, 1992; Cortes and Vapnik, 1995; Vapnik, 1995, 1998). Like multilayer perceptrons and radial-basis function networks, support vector machines can be used for pattern classification and nonlinear regression.

Basically, the support vector machine is a *linear machine* with some very nice properties. To explain how it works, it is perhaps easiest to start with the case of separable patterns that could arise in the context of pattern classification. In this context, the main idea of a support vector machine is to construct a hyperplane as the decision surface in such a way that the margin of separation between positive and negative examples is maximized. The machine achieves this desirable property by following a principled approach rooted in the statistical learning theory that is discussed in Chapter 2. More precisely, the support vector machine is an approximate implementation of the *method of structural risk minimization.* This induction principle is based on the fact that the error rate of a learning machine on test data (i.e., the generalization error rate) is bounded by the sum of the training-error rate and a term that depends on the *Vapnik–Chervonenkis (VC) dimension;* in the case of separable patterns, a support vector machine produces a value of zero for the first term and minimizes the second term. Accordingly, the support vector machine can provide a good generalization performance on pattern classification problems despite the fact that it *does not* incorporate problem-domain knowledge. This attribute is unique to support vector machines.

A notion that is central to the construction of the support vector learning algorithm is the inner-product kernel between a "support vector" $\mathbf{x}_i$ and the vector $\mathbf{x}$ drawn from the input space. The support vectors consist of a small subset of the training data extracted by the algorithm. Depending on how this inner-product kernel is generated,

we may construct different learning machines characterized by nonlinear decision surfaces of their own. In particular, we may use the support vector learning algorithm to construct the following three types of learning machines (among others):

- Polynomial learning machines
- Radial-basis function networks
- Two-layer perceptrons (i.e., with a single hidden layer)

That is, for each of these feedforward networks we may use the support vector learning algorithm to implement the learning process using a given set of training data, automatically determining the required number of hidden units. Stated in another way: Whereas the back-propagation algorithm is devised specifically to train a multilayer perceptron, the support vector learning algorithm is of a more generic nature because it has wider applicability.

Organization of the Chapter

The main body of the chapter is organized in three parts. In the first part we describe the basic ideas behind a support vector machine. Specifically, in Section 6.2 we discuss the construction of optimal hyperplanes for the simple case of linearly separable patterns. This is followed by considering the more difficult case of nonseparable patterns in Section 6.3.

In so doing, we pave the way for the second part of the chapter, which presents a detailed discussion of the support vector machine for solving pattern-recognition tasks. This is done in Section 6.4. In Section 6.5 we revisit the XOR problem to illustrate the construction of a support vector machine. In Section 6.6 we revisit the computer experiment on pattern classification that was studied in Chapters 4 and 5, thereby providing a comparative evaluation of support vector machines with multilayer perceptrons trained on the back-propagation algorithm and standard radial-basis function networks.

The last part of the chapter deals with the nonlinear regression problem. In Section 6.7 we describe a loss function that is well suited for such a problem. Then in Section 6.8 we discuss the construction of a support vector machine for nonlinear regression.

The chapter concludes with some final remarks in Section 6.9.

6.2 OPTIMAL HYPERPLANE FOR LINEARLY SEPARABLE PATTERNS

Consider the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, where $\mathbf{x}_i$ *is* the input pattern for the ith example and d_i is the corresponding desired response (target output). To begin with, we assume that the pattern (class) represented by the subset $d_i = +1$ and the pattern represented by the subset $d_i = -1$ are "linearly separable." The equation of a decision surface in the form of a hyperplane that does the separation is

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{6.1}$$

where $\mathbf{x}$ is an input vector, $\mathbf{w}$ is an adjustable weight vector, and b is a bias. We may thus write

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_i + b \geq 0 \qquad & \text{for } d_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b < 0 \qquad & \text{for } d_i = -1 \end{aligned} \tag{6.2}$$

The assumption of linearly separable patterns is made here to explain the basic idea behind a support vector machine in a rather simple setting; this assumption will be relaxed in Section 6.3.

For a given weight vector $\mathbf{w}$ and bias b, the separation between the hyperplane defined in Eq. (6.1) and the closest data point is called the *margin of separation*, denoted by ρ. The goal of a support vector machine is to find the particular hyperplane for which the margin of separation ρ is maximized. Under this condition, the decision surface is referred to as the *optimal hyperplane*. Figure 6.1 illustrates the geometric construction of an optimal hyperplane for a two-dimensional input space.

Let $\mathbf{w}_o$ and b_o denote the optimum values of the weight vector and bias, respectively. Correspondingly, the *optimal hyperplane*, representing a multidimensional linear decision surface in the input space, is defined by

$$\mathbf{w}_o^T\mathbf{x} + b_o = 0 \tag{6.3}$$

which is a rewrite of Eq. (6.1). The discriminant function

$$g(\mathbf{x}) = \mathbf{w}_o^T\mathbf{x} + b_o \tag{6.4}$$

gives an algebraic measure of the *distance* from $\mathbf{x}$ to the optimal hyperplane (Duda and Hart, 1973). Perhaps the easiest way to see this is to express $\mathbf{x}$ as

$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}_o}{\|\mathbf{w}_o\|}$$

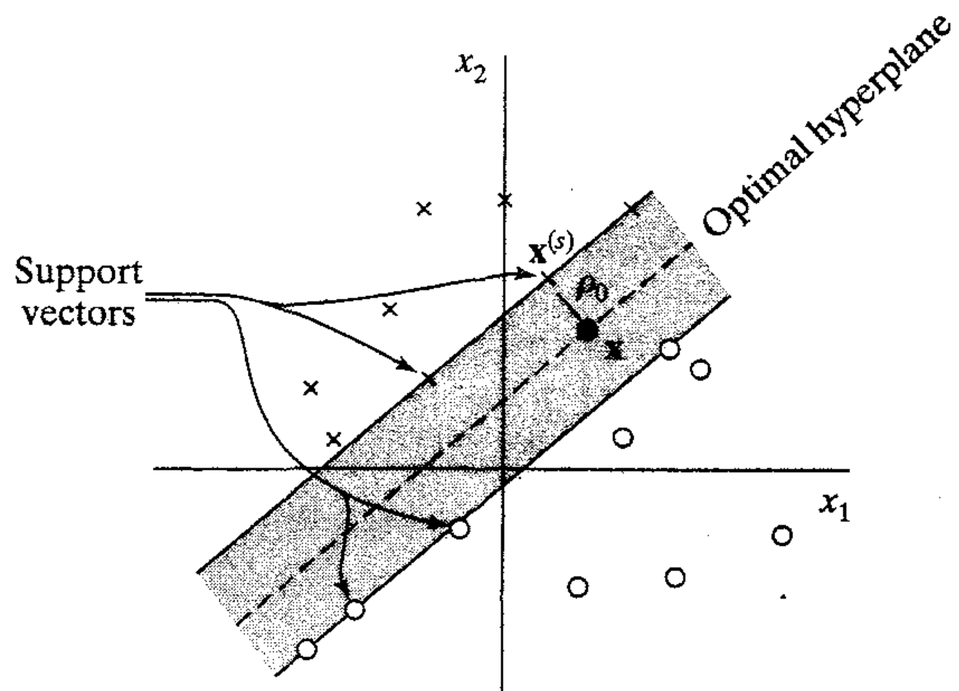

FIGURE 6.1 Illustration of the idea of an optimal hyperplane for linearly separable patterns.

where $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto the optimal hyperplane, and r is the desired algebraic distance; r is positive if $\mathbf{x}$ is on the positive side of the optimal hyperplane and negative if $\mathbf{x}$ is on the negative side. Since, by definition, $g(\mathbf{x}_p) = 0$, it follows that

$$g(\mathbf{x}) = \mathbf{w}_o^T\mathbf{x} + b_o = r\|\mathbf{w}_o\|$$

or

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}_o\|} \tag{6.5}$$

In particular, the distance from the origin (i.e., $\mathbf{x} = \mathbf{0}$) to the optimal hyperplane is given by $b_o/\|\mathbf{w}_o\|$. If $b_o > 0$, the origin is on the positive side of the optimal hyperplane; if $b_o < 0$, it is on the negative side. If $b_o = 0$, the optimal hyperplane passes through the origin. A geometric interpretation of these algebraic results is given in Fig. 6.2.

The issue at hand is to find the parameters $\mathbf{w}_o$ and b_o for the optimal hyperplane, given the training set $\mathcal{T} = \{(\mathbf{x}_i, d_i)\}$. In light of the results portrayed in Fig. 6.2, we see that the pair $(\mathbf{w}_o, b_o)$ must satisfy the constraint:

$$\begin{aligned} \mathbf{w}_o^T\mathbf{x}_i + b_o &\geq 1 \qquad \text{for } d_i = +1 \\ \mathbf{w}_o^T\mathbf{x}_i + b_o &\leq -1 \qquad \text{for } d_i = -1 \end{aligned} \tag{6.6}$$

Note that if Eq. (6.2) holds, that is, the patterns are linearly separable, we can always rescale $\mathbf{w}_o$ and b_o such that Eq. (6.6) holds; this scaling operation leaves Eq. (6.3) unaffected.

The particular data points $(\mathbf{x}_i, d_i)$ for which the first or second line of Eq. (6.6) is satisfied with the equality sign are called *support vectors*, hence the name "support vector machine." These vectors play a prominent role in the operation of this class of learning machines. In conceptual terms, the support vectors are those data points that lie closest to the decision surface and are therefore the most difficult to classify. As such, they have a direct bearing on the optimum location of the decision surface.

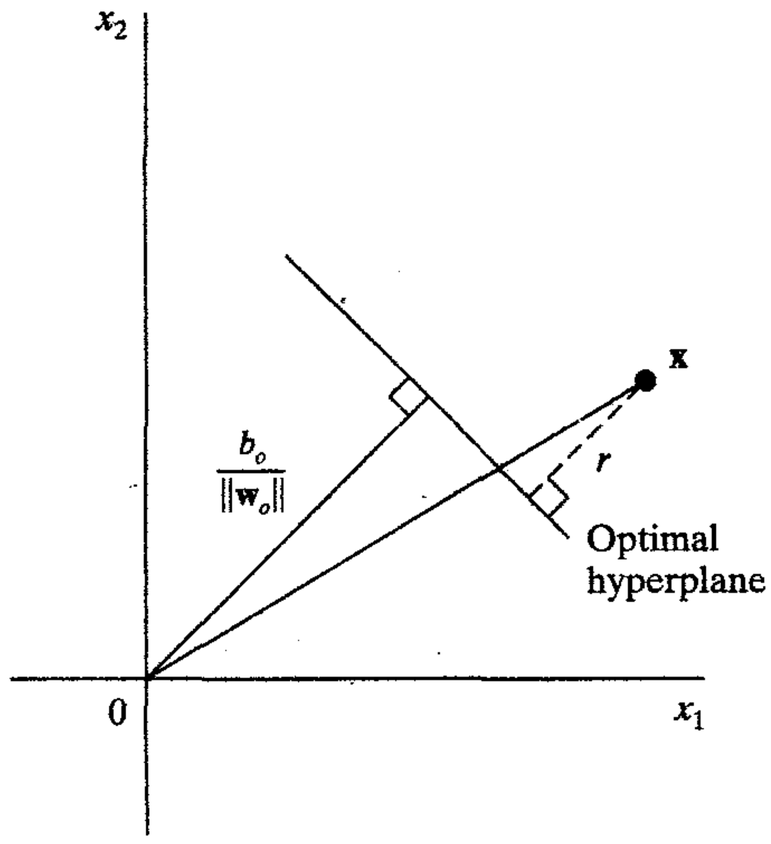

FIGURE 6.2 Geometric interpretation of algebraic distances of points to the optimal hyperplane for a two-dimensional case.

