11

Support Vector Machines

11.1 Introduction

Fisher's linear discriminant function (LDF) and related classifiers for binary and multiclass learning problems have performed well for many years and for many data sets. Recently, a brand-new learning methodology, *support vector machines (SVMs)*, has emerged (Boser, Guyon, and Vapnik, 1992), which has matched the performance of the LDF and, in many instances, has proved to be superior to it.

Development and implementation of algorithms for SVMs are currently of great interest to theoretical researchers and applied scientists in machine learning, data mining, and bioinformatics. Huge numbers of research articles, tutorials, and textbooks have been published on the topic, and annual workshops, new research journals, courses, and websites are now devoted to the subject. SVMs have been successfully applied to classification problems as diverse as handwritten digit recognition, text categorization, cancer classification using microarray expression data, protein secondary-structure prediction, and cloud classification using satellite-radiance profiles.

SVMs, which are available in both linear and nonlinear versions, involve optimization of a convex loss function under given constraints and so are unaffected by problems of local minima. This gives SVMs quite a strong competitive advantage over methods such as neural networks and decision trees. SVMs are computed using well-documented, general-purpose, mathematical programming algorithms, and their performance in many situations has been quite remarkable. Even in the face of massive data sets, extremely fast and efficient software is being designed to compute SVMs for classification.

By means of the new technology of *kernel methods*, SVMs have been very successful in building highly nonlinear classifiers. The kernel method enables us to construct linear classifiers in high-dimensional feature spaces that are nonlinearly related to input space and to carry out those computations in input space using very few parameters. SVMs have also been successful in dealing with situations in which there are many more variables than observations.

Although these advantages hold in general, we have to recognize that there will always be applications in which SVMs can get beaten in performance by a hand-crafted classification method.

In this chapter, we describe the linear and nonlinear SVM as solutions of the binary classification problem. The nonlinear SVM incorporates nonlinear transformations of the input vectors and uses the kernel trick to simplify computations. We describe a variety of kernels, including string kernels for text categorization problems. Although the SVM methodology was built specifically for binary classification, we discuss attempts to extend that methodology to multiclass classification. Finally, although the SVM methodology was originally designed to solve classification problems, we discuss how the SVM methodology has been defined for regression situations.

11.2 Linear Support Vector Machines

Assume we have available a learning set of data,

$$\mathcal{L} = \{(\mathbf{x}_i, y_i) : i = 1, 2, \ldots, n\}, \tag{11.1}$$

where $\mathbf{x}_i \in \Re^r$ and $y_i \in \{-1, +1\}$. The *binary classification problem* is to use $\mathcal{L}$ to construct a function $f : \Re^r \to \Re$ so that

$$C(\mathbf{x}) = \text{sign}(f(\mathbf{x})) \tag{11.2}$$

is a classifier. The *separating function* f then classifies each new point $\mathbf{x}$ in a test set $\mathcal{T}$ into one of two classes, Π_+ or Π_-, depending upon whether $C(\mathbf{x})$ is $+1$ (if $f(\mathbf{x}) \geq 0$) or -1 (if $f(\mathbf{x}) < 0$), respectively. The goal is to have f assign all positive points in $\mathcal{T}$ (i.e., those with $y = +1$) to Π_+ and all negative points in $\mathcal{T}$ ($y = -1$) to Π_-. In practice, we recognize that 100% correct classification may not be possible.

11.2.1 The Linearly Separable Case

First, consider the simplest situation: suppose the positive ($y_i = +1$) and negative ($y_i = -1$) data points from the learning set $\mathcal{L}$ can be separated by a hyperplane,

$$\{\mathbf{x} : f(\mathbf{x}) = \beta_0 + \mathbf{x}^\tau \boldsymbol{\beta} = 0\}, \tag{11.3}$$

where $\boldsymbol{\beta}$ is the *weight vector* with Euclidean norm $\| \boldsymbol{\beta} \|$, and β_0 is the *bias*. (Note: $b = -\beta_0$ is the *threshold*.) If this hyperplane can separate the learning set into the two given classes without error, the hyperplane is termed a *separating hyperplane*. Clearly, there is an infinite number of such separating hyperplanes. How do we determine which one is the best?

Consider any separating hyperplane. Let d_- be the shortest distance from the separating hyperplane to the nearest negative data point, and let d_+ be the shortest distance from the same hyperplane to the nearest positive data point. Then, the *margin* of the separating hyperplane is defined as $d = d_- + d_+$. If, in addition, the distance between the hyperplane and its closest observation is maximized, we say that the hyperplane is an *optimal separating hyperplane* (also known as a *maximal margin classifier*).

If the learning data from the two classes are linearly separable, there exists β_0 and $\boldsymbol{\beta}$ such that

$$\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta} \geq +1, \quad \text{if } y_i = +1, \tag{11.4}$$

$$\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta} \leq -1, \quad \text{if } y_i = -1. \tag{11.5}$$

If there are data vectors in $\mathcal{L}$ such that equality holds in (11.4), then these data vectors lie on the hyperplane H_{+1}: $(\beta_0 - 1) + \mathbf{x}^\tau \boldsymbol{\beta} = 0$; similarly, if there are data vectors in $\mathcal{L}$ such that equality holds in (11.5), then these data vectors lie on the hyperplane H_{-1}: $(\beta_0 + 1) + \mathbf{x}^\tau \boldsymbol{\beta} = 0$. Points in $\mathcal{L}$ that lie on either one of the hyperplanes H_{-1} or H_{+1}, are said to be *support vectors*. See Figure 11.1. The support vectors typically consist of a small percentage of the total number of sample points.

If $\mathbf{x}_{-1}$ lies on the hyperplane H_{-1}, and if $\mathbf{x}_{+1}$ lies on the hyperplane H_{+1}, then,

$$\beta_0 + \mathbf{x}_{-1}^\tau \boldsymbol{\beta} = -1, \quad \beta_0 + \mathbf{x}_{+1}^\tau \boldsymbol{\beta} = +1. \tag{11.6}$$

The difference of these two equations is $\mathbf{x}_{+1}^\tau \boldsymbol{\beta} - \mathbf{x}_{-1}^\tau \boldsymbol{\beta} = 2$, and their sum is $\beta_0 = -\frac{1}{2}\{\mathbf{x}_{+1}^\tau \boldsymbol{\beta} + \mathbf{x}_{-1}^\tau \boldsymbol{\beta}\}$. The perpendicular distances of the hyperplane $\beta_0 + \mathbf{x}^\tau \boldsymbol{\beta} = 0$ from the points $\mathbf{x}_{-1}$ and $\mathbf{x}_{+1}$ are

$$d_- = \frac{|\beta_0 + \mathbf{x}_{-1}^\tau \boldsymbol{\beta}|}{\| \boldsymbol{\beta} \|} = \frac{1}{\| \boldsymbol{\beta} \|}, \quad d_+ = \frac{|\beta_0 + \mathbf{x}_{+1}^\tau \boldsymbol{\beta}|}{\| \boldsymbol{\beta} \|} = \frac{1}{\| \boldsymbol{\beta} \|}, \tag{11.7}$$

