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

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

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

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

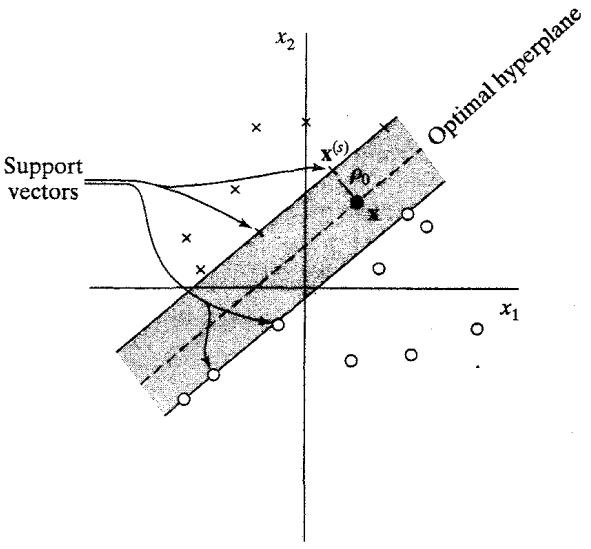

FIGURE 6.1 Illustration of the idea of an optimal hyperplane for linearly separable patterns.

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.

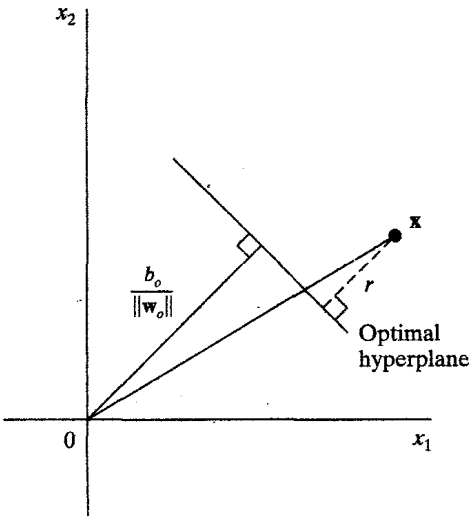

FIGURE 6.2 Geometric interpretation of algebraic distances of points to the optimal hyperplane for a two-dimensional case.

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

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

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

$$\begin{aligned} \rho &= 2r \\ &= \frac{2}{\|\mathbf{w}_o\|} \end{aligned} \tag{6.9}$$

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)

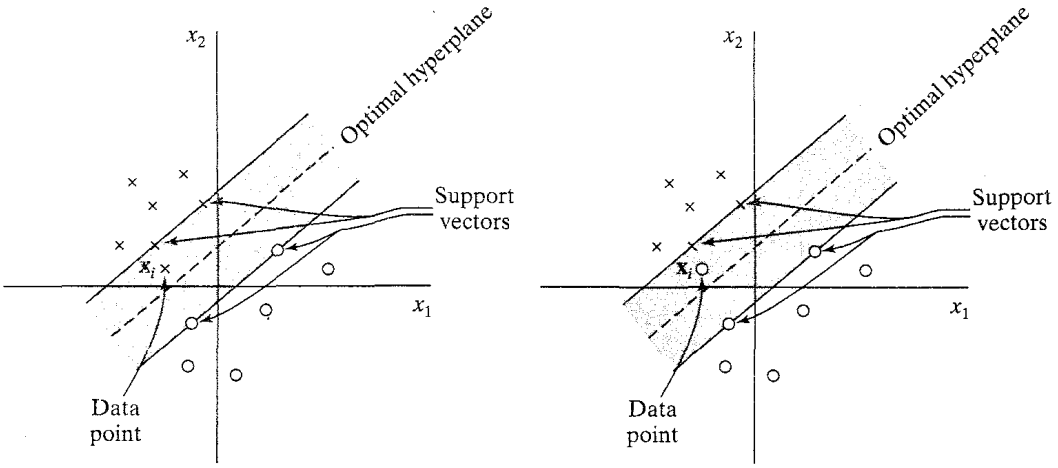

FIGURE 6.3 (a) Data point $\mathbf{x}_i$ (belonging to class $\mathscr{C}_1$) falls inside the region of separation, but on the right side of the decision surface. (b) Data point $\mathbf{x}_i$ (belonging to class $\mathscr{C}_2$) falls on the wrong side of the decision surface.

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

(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 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

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

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.