Consider a support vector $\mathbf{x}^{(s)}$ for which $d^{(s)} = +1$. Then by definition, we have

$$g(\mathbf{x}^{(s)}) = \mathbf{w}_o^T \mathbf{x}^{(s)} \mp b_o = \mp 1 \qquad \text{for } d^{(s)} = \mp 1 \tag{6.7}$$

From Eq. (6.5) the *algebraic distance* from the support vector $\mathbf{x}^{(s)}$ to the optimal hyperplane is

$$r = \frac{g(\mathbf{x}^{(s)})}{\|\mathbf{w}_o\|}$$

$$= \begin{cases} \dfrac{1}{\|\mathbf{w}_o\|} & \text{if } d^{(s)} = +1 \\ -\dfrac{1}{\|\mathbf{w}_o\|} & \text{if } d^{(s)} = -1 \end{cases} \tag{6.8}$$

where the plus sign indicates that $\mathbf{x}^{(s)}$ lies on the positive side of the optimal hyperplane and the minus sign indicates that $\mathbf{x}^{(s)}$ lies on the negative side of the optimal hyperplane. Let ρ denote the optimum value of the *margin of separation* between the two classes that constitute the training set $\mathcal{T}$. Then, from Eq. (6.8) it follows that

$$\rho = 2r$$

$$= \frac{2}{\|\mathbf{w}_o\|} \tag{6.9}$$

Equation (6.9) states that maximizing the margin of separation between classes is equivalent to minimizing the Euclidean norm of the weight vector $\mathbf{w}$.

In summary, the optimal hyperplane defined by Eq. (6.3) is *unique* in the sense that the optimum weight vector $\mathbf{w}_o$ provides the maximum possible separation between positive and negative examples. This optimum condition is attained by minimizing the Euclidean norm of the weight vector $\mathbf{w}$.

Quadratic Optimization for Finding the Optimal Hyperplane

Our goal is to develop a computationally efficient procedure for using the training sample $\mathcal{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N$ to find the optimal hyperplane, subject to the constraint

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \qquad \text{for } i = 1, 2, \ldots, N \tag{6.10}$$

This constraint combines the two lines of Eq. (6.6) with $\mathbf{w}$ used in place of $\mathbf{w}_o$.

The constrained optimization problem that we have to solve may now be stated as:

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, *find the optimum values of the weight vector* $\mathbf{w}$ *and bias* b *such that they satisfy the constraints*

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \qquad \text{for } i = 1, 2, \ldots, N$$

and the weight vector $\mathbf{w}$ *minimizes the cost function:*

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

The scaling factor 1/2 is included here for convenience of presentation. This constrained optimization problem is called the *primal problem*. It is characterized as follows:

- The cost function $\Phi(\mathbf{w})$ is a *convex* function[1] of $\mathbf{w}$.
- The constraints are *linear* in $\mathbf{w}$.

Accordingly, we may solve the constrained optimization problem using the *method of Lagrange multipliers* (Bertsekas, 1995).

First, we construct the *Lagrangian function*:

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right] \tag{6.11}$$

where the auxiliary nonnegative variables α_i are called *Lagrange multipliers*. The solution to the constrained optimization problem is determined by the *saddle point* of the Lagrangian function $J(\mathbf{w}, b, \alpha)$, which has to be *minimized* with respect to $\mathbf{w}$ and b; it also has to be *maximized* with respect to α. Thus, differentiating $J(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b and setting the results equal to zero, we get the following two *conditions of optimality*:

Condition 1: $$\frac{\partial J(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = \mathbf{0}$$

Condition 2: $$\frac{\partial J(\mathbf{w}, b, \alpha)}{\partial b} = 0$$

Application of optimality condition 1 to the Lagrangian function of Eq. (6.11) yields (after rearrangement of terms)

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \mathbf{x}_i \tag{6.12}$$

Application of optimality condition 2 to the Lagrangian function of Eq. (6.11) yields

$$\sum_{i=1}^{N} \alpha_i d_i = \mathbf{0} \tag{6.13}$$

The solution vector $\mathbf{w}$ is defined in terms of an expansion that involves the N training examples. Note, however, that although this solution is unique by virtue of the convexity of the Lagrangian, the same cannot be said about the Lagrange coefficients, α_i.

It is also important to note that at the saddle point, for each Lagrange multiplier α_i, the product of that multiplier with its corresponding constraint vanishes, as shown by

$$\alpha_i \left[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right] = 0 \qquad \text{for } i = 1, 2, \ldots, N \tag{6.14}$$

Therefore, only those multipliers exactly meeting Eq. (6.14) can assume *nonzero* values. This property follows from the *Kuhn–Tucker conditions* of optimization theory (Fletcher, 1987; Bertsekas, 1995).

As noted earlier, the primal problem deals with a convex cost function and linear constraints. Given such a constrained optimization problem, it is possible to construct another problem called the *dual problem*. This second problem has the same optimal

value as the primal problem, but with the Lagrange multipliers providing the optimal solution. In particular, we may state the following *duality theorem* (Bertsekas, 1995):

(a) If the primal problem has an optimal solution, the dual problem also has an optimal solution, and the corresponding optimal values are equal.
(b) In order for $\mathbf{w}_o$ to be an optimal primal solution and α_o to be an optimal dual solution, it is necessary and sufficient that $\mathbf{w}_o$ is feasible for the primal problem, and

$$\Phi(\mathbf{w}_o) = J(\mathbf{w}_o, b_o, \alpha_o) = \min_{\mathbf{w}} J(\mathbf{w}, b_o, \alpha_o)$$

To postulate the dual problem for our primal problem, we first expand Eq. (6.11), term by term, as follows:

$$J(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i d_i \mathbf{w}^T\mathbf{x}_i - b\sum_{i=1}^{N} \alpha_i d_i + \sum_{i=1}^{N} \alpha_i \tag{6.15}$$

The third term on the right-hand side of Eq. (6.15) is zero by virtue of the optimality condition of Eq. (6.13). Furthermore, from Eq. (6.12) we have

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \mathbf{w}^T\mathbf{x}_i = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j$$

Accordingly, setting the objective function $J(\mathbf{w}, b, \alpha) = Q(\alpha)$, we may reformulate Eq. (6.15) as

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j \tag{6.16}$$

where the α_i are nonnegative.

We may now state the dual problem:

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, find the Lagrange multipliers $\{\alpha_i\}_{i=1}^N$ that maximize the objective function

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j$$

subject to the constraints

(1) $\sum_{i=1}^{N} \alpha_i d_i = 0$

(2) $\alpha_i \geq 0 \qquad$ for $i = 1, 2, \ldots, N$

Note that the dual problem is cast entirely in terms of the training data. Moreover, the function $Q(\alpha)$ to be maximized depends *only* on the input patterns in the form of a set of dot products, $\{\mathbf{x}_i^T\mathbf{x}_j\}_{(i,j)=1}^N$.

Having determined the optimum Lagrange multipliers, denoted by $\alpha_{o,i}$, we may compute the optimum weight vector $\mathbf{w}_o$ using Eq. (6.12) and so write

$$\mathbf{w}_o = \sum_{i=1}^{N} \alpha_{o,i} d_i \mathbf{x}_i \tag{6.17}$$

To compute the optimum bias b_o, we may use the $\mathbf{w}_o$ thus obtained and take advantage of Eq. (6.7) pertaining to a positive support vector, and thus write

$$b_o = 1 - \mathbf{w}_o^T\mathbf{x}^{(s)} \qquad \text{for } d^{(s)} = 1 \tag{6.18}$$

Statistical Properties of the Optimal Hyperplane

From the statistical learning theory presented in Chapter 2, we recall that the VC dimension of a learning machine determines the way in which a nested structure of approximating functions should be used. We also recall that the VC dimension of a set of separating hyperplanes in a space of dimensionality m is equal to $m + 1$. However, in order to apply the method of structural risk minimization described in Chapter 2 we need to construct a set of separating hyperplanes of varying VC dimension such that the empirical risk (i.e., the training classification error) and the VC dimension are both minimized at the same time. In a support vector machine a structure is imposed on the set of separating hyperplanes by constraining the Euclidean norm of the weight vector $\mathbf{w}$. Specifically, we may state the following theorem (Vapnik, 1995, 1998):

> Let D denote the diameter of the smallest ball containing all the input vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. The set of optimal hyperplanes described by the equation
>
> $$\mathbf{w}_o^T\mathbf{x} + b_o = 0$$
>
> has a VC dimension h bounded from above as
>
> $$h \leq \min\left\{\left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0\right\} + 1 \tag{6.19}$$
>
> where the ceiling sign $\lceil \cdot \rceil$ means the smallest integer greater than or equal to the number enclosed within, ρ is the margin of separation equal to $2/\|\mathbf{w}_o\|$, and m_0 is the dimensionality of the input space.

This theorem tells us that we may exercise control over the VC dimension (i.e., complexity) of the optimal hyperplane, independently of the dimensionality m_0 of the input space, by properly choosing the margin of separation ρ.

Suppose then we have a nested structure described in terms of the separating hyperplanes as follows:

$$S_k = \{\mathbf{w}^T\mathbf{x} + b : \|\mathbf{w}\|^2 \leq c_k\}, \qquad k = 1, 2, \ldots \tag{6.20}$$

By virtue of the upper bound on the VC dimension h defined in Eq. (6.19), the nested structure described in Eq. (6.20) may be reformulated in terms of the margin of separation in the equivalent form

$$S_k = \left\{\left\lceil \frac{r^2}{\rho^2} \right\rceil + 1 : \rho^2 \geq a_k\right\}, \qquad k = 1, 2, \ldots \tag{6.21}$$

The a_k and c_k are constants.

From Chapter 2 we also recall that in order to achieve a good generalization capability, we should select the particular structure with the smallest VC dimension and training error, in accordance with the principle of structural risk minimization. From Eqs. (6.19) and (6.21) we see that this requirement can be satisfied by using the optimal hyperplane (i.e., the separating hyperplane with the largest margin of separation ρ). Equivalently, in light of Eq. (6.9), we should use the optimum weight vector $\mathbf{w}_o$ having the minimum Euclidean norm. Thus, the choice of the optimal hyperplane as the decision surface for a set of linearly separable patterns is not only intuitively satisfying but also in complete fulfillment of the principle of structural risk minimization of a support vector machine.

6.3 OPTIMAL HYPERPLANE FOR NONSEPARABLE PATTERNS

The discussion thus far has focused on linearly separable patterns. In this section we consider the more difficult case of nonseparable patterns. Given such a set of training data, it is not possible to construct a separating hyperplane without encountering classification errors. Nevertheless, we would like to find an optimal hyperplane that minimizes the probability of classification error, averaged over the training set.

The margin of separation between classes is said to be *soft* if a data point $(\mathbf{x}_i, d_i)$ violates the following condition (see Eq. (6.10)):

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq +1, \qquad i = 1, 2, \ldots, N$$

This violation can arise in one of two ways:

- The data point $(\mathbf{x}_i, d_i)$ falls inside the region of separation but on the right side of the decision surface, as illustrated in Fig. 6.3a.
- The data point $(\mathbf{x}_i, d_i)$ falls on the wrong side of the decision surface, as illustrated in Fig. 6.3b.

Note that we have correct classification in case 1, but misclassification in case 2.