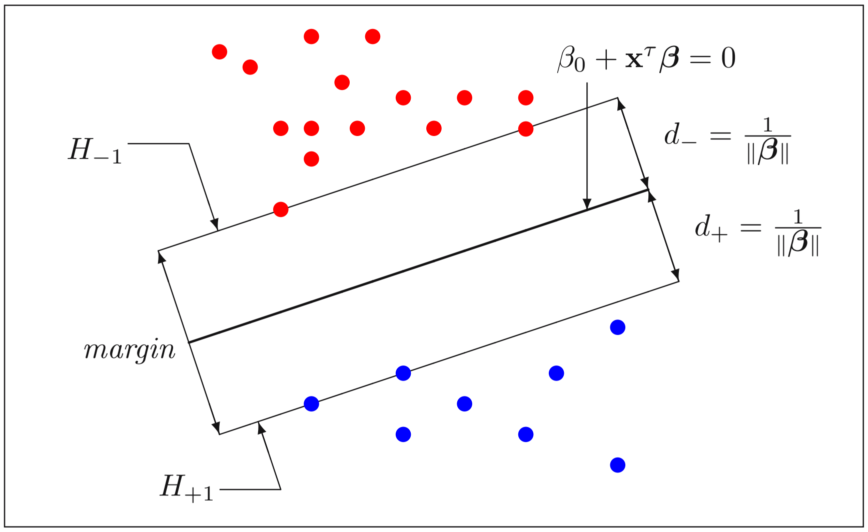
$\beta_0 + \mathbf{x}^\tau \boldsymbol{\beta} = 0$
H_{-1}
$d_- = \frac{1}{\|\boldsymbol{\beta}\|}$
$d_+ = \frac{1}{\|\boldsymbol{\beta}\|}$
margin
H_{+1}

FIGURE 11.1. *Support vector machines: the linearly separable case. The red points correspond to data points with $y_i = -1$, and the blue points correspond to data points with $y_i = +1$. The separating hyperplane is the line $\beta_0 + \mathbf{x}^\tau \boldsymbol{\beta} = 0$. The support vectors are those points lying on the hyperplanes H_{-1} and H_{+1}. The margin of the separating hyperplane is $d = 2/ \parallel \boldsymbol{\beta} \parallel$.*

respectively (see Exercise 11.1). So, the margin of the separating hyperplane is $d = 2/ \parallel \boldsymbol{\beta} \parallel$.

The inequalities (11.4) and (11.5) can be combined into a single set of inequalities,

$$y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) \geq +1, \quad i = 1, 2, \ldots, n. \tag{11.8}$$

The quantity $y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta})$ is called the *margin of* $(\mathbf{x}_i, y_i)$ *with respect to the hyperplane* (11.3), $i = 1, 2, \ldots, n$. From (11.6), we see that $\mathbf{x}_i$ is a support vector with respect to the hyperplane (11.3) if its margin equals one; that is, if

$$y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) = 1. \tag{11.9}$$

The support vectors in Figure 11.1 are identified (with circles around them). The empirical distribution of the margins of all the observations in $\mathcal{L}$ is called the *margin distribution of a hyperplane with respect to* $\mathcal{L}$. The minimum of the empirical margin distribution is the *margin of the hyperplane with respect to* $\mathcal{L}$.

The problem is to find the optimal separating hyperplane; namely, find the hyperplane that maximizes the margin, $2/ \parallel \boldsymbol{\beta} \parallel$, subject to the conditions (11.8). Equivalently, we wish to find β_0 and $\boldsymbol{\beta}$ to

$$\textit{minimize} \quad \frac{1}{2} \parallel \boldsymbol{\beta} \parallel^2, \tag{11.10}$$

$$\textit{subject to} \quad y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) \geq 1, \quad i = 1, 2, \ldots, n. \tag{11.11}$$

This is a convex optimization problem: minimize a quadratic function subject to linear inequality constraints. Convexity ensures that we have a global minimum wthout local minima. The resulting optimal separating hyperplane is called the *maximal* (or *hard*) *margin solution*.

We solve this problem using Lagrangian multipliers. Because the constraints are $y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - 1 \geq 0$, $i = 1, 2, \ldots, n$, we multiply the constraints by positive Lagrangian multipliers and subtract each such product from the objective function (11.10) to form the *primal!functional*,

$$F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) = \frac{1}{2} \parallel \boldsymbol{\beta} \parallel^2 - \sum_{i=1}^{n} \alpha_i \{y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - 1\}, \tag{11.12}$$

where

$$\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^\tau \geq \mathbf{0} \tag{11.13}$$

is the n-vector of (nonnegative) Lagrangian coefficients. We need to minimize F with respect to the *primal variables* β_0 and $\boldsymbol{\beta}$, and then maximize the resulting minimum-F with respect to the *dual variables* $\boldsymbol{\alpha}$.

The Karush–Kuhn–Tucker conditions give necessary and sufficient conditions for a solution to a constrained optimization problem. For our primal problem, β_0, $\boldsymbol{\beta}$, and $\boldsymbol{\alpha}$ have to satisfy:

$$\frac{\partial F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})}{\partial \beta_0} = -\sum_{i=1}^{n} \alpha_i y_i = 0, \tag{11.14}$$

$$\frac{\partial F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0, \tag{11.15}$$

$$y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - 1 \geq 0, \tag{11.16}$$

$$\alpha_i \geq 0, \tag{11.17}$$

$$\alpha_i \{y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - 1\} = 0, \tag{11.18}$$

for $i = 1, 2, \ldots, n$. The condition (11.18) is known as the *Karush–Kuhn–Tucker complementarity condition*.

Solving equations (11.14) and (11.15) yields

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \tag{11.19}$$

$$\boldsymbol{\beta}^* = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i. \tag{11.20}$$

Substituting (11.19) and (11.20) into (11.12) yields the minimum value of $F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha})$, namely,

$$F_D(\boldsymbol{\alpha}) = \frac{1}{2} \parallel \boldsymbol{\beta}^* \parallel^2 - \sum_{i=1}^{n} \alpha_i \{y_i(\beta_0^* + \mathbf{x}_i^\tau \boldsymbol{\beta}^*) - 1\}$$

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i^\tau\mathbf{x}_j) - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i^\tau\mathbf{x}_i) + \sum_{i=1}^{n}\alpha_i$$

$$= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i^\tau\mathbf{x}_j), \tag{11.21}$$

where we used (11.18) in the second line. Note that the primal variables have been removed from the problem. The expression (11.21) is usually referred to as the *dual functional* of the optimization problem.

