CHAPTER 6

Support Vector Machines

ORGANIZATION OF THE CHAPTER

This chapter is devoted to the study of support vector machines: a machine-learning algorithm that is perhaps the most elegant of all kernel-learning methods. Following the introductory section, Section 6.1, the rest of the chapter is organized as follows:

1. Section 6.2 discusses the construction of an optimal hyperplane for the simple case of linearly separable patterns, which is followed by consideration of the more difficult case of nonseparable patterns in Section 6.3.
2. In Section 6.4, the idea of an inner-product kernel is introduced, thereby building the framework for viewing the learning algorithm involved in the construction of a support vector machine as a kernel method. In that section, we also introduce a widely used notion known as the "kernel trick." The design philosophy of a support vector machine is summed up in Section 6.5, followed by a revisitation of the XOR problem in Section 6.6. The second part of the chapter concludes with a computer experiment on pattern classification, presented in Section 6.7.
3. Section 6.8 introduces the concept of an ε-insensitive loss function, the use of which in solving regression problems is discussed in Section 6.9.
4. Section 6.10 deals with the representer theorem, which provides insight into the formulation of an approximating function in the context of Mercer's kernels.

The chapter concludes with a summary and discussion in Section 6.11.

6.1 INTRODUCTION

In Chapter 4, we studied multilayer perceptrons trained with the back-propagation algorithm. The desirable feature of this algorithm is its simplicity, but the algorithm converges slowly and lacks optimality. In Chapter 5, we studied another class of feedforward networks known as radial-basis function networks, which we developed from interpolation theory; we then described a suboptimal two-stage procedure for its design. In this chapter, we study another category of feedforward networks known collectively as *support vector machines* (SVMs).[1]

Basically, the support vector machine is a binary learning machine with some highly elegant properties. To explain how the machine works, it is perhaps easiest to start with

the case of separable patterns that arise in the context of pattern classification. In this context, the main idea behind the machine may be summed up as follows:

> *Given a training sample, the support vector machine constructs a hyperplane as the decision surface in such a way that the margin of separation between positive and negative examples is maximized.*

This basic idea is extended in a principled way to deal with the more difficult case of nonlinearly separable patterns.

A notion that is central to the development of the support vector learning algorithm is the *inner-product kernel* between a "support vector" $\mathbf{x}_i$ and a vector $\mathbf{x}$ drawn from the input data space. Most importantly, the support vectors consist of a small subset of data points extracted by the learning algorithm from the training sample itself. Indeed, it is because of this central property that the learning algorithm, involved in the construction of a support vector machine, is also referred to as a *kernel method*. However, unlike the suboptimal kernel method described in Chapter 5, the kernel method basic to the design of a support vector machine is *optimal*, with the optimality being rooted in convex optimization. However, this highly desirable feature of the machine is achieved at the cost of increased computational complexity.

As with the design procedures discussed in Chapters 4 and 5, the support vector machine can be used to solve both pattern-classification and nonlinear-regression problems. However, it is in solving difficult pattern-classification problems where support vector machines have made their most significant impact.

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

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

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, equivalently,

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

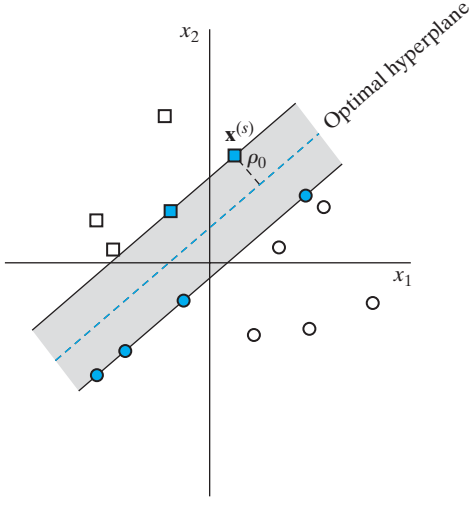


FIGURE 6.1 Illustration of the idea of an optimal hyperplane for linearly separable patterns: The data points shaded in red are support vectors.

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 following 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, if 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." All the remaining examples in the training sample are completely irrelevant. Because of their distinct property, the support 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 optimal hyperplane and are therefore the most difficult to classify. As such, they have a direct bearing on the optimum location of the decision surface.

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

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

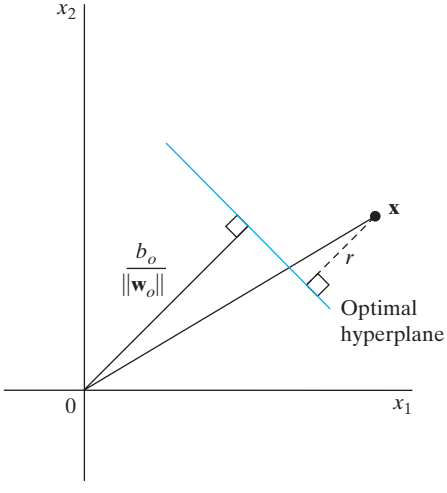


FIGURE 6.2 Geometric interpretation of algebraic distances of points to the optimal hyperplane for a two-dimensional case.

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 sample $\mathcal{T}$. Then, from Eq. (6.8), it follows that

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

Equation (6.9) states the following:

> *Maximizing the margin of separation between binary 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

The support vector machine is cleverly formulated under the umbrella of *convex optimization*[2]—hence the well-defined optimality of the machine. In basic terms, the formulation proceeds along four major steps:

1. The problem of finding the optimal hyperplane starts with a statement of the problem in the primal weight space as a constrained-optimization problem.
2. The Lagrangian function of the problem is constructed.
3. The conditions for optimality of the machine are derived.
4. The stage is finally set for solving the optimization problem in the dual space of Lagrange multipliers.

To proceed then, we first note that the training sample

$$\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^{N}$$

is embodied in the two-line constraint of Eq. (6.6). It is instructive to combine the two lines of this equation into the single line

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

With this form of the constraint at hand, we are now ready to formally state the constrained-optimization problem as follows:

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*

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

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

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

The scaling factor $\frac{1}{2}$ is included here for convenience of presentation. This constrained-optimization problem is called the *primal problem.* It is basically characterized as follows:

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

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

First, we construct the *Lagrangian function*

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

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 $J(\mathbf{w}, b, \alpha)$. A saddle point of a Lagrangian is a point where the roots are real, but of opposite signs; such a singularity is always unstable. The saddle point 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:*

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

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

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

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

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 multipliers α_i.

It is also important to note that for all the constraints that are not satisfied as equalities, the corresponding multiplier α_i must be zero. In other words, only those multipliers that exactly satisfy the condition

$$\alpha_i[d_i(\mathbf{w}^T\mathbf{x}_i + b) - 1] = 0 \tag{6.14}$$

can assume nonzero values. This property is a statement of the *Karush–Kuhn–Tucker conditions*[3] (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*

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

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

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

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 all nonnegative. Note that we have changed the notation from $J(\mathbf{w}, b, \alpha)$ to $Q(\alpha)$ so as to reflect the transformation from the primal optimization problem to its dual.

We may now state the dual problem as follows:

Given the training sample $\mathcal{T} = \{\boldsymbol{x}_i, d_i\}_{i=1}^N$, 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

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

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

Unlike the primal optimization problem based on the Lagrangian of Eq. (6.11), the dual problem defined in Eq. (6.16) 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*

$$\{\mathbf{x}_i^T \mathbf{x}_j\}_{i,j=1}^{N}$$

Typically, the support vectors constitute a subset of the training sample, which means that the solution vector is *sparse.*[4] That is to say, constraint (2) of the dual problem is satisfied with the inequality sign for all the support vectors for which the α's are nonzero, and with the equality sign for all the other data points in the training sample, for which the α's are all zero. Accordingly, having determined the optimum Lagrange multipliers, denoted by $\alpha_{o,i}$, we may compute the optimum weight vector $\mathbf{w}_o$ by using Eq. (6.12) as

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

where N_s is the number of support vectors for which the Lagrange multipliers $\alpha_{o,i}$ are all nonzero. To compute the optimum bias b_o, we may use the $\mathbf{w}_o$ thus obtained and take advantage of Eq. (6.7), which pertains to a positive support vector:

$$\begin{aligned} b_o &= 1 - \mathbf{w}_o^T \mathbf{x}^{(s)} \qquad \text{for } d^{(s)} = 1 \\ &= 1 - \sum_{i=1}^{N_s} \alpha_{o,i} d_i \mathbf{x}_i^T \mathbf{x}^{(s)} \end{aligned} \tag{6.18}$$

Recall that the support vector $\mathbf{x}^{(s)}$ corresponds to any point $(\mathbf{x}_i, d_i)$ in the training sample for which the Lagrange multiplier $\alpha_{o,i}$ is nonzero. From a numerical (practical) perspective, it is better to average Eq. (6.18) over all the support vectors—that is, over all the nonzero Lagrange multipliers.

Statistical Properties of the Optimal Hyperplane

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 the sign, ρ is the margin of separation equal to $2/\|\mathbf{w}_o\|$, and m_0 is the dimensionality of the input space.

As mentioned previously in Chapter 4, the VC dimension, short for *Vapnik–Chervonenkis dimension*, provides a measure of the complexity of a space of functions. The theorem just stated 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 made up of separating hyperplanes described by

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

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 in Eqs. (6.20) and (6.21) are constants.

Equation (6.20) states that the optimal hyperplane is a hyperplane for which the margin of separation between the positive and negative examples is the largest possible. Equivalently, Eq. (6.21) states that construction of the optimal hyperplane is realized by making the squared Euclidean norm of the weight vector $\mathbf{w}$ the smallest possible. In a sense, these two equations reinforce the statement we made previously in light of Eq. (6.9).

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 sample 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 sample.

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 correct side of the decision surface, as illustrated in Fig. 6.3a.

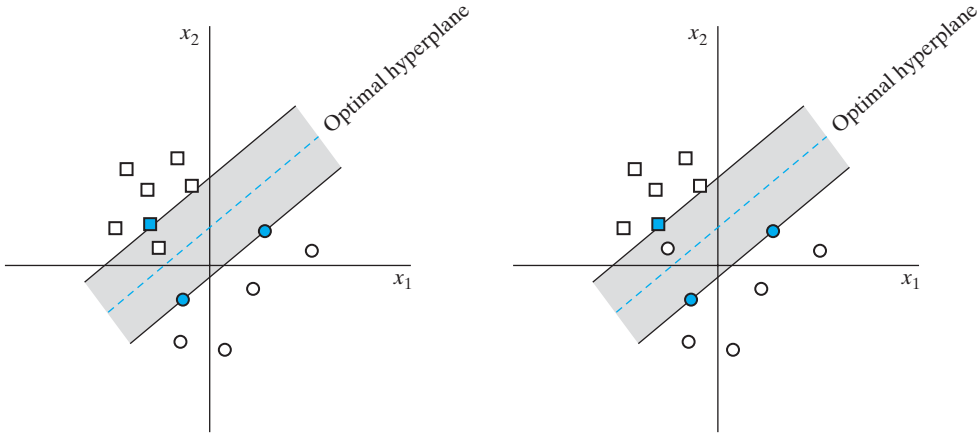


FIGURE 6.3 Soft margin hyperplane (a) Data point $\mathbf{x}_i$ (belonging to class $\mathscr{C}_1$, represented by a small square) falls inside the region of separation, but on the correct side of the decision surface. (b) Data point $\mathbf{x}_i$ (belonging to class $\mathscr{C}_2$, represented by a small circle) falls on the wrong side of the decision surface.

- 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 the first case, but misclassification in the second.