To set the stage for a formal treatment of nonseparable data points, we introduce a new set of nonnegative scalar variables, $\{\xi_i\}_{i=1}^N$, into the definition of the separating hyperplane (i.e., decision surface) as shown here:

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \qquad i = 1, 2, \ldots, N \tag{6.22}$$

The ξ_i are called *slack variables*; they measure the deviation of a data point from the ideal condition of pattern separability. For $0 \leq \xi_i \leq 1$, the data point falls inside the region of separation but on the right side of the decision surface, as illustrated in Fig. 6.3a. For $\xi_i > 1$, it falls on the wrong side of the separating hyperplane, as illustrated in Fig. 6.3b. The support vectors are those particular data points that satisfy Eq. (6.22)

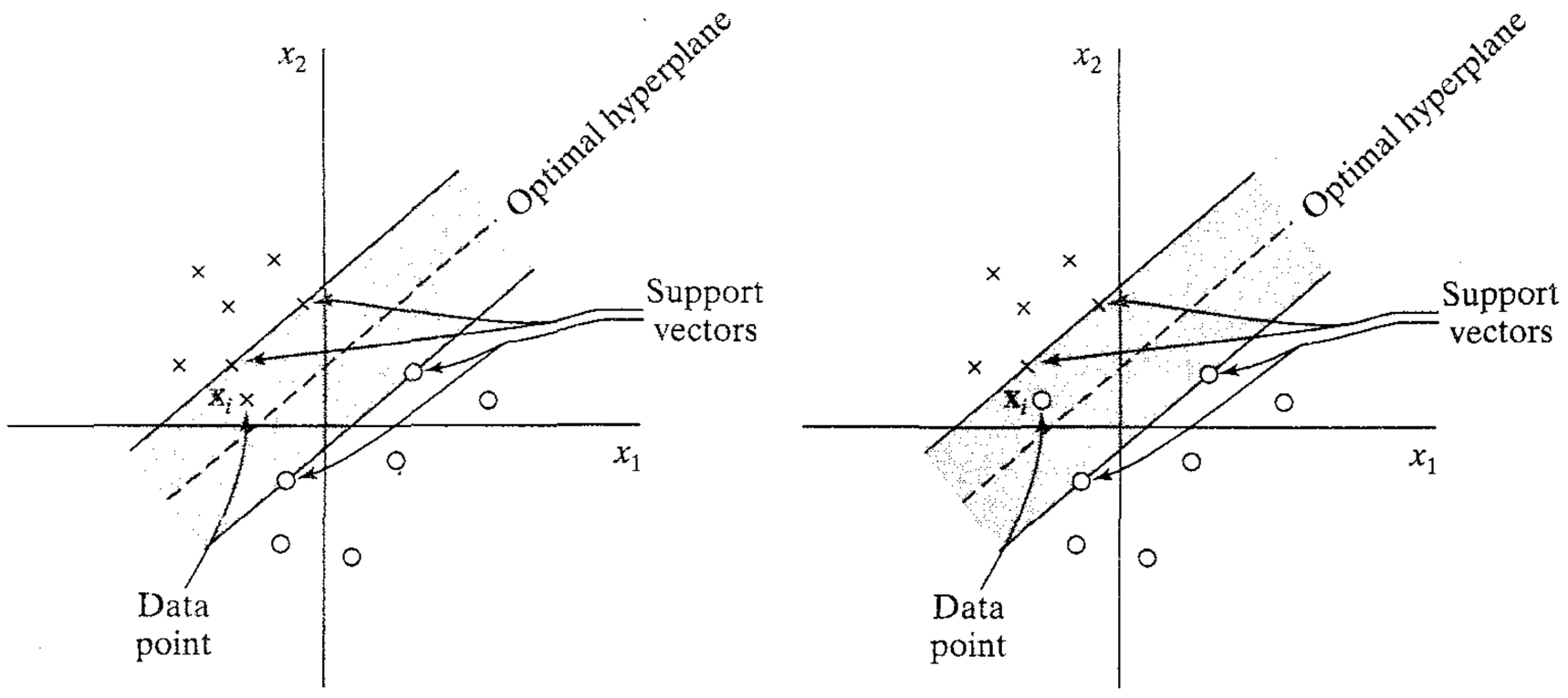

FIGURE 6.3 (a) Data point $\mathbf{x}_i$ (belonging to class $\mathscr{C}_1$) falls inside the region of separation, but on the right side of the decision surface. (b) Data point $\mathbf{x}_i$ (belonging to class $\mathscr{C}_2$) falls on the wrong side of the decision surface.

precisely even if $\xi_i > 0$. Note that if an example with $\xi_i > 0$ is left out of the training set, the decision surface would change. The support vectors are thus defined in exactly the same way for both linearly separable and nonseparable cases.

Our goal is to find a separating hyperplane for which the misclassification error, averaged on the training set, is minimized. We may do this by minimizing the functional

$$\Phi(\xi) = \sum_{i=1}^{N} I(\xi_i - 1)$$

with respect to the weight vector $\mathbf{w}$, subject to the constraint described in Eq. (6.22) and the constraint on $\|\mathbf{w}\|^2$. The function $I(\xi)$ is an *indicator function*, defined by

$$I(\xi) = \begin{cases} 0 & \text{if } \xi \leq 0 \\ 1 & \text{if } \xi > 0 \end{cases}$$

Unfortunately, minimization of $\Phi(\xi)$ with respect to $\mathbf{w}$ is a nonconvex optimization problem that is NP-complete.[2]

To make the optimization problem mathematically tractable, we approximate the functional $\Phi(\xi)$ by writing

$$\Phi(\xi) = \sum_{i=1}^{N} \xi_i$$

Moreover, we simplify the computation by formulating the functional to be minimized with respect to the weight vector $\mathbf{w}$ as follows:

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N} \xi_i \tag{6.23}$$

As before, minimizing the first term in Eq. (6.23) is related to minimizing the VC dimension of the support vector machine. As for the second term $\sum_i \xi_i$, it is an upper bound on the number of test errors. Formulation of the cost function $\Phi(\mathbf{w}, \xi)$ in Eq. (6.23) is therefore in perfect accord with the principle of structural risk minimization.

The parameter C controls the tradeoff between complexity of the machine and the number of nonseparable points; it may therefore be viewed as a form of a "regularization" parameter. The parameter C has to be selected by the user. This can be done in one of two ways:

- The parameter C is determined *experimentally* via the standard use of a training/(validation) test set, which is a crude form of resampling.
- It is determined *analytically* by estimating the VC dimension via Eq. (6.19) and then by using bounds on the generalization performance of the machine based on the VC dimension.

In any event, the functional $\Phi(\mathbf{w}, \xi)$ is optimized with respect to $\mathbf{w}$ and $\{\xi_i\}_{i=1}^N$, subject to the constraint described in Eq. (6.22), and $\xi_i \geq 0$. In so doing, the squared norm of $\mathbf{w}$ is treated as a quantity to be jointly minimized with respect to the nonseparable points rather than as a constraint imposed on the minimization of the number of nonseparable points.

The optimization problem for nonseparable patterns just stated, includes the optimization problem for linearly separable patterns as a special case. Specifically, setting $\xi_i = 0$ for all i in both Eqs. (6.22) and (6.23) reduces them to the corresponding forms for the linearly separable case.

We may now formally state the primal problem for the nonseparable case as:

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, *find the optimum values of the weight vector* $\mathbf{w}$ *and bias* b *such that they satisfy the constraint*

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \qquad \text{for } i = 1, 2, \ldots, N$$

$$\xi_i \geq 0 \quad \text{for all } i$$

and such that the weight vector $\mathbf{w}$ *and the slack variables* ξ_i *minimize the cost functional*

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

where C *is a user-specified positive parameter.*

Using the method of Lagrange multipliers and proceeding in a manner similar to that described in Section 6.2, we may formulate the dual problem for nonseparable patterns as (see Problem 6.3):

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, *find the Lagrange multipliers* $\{\alpha_i\}_{i=1}^N$ *that maximize the objective function*

$$Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j$$

subject to the constraints

$$(1)\quad \sum_{i=1}^{N}\alpha_i d_i = 0$$

$$(2)\quad 0 \leq \alpha_i \leq C \qquad \text{for } i = 1, 2, \ldots, N$$

where C *is a user-specified positive parameter.*

Note that neither the slack variables ξ_i nor their Lagrange multipliers appear in the dual problem. The dual problem for the case of nonseparable patterns is thus similar to that for the simple case of linearly separable patterns except for a minor but important difference. The objective function $Q(\alpha)$ to be maximized is the same in both cases. The nonseparable case differs from the separable case in that the constraint $\alpha_i \geq 0$ is replaced with the more stringent constraint $0 \leq \alpha_i \leq C$. Except for this modification, the constrained optimization for the nonseparable case and computations of the optimum values of the weight vector $\mathbf{w}$ and bias b proceed in the same way as in the linearly separable case. Note also that the support vectors are defined in exactly the same way as before.

The optimum solution for the weight vector $\mathbf{w}$ is given by

$$\mathbf{w}_o = \sum_{i=1}^{N_S}\alpha_{o,i}d_i\mathbf{x}_i \tag{6.24}$$

where N_S is the number of support vectors. The determination of the optimum values of the bias also follows a procedure similar to that described before. Specifically, the Kuhn–Tucker conditions are now defined by

$$\alpha_i\left[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i\right] = 0, \qquad i = 1, 2, \ldots, N \tag{6.25}$$

and

$$\mu_i\xi_i = 0, \qquad i = 1, 2, \ldots, N \tag{6.26}$$

Equation (6.25) is a rewrite of Eq. (6.14) except for the replacement of the unity term by $(1 - \xi_i)$. As for Eq. (6.26), the μ_i are Lagrange multipliers that have been introduced to enforce the nonnegativity of the slack variables ξ_i for all i. At the saddle point the derivative of the Lagrangian function for the primal problem with respect to the slack variable ξ_i is zero, the evaluation of which yields

$$\alpha_i + \mu_i = C \tag{6.27}$$

By combining Eqs. (6.26) and (6.27), we see that

$$\xi_i = 0 \quad \text{if} \quad \alpha_i < C \tag{6.28}$$

We may determine the optimum bias b_o by taking any data point $(\mathbf{x}_i, d_i)$ in the training set for which we have $0 < \alpha_{o,i} < C$ and therefore $\xi_i = 0$, and using that data point in Eq. (6.25). However, from a numerical perspective it is better to take the mean value of b_o resulting from all such data points in the training sample (Burges, 1998).

6.4 HOW TO BUILD A SUPPORT VECTOR MACHINE FOR PATTERN RECOGNITION

With the material on how to find the optimal hyperplane for nonseparable patterns at hand, we are now in a position to formally describe the construction of a support vector machine for a pattern-recognition task.

Basically, the idea of a support vector machine[3] hinges on two mathematical operations summarized here and illustrated in Fig. 6.4:

1. Nonlinear mapping of an input vector into a high-dimensional *feature space* that is hidden from both the input and output.
2. Construction of an optimal hyperplane for separating the features discovered in step 1.

The rationale for each of these two operations is explained in what follows.

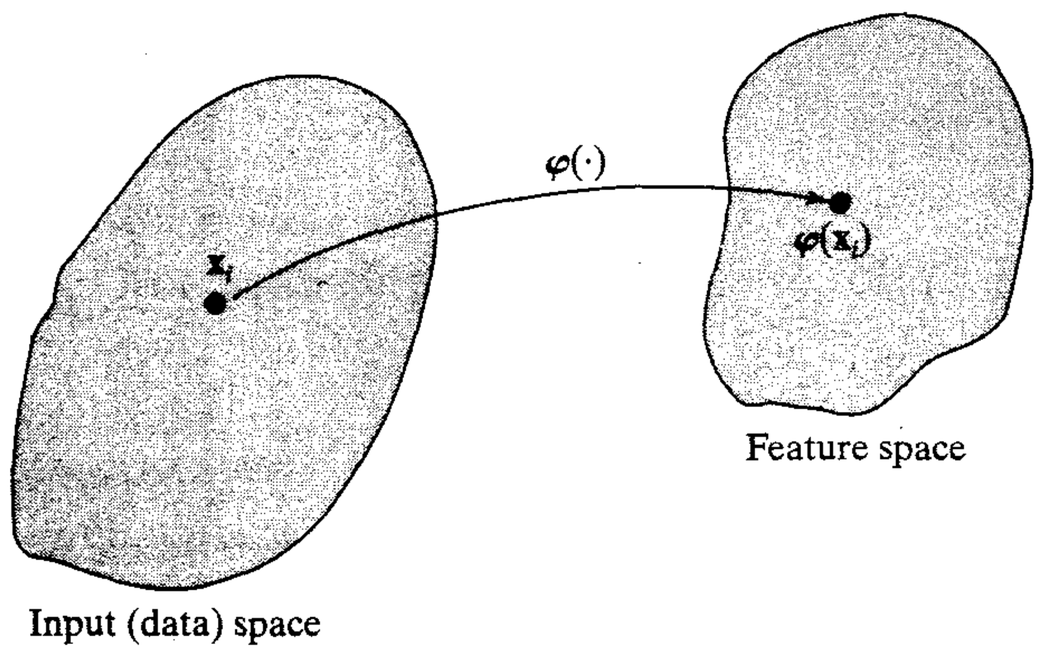

FIGURE 6.4 Nonlinear map $\varphi(\cdot)$ from the input space to the feature space.