We next find the Lagrangian multipliers $\boldsymbol{\alpha}$ by maximizing the dual functional (11.21) subject to the constraints (11.17) and (11.19). The constrained maximization problem (the "Wolfe dual") can be written in matrix notation as follows. Find $\boldsymbol{\alpha}$ to

$$\textit{maximize} \quad F_D(\boldsymbol{\alpha}) = \mathbf{1}_n^\tau\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\tau\mathbf{H}\boldsymbol{\alpha} \tag{11.22}$$

$$\textit{subject to} \quad \boldsymbol{\alpha} \geq \mathbf{0},\ \boldsymbol{\alpha}^\tau\mathbf{y} = \mathbf{0}, \tag{11.23}$$

where $\mathbf{y} = (y_1, \cdots, y_n)^\tau$ and $\mathbf{H} = (H_{ij})$ is a square $(n \times n)$-matrix with $H_{ij} = y_i y_j(\mathbf{x}_i^\tau\mathbf{x}_j)$. If $\widehat{\boldsymbol{\alpha}}$ solves this optimization problem, then

$$\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\widehat{\alpha}_i y_i\mathbf{x}_i \tag{11.24}$$

yields the optimal weight vector. If $\widehat{\alpha}_i > 0$, then, from (11.18), $y_i(\beta_0^* + \mathbf{x}_i^\tau\boldsymbol{\beta}^*) = 1$, and so $\mathbf{x}_i$ is a support vector; for all observations that are not support vectors, $\widehat{\alpha}_i = 0$. Let $sv \subset \{1, 2, \ldots, n\}$ be the subset of indices that identify the support vectors (and also the nonzero Lagrangian multipliers). Then, the optimal $\boldsymbol{\beta}$ is given by (11.24), where the sum is taken only over the support vectors; that is,

$$\widehat{\boldsymbol{\beta}} = \sum_{i\in sv}\widehat{\alpha}_i y_i\mathbf{x}_i. \tag{11.25}$$

In other words, $\widehat{\boldsymbol{\beta}}$ is a linear function only of the support vectors $\{\mathbf{x}_i, i \in sv\}$. In most applications, the number of support vectors will be small relative to the size of $\mathcal{L}$, yielding a *sparse* solution. In this case, the support vectors carry all the information necessary to determine the optimal hyperplane.

The primal and dual optimization problems yield the same solution, although the dual problem is simpler to compute and, as we shall see, is simpler to generalize to nonlinear classifiers. Finding the solution involves standard convex quadratic-programming methods, and so any local minimum also turns out to be a global minimum.

Although the optimal bias $\widehat{\beta}_0$ is not determined explicitly by the optimization solution, we can estimate it by solving (11.18) for each support vector and then averaging the results. In other words, the estimated bias of the optimal hyperplane is given by

$$\widehat{\beta}_0 = \frac{1}{|sv|} \sum_{i \in sv} \left(\frac{1 - y_i \mathbf{x}_i^\tau \widehat{\boldsymbol{\beta}}}{y_i} \right), \tag{11.26}$$

where $|sv|$ is the number of support vectors in $\mathcal{L}$.

It follows that the optimal hyperplane can be written as

$$\begin{aligned} \widehat{f}(\mathbf{x}) &= \widehat{\beta}_0 + \mathbf{x}^\tau \widehat{\boldsymbol{\beta}} \\ &= \widehat{\beta}_0 + \sum_{i \in sv} \widehat{\alpha}_i y_i (\mathbf{x}^\tau \mathbf{x}_i). \end{aligned} \tag{11.27}$$

Clearly, only support vectors are relevant in computing the optimal separating hyperplane; observations that are not support vectors play no role in determining the hyperplane and are, thus, irrelevant to solving the optimization problem. The *classification rule* is given by

$$C(\mathbf{x}) = \text{sign}\{\widehat{f}(\mathbf{x})\}. \tag{11.28}$$

If $j \in sv$, then, from (11.27),

$$y_j \widehat{f}(\mathbf{x}_j) = y_j \widehat{\beta}_0 + \sum_{i \in sv} \widehat{\alpha}_i y_i y_j (\mathbf{x}_j^\tau \mathbf{x}_i) = 1. \tag{11.29}$$

Hence, the squared-norm of the weight vector $\widehat{\boldsymbol{\beta}}$ of the optimal hyperplane is

$$\begin{aligned} \| \widehat{\boldsymbol{\beta}} \|^2 &= \sum_{i \in sv} \sum_{j \in sv} \widehat{\alpha}_i \widehat{\alpha}_j y_i y_j (\mathbf{x}_i^\tau \mathbf{x}_j) \\ &= \sum_{j \in sv} \widehat{\alpha}_j y_j \sum_{i \in sv} \widehat{\alpha}_i y_i (\mathbf{x}_i^\tau \mathbf{x}_j) \\ &= \sum_{j \in sv} \widehat{\alpha}_j (1 - y_j \widehat{\beta}_0) \\ &= \sum_{j \in sv} \widehat{\alpha}_j. \end{aligned} \tag{11.30}$$

The third line used (11.29) and the fourth line used (11.19). It follows from (11.30) that the optimal hyperplane has maximum margin $2/ \| \widehat{\boldsymbol{\beta}} \|$, where

$$\frac{1}{\| \widehat{\boldsymbol{\beta}} \|} = \left(\sum_{j \in sv} \widehat{\alpha}_j \right)^{-1/2}. \tag{11.31}$$

11.2.2 The Linearly Nonseparable Case

In real applications, it is unlikely that there will be such a clear linear separation between data drawn from two classes. More likely, there will be some overlap. We can generally expect some data from one class to infiltrate the region of space perceived to belong to the other class, and vice versa. The overlap will cause problems for any classification rule, and, depending upon the extent of the overlap, we should expect that some of the overlapping points will be misclassified.

The *nonseparable case* occurs if either the two classes are separable, but not linearly so, or that no clear separability exists between the two classes, linearly or nonlinearly. One reason for overlapping classes is the high noise level (i.e., large variances) of one or both classes. As a result, one or more of the constraints will be violated.