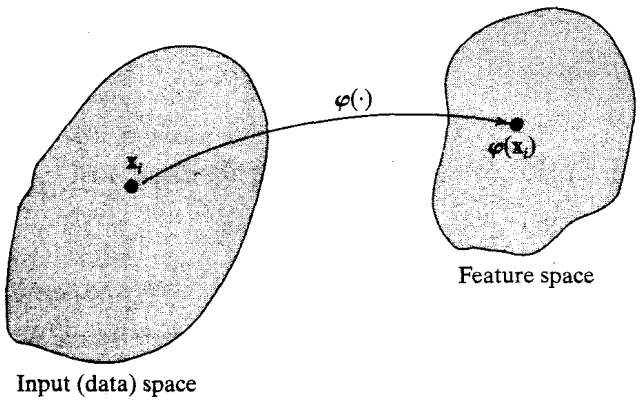

FIGURE 6.4 Nonlinear map $\varphi(\cdot)$ from the input space to the feature space.

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 $\boldsymbol{\varphi}^{\mathrm{T}}(\mathbf{x}_i)\boldsymbol{\varphi}(\mathbf{x})$ 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:

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

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

where C is a user-specified positive parameter.

Note that 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.

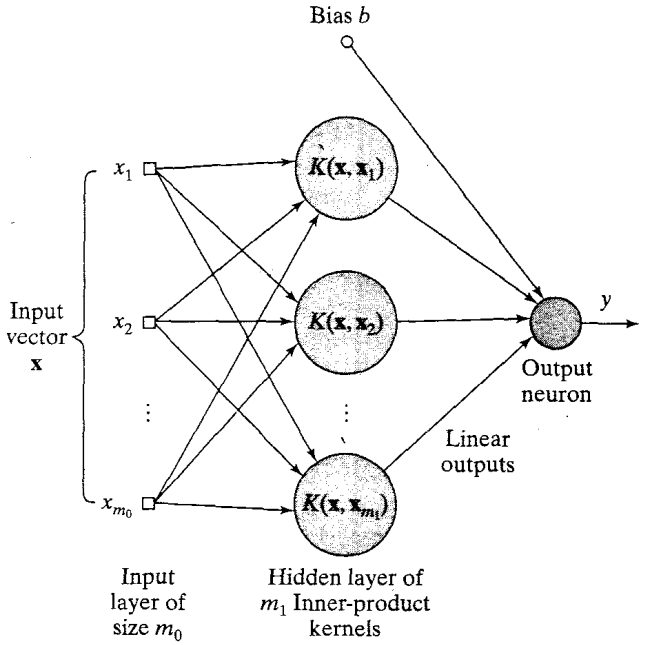

FIGURE 6.5 Architecture of support vector machine.

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:

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

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

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

Similarly,

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

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

$$\mathbf{w}_o = \frac{1}{8}[-\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) + \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)]$$

$$= \frac{1}{8}\left[-\begin{bmatrix}1\\1\\\sqrt{2}\\1\\-\sqrt{2}\\-\sqrt{2}\end{bmatrix} + \begin{bmatrix}1\\1\\-\sqrt{2}\\1\\-\sqrt{2}\\\sqrt{2}\end{bmatrix} + \begin{bmatrix}1\\1\\-\sqrt{2}\\1\\\sqrt{2}\\-\sqrt{2}\end{bmatrix} - \begin{bmatrix}1\\1\\\sqrt{2}\\1\\\sqrt{2}\\\sqrt{2}\end{bmatrix}\right]$$

$$= \begin{bmatrix}0\\0\\-1/\sqrt{2}\\0\\0\\0\end{bmatrix}$$

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

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

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

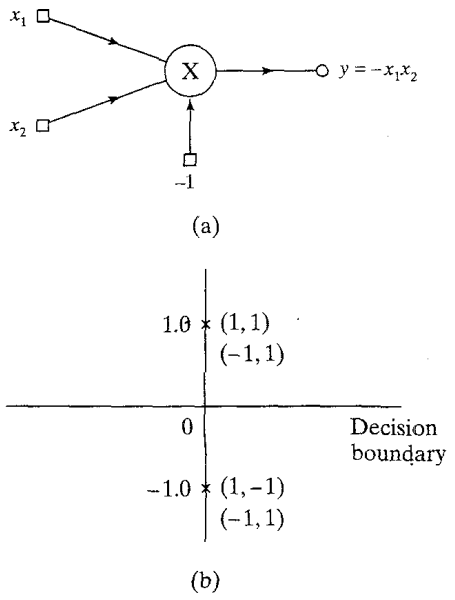

FIGURE 6.6 (a) Polynomial machine for solving the XOR problem. (b) Induced images in the feature space due to the four data points of the XOR problem.

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

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