Operation 1 is performed in accordance with Cover's theorem on the separability of patterns, which is discussed in Chapter 5. Consider an input space made up of *nonlinearly separable patterns*. Cover's theorem states that such a multidimensional space may be transformed into a new feature space where the patterns are linearly separable with high probability, provided two conditions are satisfied. First, the transformation is nonlinear. Second, the dimensionality of the feature space is high enough. These two conditions are embodied in operation 1. Note, however, Cover's theorem does *not* discuss the optimality of the separating hyperplane. It is only by using an optimal separating hyperplane that the VC dimension is minimized and generalization is achieved.

This latter matter is where the second operation comes in. Specifically, operation 2 exploits the idea of building an optimal separating hyperplane in accordance with the theory described in Section 6.3, but with a fundamental difference: The separating hyperplane is now defined as a linear function of vectors drawn from the feature space rather than the original input space. Most importantly, construction of this hyperplane is performed in accordance with the principle of structural risk minimization that is rooted in VC dimension theory. The construction hinges on the evaluation of an inner-product kernel.

Inner-Product Kernel

Let $\mathbf{x}$ denote a vector drawn from the input space, assumed to be of dimension m_0. Let $\{\varphi_j(\mathbf{x})\}_{j=1}^{m_1}$ denote a set of nonlinear transformations from the input space to the feature space: m_1 is the dimension of the feature space. It is assumed that $\varphi_j(\mathbf{x})$ is defined *a priori* for all j. Given such a set of nonlinear transformations, we may define a hyperplane acting as the decision surface as follows:

$$\sum_{j=1}^{m_1} w_j \varphi_j(\mathbf{x}) + b = 0 \tag{6.29}$$

where $\{w_j\}_{j=1}^{m_1}$ denotes a set of linear weights connecting the feature space to the output space, and b is the bias. We may simplify matters by writing

$$\sum_{j=0}^{m_1} w_j \varphi_j(\mathbf{x}) = 0 \tag{6.30}$$

where it is assumed that $\varphi_0(\mathbf{x}) = 1$ for all $\mathbf{x}$, so that w_0 denotes the bias b. Equation (6.30) defines the decision surface computed in the feature space in terms of the linear weights of the machine. The quantity $\varphi_j(\mathbf{x})$ represents the input supplied to the weight w_j *via* the feature space. Define the vector

$$\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T \tag{6.31}$$

where, by definition, we have

$$\varphi_0(\mathbf{x}) = 1 \quad \text{for all } \mathbf{x} \tag{6.32}$$

In effect, the vector $\boldsymbol{\varphi}(\mathbf{x})$ represents the "image" induced in the feature space due to the input vector $\mathbf{x}$, as illustrated in Fig. 6.4. Thus, in terms of this image we may define the decision surface in the compact form:

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = 0 \tag{6.33}$$

Adapting Eq. (6.12) to our present situation involving a feature space where we now seek "linear" separability of features, we may write

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i d_i \boldsymbol{\varphi}(\mathbf{x}_i) \tag{6.34}$$

where the *feature vector* $\boldsymbol{\varphi}(\mathbf{x}_i)$ corresponds to the input pattern $\mathbf{x}_i$ in the ith example. Therefore, substituting Eq. (6.34) in (6.33), we may define the decision surface computed in the feature space as:

$$\sum_{i=1}^{N} \alpha_i d_i \boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}) = 0 \tag{6.35}$$

The term $\boldsymbol{\varphi}^{\mathrm{T}}(\mathbf{x}_i)\varphi(\mathbf{x})$ represents the inner product of two vectors induced in the feature space by the input vector $\mathbf{x}$ and the input pattern $\mathbf{x}_i$ pertaining to the ith example. We may therefore introduce the *inner-product kernel* denoted by $K(\mathbf{x}, \mathbf{x}_i)$ and defined by

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}_i) &= \boldsymbol{\varphi}^T(\mathbf{x})\boldsymbol{\varphi}(\mathbf{x}_i) \\ &= \sum_{j=0}^{m_1} \varphi_j(\mathbf{x})\varphi_j(\mathbf{x}_i) \qquad \text{for } i = 1, 2, \ldots, N \end{aligned} \tag{6.36}$$

From this definition we immediately see that the inner-product kernel is a *symmetric function* of its arguments, as shown by

$$K(\mathbf{x}, \mathbf{x}_i) = K(\mathbf{x}_i, \mathbf{x}) \qquad \text{for all } i \tag{6.37}$$

Most importantly, we may use the inner-product kernel $K(\mathbf{x}, \mathbf{x}_i)$ to construct the optimal hyperplane in the feature space without having to consider the feature space itself in explicit form. This is readily seen by using Eq. (6.36) in (6.35), whereby the optimal hyperplane is now defined by

$$\sum_{i=1}^{N} \alpha_i d_i K(\mathbf{x}, \mathbf{x}_i) = 0 \tag{6.38}$$

Mercer's Theorem

The expansion of Eq. (6.36) for the inner-product kernel $K(\mathbf{x}, \mathbf{x}_i)$ is an important special case of *Mercer's theorem* that arises in functional analysis. This theorem may be formally stated as (Mercer, 1908; Courant and Hilbert, 1970):

> Let $K(\mathbf{x}, \mathbf{x}')$ be a continuous symmetric kernel that is defined in the closed interval $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$ and likewise for $\mathbf{x}'$. The kernel $K(\mathbf{x}, \mathbf{x}')$ can be expanded in the series
>
> $$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \varphi_i(\mathbf{x})\varphi_i(\mathbf{x}') \tag{6.39}$$
>
> with positive coefficients, $\lambda_i > 0$ for all i. For this expansion to be valid and for it to converge absolutely and uniformly, it is necessary and sufficient that the condition
>
> $$\int_{\mathbf{b}}^{\mathbf{a}} \int_{\mathbf{b}}^{\mathbf{a}} K(\mathbf{x}, \mathbf{x}')\psi(\mathbf{x})\psi(\mathbf{x}')d\mathbf{x}d\mathbf{x}' \geq 0$$
>
> holds for all $\psi(\cdot)$ for which
>
> $$\int_{\mathbf{b}}^{\mathbf{a}} \psi^2(\mathbf{x})d\mathbf{x} < \infty$$

The functions $\varphi_i(\mathbf{x})$ are called *eigenfunctions* of the expansion and the numbers λ_i are called *eigenvalues*. The fact that all of the eigenvalues are positive means that the kernal $K(\mathbf{x},\mathbf{x}')$ is *positive definite*.

In light of Mercer's theorem, we may now make the following observations:

- For $\lambda_i \neq 1$, the ith image $\sqrt{\lambda_i}\,\varphi_i(\mathbf{x})$ induced in the feature space by the input vector $\mathbf{x}$ is an eigenfunction of the expansion.
- In theory, the dimensionality of the feature space (i.e., the number of eigenvalues/eigenfunctions) can be infinitely large.

Mercer's theorem only tells us whether or not a candidate kernel is actually an inner-product kernel in some space and therefore admissible for use in a support vector machine. However, it says nothing about how to construct the functions $\varphi_i(\mathbf{x})$; we have to do that ourselves.

From the defining equation (6.23), we see that the support vector machine includes a form of regularization in an *implicit* sense. In particular, the use of a kernel $K(\mathbf{x}, \mathbf{x}')$ defined in accordance with Mercer's theorem corresponds to regularization with an operator $\mathbf{D}$ such that the kernel $K(\mathbf{x}, \mathbf{x}')$ is the Green's function of $\widetilde{\mathbf{D}}\mathbf{D}$, where $\widetilde{\mathbf{D}}$ is the adjoint of $\mathbf{D}$ (Smola and Schölkopf, 1998). Regularization theory is discussed in Chapter 5.

Optimum Design of a Support Vector Machine

The expansion of the inner-product kernel $K(\mathbf{x}, \mathbf{x}_i)$ in Eq. (6.36) permits us to construct a decision surface that is nonlinear in the input space, but *its image in the feature space is linear*. With this expansion at hand, we may now state the dual form for the constrained optimization of a support vector machine as follows:

Given the training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, *find the Lagrange multipliers* $\{\alpha_i\}_{i=1}^N$ *that maximize the objective function*

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{6.40}$$

subject to the constraints:

(1) $\displaystyle\sum_{i=1}^{N} \alpha_i d_i = 0$

(2) $0 \leq \alpha_i \leq C \qquad \text{for } i = 1, 2, \ldots, N$

where C is a user-specified positive parameter.

Note that constraint (1) arises from optimization of the Lagrangian $Q(\alpha)$ with respect to the bias $b = w_0$ for $\varphi_0(\mathbf{x}) = 1$. The dual problem just stated is of the same form as that for the case of nonseparable patterns considered in Section 6.3, except for the fact that the inner product $\mathbf{x}_i^T\mathbf{x}_j$ used therein has been replaced by the inner-product kernel $K(\mathbf{x}_i, \mathbf{x}_j)$. We may view $K(\mathbf{x}_i, \mathbf{x}_j)$ as the ij-th element of a symmetric N-by-N matrix $\mathbf{K}$, as shown by

$$\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{(i,j)=1}^{N} \tag{6.41}$$

Having found the optimum values of the Lagrange multipliers, denoted by $\alpha_{o,i}$, we may determine the corresponding optimum value of the linear weight vector, $\mathbf{w}_o$, connecting the feature space to the output space by adapting the formula of Eq. (6.17) to the new situation. Specifically, recognizing that the image $\varphi(\mathbf{x}_i)$ plays the role of input to the weight vector $\mathbf{w}$, we may define $\mathbf{w}_o$ as

$$\mathbf{w}_o = \sum_{i=1}^{N} \alpha_{o,i} d_i \varphi(\mathbf{x}_i) \tag{6.42}$$

where $\varphi(\mathbf{x}_i)$ is the image induced in the feature space due to $\mathbf{x}_i$. Note the first component of $\mathbf{w}_o$ represents the optimum bias b_o.

Examples of Support Vector Machine

The requirement on the kernel $K(\mathbf{x}, \mathbf{x}_i)$ is to satisfy Mercer's theorem. Within this requirement there is some freedom in how it is chosen. In Table 6.1 we summarize the inner-product kernels for three common types of support vector machines: polynomial learning machine, radial-basis function network, and two-layer perceptron. The following points are noteworthy:

1. The inner-product kernels for polynomial and radial-basis function types of support vector machines always satisfy Mercer's theorem. In contrast, the inner-product kernel for a two-layer perceptron type of support vector machine is somewhat restricted, as indicated in the last row of Table 6.1. This latter entry is a testament to the fact that the determination of whether or not a given kernel satisfies Mercer's theorem can indeed be a difficult matter; see Problem 6.8.
2. For all three machine types, the dimensionality of the feature space is determined by the number of support vectors extracted from the training data by the solution to the constrained optimization problem.
3. The underlying theory of a support vector machine avoids the need for heuristics often used in the design of conventional radial-basis function networks and multilayer perceptrons:

TABLE 6.1 Summary of Inner-Product Kernels

Type of support vector machine	Inner product kernel $K(\mathbf{x}, \mathbf{x}_i)$, $i = 1, 2, \ldots, N$	Comments
Polynomial learning machine	$(\mathbf{x}^T\mathbf{x}_i + 1)^p$	Power p is specified *a priori* by the user
Radial-basis function network	$\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}_i\|^2\right)$	The width σ^2, common to all the kernels, is specified *a priori* by the user
Two-layer perceptron	$\tanh(\beta_0\mathbf{x}^T\mathbf{x}_i + \beta_1)$	Mercer's theorem is satisfied only for some values of β_0 and β_1

- In the radial-basis function type of a support vector machine, the number of radial-basis functions and their centers are determined automatically by the number of support vectors and their values, respectively.
- In the two-layer perceptron type of a support vector machine, the number of hidden neurons and their weight vectors are determined automatically by the number of support vectors and their values, respectively.

Figure 6.5 displays the architecture of a support vector machine.

Irrespective of how a support vector machine is implemented, it differs from the conventional approach to the design of a multilayer perceptron in a fundamental way. In the conventional approach, model complexity is controlled by keeping the number of features (i.e., hidden neurons) small. On the other hand, the support vector machine offers a solution to the design of a learning machine by controlling model complexity independently of dimensionality, as summarized here (Vapnik, 1995, 1998):

- *Conceptual problem.* Dimensionality of the feature (hidden) space is purposely made very large to enable the construction of a decision surface in the form of a hyperplane in that space. For good generalization performance, the model complexity is controlled by imposing certain constraints on the construction of the separating hyperplane, which results in the extraction of a fraction of the training data as support vectors.
- *Computational problem.* Numerical optimization in a high-dimensional space suffers from the curse of dimensionality. This computational problem is avoided by using the notion of an inner-product kernel (defined in accordance with Mercer's theorem) and solving the dual form of the constrained optimization problem formulated in the input (data) space.