The way we cope with overlapping data is to create a more flexible formulation of the problem, which leads to a *soft-margin solution*. To do this, we introduce the concept of a nonnegative *slack variable*, ξ_i, for each observation, $(\mathbf{x}_i, y_i)$, in $\mathcal{L}$, $i = 1, 2, \ldots, n$. See Figure 11.2 for a two-dimensional example. Let

$$\boldsymbol{\xi} = (\xi_1, \cdots, \xi_n)^\tau \geq \mathbf{0}. \tag{11.32}$$

The constraints (11.11) now become $y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) + \xi_i \geq 1$ for $i = 1, 2, \ldots, n$. Data points that obey these constraints have $\xi_i = 0$. The classifier now has to find the optimal hyperplane that controls both the margin, $2/\| \boldsymbol{\beta} \|$, and some computationally simple function of the slack variables, such as

$$g_\sigma(\boldsymbol{\xi}) = \sum_{i=1}^{n} \xi_i^\sigma, \tag{11.33}$$

subject to certain constraints. The usual values of σ are 1 ("1-norm") or 2 ("2-norm"). Here, we discuss the case of $\sigma = 1$; for $\sigma = 2$, see Exercise 11.2.

The *1-norm soft-margin optimization problem* is to find β_0, $\boldsymbol{\beta}$, and $\boldsymbol{\xi}$ to

$$\textit{minimize} \quad \frac{1}{2} \| \boldsymbol{\beta} \|^2 + C \sum_{i=1}^{n} \xi, \tag{11.34}$$

$$\textit{subject to} \quad \xi_i \geq 0, \; y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) \geq 1 - \xi_i, \quad i = 1, 2, \ldots, n, \tag{11.35}$$

where $C > 0$ is a *regularization parameter*. C takes the form of a tuning constant that controls the size of the slack variables and balances the two terms in the minimizing function.

Form the primal functional, $F_P = F_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\eta})$, where

$$F_P = \frac{1}{2} \| \boldsymbol{\beta} \|^2 + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \{ y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - (1 - \xi_i) \} - \sum_{i=1}^{n} \eta_i \xi_i, \tag{11.36}$$

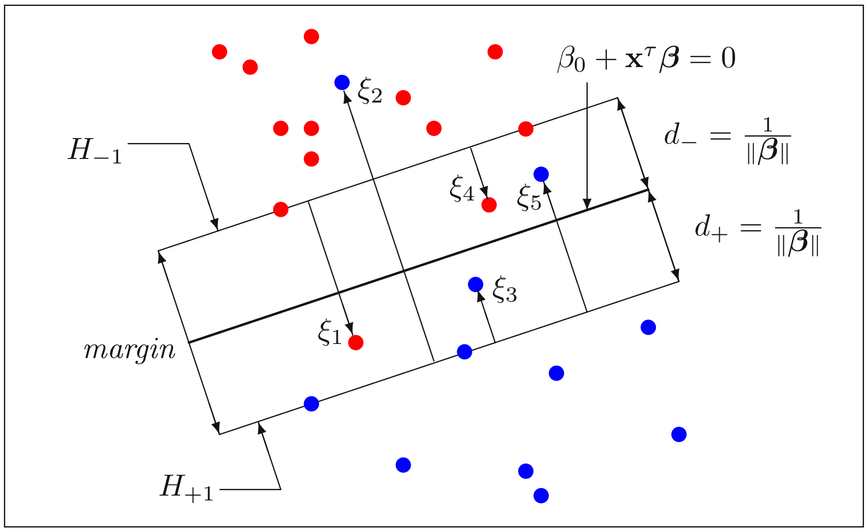
$\beta_0 + \mathbf{x}^\tau\boldsymbol{\beta} = 0$
ξ_2
H_{-1}
$d_- = \frac{1}{\|\boldsymbol{\beta}\|}$
ξ_4
ξ_5
$d_+ = \frac{1}{\|\boldsymbol{\beta}\|}$
ξ_3
ξ_1
margin
H_{+1}

FIGURE 11.2. *Support vector machines: the nonlinearly separable case. The red points correspond to data points with $y_i = -1$, and the blue points correspond to data points with $y_i = +1$. The separating hyperplane is the line $\beta_0 + \mathbf{x}^\tau\boldsymbol{\beta} = 0$. The support vectors are those circled points lying on the hyperplanes H_{-1} and H_{+1}. The slack variables ξ_1 and ξ_4 are associated with the red points that violate the constraint of hyperplane H_{-1}, and points marked by ξ_2, ξ_3, and ξ_5 are associated with the blue points that violate the constraint of hyperplane H_{+1}. Points that satisfy the constraints of the appropriate hyperplane have $\xi_i = 0$.*

with $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^\tau \geq \mathbf{0}$ and $\boldsymbol{\eta} = (\eta_1, \cdots, \eta_n)^\tau \geq \mathbf{0}$. Fix $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$, and differentiate F_P with respect to β_0, $\boldsymbol{\beta}$,and $\boldsymbol{\xi}$:

$$\frac{\partial F_P}{\partial \beta_0} = -\sum_{i=1}^n \alpha_i y_i, \qquad (11.37)$$

$$\frac{\partial F_P}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i, \qquad (11.38)$$

$$\frac{\partial F_P}{\partial \xi_i} = C - \alpha_i - \eta_i, \quad i = 1, 2, \ldots, n. \qquad (11.39)$$

Setting these derivatives equal to zero and solving yields

$$\sum_{i=1}^n \alpha_i y_i = 0, \ \boldsymbol{\beta}^* = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i, \ \alpha_i = C - \eta_i. \qquad (11.40)$$

Substituting (11.37) into (11.33) gives the dual functional,

$$F_D(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\tau \mathbf{x}_j), \qquad (11.41)$$

which, remarkably, is the same as (11.18) for the linearly separable case. From the constraints $C - \alpha_i - \eta_i = 0$ and $\eta_i \geq 0$, we have that $0 \leq \alpha_i \leq C$. In addition, we have the Karush–Kuhn–Tucker conditions:

$$\begin{aligned} y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - (1 - \xi_i) &\geq 0 && (11.42) \\ \xi_i &\geq 0, && (11.43) \\ \alpha_i &\geq 0, && (11.44) \\ \eta_i &\geq 0, && (11.45) \\ \alpha_i\{y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - (1 - \xi_i)\} &= 0, && (11.46) \\ \xi_i(\alpha_i - C) &= 0, && (11.47) \end{aligned}$$

for $i = 1, 2, \ldots, n$. From (11.47), a slack variable, ξ_i, can be nonzero only if $\alpha_i = C$. The Karush–Kuhn–Tucker complementarity conditions, (11.46) and (11.47), can be used to find the optimal bias β_0.

