

### 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  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ . 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

$$h \leq \min \left\{ \left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0 \right\} + 1 \quad (6.19)$$

where the ceiling sign  $\lceil \cdot \rceil$  means the smallest integer greater than or equal to the number enclosed within,  $\rho$  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  $\rho$ .

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

$$S_k = \{\mathbf{w}^T \mathbf{x} + b : \|\mathbf{w}\|^2 \leq c_k\}, \quad k = 1, 2, \dots \quad (6.20)$$

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

$$S_k = \left\{ \left\lceil \frac{r^2}{\rho^2} \right\rceil + 1 : \rho^2 \geq a_k \right\}, \quad k = 1, 2, \dots \quad (6.21)$$

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  $\rho$ ). 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)):

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

This violation can arise in one of two ways:

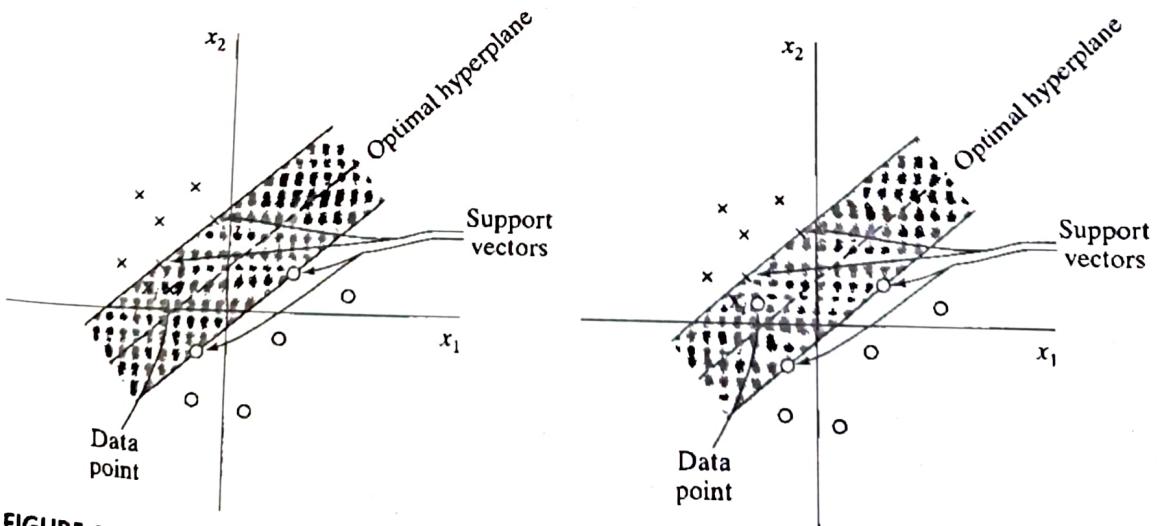
- The data point  $(\mathbf{x}_i, d_i)$  falls inside the region of separation but on the right side of the decision surface, as illustrated in Fig. 6.3a.
- The data point  $(\mathbf{x}_i, d_i)$  falls on the wrong side of the decision surface, as illustrated in Fig. 6.3b.

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:

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

The  $\xi_i$  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  $C_1$ ) falls inside the region of separation, but on the right side of the decision surface. (b) Data point  $\mathbf{x}_i$  (belonging to class  $C_2$ ) 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  $w$ , subject to the constraint described in Eq. (6.22) and the constraint on  $\|w\|^2$ . 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  $w$  is a nonconvex optimization problem that is NP-complete.<sup>2</sup>

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  $w$  as follows:

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

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  $\Phi(w, \xi)$  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  $\Phi(w, \xi)$  is optimized with respect to  $w$  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  $w$  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.

## Chapter 6 Support Vector Machines

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*

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

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

*and such that the weight vector  $\mathbf{w}$  and the slack variables  $\xi_i$  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*

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

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

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

Note that neither the slack variables  $\xi_i$  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

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

and

$$\mu_i \xi_i = 0, \quad i = 1, 2, \dots, N \quad (6.26)$$

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  $\mu_i$  are Lagrange multipliers that have been introduced to enforce the nonnegativity of the slack variables  $\xi_i$  for all  $i$ . At the saddle point the derivative of the Lagrangian function for the primal problem with respect to the slack variable  $\xi_i$  is zero, the evaluation of which yields

$$\alpha_i + \mu_i = C \quad (6.27)$$

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

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

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).