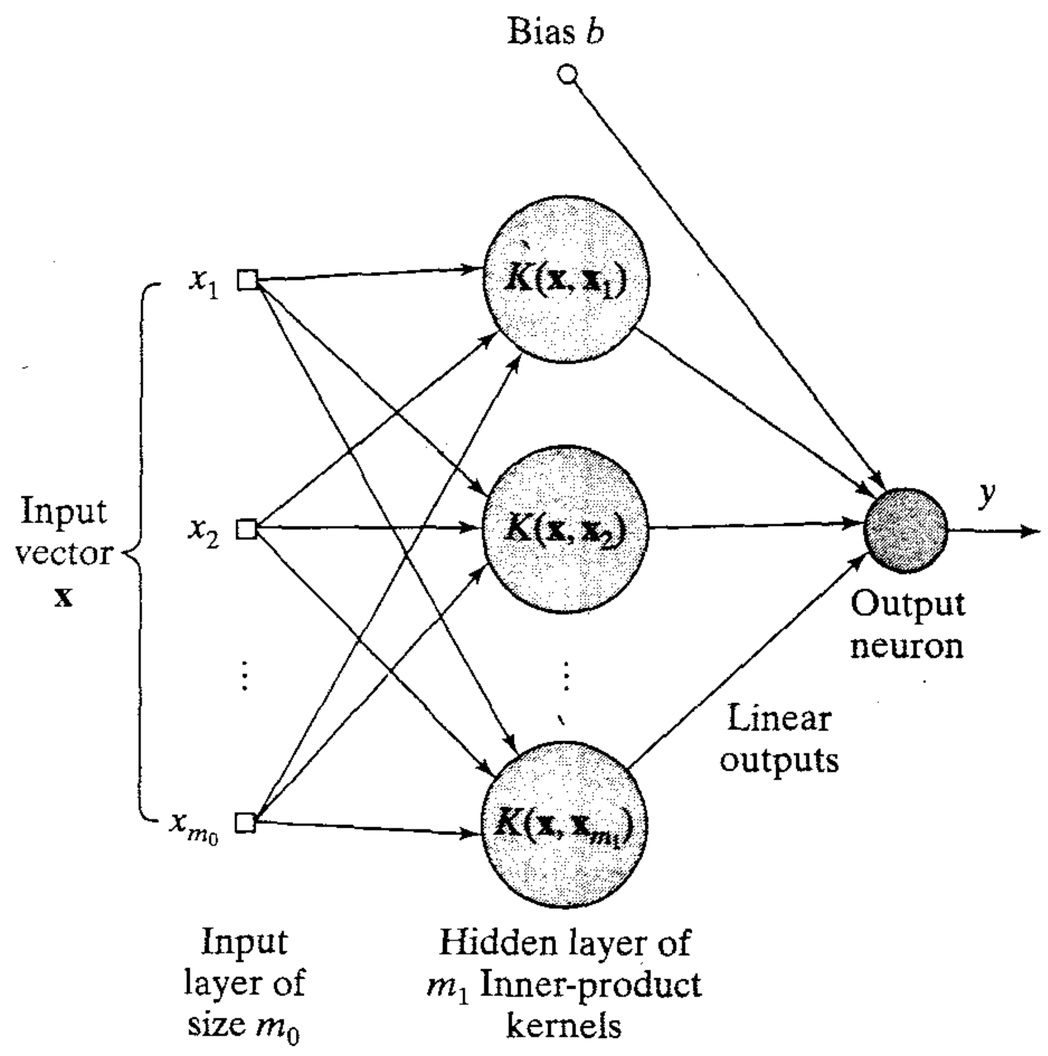

FIGURE 6.5 Architecture of support vector machine.

6.5 EXAMPLE: XOR PROBLEM (REVISITED)

To illustrate the procedure for the design of a support vector machine, we revisit the XOR (Exclusive OR) problem discussed in Chapters 4 and 5. Table 6.2 presents a summary of the input vectors and desired responses for the four possible states.

To proceed, let (Cherkassky and Mulier, 1998)

$$K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}^T\mathbf{x}_i)^2 \tag{6.43}$$

With $\mathbf{x} = [x_1, x_2]^T$ and $\mathbf{x}_i = [x_{i1}, x_{i2}]^T$, we may thus express the inner-product kernel $K(\mathbf{x}, \mathbf{x}_i)$ in terms of *monomials* of various orders as follows:

$$K(\mathbf{x}, \mathbf{x}_i) = 1 + x_1^2 x_{i1}^2 + 2x_1x_2x_{i1}x_{i2} + x_2^2 x_{i2}^2 + 2x_1x_{i1} + 2x_2x_{i2}$$

The image of the input vector $\mathbf{x}$ induced in the feature space is therefore deduced to be

$$\boldsymbol{\varphi}(\mathbf{x}) = \left[1, x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2\right]^T$$

Similarly,

$$\boldsymbol{\varphi}(\mathbf{x}_i) = \left[1, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}\right]^T, \qquad i = 1, 2, 3, 4$$

From Eq. (6.41), we also find that

$$\mathbf{K} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

The objective function for the dual form is therefore (see Eq. (6.40))

$$Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4$$

$$+ 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2)$$

Optimizing $Q(\alpha)$ with respect to the Lagrange multipliers yields the following set of simultaneous equations:

$$\begin{aligned} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 &= 1 \\ -\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 &= 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 &= 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 &= 1 \end{aligned}$$

TABLE 6.2 XOR Problem

Input vector, $\mathbf{x}$	Desired response, d
$(-1, -1)$	-1
$(-1, +1)$	$+1$
$(+1, -1)$	$+1$
$(+1, +1)$	-1

Hence, the optimum values of the Lagrange multipliers are

$$\alpha_{o,1} = \alpha_{o,2} = \alpha_{o,3} = \alpha_{o,4} = \frac{1}{8}$$

This result indicates that in this example all four input vectors $\{\mathbf{x}_i\}_{i=1}^{4}$ are support vectors. The optimum value of $Q(\alpha)$ is

$$Q_o(\alpha) = \frac{1}{4}$$

Correspondingly, we may write

$$\frac{1}{2}\|\mathbf{w}_o\|^2 = \frac{1}{4}$$

or

$$\|\mathbf{w}_o\| = \frac{1}{\sqrt{2}}$$

From Eq. (6.42), we find that the optimum weight vector is

$$\begin{aligned}\mathbf{w}_o &= \frac{1}{8}[-\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) + \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)] \\ &= \frac{1}{8}\left[-\begin{bmatrix}1\\1\\\sqrt{2}\\1\\-\sqrt{2}\\-\sqrt{2}\end{bmatrix} + \begin{bmatrix}1\\1\\-\sqrt{2}\\1\\-\sqrt{2}\\\sqrt{2}\end{bmatrix} + \begin{bmatrix}1\\1\\-\sqrt{2}\\1\\\sqrt{2}\\-\sqrt{2}\end{bmatrix} - \begin{bmatrix}1\\1\\\sqrt{2}\\1\\\sqrt{2}\\\sqrt{2}\end{bmatrix}\right] \\ &= \begin{bmatrix}0\\0\\-1/\sqrt{2}\\0\\0\\0\end{bmatrix}\end{aligned}$$

The first element of $\mathbf{w}_o$ indicates that the bias b is zero.

The optimal hyperplane is defined by (see Eq. (6.33))

$$\mathbf{w}_o^T\boldsymbol{\varphi}(\mathbf{x}) = 0$$

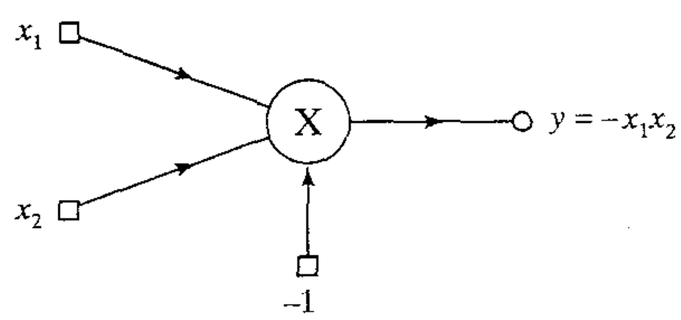

FIGURE 6.6 (a) Polynomial machine for solving the XOR problem. (b) Induced images in the feature space due to the four data points of the XOR problem.

That is,

$$\left[0, 0, \frac{-1}{\sqrt{2}}, 0, 0, 0\right]\begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = 0$$

which reduces to

$$-x_1x_2 = 0$$

The polynomial form of support vector machine for the XOR problem is as shown in Fig. 6.6a. For both $x_1 = x_2 = -1$ and $x_1 = x_2 = +1$, the output $y = -1$; and for both $x_1 = -1, x_2 = +1$ and $x_1 = +1$ and $x_2 = -1$, we have $y = +1$. Thus the XOR problem is solved as indicated in Fig. 6.6b.

6.6 COMPUTER EXPERIMENT

In this computer experiment we revisit the pattern-classification problem that we studied in Chapters 4 and 5. The experiment involved the classification of two overlapping two-dimensional Gaussian distributions labeled 1 (class $\mathscr{C}_1$) and 2 (Class $\mathscr{C}_2$). The scatter plots for these two sets of data are shown in Fig. 4.14. The probability of correct classification produced by the Bayesian (optimum) classifier is calculated to be

$$p_c = 81.51 \text{ percent}$$

TABLE 6.3 Summary of the Results of Two-Class Pattern-Classification Experiment Using the Support Vector Machine

	Common width, $\sigma^2 = 4$ Regularization parameter, $C = 0.1$				
Probability of correct classification, p_c	81.22	81.28	81.55	81.49	81.45
Number of support vectors, N_S	298	287	283	287	286

Table 6.3 presents the summary of the results obtained from a computer experiment performed on this data set using the support vector machine. For the inner-product kernel, we used the radial-basis function:

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right), \qquad i = 1, 2, \ldots, N$$

where the same width $\sigma^2 = 4$ was used for all points in the data set. The machine was trained on a total of $N = 500$ data points drawn at random from the population of data representing the two classes. The value used for the regularization parameter was $C = 0.1$.

The results presented in Table 6.3 pertain to five different trials of the experiment, with each trial involving the use of 500 data points for training and 32,000 data points for testing. The probability of correct classification, averaged over these five trials, is 81.40 percent. This average is almost equal to that realized by the Bayesian classifier. The fact that this optimum result was exceeded by 0.05 percent on one of the trials is attributed to experimental errors.

The near-perfect classification performance attained by the support vector machine is further confirmed by the decision boundary shown in Fig. 6.7, which was achieved by one of the five realizations of the machine picked at random. In this figure we have also included the decision boundary for the Bayesian classifier, which consists of a circle of center $\mathbf{x}_c = [-2/3, 0]^T$ and radius $r = 2.34$. Figure 6.6 clearly demonstrates that the support vector machine is capable of constructing a decision boundary between the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$ that is almost as good as the optimum decision boundary.

Returning to the summary of results presented in Table 6.3, the second row displays the sizes of the five different realizations of the support vector machine. These results indicate that for this experiment, the support vector machine learning algorithm selected close to 60 percent of the data points as support vectors.

In the case of nonseparable patterns, all the training errors give rise to support vectors of their own; this follows from the Kuhn–Tucker conditions. For the present experiment, the classification error is about 20 percent. With a sample size of 500, we therefore find that about one third of the support vectors were in fact due to classification errors.