We can write the dual maximization problem in matrix notation as follows. Find $\boldsymbol{\alpha}$ to

$$\text{maximize} \quad F_D(\boldsymbol{\alpha}) = \mathbf{1}_n^\tau \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\tau \mathbf{H} \boldsymbol{\alpha} \qquad (11.48)$$

$$\text{subject to} \quad \boldsymbol{\alpha}^\tau \mathbf{y} = 0, \ \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_n. \qquad (11.49)$$

The only difference between this optimization problem and that for the linearly separable case, (11.22) and (11.23), is that, here, the Lagrangian coefficients α_i, $i = 1, 2, \ldots, n$, are each bounded above by C; this upper bound restricts the influence of each observation in determining the solution. This type of constraint is referred to as a *box constraint* because $\boldsymbol{\alpha}$ is constrained by the box of side C in the positive orthant. From (11.49), we see that the *feasible region* for the solution to this convex optimization problem is the intersection of the hyperplane $\boldsymbol{\alpha}^\tau \mathbf{y} = 0$ with the box constraint $\mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_n$. If $C = \infty$, then the problem reduces to the hard-margin separable case.

If $\widehat{\boldsymbol{\alpha}}$ solves this optimization problem, then,

$$\widehat{\boldsymbol{\beta}} = \sum_{i \in sv} \widehat{\alpha}_i y_i \mathbf{x}_i \qquad (11.50)$$

yields the optimal weight vector, where the set sv of support vectors contains those observations in $\mathcal{L}$ which satisfy the constraint (11.42).

11.3 Nonlinear Support Vector Machines

So far, we have discussed methods for constructing a linear SVM classifier. But what if a linear classifier is not appropriate for the data set in question? Can we extend the idea of linear SVM to the nonlinear case? The key to constructing a nonlinear SVM is to observe that the observations in $\mathcal{L}$ only enter the dual optimization problem through the inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^\tau \mathbf{x}_j$, $i, j = 1, 2, \ldots, n$.

11.3.1 Nonlinear Transformations

Suppose we transform each observation, $\mathbf{x}_i \in \Re^r$, in $\mathcal{L}$ using some nonlinear mapping $\mathbf{\Phi} : \Re^r \rightarrow \mathcal{H}$, where $\mathcal{H}$ is an $N_{\mathcal{H}}$-dimensional feature space. The nonlinear map $\mathbf{\Phi}$ is generally called the *feature map* and the space $\mathcal{H}$ is called the *feature space*. The space $\mathcal{H}$ may be very high-dimensional, possibly even infinite dimensional. We will generally assume that $\mathcal{H}$ is a Hilbert space of real-valued functions on $\Re$ with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$.

Let

$$\mathbf{\Phi}(\mathbf{x}_i) = (\phi_1(\mathbf{x}_i), \cdots, \phi_{N_{\mathcal{H}}}(\mathbf{x}_i))^\tau \in \mathcal{H}, \quad i = 1, 2, \ldots, n. \qquad (11.51)$$

The transformed sample is then $\{\mathbf{\Phi}(\mathbf{x}_i), y_i\}$, where $y_i \in \{-1, +1\}$ identifies the two classes. If we substitute $\mathbf{\Phi}(\mathbf{x}_i)$ for $\mathbf{x}_i$ in the development of the linear SVM, then data would only enter the optimization problem by way of the inner products $\langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle = \mathbf{\Phi}(\mathbf{x}_i)^\tau \mathbf{\Phi}(\mathbf{x}_j)$. The difficulty in using nonlinear transformations in this way is computing such inner products in high-dimensional space $\mathcal{H}$.

11.3.2 The "Kernel Trick"

The idea behind nonlinear SVM is to find an optimal separating hyperplane (with or without slack variables, as appropriate) in high-dimensional feature space $\mathcal{H}$ just as we did for the linear SVM in input space. Of course, we would expect the dimensionality of $\mathcal{H}$ to be a huge impediment to constructing an optimal separating hyperplane (and classification rule) because of the curse of dimensionality. The fact that this does not become a problem in practice is due to the "kernel trick," which was first applied to SVMs by Cortes and Vapnik (1995).

The so-called kernel trick is a wonderful idea that is widely used in algorithms for computing inner products of the form $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ in feature space $\mathcal{H}$. The trick is that instead of computing these inner products in $\mathcal{H}$, which would be computationally expensive because of its high dimensionality, we compute them using a nonlinear kernel function, $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$, in input space, which helps speed up the computations. Then, we just compute a *linear* SVM, but where the computations are carried out in some other space.

11.3.3 Kernels and Their Properties

A *kernel* K is a function $K : \Re^r \times \Re^r \rightarrow \Re$ such that, for all $\mathbf{x}, \mathbf{y} \in \Re^r$,

$$K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle. \tag{11.52}$$

The kernel function is designed to compute inner-products in $\mathcal{H}$ by using only the original input data. Thus, wherever we see the inner product $\langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$, we substitute the kernel function $K(\mathbf{x}, \mathbf{y})$. The choice of K implicitly determines both Φ and $\mathcal{H}$. The big advantage to using kernels as inner products is that if we are given a kernel function K, then we do not need to know the explicit form of Φ.

We require that the kernel function be symmetric, $K(\mathbf{x}, \mathbf{y}) = K(\mathbf{y}, \mathbf{x})$, and satisfy an inequality, $[K(\mathbf{x}, \mathbf{y})]^2 \leq K(\mathbf{x}, \mathbf{x})K(\mathbf{y}, \mathbf{y})$. derived from the Cauchy–Schwarz inequality. If $K(\mathbf{x}, \mathbf{x}) = 1$ for all $\mathbf{x} \in \Re^r$, this implies that $\|\Phi(\mathbf{x})\|_{\mathcal{H}} = 1$. A kernel K is said to have the *reproducing property* if, for any $f \in \mathcal{H}$,

$$\langle f(\cdot), K(\mathbf{x}, \cdot) \rangle = f(\mathbf{x}). \tag{11.53}$$

If K has this property, we say it is a *reproducing kernel.* K is also called the *representer of evaluation.* In particular, if $f(\cdot) = K(\cdot, \mathbf{x})$, then,

$$\langle K(\mathbf{x}, \cdot), K(\mathbf{y}, \cdot) \rangle = K(\mathbf{x}, \mathbf{y}). \tag{11.54}$$

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be any set of n points in $\mathcal{R}^r$. Then, the $(n \times n)$-matrix $\mathbf{K} = (K_{ij})$, where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = 1, 2, \ldots, n$, is called the *Gram* (or *kernel*) matrix of K with respect to $\mathbf{x}_1, \ldots, \mathbf{x}_n$. If the Gram matrix $\mathbf{K}$ satisfies $\mathbf{u}^\tau \mathbf{K} \mathbf{u} \geq 0$, for any n-vector $\mathbf{u}$, then it is said to be *nonnegative-definite* with nonnegative eigenvalues, in which case we say that K is a nonnegative-definite kernel[1] (or *Mercer kernel*).