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:

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

The ξ_i are called *slack variables*; they measure the deviation of a data point from the ideal condition of pattern separability. For $0 < \xi_i \leq 1$, the data point falls inside the region of separation, but on the correct 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) precisely even if $\xi_i > 0$. Moreover, there can be support vectors satisfying the condition $\xi_i = 0$. Note that if an example with $\xi_i > 0$ is left out of the training sample, the decision surface will 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 over the training sample, 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.[5]

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 the support vector machine. As for the second term, $\sum_i \xi_i$, it is an upper bound on the number of test errors.

The parameter C controls the tradeoff between complexity of the machine and the number of nonseparable points; it may therefore be viewed as the *reciprocal* of a parameter commonly referred to as the "regularization" parameter.[6] When the parameter C is assigned a large value, the implication is that the designer of the support vector machine has high confidence in the quality of the training sample $\mathcal{T}$. Conversely, when C is assigned a small value, the training sample $\mathcal{T}$ is considered to be noisy, and less emphasis should therefore be placed on it.

In any event, the parameter C has to be selected by the user. It may be determined *experimentally* via the standard use of a training (validation) sample, which is a crude form of resampling; the use of cross-validation for optimum selection of regularization parameter (i.e., $1/C$) is discussed in Chapter 7.

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 follows:

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

$$d_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \qquad \text{for } i = 1, 2, \ldots, N \tag{6.24}$$

$$\xi_i \geq 0 \qquad \text{for all } i \tag{6.25}$$

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

$$\Phi(\mathbf{w}, \xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \tag{6.26}$$

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 follows (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*

$$Q(\alpha) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j \mathbf{x}_i^T\mathbf{x}_j \tag{6.27}$$

subject to the constraints

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

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

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

Note that neither the slack variables ξ_i nor their own 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 $\alpha_i \ge 0$ is replaced with the more stringent constraint $0 \le \alpha_i \le C$. 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.

Unbounded Support Vectors

For a prescribed parameter C, a data point $(\mathbf{x}_i, d_i)$ for which the condition $0 < \alpha_i < C$ holds is said to be an *unbounded*, or *free support vector*. When $\alpha_i = C$, we find that

$$d_i F(\mathbf{x}_i) \le 1, \qquad \alpha_i = C$$

where $F(\mathbf{x}_i)$ is the approximating function realized by the support vector machine for the input $\mathbf{x}_i$. On the other hand, when $\alpha_i = 0$, we find that

$$d_i F(\mathbf{x}_i) \ge 1, \qquad \alpha_i = 0$$

In light of these two arguments, it follows that for unbounded support vectors, we have

$$d_i F(\mathbf{x}_i) = 1$$

Unfortunately, the converse argument does not hold; that is, even if we know that $d_i F(\mathbf{x}_i) = 1$ for a particular data point $(\mathbf{x}_i, d_i)$, this condition does not necessarily tell us anything about the corresponding Lagrange multiplier α_i.

Consequently, there is a distinct possibility of degeneracy (i.e., reduced optimality conditions) in the solution to a pattern-classification problem computed by the support vector machine. By this statement, we mean that a point $(\mathbf{x}_i, d_i)$ that satisfies the margin requirement exactly has no constraint on the possible value of the associated α_i.

In Rifkin (2002), it is argued that the number of unbounded support vectors is the primary reason for how difficult, in a computational sense, the training of a support vector machine can be.

Underlying Philosophy of a Support Vector Machine for Pattern Classification

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 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;

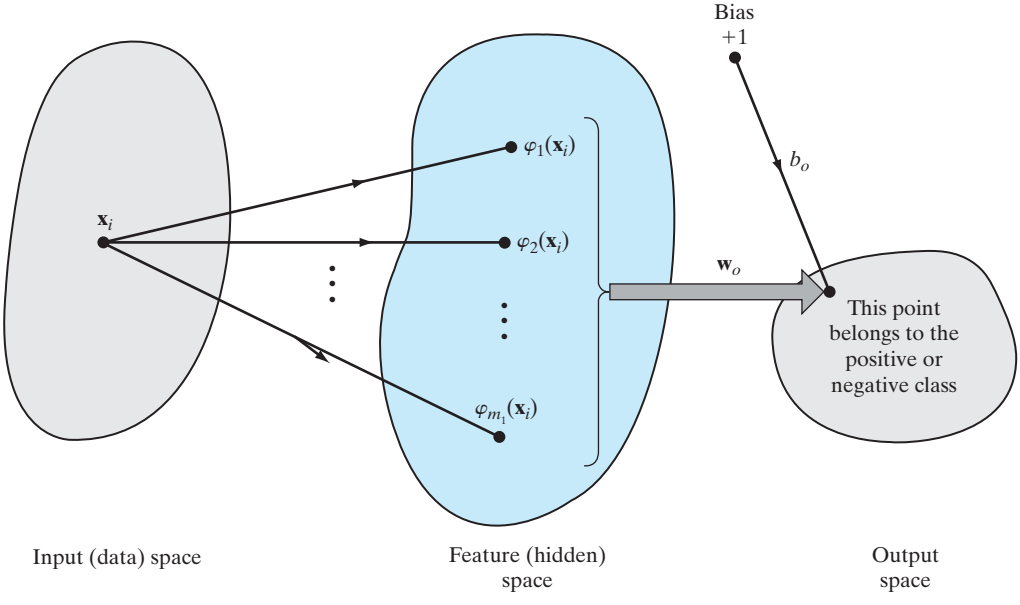


FIGURE 6.4 Illustrating the two mappings in a support vector machine for pattern classification: (i) nonlinear mapping from the input space to the feature space; (ii) linear mapping from the feature space to the output space.

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 the upcoming text.

One last important comment is in order. The number of features constituting the hidden space in Fig. 6.4 is determined by the number of support vectors. Thus, SVM theory provides an analytic approach for determining the optimum size of the feature (hidden) space, thereby assuring optimality of the classification task.

6.4 THE SUPPORT VECTOR MACHINE VIEWED AS A KERNEL MACHINE

Inner-Product Kernel

Let $\mathbf{x}$ denote a vector drawn from the input space of dimension m_0. Let $\{\varphi_j(\mathbf{x})\}_{j=1}^{\infty}$ denote a set of nonlinear functions that, between them, transform the input space of dimension m_0 to a feature space of infinite dimensionality. Given this transformation, we may define a hyperplane acting as the decision surface in accordance with the formula

$$\sum_{j=1}^{\infty} w_j \varphi_j(\mathbf{x}) = 0 \tag{6.28}$$

where $\{w_j\}_{j=1}^{\infty}$ denotes an infinitely large set of weights that transforms the feature space to the output space. It is in the output space where the decision is made on whether the input vector $\mathbf{x}$ belongs to one of two possible classes, positive or negative. For convenience of presentation, we have set the bias to zero in Eq. (6.28). Using matrix notation, we may rewrite this equation in the compact form

$$\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) = 0 \tag{6.29}$$

where $\boldsymbol{\phi}(\mathbf{x})$ is the *feature vector* and $\mathbf{w}$ is the corresponding *weight vector.*

As in Section 6.3, we seek "linear separability of the transformed patterns" in the feature space. With this objective in mind, we may adapt Eq. (6.17) to our present situation by expressing the weight vector as

$$\mathbf{w} = \sum_{i=1}^{N_s} \alpha_i d_i \boldsymbol{\phi}(\mathbf{x}_i) \tag{6.30}$$

where the feature vector is expressed as

$$\boldsymbol{\phi}(\mathbf{x}_i) = [\varphi_1(\mathbf{x}_i), \varphi_2(\mathbf{x}_i), \ldots]^T \tag{6.31}$$

and N_s is the number of support vectors. Hence, substituting Eq. (6.29) into Eq. (6.30), we may express the decision surface in the output space as

$$\sum_{i=1}^{N_s} \alpha_i d_i \boldsymbol{\phi}^T(\mathbf{x}_i) \boldsymbol{\phi}(\mathbf{x}) = 0 \tag{6.32}$$

We now immediately see that the scalar term $\boldsymbol{\phi}^T(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x})$ in Eq. (6.32) represents an *inner product.* Accordingly, let this inner-product term be denoted as the scalar

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}_i) &= \boldsymbol{\phi}^T(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}) \\ &= \sum_{j=1}^{\infty} \varphi_j(\mathbf{x}_i)\varphi_j(\mathbf{x}), \qquad i = 1, 2, ..., N_s \end{aligned} \tag{6.33}$$

Correspondingly, we may express the optimal decision surface (hyperplane) in the output space as

$$\sum_{i=1}^{N_s} \alpha_i d_i k(\mathbf{x}, \mathbf{x}_i) = 0 \tag{6.34}$$

The function $k(\mathbf{x}, \mathbf{x}_i)$ is called the *inner-product kernel*,[7] or simply the *kernel*, which is formally defined as follows (Shawe-Taylor and Cristianini, 2004):

The kernel $k(\mathbf{x}, \mathbf{x}_i)$ is a function that computes the inner product of the images produced in the feature space under the embedding $\boldsymbol{\phi}$ of two data points in the input space.

Following the definition of a kernel introduced in Chapter 5, we may state that the kernel $k(\mathbf{x}, \mathbf{x}_i)$ is a function that has two basic properties[8]:

Property 1. *The function $k(\mathbf{x}, \mathbf{x}_i)$ is symmetric about the center point $\mathbf{x}_i$, that is,*

$$k(\mathbf{x}, \mathbf{x}_i) = k(\mathbf{x}_i, \mathbf{x}) \qquad \text{for all } \mathbf{x}_i$$

and it attains its maximum value at the point $\mathbf{x} = \mathbf{x}_i$.

Note, however, that the maximum need not exist; for example, $k(\mathbf{x}, \mathbf{x}_i) = \mathbf{x}^T\mathbf{x}_i$, as a kernel does not have a maximum.

Property 2. *The total volume under the surface of the function $k(\mathbf{x}, \mathbf{x}_i)$ is a constant.*

If the kernel $k(\mathbf{x}, \mathbf{x}_i)$ is appropriately scaled to make the constant under property 2 equal to unity, then it will have properties similar to those of the probability density function of a random variable.

The Kernel Trick

Examining Eq. (6.34), we may now make two important observations:

1. Insofar as pattern classification in the output space is concerned, specifying the kernel $k(\mathbf{x}, \mathbf{x}_i)$ is *sufficient*; in other words, we need never explicitly compute the weight vector $\mathbf{w}_o$; it is for this reason that the application of Eq. (6.33) is commonly referred to as the *kernel trick*.
2. Even though we assumed that the feature space could be of infinite dimensionality, the linear equation of Eq. (6.34), defining the optimal hyperplane, consists of a *finite* number of terms that is equal to the number of training patterns used in the classifier.

It is in light of observation 1 that the support vector machine is also referred to as a *kernel machine*. For pattern classification, the machine is parameterized by an N-dimensional vector whose ith term is defined by the product $\alpha_i d_i$ for $i = 1, 2, ..., N$.

We may view $k(\mathbf{x}_i, \mathbf{x}_j)$ as the ij-th element of the symmetric N-by-N matrix

$$\mathbf{K} = \{k(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^{N} \tag{6.35}$$

The matrix $\mathbf{K}$ is a nonnegative definite matrix called the *kernel matrix*; it is also referred to simply as the *Gram*. It is nonnegative definite or positive semidefinite in that it satisfies the condition

$$\mathbf{a}^T\mathbf{K}\mathbf{a} \geq 0$$

for any real-valued vector $\mathbf{a}$ whose dimension is compatible with that of $\mathbf{K}$.