Summarizing Remarks

Comparing the results of this simple computer experiment on the support vector machine with the corresponding results reported in Section 4.8 on the multilayer perceptron trained on the same data sample using the back-propagation algorithm, we can make the following observations:

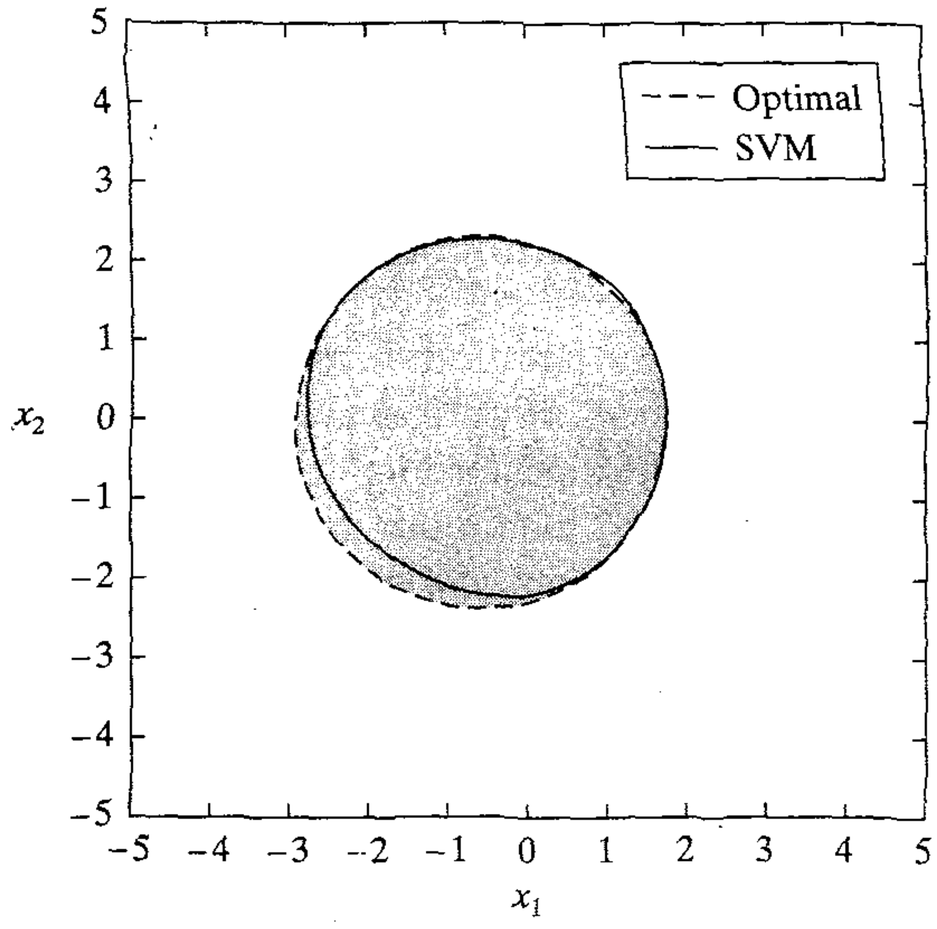

FIGURE 6.7 Decision surface for the computer experiment on pattern classification.

1. The support vector machine has the inherent ability to solve a pattern-classification problem in a manner *close to the optimum* for the problem of interest. Moreover, it is able to achieve such a remarkable performance with *no* problem domain knowledge built into the design of the machine.

2. The multilayer perceptron trained using the back-propagation algorithm, on the other hand, provides a *computationally efficient* solution to the pattern-classification problem of interest. For the two-class experiment described here we were able to realize a probability of correct classification of about 79.70 percent using a multilayer perceptron with only two hidden neurons.

In making this summary, we highlighted the individual virtues of these two approaches to pattern classification. However, for a balanced summary we must also identify their individual shortcomings. In the case of a support vector machine, the near-to-perfect classification performance is achieved at the cost of a significant demand on computational complexity. On the other hand, for a multilayer perceptron trained on the back-propagation algorithm to attain a classification performance comparable to that of the support vector machine for the same pattern-classification tasks, we need to do two things: build problem-domain knowledge into the design of the multilayer perceptron, and tune a multitude of design parameters, a practice that can be excruciating for difficult learning tasks.

6.7 ϵ-INSENSITIVE LOSS FUNCTION

Up to this point in the chapter we have focused on the use of support vector machines for solving pattern-recognition tasks. We now consider the use of support vector machines to solve nonlinear regression problems. To prepare for this discussion we will first address the issue of a suitable optimization criterion for this latter class of learning tasks.

In Chapter 4 on multilayer perceptrons and Chapter 5 on radial-basis function networks, we used a quadratic loss function as the criterion for optimizing these networks.

The main reason for using this criterion is mathematical, that is, for computational convenience. However, a least-squares estimator is sensitive to the presence of outliers (i.e., observations that are improbably large for a nominal model), and it performs poorly when the underlying distribution of the additive noise has a long tail. To overcome these limitations, we need a *robust* estimator that *is insensitive to small changes in the model.*

With robustness as a design goal, any quantitative measure of robustness must be concerned with the maximum degradation of performance that is possible for an ϵ-deviation from the nominal noise model. According to this viewpoint, an *optimal robust estimation procedure* minimizes the maximum degradation, and will therefore be a *minimax* procedure of some kind (Huber, 1981). For the case when the additive noise has a probability density function that is symmetric about the origin, the minimax procedure[4] for solving the nonlinear regression problem uses the absolute error as the quantity to be minimized (Huber, 1964). That is, the loss function has the form

$$L(d, y) = |d - y| \tag{6.44}$$

where d is the desired response and y is the estimator output.

To construct a support vector machine for approximating a desired response d, we may use an extension of the loss function in Eq. (6.44), originally proposed in Vapnik (1995, 1998), as described here

$$L_\epsilon(d, y) = \begin{cases} |d - y| - \epsilon, & \text{for } |d - y| \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{6.45}$$

where ϵ is a prescribed parameter. The loss function $L_\epsilon(d, y)$ is called the *ϵ-insensitive loss function*. It is equal to zero if the absolute value of the deviation of the estimator output y from the desired response d is less than or equal to zero, and it is equal to the absolute value of the deviation minus ϵ otherwise. The loss function of Eq. (6.44) is a special case of the ϵ-insensitive loss function for $\epsilon = 0$. Figure 6.8 illustrates the dependence of $L_\epsilon(d, y)$ on the error $d - y$.

6.8 SUPPORT VECTOR MACHINES FOR NONLINEAR REGRESSION

Consider a *nonlinear regressive model* in which the dependence of a scalar d on a vector $\mathbf{x}$ is described by

$$d = f(\mathbf{x}) + \nu \tag{6.46}$$

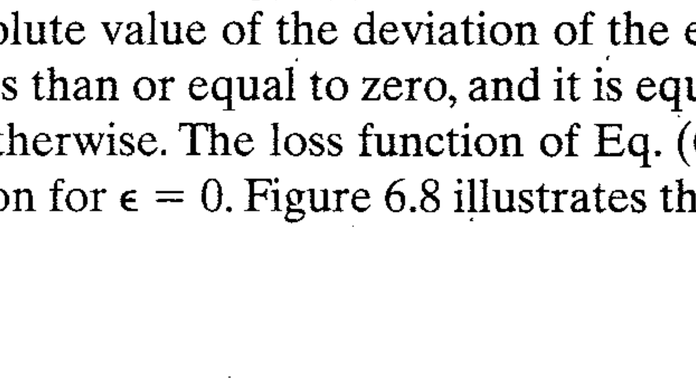

FIGURE 6.8 ϵ-insensitive loss function.

The scalar-valued nonlinear function $f(\mathbf{x})$ is defined by the conditional expectation $E[D|\mathbf{x}]$, as discussed in Chapter 2; D is a random variable with a realization denoted by d. The additive noise term v is statistically independent of the input vector $\mathbf{x}$. The function $f(\cdot)$ and the statistics of noise v are unknown. All that we have available is a set of training data $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, where $\mathbf{x}_i$ is a sample value of the input vector $\mathbf{x}$ and d_i is the corresponding value of the model output d. The problem is to provide an estimate of the dependence of d on $\mathbf{x}$.

To proceed, we postulate an estimate of d, denoted by y, which is expanded in terms of a set of nonlinear basis functions $\{\varphi_j(\mathbf{x})\}_{j=0}^{m_1}$ as follows:

$$\begin{aligned} y &= \sum_{j=0}^{m_1} w_j \varphi_j(\mathbf{x}) \\ &= \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) \end{aligned} \tag{6.47}$$

where

$$\boldsymbol{\varphi}(\mathbf{x}) = [\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T$$

and

$$\mathbf{w} = [w_0, w_1, \ldots, w_{m_1}]^T$$

As before, it is assumed that $\varphi_0(\mathbf{x}) = 1$, so that the weight w_0 represents the bias b. The issue to be resolved is to minimize the empirical risk

$$R_{\text{emp}} = \frac{1}{N} \sum_{i=1}^{N} L_\epsilon(d_i, y_i) \tag{6.48}$$

subject to the inequality

$$\|\mathbf{w}\|^2 \leq c_0 \tag{6.49}$$

where c_0 is a constant. The ϵ-insensitive loss function $L_\epsilon(d_i, y_i)$ is as previously defined in Eq. (6.45). We may reformulate this constrained optimization problem by introducing two sets of nonnegative *slack variables* $\{\xi_i\}_{i=1}^{N}$ and $\{\xi_i'\}_{i=1}^{N}$ that are defined as follows:

$$d_i - \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) \leq \epsilon + \xi_i, \qquad i = 1, 2, \ldots, N \tag{6.50}$$

$$\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) - d_i \leq \epsilon + \xi_i', \qquad i = 1, 2, \ldots, N \tag{6.51}$$

$$\xi_i \geq 0, \qquad i = 1, 2, \ldots, N \tag{6.52}$$

$$\xi_i' \geq 0, \qquad i = 1, 2, \ldots, N \tag{6.53}$$

The slack variables ξ_i and ξ_i' describe the ϵ-insensitive loss function defined in Eq. (6.45). This constrained optimization problem may therefore be viewed as equivalent to that of minimizing the cost functional

$$\Phi(\mathbf{w}, \xi, \xi') = C\left(\sum_{i=1}^{N} (\xi_i + \xi_i')\right) + \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{6.54}$$

subject to the constraints of Eqs. (6.50) to (6.53). By incorporating the term $\mathbf{w}^T\mathbf{w}/2$ in the functional $\Phi(\mathbf{w}, \xi, \xi')$ of Eq. (6.54), we dispense with the need for the inequality

constraint of Eq. (6.49). The constant C in Eq. (6.54) is a user-specified parameter. Accordingly, we may define the Lagrangian function:

$$\begin{aligned} J(\mathbf{w}, \xi, \xi', \alpha, \alpha', \gamma, \gamma') = C \sum_{i=1}^{N} (\xi_i + \xi_i') + \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) - d_i + \epsilon + \xi_i\right] \\ - \sum_{i=1}^{N} \alpha_i' \left[d_i - \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + \epsilon + \xi_i'\right] \\ - \sum_{i=1}^{N} (\gamma_i\xi_i + \gamma_i'\xi_i') \end{aligned} \tag{6.55}$$

where the α_i and the α_i' are the Lagrange multipliers. The last term on the right-hand side of Eq. (6.55), involving γ_i and γ_i', is included to ensure that the optimality constraints on the Lagrange multipliers α_i and α_i' assume variable forms. The requirement is to minimize $J(\mathbf{w}, \xi, \xi' \alpha, \alpha', \gamma, \gamma')$ with respect to the weight vector $\mathbf{w}$ and slack variables ξ and ξ'; it must also be maximized with respect to α and α' and also with respect to γ and γ'. By carrying out this optimization we have in respective ways:

$$\mathbf{w} = \sum_{i=1}^{N} (\alpha_i - \alpha_i')\boldsymbol{\varphi}(\mathbf{x}_i) \tag{6.56}$$

$$\gamma_i = C - \alpha_i \tag{6.57}$$

and

$$\gamma_i' = C - \alpha_i' \tag{6.58}$$

The optimization of $J(\mathbf{w}, \xi, \xi', \alpha, \alpha', \gamma, \gamma')$ just described is the primal problem for regression. To formulate the corresponding dual problem, we substitute Eqs. (6.56) through (6.58) in Eq. (6.55), and thus get the convex functional (after simplifying terms):

$$\begin{aligned} Q(\alpha_i, \alpha_i') = \sum_{i=1}^{N} d_i(\alpha_i - \alpha_i') - \epsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i') \\ - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_i')(\alpha_j - \alpha_j')K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{6.59}$$

where $K(\mathbf{x}_i, \mathbf{x}_j)$ is the inner-product kernel defined in accordance with Mercer's theorem:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}^T(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x}_j)$$

The solution to our constrained optimization problem is thus obtained by maximizing $Q(\alpha, \alpha')$ with respect to the Lagrange multipliers α and α', subject to a new set of constraints that incorporates the constant C included in the definition of the function $\Phi(\mathbf{w}, \xi, \xi')$ of Eq. (6.54).