If K is a specific Mercer kernel on $\mathcal{R}^r \times \mathcal{R}^r$, we can always construct a unique Hilbert space $\mathcal{H}_K$, say, of real-valued functions for which K is its reproducing kernel. We call $\mathcal{H}_K$ a (real) *reproducing kernel Hilbert space* (rkhs). We write the inner-product and norm of $\mathcal{H}_K$ by $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$ (or just $\langle \cdot, \cdot \rangle$ when K is understood) and $\| \cdot \|_{\mathcal{H}_K}$, respectively.

11.3.4 Examples of Kernels

An example of a kernel is the *inhomogeneous polynomial kernel of degree d*,

$$K(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + c)^d, \quad \mathbf{x}, \mathbf{y} \in \Re^r, \tag{11.55}$$

[1]In the machine-learning literature, nonnegative-definite matrices and kernels are usually referred to as positive-definite matrices and kernels, respectively.

TABLE 11.1. *Kernel functions, $K(\mathbf{x}, \mathbf{y})$, where $\sigma > 0$ is a scale parameter, $a, b, c \geq 0$, and d is an integer. The Euclidean norm is $\|\mathbf{x}\|^2 = \mathbf{x}^\tau \mathbf{x}$.*

Kernel	$K(\mathbf{x}, \mathbf{y})$
Polynomial of degree d	$(\langle \mathbf{x}, \mathbf{y} \rangle + c)^d$
Gaussian radial basis function	$\exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right\}$
Laplacian	$\exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right\}$
Thin-plate spline	$\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right)^2 \log_e\left\{\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right\}$
Sigmoid	$\tanh(a\langle \mathbf{x}, \mathbf{y} \rangle + b)$

where c and d are parameters. The homogeneous form of the kernel occurs when $c = 0$ in (12.55). If $d = 1$ and $c = 0$, the feature map reduces to the identity. Usually, we take $c > 0$. A simple nonlinear map is given by the case $r = 2$ and $d = 2$. If $\mathbf{x} = (x_1, x_2)^\tau$ and $\mathbf{y} = (y_1, y_2)^\tau$, then,

$$K(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + c)^2 = (x_1 y_1 + x_2 y_2 + c)^2 = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle,$$

where $\Phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2, \sqrt{2c}x_1, \sqrt{2}x_2, c)^\tau$ and similarly for $\Phi(\mathbf{y})$. In this example, the function $\Phi(\mathbf{x})$ consists of six features ($\mathcal{H} = \Re^6$), all monomials having degree at most 2. For this kernel, we see that c controls the magnitudes of the constant term and the first-degree term.

In general, there will be $\dim(\mathcal{H}) = \binom{r+d}{d}$ different features, consisting of all monomials having degree at most d. The dimensionality of $\mathcal{H}$ can rapidly become very large: for example, in visual recognition problems, data may consist of 16×16 pixel images (so that each image is turned into a vector of dimension $r = 256$); if $d = 2$, then $\dim(\mathcal{H}) = 33{,}670$, whereas if $d = 4$, we have $\dim(\mathcal{H}) = 186{,}043{,}585$.

Other popular kernels, such as the *Gaussian radial basis function (RBF)*, the *Laplacian kernel*, the *thin-plate spline kernel*, and the *sigmoid kernel*, are given in Table 11.1. Strictly speaking, the sigmoid kernel is not a kernel (it satisfies Mercer's conditions only for certain values of a and b), but it has become very popular in that role in certain situations (e.g., two-layer neural networks).

The Gaussian RBF, Laplacian, and thin-plate spline kernels are examples of *translation-invariant* (or *stationary*) *kernels* having the general form

$K(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$, where $k : \Re^r \to \Re$. The polynomial kernel is an example of a nonstationary kernel. A stationary kernel $K(\mathbf{x}, \mathbf{y})$ is *isotropic* if it depends only upon the distance $\delta = \|\mathbf{x} - \mathbf{y}\|$, i.e., if $K(\mathbf{x}, \mathbf{y}) = k(\delta)$, scaled to have $k(0) = 1$.

It is not always obvious which kernel to choose in any given application. Prior knowledge or a search through the literature can be helpful. If no such information is available, the best approach is to try either a Gaussian RBF, which has only a single parameter (σ) to be determined, or a polynomial kernel of low degree ($d = 1$ or 2). If necessary, more complicated kernels can then be applied to compare results.

String Kernels for Text Categorization

Text categorization is the assignment of natural-language text (or hypertext) documents into a given number of predefined categories based upon the content of those documents (see Section 2.2.1). Although manual categorization of text documents is currently the norm (e.g., using folders to save files, e-mail messages, URLs, etc.), some text categorization is automated (e.g., filters for spam or junk mail to help users cope with the sheer volume of daily e-mail messages). To reduce costs of text categorization tasks, we should expect a greater degree of automation to be present in the future.

In text-categorization problems, *string kernels* have been proposed based upon ideas derived from bioinformatics (see, e.g., Lodhi, Saunders, Shawe-Taylor,Cristianini, and Watkins, 2002).

Let $\mathcal{A}$ be a finite alphabet. A "string"

$$s = s_1 s_2 \cdots s_{|s|} \tag{11.56}$$

is a finite sequence of elements of $\mathcal{A}$, including the empty sequence, where $|s|$ denotes the length of s. We call u a *subsequence* of s (written $u = s(\mathbf{i})$) if there are indices $\mathbf{i} = (i_1, i_2, \cdots, i_{|u|})$, with $1 \leq i_1 < \cdots < i_{|u|} \leq |s|$, such that $u_j = s_{i_j}$, $j = 1, 2, \ldots, |u|$. If the indices $\mathbf{i}$ are contiguous, we say that u is a *substring* of s. The length of u in s is

$$\ell(\mathbf{i}) = i_{|u|} - i_1 + 1, \tag{11.57}$$

which is the number of elements of s overlaid by the subsequence u. For example, let s be the string "cat" ($s_1 = c, s_2 = a, s_3 = t$, $|s| = 3$), and consider all possible 2-symbol sequences, "ca," "ct," and "at," derived from s. For the string $u = \text{ca}$, we have that $u_1 = \text{c} = s_1, u_2 = \text{a} = s_2$, whence, $u = s(\mathbf{i})$, where $\mathbf{i} = (i_1, i_2) = (1, 2)$. Thus, $\ell(\mathbf{i}) = 2$. Similarly, for the subsequence $u = \text{ct}$, $u_1 = \text{c} = s_1, u_2 = \text{t} = s_3$, whence, $\mathbf{i} = (i_1, i_2) = (1, 3)$, and $\ell(\mathbf{i}) = 3$. Also, the subsequence $u = \text{at}$ has $u_1 = \text{a} = s_2, u_2 = \text{t} = s_3$, whence, $\mathbf{i} = (2, 3)$, and $\ell(\mathbf{i}) = 2$.