Mercer's Theorem

The expansion of Eq. (6.33) for the symmetric 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 follows (Mercer, 1909; 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

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \varphi_i(\mathbf{x})\varphi_i(\mathbf{x}') \tag{6.36}$$

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

$$\int_{\mathbf{b}}^{\mathbf{a}} \int_{\mathbf{b}}^{\mathbf{a}} k(\mathbf{x}, \mathbf{x}')\psi(\mathbf{x})\psi(\mathbf{x}')d\mathbf{x}d\mathbf{x}' \geq 0 \tag{6.37}$$

holds for all $\psi(\cdot)$, for which we have

$$\int_{\mathbf{b}}^{\mathbf{a}} \psi^2(\mathbf{x})d\mathbf{x} < \infty \tag{6.38}$$

where $\mathbf{a}$ and $\mathbf{b}$ are the constants of integretion.

The features $\varphi_i(\mathbf{x})$ 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 kernel $k(\mathbf{x}, \mathbf{x}')$ is *positive definite*. This property, in turn, means that we have a complex problem that can be solved efficiently for the weight vector $\mathbf{w}$, as discussed next.

Note, however, that Mercer's theorem tells us only whether a candidate kernel is actually an inner-product kernel in some space and therefore admissible for use in a support vector machine. It says nothing about how to construct the functions $\varphi_i(\mathbf{x})$; we have to do that ourselves. Nevertheless, Mercer's theorem is important because it places a limit on the number of admissible kernels. Note also that the expansion of Eq. (6.33) is a special case of Mercer's theorem, since all the eigenvalues of this expansion are unity. It is for this reason that an inner-product kernel is also referred to as a *Mercer kernel*.

6.5 DESIGN OF SUPPORT VECTOR MACHINES

The expansion of the kernel $k(\mathbf{x}, \mathbf{x}_i)$ in Eq. (6.33) permits us to construct a decision surface that is nonlinear in the input space, but whose 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*

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{6.39}$$

subject to the constraints

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

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

where C is a user-specified positive parameter.

Constraint (1) arises from optimization of the Lagrangian $Q(\alpha)$ with respect to the bias b, which is a rewrite of Eq. (6.13). 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 $\mathbf{x}_i^T \mathbf{x}_j$ has been replaced by the Mercer kernel $k(\mathbf{x}, \mathbf{x}_i)$.

Examples of Support Vector Machines

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 the kernel is chosen. In Table 6.1, we summarize the 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 Mercer kernels for polynomial and radial-basis-function types of support vector machines always satisfy Mercer's theorem. In contrast, the Mercer kernel for

TABLE 6.1 Summary of Mercer Kernels

Type of support vector machine	Mercer 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

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 a given kernel satisfies Mercer's theorem can indeed be a difficult matter.

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.
4. 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.

Figure 6.5 displays the architecture of a support vector machine, where m_1 denotes the size of the hidden layer (i.e., feature space).

Regardless 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, 1998; Schölkopf and Smola, 2002):

- *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

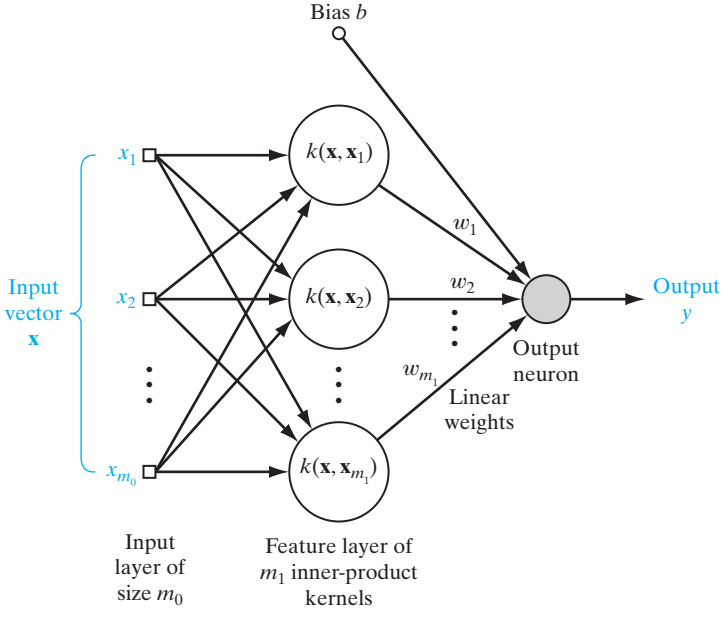


FIGURE 6.5
Architecture of support vector machine, using a radial-basis function network.

hyperplane in that space. For good generalization performance, the model complexity is controlled by imposing certain constraints on the construction of the separating soft-margin hyperplane, which results in the extraction of a fraction of the training data as support vectors.

- *Computational problem.* The need to compute the weight vector and the bias in the output layer of the RBF network is avoided by using the kernel trick.

6.6 XOR PROBLEM

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, define the following kernel (Cherkassky and Mulier, 1998):

$$k(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}^T\mathbf{x}_i)^2 \tag{6.40}$$

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

$$k(\mathbf{x}, \mathbf{x}_i) = 1 + x_1^2x_{i1}^2 + 2x_1x_2x_{i1}x_{i2} + x_2^2x_{i2}^2 + 2x_1x_{i1} + 2x_2x_{i2} \tag{6.41}$$

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

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

Similarly,

$$\boldsymbol{\phi}(\mathbf{x}_i) = [1, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}]^T, \qquad i = 1, 2, 3, 4 \tag{6.42}$$

Using the definition of Eq. (6.35), we obtain the Gram

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

TABLE 6.2 XOR Problem

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

The objective function for the dual form of optimization is therefore as follows (see Eq. (6.39)):

$$\begin{aligned} Q(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 \\ + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2) \end{aligned} \tag{6.43}$$

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

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

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. (6.30), we find that the optimum weight vector is

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

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

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

The optimal hyperplane is defined by

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

Expanding the inner product $\mathbf{w}_o^T \boldsymbol{\phi}(\mathbf{x})$ yields:

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

which reduces to

$$-x_1x_2 = 0$$

The polynomial form of support vector machine for the XOR problem is therefore 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.

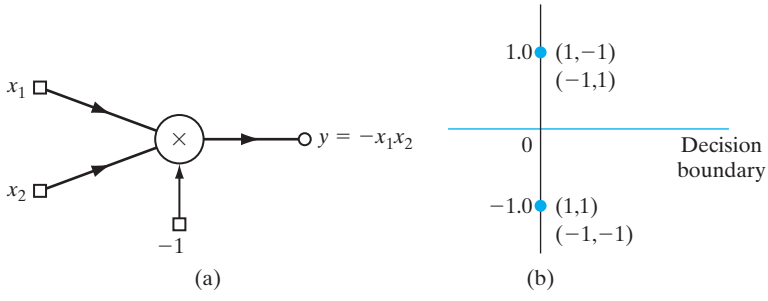


FIGURE 6.6 (a) Polynomial machine for solving the XOR problem. (b) Induced images in the feature space due to the four data points of the XOR problem.

6.7 COMPUTER EXPERIMENT: PATTERN CLASSIFICATION

In this section, we continue the sequence of pattern-classification experiments based on the double-moon problem depicted in Fig. 1.8. This time, we use the nonlinear support vector machine with a single hidden layer. The experiment was repeated for two different settings of the vertical separation between the two moons of Fig. 1.8, namely, $d = -6.0$ and $d = -6.5$. The parameter C was set equal to infinity for both parts of the experiment. The training sample consisted of 300 data points, and the test sample consisted of 2,000 data points. The training data were preprocessed in the same manner as described in Section 1.5.

The scenario corresponding to the distance $d = -6.0$ was chosen for the first part of the experiment so as to provide an illustrative way of comparing the SVM with the "K-means, RLS" algorithm used to train the RBF network in Chapter 5, for which a small number of classification errors was reported. Figure 6.7 presents the corresponding results of the experiment using the SVM with $d = -6.0$. Part (a) of the figure presents the training result, displaying the support vectors and the optimal decision boundary computed by the algorithm. For this part of the experiment, there were no classification errors when the machine was tested with data not seen before, as shown in part (b) of the figure.

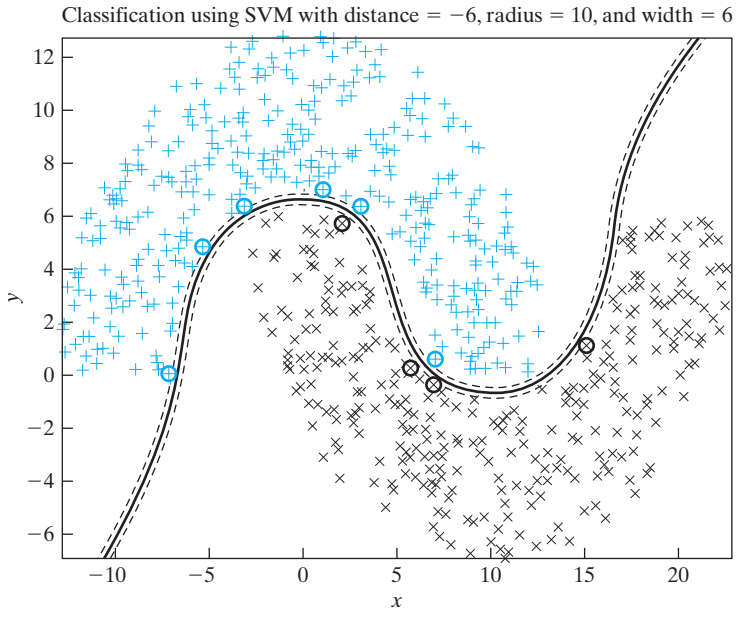
Figure 6.8 presents the results of the second part of the experiment, which used the SVM for a more difficult scenario for which the vertical separation between the two moons was reduced to $d = -6.5$. Once again, part (a) of the figure presents the training result, showing the support vectors and the decision boundary computed by the algorithm. Part (b) of the figure shows the corresponding test result. This time, there were 11 classification errors in the test sample of 2,000 data points, representing a classification error of 0.55%.

As already stated, both parts of the experiment were performed using the common value $C = \infty$. In this context, the following two points are noteworthy:

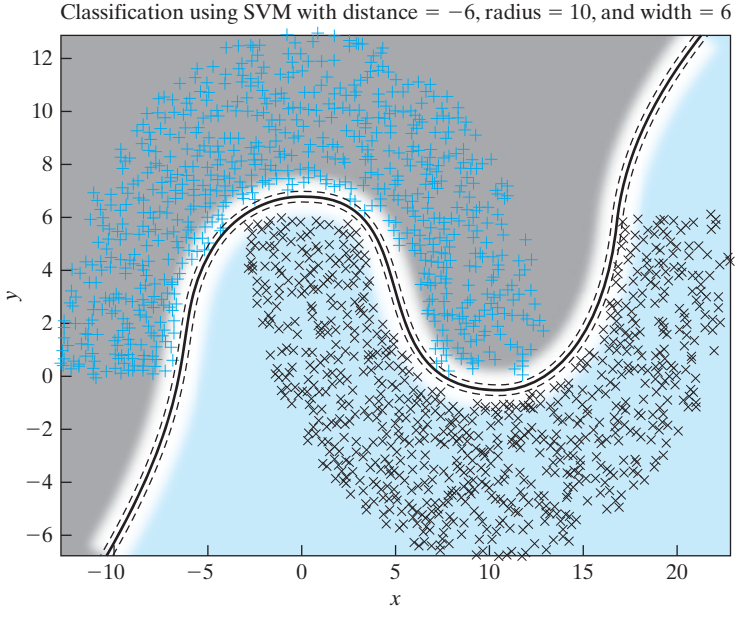
1. For $d = -6.0$, the two moons are perfectly nonlinearly separable; this observation is confirmed by the complete absence of classification errors on test data, as demonstrated in Fig. 6.7b.
2. For $d = -6.5$, the two moons of Fig. 1.8 overlap slightly. Consequently, they are no longer separable, which is confirmed by the small number of classification errors that were found on test data in Fig. 6.8b. In this second part of the experiment, no attempt was made to find the optimal value of parameter C so as to minimize the classification error rate; this issue is addressed in Problem 6.24.