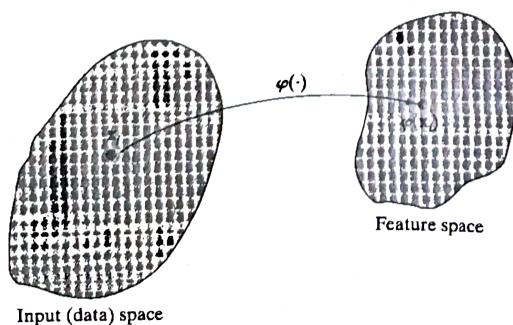
## 6.4 HOW TO BUILD A SUPPORT VECTOR MACHINE FOR PATTERN RECOGNITION

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<sup>3</sup> 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  $\phi(\cdot)$  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 *non-linearly 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:

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

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

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

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

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

where, by definition, we have

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

In effect, the vector  $\varphi(\mathbf{x})$  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:

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

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

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

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

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

The term  $\varphi^T(\mathbf{x}_i) \varphi(\mathbf{x})$  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  $i$ th example. We may therefore introduce the inner-product kernel denoted by  $K(\mathbf{x}, \mathbf{x}_i)$  and defined by

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

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

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

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

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

### 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

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \varphi_i(\mathbf{x}) \varphi_i(\mathbf{x}') \quad (6.39)$$

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  $\lambda_i$  are called *eigenvalues*. The fact that all of the eigenvalues are positive means that the kernel  $K(\mathbf{x}, \mathbf{x}')$  is *positive definite*.

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

- For  $\lambda_i \neq 1$ , the  $i$ th 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.

Merger'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  $\tilde{\mathbf{D}}\mathbf{D}$ , where  $\tilde{\mathbf{D}}$  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*

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

*subject to the constraints:*

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

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

*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

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

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

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

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 $K(\mathbf{x}, \mathbf{x}_i), i = 1, 2, \dots, N$	Comments
Polynomial learning machine	$(\mathbf{x}^T \mathbf{x}_i + 1)^p$	Power $p$ is specified <i>a priori</i> by the user
Radial-basis function network	$\exp\left(-\frac{1}{2\sigma^2} \ \mathbf{x} - \mathbf{x}_i\ ^2\right)$	The width $\sigma^2$ , common to all the kernels, is specified <i>a priori</i> by the user
Two-layer perceptron	$\tanh(\beta_0 \mathbf{x}^T \mathbf{x}_i + \beta_1)$	Merger's theorem is satisfied only for some values of $\beta_0$ and $\beta_1$

- 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.

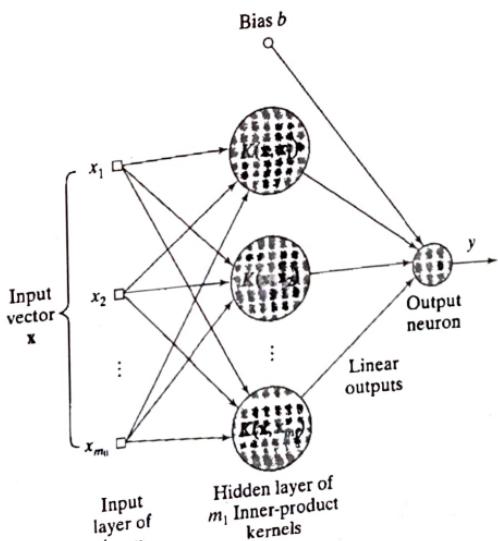


FIGURE 6.5 Architecture of support vector machine.

## 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)

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

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:

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

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

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

Similarly,

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

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))

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

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

$$9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1$$

$$-\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1$$

$$-\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1$$

$$\alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1$$

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

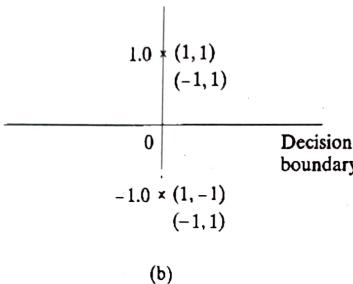
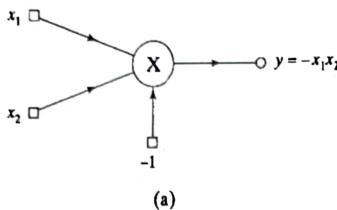
$$\|\mathbf{w}_o\| = \frac{1}{\sqrt{2}}$$

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 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,

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

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  $x_1 = +1, x_2 = -1$ , 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  $\mathcal{C}_1$ ) and 2 (Class  $\mathcal{C}_2$ ). 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:

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

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.

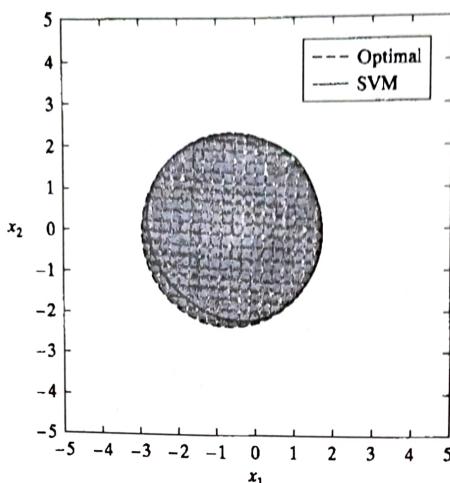
The near-perfect classification performance attained by the support vector machine is further confirmed by the decision boundary shown in Fig. 6.7, which was achieved by one of the five realizations of the machine picked at random. In this figure we have also included the decision boundary for the Bayesian 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  $\mathcal{C}_1$  and  $\mathcal{C}_2$  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 $\epsilon$ -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.