If $D = \mathcal{A}^m$ is the set of all finite strings of length at most m from $\mathcal{A}$, then, the feature space for a string kernel is $\Re^D$. The feature map Φ_u, operating on a string $s \in \mathcal{A}^m$, is characterized in terms of a given string $u \in \mathcal{A}^m$. To deal with noncontiguous subsequences, define $\lambda \in (0,1)$ as the *drop-off rate* (or *decay factor*); we use λ to weight the interior gaps in the subsequences. The degree of importance we put into a contiguous subsequence is reflected in how small we take the value of λ. The value $\Phi_u(s)$ is computed as follows: identify all subsequences (indexed by $\mathbf{i}$) of s that are identical to u; for each such subsequence, raise λ to the power $\ell(\mathbf{i})$; and then sum the results over all subsequences. Because $\lambda < 1$, larger values of $\ell(\mathbf{i})$ carry less weight than smaller values of $\ell(\mathbf{i})$. We write

$$\Phi_u(s) = \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{\ell(\mathbf{i})}, \quad u \in \mathcal{A}^m. \tag{11.58}$$

In our example above, $\Phi_{\text{ca}}(\text{cat}) = \lambda^2$, $\Phi_{\text{ct}}(\text{cat}) = \lambda^3$, and $\Phi_{\text{at}}(\text{cat}) = \lambda^2$.

Two documents are considered to be "similar" if they have many subsequences in common: the more subsequences they have in common, the more similar they are deemed to be. Note that the degree of contiguity present in a subsequence determines the weight of that substring in the comparison; the closer the subsequence is to a contiguous substring, the more it should contribute to the comparison.

Let s and t be two strings. The kernel associated with the feature maps corresponding to s and t is given by the sum of inner products for all *common* substrings of length m,

$$\begin{aligned} K_m(s,t) &= \sum_{u\in\mathcal{D}} \langle \Phi_u(s), \Phi_u(t) \rangle \\ &= \sum_{u\in\mathcal{D}} \sum_{\mathbf{i}:u=s(\mathbf{i})} \sum_{\mathbf{j}:u=s(\mathbf{j})} \lambda^{\ell(\mathbf{i})+\ell(\mathbf{j})}. \end{aligned} \tag{11.59}$$

The kernel (11.59) is called a *string kernel* (or a *gap-weighted subsequences kernel*). For the example, let t be the string "car" ($t_1 = \text{c}, t_2 = \text{a}, t_3 = \text{r}$, $|t| = 3$). Note that the strings "cat" and "car" are both substrings of the string "cart." The three 2-symbol substrings of t are "ca," "cr," and "ar." For these substrings, we have that $\Phi_{\text{ca}}(\text{car}) = \lambda^2, \Phi_{\text{cr}}(\text{car}) = \lambda^3$, and $\Phi_{\text{ar}}(\text{car}) = \lambda^2$. The inner product (11.62) is given by $K_2(\text{cat},\text{car}) = \langle \Phi_{\text{ca}}(\text{cat}), \Phi_{\text{ca}}(\text{car}) \rangle = \lambda^4$.

The feature maps in feature space are usually normalized to remove any bias introduced by document length. This is equivalent to normalizing the kernel (11.59),

$$K^*_m(s,t) = \frac{K_m(s,t)}{\sqrt{K_m(s,s)K_m(t,t)}}. \tag{11.60}$$

For our example, $K_2(\text{cat}, \text{cat}) = \langle\Phi_{\text{ca}}(\text{cat}), \Phi_{\text{ca}}(\text{cat})\rangle + \langle\Phi_{\text{ct}}(\text{cat}), \Phi_{\text{ct}}(\text{cat})\rangle + \langle\Phi_{\text{at}}(\text{cat}), \Phi_{\text{at}}(\text{cat})\rangle = \lambda^6 + 2\lambda^4$, and, similarly, $K_2(\text{car}, \text{car}) = \lambda^6 + 2\lambda^4$, whence, $K_2^*(\text{cat}, \text{car}) = \lambda^4/(\lambda^6 + 2\lambda^4) = 1/(\lambda^2 + 2)$.

The parameters of the string kernel (11.59) are m and λ. The choices of $m = 5$ and $\lambda = 0.5$ have been found to perform well on segments of certain data sets (e.g., on subsets of the Reuters-21578 data) but do not fare as well when applied to the full data set.

11.3.5 Optimizing in Feature Space

Let K be a kernel. Suppose, first, that the observations in $\mathcal{L}$ are linearly separable in the feature space corresponding to the kernel K. Then, the dual optimization problem is to find $\boldsymbol{\alpha}$ and β_0 to

$$\textit{maximize} \quad F_D(\boldsymbol{\alpha}) = \mathbf{1}_n^\tau \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\tau \mathbf{H} \boldsymbol{\alpha} \tag{11.61}$$

$$\textit{subject to} \quad \boldsymbol{\alpha} \geq \mathbf{0}, \ \boldsymbol{\alpha}^\tau \mathbf{y} = 0, \tag{11.62}$$

where $\mathbf{y} = (y_1, \cdots, y_n)^\tau$, $\mathbf{H} = (H_{ij})$, and

$$H_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) = y_i y_j K_{ij}, \quad i, j = 1, 2, \ldots, n. \tag{11.63}$$

Because K is a kernel, the Gram matrix $\mathbf{K} = (K_{ij})$ is nonnegative-definite, and so is the matrix $\mathbf{H}$ with elements (11.63). Hence, the functional $F_D(\boldsymbol{\alpha})$ is convex (see Exercise 11.8). So, there is a unique solution to this constrained optimization problem. If $\widehat{\boldsymbol{\alpha}}$ and $\widehat{\beta}_0$ solve this problem, then, the SVM decision rule is sign$\{\widehat{f}(\mathbf{x})\}$, where

$$\widehat{f}(\mathbf{x}) = \widehat{\beta}_0 + \sum_{i \in sv} \widehat{\alpha}_i y_i K(\mathbf{x}, \mathbf{x}_i) \tag{11.64}$$

is the optimal separating hyperplane in the feature space corresponding to the kernel K.