6.8 REGRESSION: ROBUSTNESS CONSIDERATIONS

Up to this point in the chapter, we have focused attention on the use of support vector machines for solving pattern-recognition problems. In this section, we prepare the stage for studying the use of support vector machines to solve regression problems. To do this, we will first address the issue of a suitable optimization criterion with *robustness* as a primary objective. With this objective in mind, we need a model that is insensitive to small changes in the model parameters, which is addressed next.

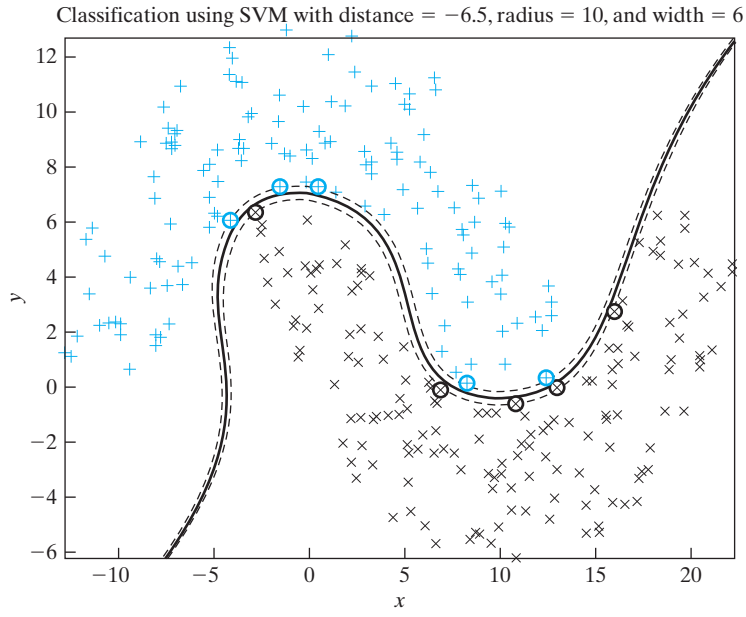


(a) Training result

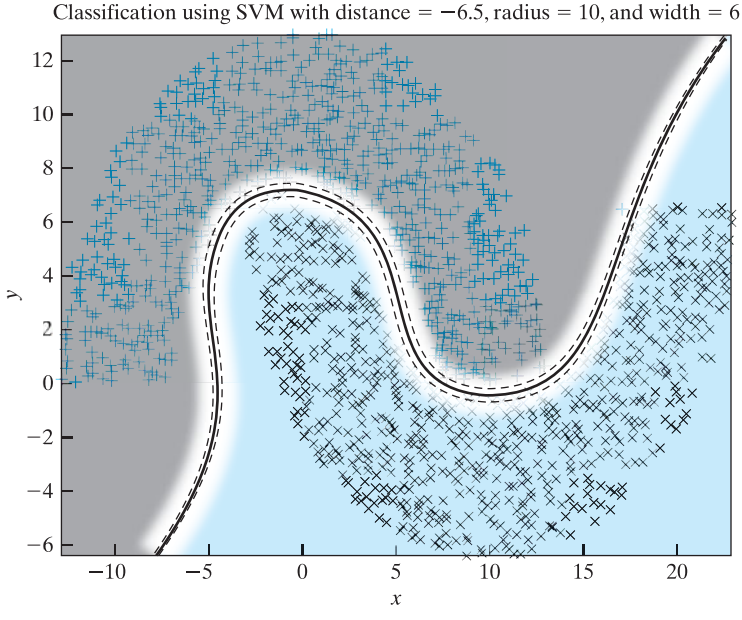


(b) Testing result

FIGURE 6.7 Experiment on SVM for the double-moon of Fig. 1.8 with distance $d = -6$.



(a) Training result



(b) Testing result

FIGURE 6.8 Experiment on SVM for the double-moon of Fig. 1.8 with distance $d = -6.5$.

ε-Insensitive Loss Function

With robustness as a design goal, any quantitative measure of robustness should 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 is therefore a *minimax* procedure[9] 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 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 $y = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})$ is the corresponding 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), in the form

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

where ε is a prescribed parameter. The loss function $L_\varepsilon(d, y)$ 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; otherwise, it is equal to the absolute value of the deviation minus ε. The loss function of Eq. (6.44) is a special case of the ε-insensitive loss function for $\varepsilon = 0$. Parts a and b of Figure 6.9 illustrate the *ε-insensitive tube* and the corresponding dependence of the loss $L_\varepsilon(d, y)$ on the error $(d - y)$.

With the ε-insensitive loss function of Eq. (6.45) as a basis for robustification, the stage is set for applying SVM theory to solve linear regression problems, as discussed next.

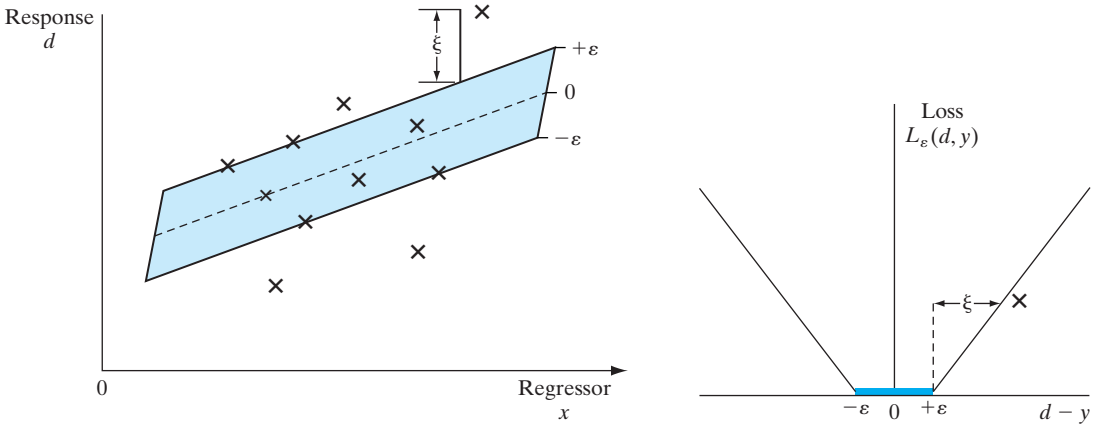


FIGURE 6.9 Linear regression (a) Illustrating an ε-insensitive tube of radius ε, fitted to the data points shown as ×'s. (b) The corresponding plot of the ε-insensitive loss function.

6.9 OPTIMAL SOLUTION OF THE LINEAR REGRESSION PROBLEM

Consider a linear regression model, in which the dependence of a scalar observable d on a regressor $\mathbf{x}$ is described by

$$d = \mathbf{w}^T\mathbf{x} + b \tag{6.46}$$

where the parameter vector $\mathbf{w}$ and the bias b are both unknown. The problem is to compute estimates of $\mathbf{w}$ and b, given the training sample $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$, where the data are statistically independent and identically distributed (iid).

Given the training sample $\mathcal{T}$, consider the *risk functional*

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}|y_i - d_i|_\varepsilon \tag{6.47}$$

where the summation accounts for the ε-insensitive training error and C is a constant that determines the tradeoff between the training error and the penalizing term $\|\mathbf{w}\|^2$. The y_i is the estimator output produced in response to the input example $\mathbf{x}_i$. The requirement is to do the following:

Minimize the risk functional of Eq. (6.47) subject to the following constraints:

$$d_i - y_i \le \varepsilon + \xi_i \tag{6.48}$$

$$y_i - d_i \le \varepsilon + \xi_i' \tag{6.49}$$

$$\xi_i \ge 0 \tag{6.50}$$

$$\xi_i' \ge 0 \tag{6.51}$$

for $i = 1, 2, \ldots, N$. The ξ_i and ξ_i' are two sets of nonnegative slack variables that describe the ε-sensitive loss function in Eq. (6.45).

To solve this optimization problem for the Lagrange multipliers α_i and α_i', we will proceed in a manner similar to that pursued in Section 6.2 for the design of a support vector machine for linearly separable patterns. First, we construct a Lagrangian function (including the constraints), and then we go on introduce the corresponding dual set of variables. Specifically, we first to write

$$\begin{aligned} J(\mathbf{w}, \xi, \xi_i', \alpha, \alpha', \gamma, \gamma') = {} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i') - \sum_{i=1}^{N}(\gamma_i\xi_i + \gamma_i'\xi_i') \\ & - \sum_{i=1}^{N}\alpha_i(\mathbf{w}^T\mathbf{x}_i + b - d_i + \varepsilon + \xi_i) \\ & - \sum_{i=1}^{N}\alpha_i'(d_i - \mathbf{w}^T\mathbf{x}_i - b + \varepsilon + \xi_i') \end{aligned} \tag{6.52}$$

where, as before, the α_i and α_i' are the Lagrange multipliers. The new multipliers γ_i and γ_i' are introduced in Eq. (6.52) to ensure that the optimality constraints on the Lagrange multipliers α_i and α_i' assume variable forms. The requirement is to minimize the Lagrangian function of Eq. (6.52) with respect to the regression model's parameters $\mathbf{w}$ and b, as well as the slack variables ξ and ξ'.

Carrying out this optimization as just stated, and setting the resulting derivatives equal to zero, we respectively obtain

$$\hat{\mathbf{w}} = \sum_{i=1}^{N} (\alpha_i - \alpha_i')\mathbf{x} \tag{6.53}$$

$$\sum_{i=1}^{N} (\alpha_i - \alpha_i') = 0 \tag{6.54}$$

$$\alpha_i + \gamma_i = C, \qquad i = 1, 2, ..., N \tag{6.55}$$

$$\alpha_i' + \gamma_i' = C, \qquad i = 1, 2, ..., N \tag{6.56}$$

The *support-vector expansion* of Eq. (6.53) defines the desired parameter estimate $\hat{\mathbf{w}}$ in terms of the computed Lagrange multipliers α_i and α_i'. To find the corresponding estimate of the bias, denoted by $\hat{b}$, we exploit the *Karush–Kuhn–Tucker conditions*. From the discussion presented in Section 6.2, we infer that in order to conform to these conditions, for all the constraints that are not satisfied as equalities, the corresponding variables of the dual problem must vanish. For the problem at hand we thus have two sets of constraints:

- One set is described by the inequalities of Eqs. (6.48) and (6.49), for which the dual variables are α_i and α_i', respectively.
- The second set is described by the inequalities of Eqs. (6.50) and (6.51), for which the dual variables are γ_i and γ_i', respectively; from Eqs. (6.55) and (6.56), we find that $\gamma_i = C - \alpha_i$ and $\gamma_i' = C - \alpha_i'$.