We may now state the dual problem for nonlinear regression using a support vector machine as follows:

Given the training sample $\{(\boldsymbol{x}_i, d_i)\}_{i=1}^N$ *find the Lagrange multipliers* $\{\alpha_i\}_{i=1}^N$ and $\{\alpha_i'\}_{i=1}^N$ *that maximize the objective function*

$$Q(\alpha_i, \alpha_i') = \sum_{i=1}^{N} d_i(\alpha_i - \alpha_i') - \epsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i')$$
$$- \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_i')(\alpha_j - \alpha_j') K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to the following constraints:

(1) $\sum_{i=1}^{N} (\alpha_i - \alpha_i') = 0$

(2) $0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, N$
$0 \leq \alpha_i' \leq C, \quad i = 1, 2, \ldots, N$

where C is a user-specified constant.

Constraint (1) arises from optimization of the Lagrangian with respect to the bias $b = w_0$ for $\varphi_0(\mathbf{x}) = 1$. Thus, having obtained the optimum values of α_i, and α_i', we may then use Eq. (6.56) to determine the optimum value of the weight vector $\mathbf{w}$ for a prescribed map $\varphi(\mathbf{x})$. Note that, as in the solution to the pattern-recognition problem, only some of the coefficients in the expansion of Eq. (6.56) have values different from zero; in particular, the data points for which $\alpha_i \neq \alpha_i'$ define the support vectors for the machine.

The two parameters ϵ and C are free parameters that control the VC dimension of the approximating function

$$F(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$$
$$= \sum_{i=1}^{N} (\alpha_i - \alpha_i') K(\mathbf{x}, \mathbf{x}_i) \tag{6.60}$$

Both ϵ and C must be selected by the user. In a conceptual sense, the choice of ϵ and C raises the same issues of complexity control as the choice of parameter C for pattern classification. In practice, however, complexity control for regression is a more difficult problem for the following reasons:

- The parameters ϵ and C must be *tuned simultaneously*.
- Regression is intrinsically more difficult than pattern classification.

A principled approach for the selection of ϵ and C is still an open research area.

Finally, as with a support vector machine for pattern recognition, a support vector machine for nonlinear regression may be implemented in the form of a polynomial learning machine, radial-basis function network, or two-layer perceptron. The inner product kernels for these three methods of implementation are presented in Table 6.1.

6.9 SUMMARY AND DISCUSSION

The support vector machine is an elegant and highly principled learning method for the design of a feedforward network with a single hidden layer of nonlinear units. Its derivation follows the principle of structural risk minimization that is rooted in VC

dimension theory, which makes its derivation even more profound. As the name implies, the design of the machine hinges on the extraction of a subset of the training data that serves as support vectors and therefore represents a stable characteristic of the data. The support vector machine includes the polynomial learning machine, radial-basis function network, and two-layer perceptron as special cases. Thus, although these methods provide different representations of intrinsic statistical regularities contained in the training data, they all stem from a common root in a support vector machine setting.

Unlike the popular back-propagation algorithm, the support vector learning algorithm operates only in a batch mode. There is another important difference between these two algorithms. The back-propagation algorithm minimizes a quadratic loss function, regardless of what the learning task is. In contrast, the support vector learning algorithm for pattern recognition is quite different from that for nonlinear regression, as indicated here:

- When performing a pattern-recognition task, the support vector learning algorithm minimizes the number of training samples that fall inside the margin of separation between positive and negative examples; this is only approximately true, since the slack variables ξ_i are used instead of the indicator function $I(\xi_i - 1)$. Although such a criterion is not exactly the same as that of minimizing the probability of classification error, it is considered to be more appropriate than the mean-square error criterion that is behind the back-propagation algorithm.
- When performing a nonlinear regression task, the support vector learning algorithm minimizes an ϵ-insensitive loss function that is an extension of the mean absolute error criterion of minimax theory. The algorithm is thereby robustified.

Whatever the learning task, the support vector machine provides a method for controlling model complexity independently of dimensionality. In particular, the model complexity problem is solved in a high-dimensional space by using a penalized hyperplane defined in the feature (hidden) space as the decision surface; the result is good generalization performance. The curse of dimensionality is bypassed by focusing on the dual problem for performing the constrained optimization problem. An important reason for using the dual setting is to avoid having to define and compute the parameters of the optimal hyperplane in a data space of possibly high dimensionality.

Ordinarily, the training of a support vector machine consists of a quadratic programming problem[5] that is attractive for two reasons:

- It is guaranteed to find a global extremum of the error surface, where the error refers to the difference between the desired response and the output of the support vector machine.
- The computation can be performed efficiently.

Most importantly, by using a suitable inner-product kernel, the support vector machine automatically computes all the important network parameters pertaining to that choice of a kernel. For example, in the case of a radial-basis function network, the kernel is a Gaussian function. For this method of implementation, the number of radial-

basis functions and their centers, and the linear weights and bias levels are all computed automatically. The centers of the radial-basis functions are defined by the support vectors picked by the quadratic optimization strategy. The support vectors are typically a fraction of the total number of examples constituting the training sample. We may thus view the design of an RBF network obtained using the support-vector machine learning procedure as a *sparse* version of the corresponding design resulting from the use of a strict interpolation strategy described in the previous chapter.

Several commercial optimization libraries[6] can be used to solve the quadratic programming problem. However, these libraries are of limited use. The memory requirements of the quadratic programming problem grow with the square of the size of the training sample. Consequently, in real-life applications that may involve several thousand data points, the quadratic programming problem cannot be solved by a straightforward use of a commercial optimization library. Osuna et al. (1997) have developed a novel decomposition algorithm that attains optimality by solving a sequence of much smaller subproblems. In particular, the decomposition algorithm takes advantage of the support vector coefficients that are active on either side of their bounds defined by $\alpha_i = 0$ or $\alpha_i = C$. It is reported therein that the decomposition algorithm performs satisfactorily in applications with as many as 100,000 data points.

In terms of running time, support vector machines are currently slower than other neural networks (e.g., multilayer perceptrons trained with the back-propagation algorithm) for a similar generalization performance. There are two reasons for this slower behavior:

1. There is no control over the number of data points selected by the learning algorithm for use as support vectors.
2. There is no provision for incorporating prior knowledge about the task at hand into the design of the learning machine.

Modifications of the support vector machine intended to deal with these shortcomings are now briefly discussed.

The issue of how to control the selection of support vectors is a difficult one, particularly when the patterns to be classified are nonseparable and the training data are noisy. In general, attempts to remove known errors from the data before training or to remove them from the expansion after training will not give the same optimal hyperplane because the errors are needed for penalizing nonseparability. In Osuna and Girosi (1998), the problem of reducing the running time of a support vector machine for pattern classification has been investigated. Two novel approaches for dealing with this problem are described:

- The support vector machine is itself used as a nonlinear regression tool to approximate the decision surface (separating the classes) with a user-specified accuracy.
- The procedure for training the support vector machine is reformulated to yield the same exact decision surface while using a smaller number of basis functions.

In the first approach, the solution is simplified by approximating it with a linear combination of a *subset* of the basis functions. The resulting machine is a natural extension of

the support vector machine for function approximation. This extension is designed to find the minimum of a cost functional of the following form:

$$\mathscr{E}(F) = \sum_{i=1}^{N} |d_i - F(\mathbf{x}_i)|_\epsilon + \frac{1}{2C}\psi(F)$$

where $F(\cdot)$ is the approximating function, $\psi(\cdot)$ is a smoothness functional, and $|x|_\epsilon$ is the ϵ-insensitive cost function defined by

$$|x|_\epsilon = \begin{cases} 0 & \text{if } |x| < \epsilon \\ |x| - \epsilon & \text{otherwise} \end{cases}$$

The ϵ-insensitive cost function has the effect of making the solution robust to outliers and insensitive to errors below a certain threshold ϵ. The minimum of the cost functional $\mathscr{E}(F)$ has the form

$$F(\mathbf{x}) = \sum_{i=1}^{N} c_i G(\mathbf{x}, \mathbf{x}_i)$$

where $G(\cdot,\cdot)$ is a kernel that depends on the particular choice of the smoothness function $\psi(\cdot)$, and the coefficients c_i are computed by solving a quadratic programming problem. The solution is typically *sparse*; that is, only a small number of the c_i will be different from zero, and their number is controlled by the parameter ϵ. In the second approach, the primal problem is reformulated in such a way that it has the same initial structure as the original primal problem, but with one difference: The inner-product kernel $K(\mathbf{x}, \mathbf{x}')$ is now incorporated in the formulation. Both approaches also lend themselves to use for reducing the complexity of support vector machines for nonlinear regression.

Finally, turning to the issue of prior knowledge, it is widely recognized that the performance of a learning machine can be improved by incorporating prior knowledge about the task to be learned in the design of the machine (Abu-Mostafa, 1995). In general, two different ways of exploiting prior knowledge have been pursued in the literature:

- An additional term is included in the cost function, thereby forcing the learning machine to construct a function that incorporates the prior knowledge. This is precisely what is done in the use of regularization.
- Virtual examples are generated from the given training sample. The motivation here is that the learning machine can extract the prior knowledge more readily from the artificially enlarged training data.

In the second approach, the learning process may be slowed down due to correlations in the artificial data and the increased size of the training data set. However, the second approach has an advantage over the first approach because it may be readily implemented for all kinds of prior knowledge and learning machines. One way of implementing the second approach is to proceed as follows (Schölkopf et al., 1996):

1. A support vector machine is trained on the given data to extract a set of support vectors in the usual way.

2. Artificial examples, called *virtual support vectors*, are generated by applying prior knowledge in the form of desired invariance transformations to the support vectors obtained under step 1.
3. Another support vector machine is trained on the artificially enlarged set of examples.

This method has the potential of yielding a significant gain in classification accuracy at a moderate cost in time: It requires two training runs instead of a single one, but it constructs classification rules using more support vectors.

NOTES AND REFERENCES

1. Let $\mathscr{C}$ be a subset of $\mathbb{R}^m$. The subset $\mathscr{C}$ is said to be *convex* if

$$\alpha x + (1 - \alpha)y \in \mathscr{C} \quad \text{for all} \quad (x, y) \in \mathscr{C} \quad \text{and} \quad \alpha \in [0, 1]$$

A function $f: \mathscr{C} \to \mathbb{R}$ is said to be a *convex function* if

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha) f(y) \quad \text{for all} \quad (x, y) \in \mathscr{C} \quad \text{and} \quad \alpha \in [0, 1]$$

2. With computational complexity as the issue of interest, we may identify two classes of algorithms:
   - *Polynomial time algorithms*, which require a running time that is a polynomial function of the problem size. For example, the fast Fourier transform (FFT) algorithm, commonly used for spectrum analysis, is a polynomial time algorithm as it requires a running time of order $n\log n$, where n is a measure of the problem size.
   - *Exponential time algorithms*, which require a running time that is an exponential function of the problem size. For example, an exponential time algorithm may take time 2^n, where n is a measure of the problem size.

   On this basis we may view polynomial time algorithms as "efficient" algorithms and exponential time algorithms as "inefficient" algorithms.

   There are many computational problems that arise in practice, for which no efficient algorithms have been devised. Many if not all of these seemingly intractable problems are said to belong to a class of problems referred to as *NP-complete problems*. The term "NP" stands for "nondeterministic polynomial."

   For more detailed discussion of NP-complete problems, see Cook (1971), Garey and Johnson (1979), and Cormen et al. (1990).
3. The idea of an inner-product kernel was first used in Aizerman et al. (1964a, 1964b) in the formulation of the method of potential functions, which represents the forerunner to radial-basis function networks. About the same time, Vapnik and Chervonenkis (1965) developed the idea of an optimal hyperplane. The combined use of these two powerful concepts in formulating the support vector machine was pioneered by Vapnik and coworkers in 1992; see Boser, Guyon and Vapnik (1992) and Cortes and Vapnik (1995). A full mathematical account of the support vector machine was first described in Vapnik (1995), and subsequently in a more expanded form in Vapnik (1998).
4. Huber's minimax theory is based on neighborhoods that are not global by virtue of excluding asymmetric distributions. Nevertheless, this theory deals successfully with a large part of traditional statistics, particularly, regression.