Accordingly, applying the Karush–Kuhn–Tucker conditions to these four constraints in accordance with their pertinent dual variables, we respectively obtain

$$\alpha_i(\varepsilon + \xi_i + d_i - y_i) = 0 \tag{6.57}$$

$$\alpha_i'(\varepsilon + \xi_i' - d_i + y_i) = 0 \tag{6.58}$$

$$(C - \alpha_i)\xi_i = 0 \tag{6.59}$$

$$(C - \alpha_i')\xi_i' = 0 \tag{6.60}$$

Examining these four equations, we draw three important conclusions:

1. Equations (6.59) and (6.60) tell us that the examples $(\mathbf{x}_i, d_i)$ for which $\alpha_i = C$ and $\alpha_i' = C$ are the only ones that can lie outside the slack variables $\xi_i > 0$ and $\xi_i' > 0$; these slack variables correspond to points lying outside the ε-insensitive tube centered around the regression function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, as depicted in Fig. 6.10a.
2. Multiplying Eq. (6.57) by α_i', multiplying Eq. (6.58) by α_i, and then adding the resulting equations, we obtain

$$\alpha_i\alpha_i'(2\varepsilon + \xi_i + \xi_i') = 0$$

Hence, for $\varepsilon > 0$, and with $\xi_i > 0$ and $\xi_i' > 0$, we have the condition

$$\alpha_i\alpha_i' = 0$$

from which it follows that there can never be a situation where the pair of Lagrange multipliers α_i and α_i' are both simultaneously nonzero.

3. From Eqs. (6.59) and (6.60), we respectively observe that

$$\xi_i = 0 \qquad \text{for } 0 < \alpha_i < C$$
$$\xi_i' = 0 \qquad \text{for } 0 < \alpha_i' < C$$

Under these two conditions, the respective Eqs. (6.57) and (6.58) show us that

$$\varepsilon - d_i + y_i = 0, \quad \text{for } 0 < \alpha_i < C \tag{6.61}$$
$$\varepsilon + d_i - y_i = 0, \quad \text{for } 0 < \alpha_i' < C \tag{6.62}$$

With Eqs. (6.61) and (6.62) at hand, we can now compute the bias estimate $\hat{b}$. First, we recognize that the output of the optimum estimator of the regression function is defined by

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

For the example $\mathbf{x}_i$ as the input, we write

$$y = \hat{\mathbf{w}}^T\mathbf{x}_i + \hat{b} \tag{6.63}$$

Substituting Eq. (6.63) into Eqs. (6.61) and (6.62) and then solving for $\hat{b}$, we respectively obtain

$$\hat{b} = d_i - \hat{\mathbf{w}}^T\mathbf{x}_i - \varepsilon \qquad \text{for } 0 < \alpha_i < C \tag{6.64}$$

and

$$\hat{b} = d_i - \hat{\mathbf{w}}^T\mathbf{x}_i + \varepsilon \qquad \text{for } 0 < \alpha_i' < C \tag{6.65}$$

Hence, knowing $\hat{\mathbf{w}}$ from Eq. (6.53) and given both ε and d_i, we may compute the bias estimate $\hat{b}$.

For the computation of $\hat{b}$, in theory, we may use any Lagrange multiplier that lies inside the range $(0, C)$. However, in practice, it is prudent to use the average value computed over all the Lagrange multipliers in that range.

Sparseness of the Support Vector Expansion

From Eqs. (6.57) and (6.58), we find that for all the examples that lie inside the ε-insensitive tube, we have

$$|d_i - y_i| \geq \varepsilon$$

Under this condition, the factors inside the parentheses of both equations are nonvanishing; hence, for Eqs. (6.57) and (6.58) to hold (i.e., for the Karush–Kuhn–Tucker conditions to be satisfied), we do not need all the examples $\mathbf{x}_i$ to compute the desired estimate $\hat{\mathbf{w}}$. In other words, the computed support vector expansion of Eq. (6.53) is *sparse*.

The examples for which the Lagrange multipliers α_i and α_i' are nonvanishing define the support vectors. Insofar as the solution of Eq. (6.53) is concerned, it is geometrically plausible that the examples that lie inside the ε-insensitive tube do not contribute to this solution. The implication of this statement is that those particular examples do not contain meaningful information about the solution (Schölkopf and Smola, 2002).

6.10 THE REPRESENTER THEOREM AND RELATED ISSUES

We complete the discussion of kernel machines (inclusive of support vector machines), be they linear or nonlinear, by establishing the Representer Theorem, which adds a great deal of insight into our understanding of this important class of learning machines. To pave the way for proving this theorem, we will first describe what is meant by a Hilbert space and then by a reproducing-kernel Hilbert space.

Hilbert Space[10]

Let $\{\mathbf{x}_k\}_{k=1}^{\infty}$ be an *orthonormal basis* for an *inner-product space* $\mathcal{F}$ that is assumed to be of infinite dimensionality. As a reminder, two vectors $\mathbf{x}_j$ and $\mathbf{x}_k$ are said to be *orthonormal* if they satisfy the twofold condition

$$\mathbf{x}_j^T \mathbf{x}_k = \begin{cases} 1 & \text{for } j = k \\ 0 & \text{otherwise} \end{cases} \tag{6.66}$$

where the first part pertains to the *normalization property* and the second to the *orthogonality property*. The space $\mathcal{F}$ so defined is called a *pre-Hilbert space*. The *normed space*, in which every vector has a finite Euclidean norm (length), is a special case of the pre-Hilbert space.

Let $\mathcal{H}$ be the largest and most inclusive space of vectors for which the infinite set $\{\mathbf{x}_k\}_{k=1}^{\infty}$ is a basis. Then, vectors not necessarily lying in the space $\mathcal{F}$ represented in the form

$$\mathbf{x} = \sum_{k=1}^{\infty} a_k \mathbf{x}_k \tag{6.67}$$

are said to be spanned by the basis $\{\mathbf{x}_k\}_{k=1}^{\infty}$; the a_k are the coefficients of the representation. Define the new vector

$$\mathbf{y}_n = \sum_{k=1}^{n} a_k \mathbf{x}_k \tag{6.68}$$

Another vector $\mathbf{y}_m$ may be similarly defined. For $n > m$, we may express the squared Euclidean distance between the vectors $\mathbf{y}_n$ and $\mathbf{y}_m$ as

$$\begin{aligned} \|\mathbf{y}_n - \mathbf{y}_m\|^2 &= \left\| \sum_{k=1}^{n} a_k \mathbf{x}_k - \sum_{k=1}^{m} a_k \mathbf{x}_k \right\|^2 \\ &= \left\| \sum_{k=m+1}^{n} a_k \mathbf{x}_k \right\|^2 \\ &= \sum_{k=m+1}^{n} a_k^2 \end{aligned} \tag{6.69}$$

where, in the last line, we invoked the twofold orthonormality condition of Eq. (6.66).

In view of Eq. (6.69), we infer the following:

1. $\sum_{k=m+1}^{n} a_k^2 \to 0$ as both $n, m \to \infty$.
2. $\sum_{k=1}^{m} a_k^2 < \infty$

Moreover, for some positive ε we may pick an integer m large enough to satisfy the inequality

$$\sum_{m+1}^{\infty} a_k^2 < \varepsilon$$

Since

$$\sum_{k=1}^{\infty} a_k^2 = \sum_{k=1}^{m} a_k^2 + \sum_{k=m+1}^{\infty} a_k^2$$

it therefore follows that

$$\sum_{k=1}^{\infty} a_k^2 < \infty \tag{6.70}$$

A sequence of vectors $\{\mathbf{y}_k\}_{k=1}^{n}$ in a normed space, for which the Euclidean distance between $\mathbf{y}_n$ and $\mathbf{y}_m$ satisfies the condition

$$\|\mathbf{y}_n - \mathbf{y}_m\| < \varepsilon \quad \text{for any } \varepsilon > 0 \text{ and all } m, n > M,$$

is a convergent sequence; such a sequence is called a *Cauchy sequence.* Note that all convergent sequences are Cauchy sequences, but not all Cauchy sequences are convergent.

Consequently, a vector $\mathbf{x}$ can be expanded on the basis $\{\mathbf{x}_k\}_{k=1}^{\infty}$ if, and only if, $\mathbf{x}$ is a linear combination of the basis vectors and the associated coefficients $\{a_k\}_{k=1}^{\infty}$ are square summable. Conversely, the square summability of the set of coefficients $\{a_k\}_{k=1}^{\infty}$ implies that the squared Euclidean distance $\|\mathbf{y}_n - \mathbf{y}_m\|^2$ approaches zero as both n and m approach infinity, and the convergent sequence $\{\mathbf{y}_n\}_{n=1}^{\infty}$ is a Cauchy sequence.

From this discussion, it is apparent that the space $\mathcal{H}$ is more "complete" than the inner-product space $\mathcal{F}$. We may therefore make the following important statement:

> *An inner-product space $\mathcal{H}$ is complete if every Cauchy sequence of vectors taken from the space $\mathcal{H}$ converges to a limit in $\mathcal{H}$; a complete inner-product space is called a* Hilbert space.

Indeed, it is in view of this statement, that the inner-product space $\mathcal{F}$, in terms of which we started the discussion, is referred to as pre-Hilbert space.

Reproducing-Kernel Hilbert Space[11]

Consider a Mercer kernel $k(\mathbf{x}, \cdot)$, where the vector $\mathbf{x} \in \mathcal{X}$, and let $\mathcal{F}$ be any vector space of all real-valued functions of $\mathbf{x}$ that are generated by the kernel $k(\mathbf{x}, \cdot)$.

Suppose now two functions $f(\cdot)$ and $g(\cdot)$ are picked from the space $\mathcal{F}$ that are respectively represented by

$$f(\cdot) = \sum_{i=1}^{l} a_i k(\mathbf{x}_i, \cdot) \tag{6.71}$$

and

$$g(\cdot) = \sum_{j=1}^{n} b_j k(\tilde{\mathbf{x}}_j, \cdot) \tag{6.72}$$

where the a_i and the b_j are expansion coefficients and both $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_j \in \mathcal{X}$ for all i and j.

Given the functions $f(\cdot)$ and $g(\cdot)$, we now introduce the *bilinear form*

$$\begin{aligned} \langle f, g \rangle &= \sum_{i=1}^{l} \sum_{j=1}^{n} a_i k(\mathbf{x}_i, \tilde{\mathbf{x}}_j) b_j \\ &= \mathbf{a}^T \mathbf{K} \mathbf{b} \end{aligned} \tag{6.73}$$

where $\mathbf{K}$ is the Gram, or kernel matrix, and in the first line of the equation we made use of the relation

$$\langle k(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot) \rangle = k(\mathbf{x}_i, \mathbf{x}_j) \tag{6.74}$$

The first line of Eq. (6.73) may now be rewritten in the simplified form

$$\begin{aligned} \langle f, g \rangle &= \sum_{i=1}^{l} a_i \sum_{j=1}^{n} b_j k(\mathbf{x}_i, \tilde{\mathbf{x}}_j) \\ &= \sum_{i=1}^{l} a_i \underbrace{\sum_{j=1}^{n} b_j k(\tilde{\mathbf{x}}_j, \mathbf{x}_i)}_{g(\mathbf{x}_i)} \\ &= \sum_{i=1}^{l} a_i g(\mathbf{x}_i) \end{aligned} \tag{6.75}$$

where, in the second line, we used the symmetric property of the Mercer kernel. Similarly, we may write

$$\langle f, g \rangle = \sum_{j=1}^{n} b_j f(\tilde{\mathbf{x}}_j) \tag{6.76}$$

The definition of the bilinear form $\langle f, g \rangle$ introduced in Eq. (6.73) is independent of how the functions $f(\cdot)$ and $g(\cdot)$ are represented. We say so because the summation $\sum_{i=1}^{l} a_i g(\mathbf{x}_i)$ in Eq. (6.75) is invariant with respect to changes in the index n, the coefficient vector $\mathbf{b}$, and the n-dimensional vector $\tilde{\mathbf{x}}_j$. A similar statement applies to the summation $\sum_{j=1}^{n} b_j f(\tilde{\mathbf{x}}_j)$ in Eq. (6.76).

Furthermore, from the definition of Eq. (6.73), we readily derive the following three properties:

Property 1. Symmetry For all functions f and g in the space $\mathcal{F}$, the term $\langle f, g \rangle$ is symmetric, as shown by

$$\langle f, g \rangle = \langle g, f \rangle \tag{6.77}$$

Property 2. Scaling and distributive property For any pair of constants c and d and any set of functions f, g, and h in the space $\mathcal{F}$, we have

$$\langle (cf + dg), h \rangle = c \langle f, h \rangle + d \langle g, h \rangle \tag{6.78}$$

Property 3. Squared norm For any real-valued function f in the space $\mathcal{F}$, if we evaluate Eq. (6.73) for f acting all by itself, we obtain the following *squared norm*, or *quadratic metric*:

$$\begin{aligned} \|f\|^2 &= \langle f, f \rangle \\ &= \mathbf{a}^T \mathbf{K} \mathbf{a} \end{aligned}$$

Since the Gram is nonnegative definite, the squared norm has the property

$$\|f\|^2 \geq 0 \tag{6.79}$$

By virtue of the fact that for all real-valued functions f and g in the space $\mathcal{F}$, the bilinear term $\langle f, g \rangle$ satisfies the symmetry, scaling, and distributive properties as well as the property that the norm $\|f\|^2 = \langle f, f \rangle$ is nonnegative, we may now formally state that the $\langle f, g \rangle$ introduced in Eq. (6.73) is indeed an *inner product*; moreover it is an inner product that must also satisfy the condition $\langle f, g \rangle = 0$ if, and only if, $f = 0$. In other words, the space $\mathcal{F}$ embracing the functions f and g is an inner-product space.

There is one additional property that follows directly from Eq. (6.75). Specifically, setting

$$g(\cdot) = k(\mathbf{x}, \cdot)$$

in Eq. (6.75), we obtain

$$\begin{aligned} \langle f, k(\mathbf{x}, \cdot) \rangle &= \sum_{i=1}^{l} a_i k(\mathbf{x}, \mathbf{x}_i) \\ &= \sum_{i=1}^{l} a_i k(\mathbf{x}_i, \mathbf{x}), \qquad k(\mathbf{x}, \mathbf{x}_i) = k(\mathbf{x}_i, \mathbf{x}) \\ &= f(\mathbf{x}) \end{aligned} \tag{6.80}$$

For obvious reasons, this property of the Mercer kernel $k(\mathbf{x}, \cdot)$ is known as the *reproducing property*.

The kernel $k(\mathbf{x}, \mathbf{x}_i)$, representing a function of the two vectors $\mathbf{x}, \mathbf{x}_i \in \mathcal{X}$, is called a reproducing kernel of the vector space $\mathcal{F}$ if it satisfies the following two conditions (Aronszajn, 1950):

1. For every $\mathbf{x}_i \in \mathcal{X}$, $k(\mathbf{x}, \mathbf{x}_i)$ as a function of the vector $\mathbf{x}$ belongs to $\mathcal{F}$.
2. It satisfies the reproducing property.

These two conditions are indeed satisfied by the Mercer kernel, thereby endowing it with the designation "reproducing kernel." If the inner-product (vector) space $\mathcal{F}$, in which the reproducing kernel space is defined, is also *complete*, then we may go one step further and speak of a "reproducing-kernel Hilbert space."

To justify the property of completeness, consider a fixed input vector $\mathbf{x}$ and a pair of Cauchy sequences $\{f_n(\mathbf{x})\}_{n=1}^{\infty}$ and $\{f_m(\mathbf{x})\}_{m=1}^{\infty}$, where $n > m$. Then, applying the reproducing property of Eq. (6.80) to both $f_n(\mathbf{x})$ and $f_m(\mathbf{x})$, we may write

$$f_n(\mathbf{x}) - f_m(\mathbf{x}) = \langle f_n(\cdot) - f_m(\cdot)\rangle k(\mathbf{x}, \cdot)$$

where the right-hand side is an inner product. By invoking the *Cauchy–Schwarz inequality*,[12] we obtain

$$(f_n(\mathbf{x}) - f_m(\mathbf{x}))^2 \le \langle f_n(\cdot) - f_m(\cdot)\rangle^2 \underbrace{k(\mathbf{x}, \cdot)k(\mathbf{x}, \cdot)}_{k(\mathbf{x}, \mathbf{x})} \tag{6.81}$$

It follows, therefore, that $f_n(\mathbf{x})$ is a bounded Cauchy sequence, which converges toward some real-valued function f in the space $\mathcal{F}$. Finally, if we define the function

$$y(\mathbf{x}) = \lim_{n\to\infty} f_n(\mathbf{x})$$

and complete the space $\mathcal{F}$ by adding to it all such convergent Cauchy sequences, we obtain the *Hilbert space* $\mathcal{H}$. We have thus demonstrated that each Mercer kernel $k(\mathbf{x}, \cdot)$ defines a Hilbert space $\mathcal{H}$, where the value of the function $f(\mathbf{x})$ is reproduced by the inner product of $f(\mathbf{x})$ with $k(\mathbf{x}, \cdot)$. The Hilbert space so defined is called a *reproducing-kernel Hilbert space*, for which we use the acronym RKHS hereafter.

The analytic power of RKHS is expressed in an important theorem considered next.

Formulation of the The Representer Theorem[13]

Define a space $\mathcal{H}$ as the RKHS induced by a Mercer kernel $k(\mathbf{x}, \cdot)$. Given any real-valued function $f(\cdot) \in \mathcal{H}$, we may decompose it into the sum of two components, both of which naturally lie in the space $\mathcal{H}$:

- One component is contained in the span of the kernel functions $k(\mathbf{x}_1, \cdot), k(\mathbf{x}_2, \cdot), \ldots, k(\mathbf{x}_l, \cdot)$; denoting this component by $f_{\|}(\mathbf{x})$, we may use Eq. (6.71) to represent it as

$$f_{\|}(\cdot) = \sum_{i=1}^{l} a_i k(\mathbf{x}_i, \cdot)$$

- The second component is *orthogonal* to the span of the kernel functions; it is denoted by $f_{\perp}(\mathbf{x})$.

We may thus express the function $f(\cdot)$ as

$$\begin{aligned} f(\cdot) &= f_{\|}(\cdot) + f_{\perp}(\cdot) \\ &= \sum_{i=1}^{l} a_i k(\mathbf{x}_i, \cdot) + f_{\perp}(\cdot) \end{aligned} \tag{6.82}$$

Applying the distributive property of Eq. (6.78) to Eq. (6.82), we obtain

$$\begin{aligned} f(\mathbf{x}_j) &= \langle f(\cdot), k(\mathbf{x}_j, \cdot)\rangle_{\mathcal{H}} \\ &= \left\langle \sum_{i=1}^{l} a_i k(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot) \right\rangle_{\mathcal{H}} + \left\langle k(\mathbf{x}_j, \cdot), f_{\perp} \right\rangle_{\mathcal{H}} \end{aligned}$$

With $f_\perp$ being orthogonal to the span of the kernel functions, the second term is zero; this equation therefore reduces to

$$\begin{aligned} f(\mathbf{x}_j) &= \left\langle \sum_{i=1}^{l} a_i k(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot) \right\rangle_{\mathcal{H}} \\ &= \sum_{i=1}^{l} a_i k(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{6.83}$$

Equation (6.83) is a mathematical statement of the *representer theorem*:

> *Any function defined in an RKHS can be represented as a linear combination of Mercer kernel functions.*

However, there is more to be said.

Generalized Applicability of the Representer Theorem

An important property of the representer theorem is that the expansion given in Eq. (6.83) is the minimizer of the *regularized* empirical risk (cost function)

$$\mathcal{C}(f) = \frac{1}{2N} \sum_{n=1}^{N} (d(n) - f(\mathbf{x}(n))^2 + \Omega(\|f\|_{\mathcal{H}}) \tag{6.84}$$

where $\{\mathbf{x}(n), d(n)\}_{n=1}^{N}$ is the training sample, f is the unknown function to be estimated, and $\Omega(\|f\|_{\mathcal{H}})$ is the regularizing function (Schölkopf and Smola, 2002). For the theorem to hold, the regularizing function must be a strictly monotonic increasing function of its argument; hereafter, this requirement is referred to simply as the *monotonicity condition*.

The first term on the right-hand side of Eq. (6.84) is the standard error term, which is a quadratic function in f. Hence, the expansion of Eq. (6.83) is the minimizer of this term through the use of fixed $a_i \in \mathbb{R}$.

To prove that this expansion is also the minimizer of the regularized part of the empirical risk $\mathcal{C}(f)$, we proceed in three steps as follows:

1. Let $f_\perp$ denote the orthogonal compliment to the span of the kernel functions $\{k(x_i, \cdot)\}_{i=1}^{l}$. Then, since, according to Eq. (6.82), every function can be expressed as a kernel expansion on the training data plus $f_\perp$, we may write

$$\Omega(\|f\|_{\mathcal{H}}) = \Omega\left(\left\| \sum_{i=1}^{l} a_i k(x_i, \cdot) + f_\perp(\cdot) \right\|_{\mathcal{H}} \right) \tag{6.85}$$

For mathematical convenience, we prefer to work with the new function

$$\widetilde{\Omega}(\|f\|_{\mathcal{H}}^2) = \Omega(\|f\|_{\mathcal{H}}) \tag{6.86}$$

rather than the original regularizing function $\Omega(\|f\|_{\mathcal{H}})$. This move is permissible because a quadratic function is strictly monotonic on the infinite interval $[0, \infty)$. Hence, $\widetilde{\Omega}(\|f\|_{\mathcal{H}}^2)$ is strictly monotonic on $[0, \infty)$ if, and only if, $\Omega(\|f\|_{\mathcal{H}})$ also satisfies the monotonicity condition. For all $f_\perp$, we may thus write

$$\widetilde{\Omega}(\|f\|_{\mathcal{H}}^2) = \widetilde{\Omega}\left(\left\| \sum_{i=1}^{l} a_i k(x_i, \cdot) + f_\perp(\cdot) \right\|_{\mathcal{H}}^2 \right) \tag{6.87}$$

2. Applying the Pythagorean decomposition to the argument of $\tilde{\Omega}$ on the right-hand side of Eq. (6.87), we may go on to write

$$\begin{aligned}\tilde{\Omega}(\|f\|_{\mathcal{H}}^2) &= \tilde{\Omega}\left(\left\|\sum_{i=1}^{l} a_i k(x_i, \cdot)\right\|_{\mathcal{H}}^2 + \|f_\perp\|_{\mathcal{H}}^2\right) \\ &\geq \tilde{\Omega}\left(\left\|\sum_{i=1}^{l} a_i k(x_i, \cdot)\right\|_{\mathcal{H}}^2\right)\end{aligned}$$

For the optimum condition, we must set $f_\perp = 0$, the use of which in this equation yields the equality

$$\tilde{\Omega}(\|f\|_{\mathcal{H}}^2) = \tilde{\Omega}\left(\left\|\sum_{i=1}^{l} a_i k(x_i, \cdot)\right\|_{\mathcal{H}}^2\right) \tag{6.88}$$

3. Finally, in light of the definition introduced in Eq. (6.86), we have the desired result

$$\Omega(\|f\|_{\mathcal{H}}) = \Omega\left(\left\|\sum_{i=1}^{l} a_i k(x_i, \cdot)\right\|_{\mathcal{H}}\right) \tag{6.89}$$

It follows therefore that, for fixed $a_i \in \mathbb{R}$, the representer theorem is also the minimizer of the regularizing function $\Omega(\|f\|_{\mathcal{H}})$, provided that the monotonicity condition is satisfied.