5. In Schurmars (1997), the use of linear programming is explored by adopting the L_1 norm, $\|\mathbf{w}\|_1$, in place of the L_2 norm, $\|\mathbf{w}\|_2$, that is used in support vector machines. The L_1 norm of the weight vector $\mathbf{w}$ is defined by

$$\|\mathbf{w}\|_1 = \sum_i |w_i|$$

where w_i is the ith element of $\mathbf{w}$. Maximum margin classification using the L_1 norm appears to be biased toward axis-oriented hyperplanes, that is, toward weight vectors with few nonzero elements.

6. Commercial libraries for quadratic programming include the following:
   - MINOS5.4: (Murtagh and Saunders, 1978)
   - LSSOL (Gill et al., 1986)
   - LOQO (Vanderbei, 1994)
   - QPOPT and SQOPT (Gill and Murray, 1991).

PROBLEMS

Optimal separating hyperplane

6.1 Consider the case of a hyperplane for linearly separable patterns, which is defined by the equation

$$\mathbf{w}^T\mathbf{x} + b = 0$$

where $\mathbf{w}$ denotes the weight vector, b denotes the bias, and $\mathbf{x}$ denotes the input vector. The hyperplane is said to correspond to a *canonical pair* $(\mathbf{w}, b)$ if, for the set of input patterns $\{\mathbf{x}_i\}_{i=1}^N$, the additional requirement

$$\min_{i=1,2,\ldots,N} |\mathbf{w}^T\mathbf{x}_i + b| = 1$$

is satisfied. Show that this requirement leads to a margin of separation between the two classes equal to $2/\|\mathbf{w}\|$.

6.2 Justify the following statement in the context of nonseparable patterns: Misclassification implies nonseparability of patterns, but the converse is *not* necessarily true.

6.3 Starting with the primal problem for the optimization of the separating hyperplane for nonseparable patterns, formulate the dual problem as described in Section 6.3.

6.4 In this problem we explore the "leave-one-out method," discussed in Chapter 4, for estimating the expected test error produced by an optimal hyperplane for the case of nonseparable patterns. Discuss the various possibilities that can arise in the use of this method by eliminating any one pattern from the training sample and constructing a solution based on the remaining patterns.

6.5 The location of the optimal hyperplane in the data space is determined by the data points selected as support vectors. If the data are noisy, one's first reaction might be to question the robustness of the margin of separation to the presence of noise. Yet careful study of the optimal hyperplane reveals that the margin of separation is actually robust to noise. Discuss the rationale for this robust behavior.

Inner-product kernel

6.6 The inner-product kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ is evaluated over a training sample $\mathcal{T}$ of size N, yielding the N-by-N matrix:

$$\mathbf{K} = \{K_{ij}\}_{(i,j)=1}^N$$

where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. The matrix $\mathbf{K}$ is positive in that all of its elements have positive values. Using the similarity transformation:

$$\mathbf{K} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues and $\mathbf{Q}$ is a matrix made up of the corresponding eigenvectors, formulate an expression for the inner-product kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ in terms of the eigenvalues and eigenvectors of matrix $\mathbf{K}$. What conclusions can you draw from this representation?

6.7 **(a)** Prove the *unitary invariance property* of the inner-product kernel $K(\mathbf{x}, \mathbf{x}_i)$; that is,

$$K(\mathbf{x}, \mathbf{x}_i) = K(\mathbf{Q}\mathbf{x}, \mathbf{Q}\mathbf{x}_i)$$

where $\mathbf{Q}$ is a unitary matrix defined by

$$\mathbf{Q}^{-1} = \mathbf{Q}^T$$

(b) Demonstrate that all three inner-product kernels described in Table 6.1 satisfy this property.

6.8 The inner-product kernel for a two-layer perceptron is defined by

$$K(\mathbf{x}, \mathbf{x}_i) = \tanh(\beta_0 \mathbf{x}^T \mathbf{x}_i + \beta_1)$$

Explore some values for the constants β_0 and β_1 for which Mercer's theorem is not satisfied.

Pattern classification

6.9 The inner-product kernel for a polynomial learning machine used to solve the XOR problem is defined by

$$K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}^T \mathbf{x}_i)^p$$

What is the minimum value of power p for which the XOR problem is solved? Assume that p is a positive integer. What is the result of using a value for p larger than the minimum?

6.10 Figure P6.10 shows the XOR function operating on a three-dimensional pattern $\mathbf{x}$, as described here

$$\text{XOR}(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$$

where the symbol $\oplus$ denotes the Exclusive OR Boolean function operator. Design a polynomial learning machine to separate the two classes of points represented by the output of this operator.

6.11 Throughout the chapter we discussed the use of a support vector machine for binary classification. Discuss how a support vector machine can be used to solve an M-ary pattern-classification problem, where $M > 2$.

Nonlinear regression

6.12 The dual problem described in Section 6.8 for using a support vector machine to solve the nonlinear regression problem includes the following constraint:

$$\sum_{i=1}^{N} (\alpha_i - \alpha_i') = 0$$

where the α_i and α_i' are the Lagrange multipliers. Show that this constraint arises from minimization of the Lagrangian with respect to the bias b, that is, the first element w_0 of the weight vector $\mathbf{w}$ corresponding to $\varphi_0(\mathbf{x}) = 1$.

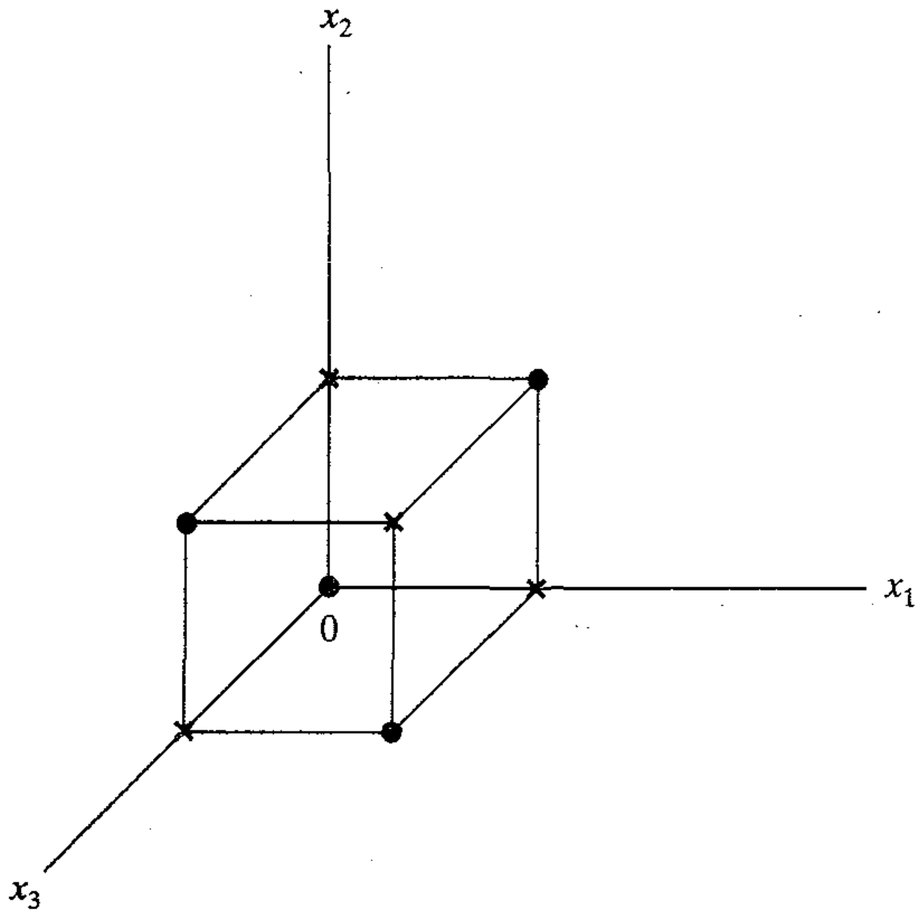

FIGURE P6.10

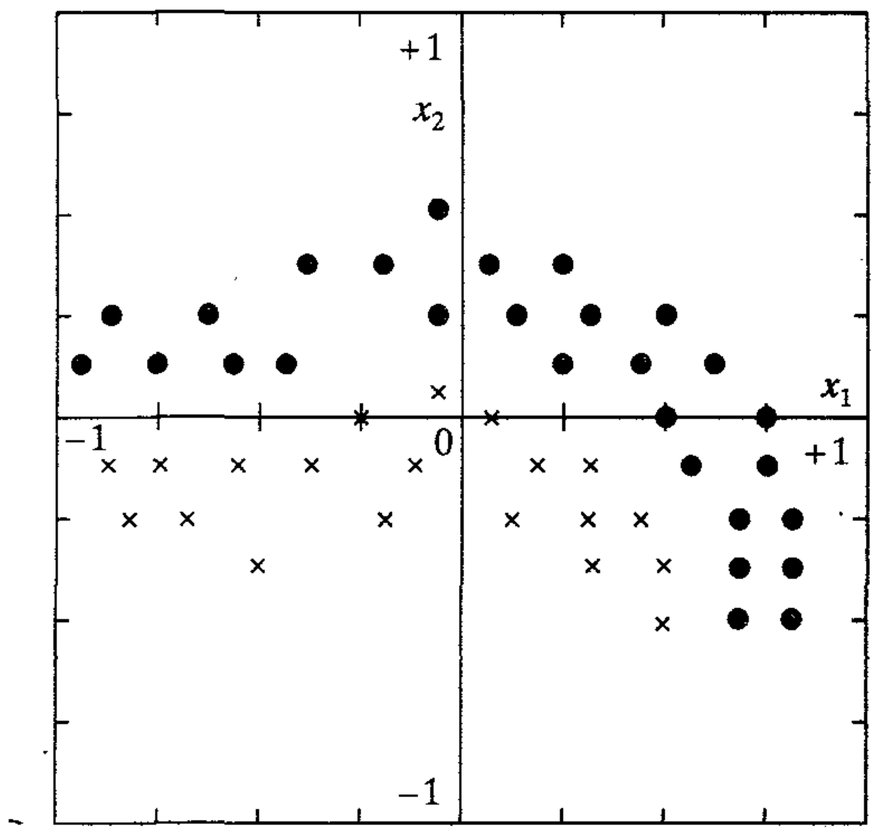

FIGURE P6.14

Virtues and limitations

6.13 **(a)** Compare the virtues and limitations of support vector machines with those of radial-basis function (RBF) networks with respect to the following tasks:
(1) Pattern classification
(2) Nonlinear regression
(b) Do the same for support vector machines versus multilayer perceptrons trained using the back-propagation algorithm.

Computer experiments

6.14 Figure P6.14 shows a set of data points corresponding to two classes, $\mathscr{C}_1$ and $\mathscr{C}_2$. Both coordinates, x_1 and x_2, range from -1 to $+1$. Using a single radial-basis function kernel

$$K(\mathbf{x}, \mathbf{t}) = \exp(-\|\mathbf{x} - \mathbf{t}\|^2)$$

construct the optimal hyperplane and identify the support vectors for this data set.

6.15 The computer experiment described in Section 6.6 was for the classification of two overlapping Gaussian distributions. The following "regularization" parameter was used in that experiment: $C = 0.1$. The common width of the radial-basis functions used for constructing the inner-product kernels was $\sigma^2 = 4$. Repeat the computer experiment described in that section for the following two values of the regularization parameter:
(a) $C = 0.05$
(b) $C = 0.2$
Comment on your results in light of the findings reported in Section 6.6.

6.16 In applying radial-basis function networks to nonlinear regression problems, we often find that the use of an unlocalized basis function such as the multiquadric results in a more accurate solution than the use of a localized basis function such as the Gaussian function. It may be conjectured that a similar situation arises in the case of support vector machines, because the use of an (unbounded) polynomial learning machine may prove more accurate than a (bounded) radial-basis function machine. Using a computer experiment on a nonlinear regression problem, explore the validity of this conjecture.