In treating the composition of the standard error and regularizing terms as one whole entity, there will be a trade-off between these two terms. In any case, for some fixed $a_i \in \mathbb{R}$, the representer theorem described by the expansion of Eq. (6.83) will serve as the minimizer of the regularized empirical risk of Eq. (6.84), thereby establishing general applicability of the representer theorem (Schölkopf and Smola, 2002).

In the next chapter, we will make extensive use of this important theorem in the study of regularization theory.

6.11 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 method of structural risk minimization (SRM) that is rooted in VC dimension theory, which makes its derivation even more profound; SRM was discussed in Chapter 4. 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 models of intrinsic statistical regularities contained in the training data, they all stem from a common root in a support vector machine setting.

One other distinctive property of the support vector machine is that it is a kernal method of the batch-learning kind.[14]

Computational Considerations

The asymptotic behavior of a support vector machine grows linearly with the number of training examples, N. It follows therefore that the computational cost of using the machine for solving pattern recognition and regression problems has both a quadratic and a cubic component. Specifically, when the parameter C is small, the computational cost grows like N^2; and when C is large, the computational cost grows like N^3 (Bottou and Lin, 2007).

To alleviate this problem, several commercial optimization libraries have been developed to solve the quadratic-programming (QP) problem. However, these libraries are of limited use. The memory requirements of the QP 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 QP problem cannot be solved by the straightforward use of a commercial optimization library. The problem is complicated further by the fact that, in general, the solution to an SVM problem is quite *sparse*, because the weight vector $\mathbf{w}$ in the output layer of the machine consists of few nonzero elements relative to the number of data points in the training sample. Accordingly, direct attempts to solve the QP problem in a support vector machine will *not* scale to large problem sizes. To mitigate this difficulty, several innovations have been described in the literature, as summarized here:[15]

1. 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.
2. Platt (1999) extended Osuna's methodology by introducing an algorithm called *sequential minimal optimization* (SMO), which breaks a large QP problem into a series of very small QP subproblems that are solvable analytically, thereby eliminating the need for a numerical QP library. The computation time of SMO is dominated by kernel evaluation; hence, the use of kernel optimizations can be accelerated.
3. Joachims (1999) introduced several key innovations of his own. Specifically, a large SVM problem is decomposed into a series of smaller ones, but in a more principled manner than that of Osuna. Another important innovation introduced is the notion of *shrinking*: If a point is not an unbounded support vector, has not been for a long time, and there is little evidence for it becoming one, then, with high probability, that point may be removed from further scruntiny, thereby saving computation time.
4. Rifkin (2002) developed a computational procedure called the *SvmFu algorithm*, which may be viewed as a synthesis of the ideas proposed by Osuna, Platt, and Joachims. Specifically, the advantages of each of those three procedures were combined with some new features. It is claimed that with SvmFu a large problem may be solved as a sequence of subproblems which are small enough that their associated Hessian matrices can fit in memory.

5. Drineas and Mahoney (2005) have developed an algorithm to compute an easily interpretable low-rank approximation to the N-by-N Gram, or kernel matrix, in such a way that the computation of interest may be performed more rapidly. The relationships of the new algorithm with the Nyström method from integral equation theory are discussed therein.
6. Hush et al. (2006) describe polynomial-time algorithms that produce approximate solutions with guaranteed accuracy for a class of quadratic-programming problems that arise in the design of support vector machine classifiers. The algorithms employ a two-stage process. The first stage produces an approximate solution to a dual quadratic-programming problem, and the second stage maps this approximate dual solution to an approximate primal solution.

Curse of Dimensionality

As is the case for a multilayer perceptron, the intrinsic complexity of a support vector machine as an approximating function *increases exponentially* with m_0, where m_0 denotes the dimensionality of the input space. Moreover, the intrinsic complexity of the machine decreases exponentially with s, where s denotes the smoothness index which measures the number of constraints imposed on the approximating function. Accordingly, the smoothness index of the approximating function acts as a *corrective measure* against the curse of dimensionality. We may therefore say that a support vector machine will provide a good approximation to a dimensionally high function, provided that the function of interest is correspondingly smooth.

Concluding Remarks

The support vector machine (SVM) has established itself as the most widely used kernel-learning algorithm. Indeed, we may go on to say that in the machine-learning literature, support vector machines represent the state-of-the-art by virtue of their good generlization performance, relative ease of use, and rigorous theoretical foundations. Moreover, in a practical context, they are capable of delivering robust performance in solving pattern-recognition and regression problems.

However, the major limitation of support vector machines is the fast increase in their computing and storage requirements with respect to the number of training examples. These severe requirements tend to leave many large-scale learning problems beyond the reach of support vector machines. The core of this practical limitation lies in the quadratic programming routine that is an integral part of the SVM optimization theory. To mitigate this practical difficulty, a great deal of effort has been devoted to accelerate the SVM solver through a variety of parallel-implementation techniques beyond the decomposition procedures described above (Durdanovic et al., 2007; Yom-Tov (2007).

NOTES AND REFERENCES

1. The support vector machine was pioneered by Vapnik; the first description of the machine was presented by Boser, Guyon, and Vapnik in 1992. The most comprehensive and detailed

description of this new class of learning machines appeared in Vapnik's 1998 book entitled "Statistical Learning Theory," which is already a classic in the field.

The paper entitled "On the Mathematical Foundations of Learning," by Cucker and Smale (2001), presents a mathematically rigorous treatment of supervised learning theory, with emphasis on the relationship of approximation to learning and the primary role of inductive inference.

Comprehensive treatments of kernel machines, including support vector machines, are presented in the books by Schölkopf and Smola (2002), Herbrich (2002), and Shawe-Taylor and Cristianini (2004).

2. *Convex optimization* is a special class of optimization techniques that include least-squares and linear programming, for which a complete theory is already available. Moreover, problems that lend themselves to convex optimization go beyond least-squares and linear programs. The advantages to be gained in formulating a problem as a convex-optimization problem include

- solutions that are reliable and efficient, and
- theoretical advantages, exemplified by the formulation of a dual problem, the solution of which is more computationally efficient and conceptually transparent than that of the original problem.

For a detailed treatment of convex analysis and optimization, see the books by Boyd and Vandenbergh (2004) and Bertsekas et al. (2003).

3. In any optimization problem with a differentiable objective function and constraints for which duality applies, the primal and dual solutions must satisfy the *Karush–Kuhn–Tucker (KKT) conditions*. These conditions are named for Karush (1939) and Kuhn and Tucker (1951). The survey paper by Kuhn (1976) gives a historical account of solving inequality-constrained problems, in which convex optimization plays a major role.

4. The relationship between sparse approximation and support vector expansion was first discussed in Girosi (1998) and Vapnik (1998).

Steinwart (2003) presents a detailed mathematical discussion of the *sparseness* that arises in solving pattern-recognition problems by using support vector machines; in particular, this paper establishes (asymptotically) lower bounds on the number of support vectors. Along the way, several results are proved that are of importance for the understanding of support vector machines. The paper addresses three admissible loss functions:

i. the *hinge loss*, defined by $L(d, y) = \max(0, 1 - dy)$;
ii. the *squared hinge loss*, defined by $L(d, y) = [\max(0, 1 - dy)]^2$;
iii. the *least-squares loss*, defined by $L(d, y) = (1 - dy)^2$.

The corresponding SVM classifiers are denoted by L_1, L_2, and LS. The variables d and y denote the desired response and the corresponding response computed by the support vector machine for a given input example, respectively.

The design of support vector machines by using the least-squares loss is treated in a great deal of detail in the book entitled "Least-Squares Support Vector Machines," by Suykens et al. (2002).

5. With computational complexity as the issue of interest, we may identify two classes of algorithms:

- *Polynomial time algorithms*, which require a running time that a is 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).

6. The reciprocal of the parameter C plays exactly the same role as the regularization parameter in regularized least-squares estimation. We have adhered to the use of the parameter C in describing the theory of support vector machines largely to be consistent with the early development of this new class of kernel machines.

7. The idea of an inner-product kernel was first described in Aizerman et al. (1964a, 1964b) in the formulation of the method of potential functions, which represents the forerunner to radial-basis-function networks. At about the same time, Vapnik and Chervonenkis (1964) developed the idea of an optimal hyperplane. The combined use of these two powerful concepts in formulating the support vector machine first appeared in Boser et al. (1992).

8. For discussions on additional properties of kernels over and above those presented in Section 6.4 under property 1 and property 2, see the books by Schölkopf and Smola (2002), Herbrich (2002), and Shawe-Taylor and Cristianini (2004).

9. To describe minimax theory, consider a function $f(x, z)$, where $x \in \mathcal{X}$ and $z \in \mathcal{Z}$. The requirement in this theory is either

$$\text{to minimize } \sup_{z \in \mathcal{Z}} f(x, z)$$
$$\text{subject to } x \in \mathcal{X}$$

or, alternatively,

$$\text{to maximize } \inf_{x \in \mathcal{X}} f(x, z)$$
$$\text{subject to } z \in \mathcal{Z}$$

The application of minimax theory arises, for example, in the study of worst-case designs, which are of engineering importance. For a discussion of this theory, see Bertsekas et al. (2003).

Huber's minimax theory is based on neighborhoods that are not global by virtue of their exclusion of asymmetric distributions. Nevertheless, this theory deals successfully with a large part of traditional statistics, particularly regression.

10. The Hilbert space is discussed in the books by Dorny (1975) and Debnath and Mikusiński (1990).

11. The original paper on reproducing-kernel Hilbert space (RKHS) is Aronszajn (1950), which is a classic; it is also discussed in the books by Shawe-Taylor and Cristianini (2004), Schölkopf and Smola (2002), and Herbrich (2002).

12. Let $\mathbf{x}$ and $\mathbf{y}$ be any two elements of an inner-product space $\mathcal{F}$. According to the *Cauchy–Schwarz inequality*, we have

$$\langle \mathbf{x}, \mathbf{y} \rangle^2 \leq \|\mathbf{x}\|^2 \cdot \|\mathbf{y}\|^2$$

the proof of which is straightforward. The inequality states that the squared inner product of **x** and **y** is less than or equal to the product of the squared Euclidean length of **x** and that of **y**. The version of the inequality presented in Eq. (6.81) is an adaptation of this statement that is made to suit the problem considered in establishing the reproducing-kernel Hilbert space.

13. In a historical context, the celebrated representer theorem was first described in Kimeldorf and Wahba (1971) for solving practical problems in statistical estimation based on squared-loss (cost) functions; see also the book by Wahba (1990). Generalized applicability of the representer theorem to regularized cost functions was addressed for the first-time in Schölkopf and Smola (2002).

14. In contrast to the support vector machine that is of a batch-learning kind, the *kernel LMS algorithm*, due to Liu et al. (2008), is of an on-line learning kind. This new algorithm embodies ideas from the least-mean-square (LMS) algorithm, discussed in Chapter 3, and the reproducting-kernel Hilbert space (RKHS), discussed in this chapter; these ideas are integrated together in a composite fashion. In particular, the kernel trick is used to permit learning on iteration-by-iteration basis.

15. An overview of quadratic programming optimization methods is presented in Bottou and Lin (2007).

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 **w** denotes the weight vector, b denotes the bias, and **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.

Mercer Kernels

6.6 The Mercer kernel $k(x_i, \mathbf{x}_j)$ 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 $k_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$. Assume that 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 made up of eigenvalues and $\mathbf{Q}$ is a matrix made up of the corresponding eigenvectors, formulate an expression for the Mercer 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)** Demonstrate that all three Mercer kernels described in Table 6.1 satisfy the *unitary invariance property*:

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

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

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

(b) Does this property hold in general?

6.8 **(a)** Show that Mercer kernels are all positive definite.

(b) Consider a Mercer kernel, denoted by $k(\mathbf{x}_i, \mathbf{x}_j)$. Such a kernel satisfies the *Cauchy–Schwarz inequality*:

$$k(\mathbf{x}_i, \mathbf{x}_j)k(\mathbf{x}_j, \mathbf{x}_i) \leq k(\mathbf{x}_i, \mathbf{x}_i)k(\mathbf{x}_j, \mathbf{x}_j)$$

Demonstrate this property of Mercer kernels by considering the determinant of a two-by-two Gram $\mathbf{K}$.

6.9 Consider the Gaussian kernel

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

where no $\mathbf{x}_i$ and $\mathbf{x}_j$ are the same. Show that the Gram

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & ... & k(\mathbf{x}_1, \mathbf{x}_N) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2)) & ... & k(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & k(\mathbf{x}_N, \mathbf{x}_2) & ... & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

has *full rank*—that is, any two columns of the matrix $\mathbf{K}$ are linearly independent in an algebraic sense.

6.10 The *Mahalanobis kernel* is defined by

$$k(\mathbf{x}, \mathbf{x}_i) = \exp(-(\mathbf{x} - \mathbf{x}_i)^T \sum{}^{-1}(\mathbf{x} - \mathbf{x}_i))$$

where the M-dimensional input vector $\mathbf{x} \in \mathcal{X}$, and $i = 1, 2, ..., N$, and the M-by-M matrix

$$\sum = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_M^2)$$

where $\sigma_1, \sigma_2, \ldots, \sigma_M$ are all positive. A distinct property of this kernel, compared with the Gaussian kernel, is that each axis of the M-dimensional input space $\mathcal{X}$ has a "smoothing" parameter (i.e., a particular σ) of its own.

To illustrate this property, consider the function

$$F(\mathbf{x}) = \sum_{i=1}^{N} a_i \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma_i^2}\right)$$

which may be viewed as a *density estimator* (Herbrich, 2002). Given $a_i = 1$ and $\sigma_i = \sigma$ for all i, $M = 2$, and $N = 20$, plot the function $F(\mathbf{x})$ versus the coordinates x_1 and x_2 for the following values:

(i) $\sigma = 0.5$
(ii) $\sigma = 0.7$
(iii) $\sigma = 1.0$
(iv) $\sigma = 2.0$

Comment on your results.

6.11 A joint probability density function $p_{X_1, X_2}(x_1, x_2)$ over an $\mathcal{X}$-by-$\mathcal{X}$ product space is said to be a *P-matrix* provided that it satisfies the finitely nonnegative definite (i.e., positive semi-definite) property (Shawe-Taylor and Cristianini, 2004).

By considering the simple case of a two-element set $\mathbf{X} = \{X_1, X_2\}$ of random variables, demonstrate validity of the following statement: All P-kernels are joint distributions, but not all joint distributions are P-kernels.

Pattern classification

6.12 The margin plays a key role in the design of support vector machines. Identify the important properties of the margin in solving pattern-classification problems.

6.13 Using the formula of Eq. (6.17), show that the margin of linearly separable patterns can be expressed in terms of the Lagrange multipliers as

$$\rho = \frac{2}{\left(\sum_{i=1}^{N_s} \alpha_i\right)^{1/2}}$$

where N_s is the number of support vectors.

6.14 Consider a training sample $\{\mathbf{x}_i, d_i\}_{i=1}^{N}$ that consists of positive and negative examples that are linearly separable. Justify the following statement:

The support vectors contain all the information needed to classify the positive and negative examples.

6.15 Figure P6.15 shows a data set that consists of nonlinearly separable positive and negative examples. Specifically, the decision boundary separating the positive from negative examples is an *ellipse*. Find the transformation that maps the input space into the feature space such that the positive and negative examples become linearly separable in the feature space.

6.16 The Mercer kernel for a polynomial learning machine used to solve the XOR problem is defined by

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

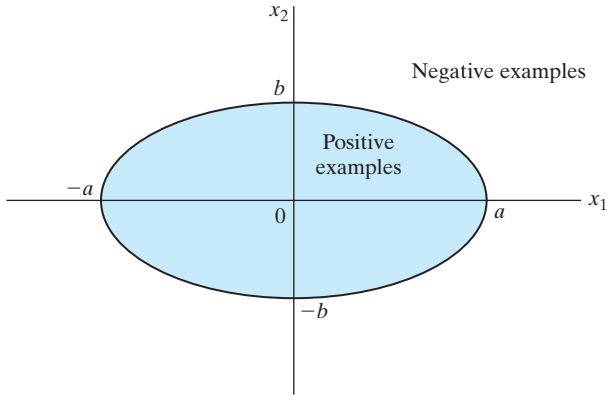


FIGURE P6.15

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?

6.17 Figure P6.17 shows the XOR function operating on a three-dimensional pattern $\mathbf{x}$ as described by the relationship

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

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.

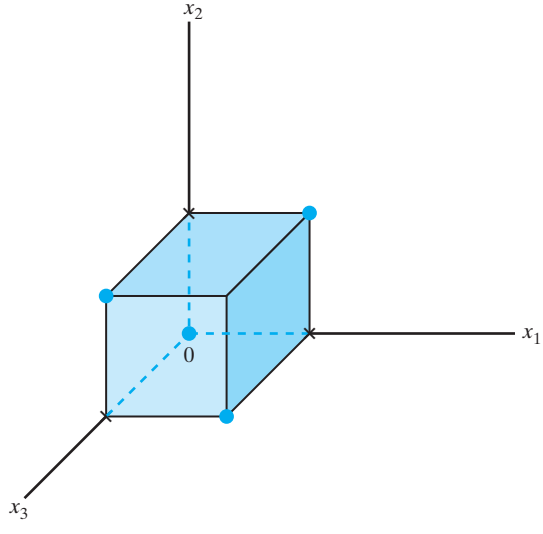


FIGURE P6.17

Sparsity

6.18 Justify the following statement:

A support vector machine solver is sparse, but the Gram associated with the machine is rarely sparse.

6.19 The quadratic programming routine in a support vector machine solver provides the basis for splitting the training examples into three categories. Define these three categories, and use a two-dimensional figure to illustrate how the splitting is performed.

Metrics

6.20 With different algortihms being developed for accelerating support vector machine solvers, it is important that we formulate metrics for comparing the performance of these different algorithms. Develop a set of metrics that could be used for dealing with this practical issue.

Reproducing-kernel Hilbert space

6.21 Let $k(\mathbf{x}_i, .)$ and $k(\mathbf{x}_j, .)$ denote a pair of kernels, where $i, j = 1, 2, \ldots, N$. The vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ have the same dimensionality. Show that

$$k(\mathbf{x}_i, .)k(\mathbf{x}_j, .) = k(\mathbf{x}_i, \mathbf{x}_j)$$

where the expression on the left-hand side is an inner-product kernel.

6.22 Equations (6.77), (6.78), and (6.79) describe three important properties of the inner product $\langle f, g \rangle$, defined in Eq. (6.75). Prove the properties described in those three equations.

6.23 Justify the following statement:

If a reproducing kernel $k(\mathbf{x}, \mathbf{x}')$ exists, then that kernel is unique.

Computer experiments

6.24 This experiment investigates the scenario where the two moons in Fig. 1.8 overlap and are therefore nonseparable.

(a) Repeat the second part of the experiment in Fig. 6.7, for which the vertical separation between the two moons was fixed at $d = -6.5$. Experimentally, determine the value of parameter C for which the classification error rate is reduced to a minimum.

(b) Reduce the vertical separation between the two moons further by setting $d = -6.75$, for which the classification error rate is expected to be higher than that for $d = -6.5$. Experimentally, determine the value of parameter C for which the error rate is reduced to a minimum.

Comment on the results obtained for both parts of the experiment.

6.25 Among the supervised-learning algorithms studied thus far, the support vector machine stands out as the most powerful. In this problem, the performance of the support vector machine is to be challenged by using it to classify the two multicircular regions that constitute the "tightly fisted" structure shown in Fig. P6.24. The radii of the three concentric circles in this figure are $d_1 = 0.2$, $d_2 = 0.5$, and $d_3 = 0.8$.

(a) Generate 100 epochs, each of which consists of 200 randomly distributed training examples, and an equal number of test data for the two regions of Fig. P6.24.

(b) Train a support vector machine, assigning the value $C = 500$. Hence, construct the decision boundary computed by the machine.

(c) Test the network and thereby determine the classification error rate.

(d) Repeat the experiment for $C = 100$ and $C = 2{,}500$.

Comment on your results.

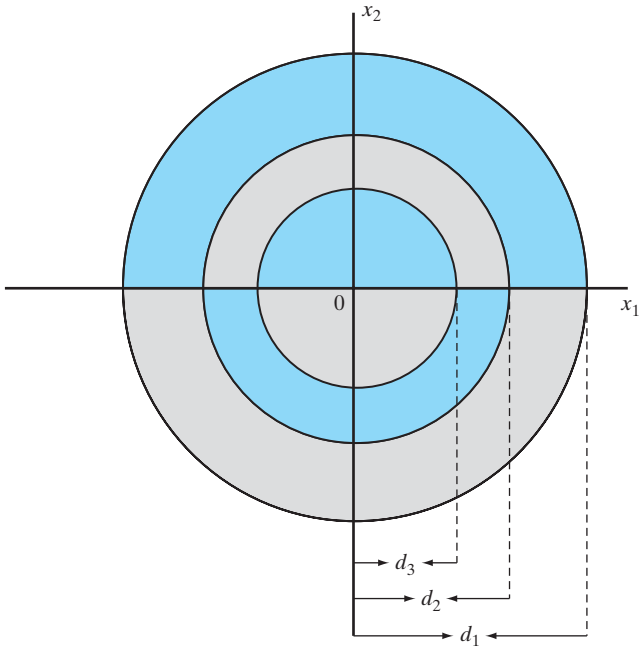


FIGURE P6.25

CHAPTER 7

Regularization Theory

ORGANIZATION OF THE CHAPTER

In this chapter, we focus attention on the many facets of regularization theory, which is at the core of all neural-network and machine-learning algorithms. Following the motivational material presented in Section 7.1, the rest of the chapter is organized as follows:

1. Section 7.2 addresses the issue of ill-posed inverse problems.
2. Section 7.3 develops Tikhonov's regularization theory, which provides the mathematical basis for the regularization of supervised-learning algorithms. This part of the chapter also includes Section 7.4, which focuses on *regularization networks* whose hidden layer has the same size as that of the training sample. Section 7.5 discusses a class of generalized radial-basis-function networks whose hidden layer is constrained to be a subset of that characterizing regularization networks. The regularized least-squares estimator is revisited in Section 7.6 as a special case of this class of generalized RBF networks. Then, in Section 7.7 we show how the insightful ideas derived from regularized least-squares estimation can be exploited in the regularization of other estimators that do not lend themselves to the application of Tikhonov's regularization theory.
3. Section 7.8 describes a procedure, based on cross-validation, for estimating the regularization parameter.
4. The last part of the chapter begins with a discussion of semisupervised learning in Section 7.9. Then, the basic ideas behind manifold regularization are discussed in Sections 7.10 through 7.12. Section 7.13 introduces spectral graph theory. Section 7.14 discusses *generalization of the representer theorem* in light of the manifold regularization theory. Section 7.15 exploits spectral graph theory on the regularized least-squares estimator (using labeled and unlabeled examples), as an illustrative application of the generalized regularization theory. In Section 7.16, we present a computer experiment on semisupervised learning, using least-squares estimation.

Section 7.17 concludes the chapter with a summary and discussion.

7.1 INTRODUCTION

In looking over the supervised-learning algorithms derived in previous chapters of the book, we find that despite the differences in their compositions, they do share a