In the nonseparable case, using the kernel K, the dual problem of the 1-norm soft-margin optimization problem is to find $\boldsymbol{\alpha}$ to

$$\textit{maximize} \quad F_D^*(\boldsymbol{\alpha}) = \mathbf{1}_n^\tau \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\tau \mathbf{H} \boldsymbol{\alpha} \tag{11.65}$$

$$\textit{subject to} \quad \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}_n, \ \boldsymbol{\alpha}^\tau \mathbf{y} = 0, \tag{11.66}$$

where $\mathbf{y}$ and $\mathbf{H}$ are as above. For an optimal solution, the Karush–Kuhn–Tucker conditions, (11.42)–(11.47), must hold for the primal problem. So, a solution, $\boldsymbol{\alpha}$, to this problem has to satisfy all those conditions. Fortunately, it suffices to check a simpler set of conditions: we have to check that $\boldsymbol{\alpha}$

satisfies (11.66) and that (11.42) holds for all points where $0 \leq \alpha_i < C$ and $\xi_i = 0$, and also for all points where $\alpha_i = C$ and $\xi_i \geq 0$.

11.3.6 Grid Search for Parameters

We need to determine two parameters when using a Gaussian RBF kernel, namely, the cost, C, of violating the constraints and the kernel parameter $\gamma = 1/\sigma^2$. The parameter C in the box constraint can be chosen by searching a wide range of values of C using either CV (usually, 10-fold) on $\mathcal{L}$ or an independent validation set of observations. In practice, it is usual to start the search by trying several different values of C, such as 10, 100, 1,000, 10,000, and so on. A initial grid of values of γ can be selected by trying out a crude set of possible values, say, 0.00001, 0.0001, 0.001, 0.01, 0.1, and 1.0.

When there appears to be a minimum CV misclassification rate within an interval of the two-way grid, we make the grid search finer within that interval. Armed with a two-way grid of values of (C, γ), we apply CV to estimate the generalization error for each cell in that grid. The (C, γ) that has the smallest CV misclassification rate is selected as the solution to the SVM classification problem.

11.3.7 Example: E-mail or Spam?

This example (`spambase`) was described in Section 8.4, where we applied LDA and QDA to a collection of 4,601 messages, comprising 1,813 spam e-mails and 2,788 non-spam e-mails. There are 57 variables (attributes) and each message is labeled as one of the two classes `email` or `spam`.

Here we apply nonlinear SVM (R package `libsvm`) using a Gaussian RBF kernel to the 4,601 messages. The SVM solution depends upon the cost C of violating the constraints and the variance, σ^2, of the Gaussian RBF kernel. After applying a trial-and-error method, we used the following grid of values for C amd $\gamma = 1/\sigma^2$:

C = 10, 80, 100, 200, 500, 1,000,

$\gamma = 0.00001(0.00001)0.0001(0.0001)0.002(0.001)0.01(0.01)0.04.$

In Figure 11.3, we plot the values of the 10-fold CV misclassification rate against the values of γ listed above, where each curve (connected set of points) represents a different value of C. For each C, we see that the CV/10 misclassification curves have similar shapes: a minimum value for γ very close to zero, and for values of γ away from zero, the curve trends upwards. In this initial search, we find a minimum CV/10 misclassification rate of 8.06% at $(C, \gamma) = (500, 0.0002)$ and $(1{,}000, 0.0002)$. We see that the general

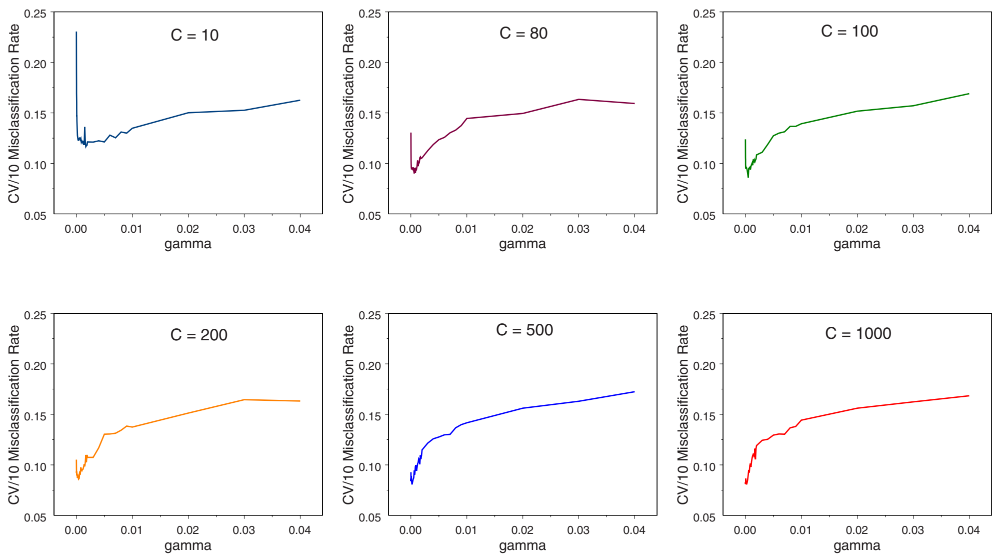

FIGURE 11.3. *SVM cross-validation misclassification rate curves for the spambase data. Initial grid search for the minimum 10-fold CV misclassification rate using* $0.00001 \leq \gamma \leq 0.04$*. The curves correspond to* C *= 10 (dark blue), 80 (brown), 100 (green), 200 (orange), 500 (light blue), and 1,000 (red). Within this intial grid search, the minimum CV/10 misclassification rate is 8.06%, which occurs at* (C, γ) *= (500, 0.0002) and (1,000, 0.0002).*

level of the misclassification rate tends to decrease as C increases and γ decreases together.

A detailed investigation of $C > 1000$ and γ close to zero reveals a minimum CV/10 misclassification rate of 6.91% at $C = 11,000$ and $\gamma = 0.00001$, corresponding to the following 10 CV estimates of the true classification rate:

0.9043, 0.9478, 0.9304, 0.9261, 0.9109,

0.9413, 0.9326, 0.9500. 0.9326, 0.9328.

This solution has 931 support vectors (482 e-mails, 449 spam), which means that a large percentage (79.8%) of the messages (82.7% of the e-mails and 75.2% of the spam) are not support points. Of the 4,601 messages, 2,697 e-mails and 1,676 spam are correctly classified (228 misclassified), yielding an apparent error rate of 4.96%.

This example turns out to be more computationally intensive than are the other binary-classification examples discussed in this chapter. Although the value of γ has very little effect on the speed of computating the 10-fold CV error rate, the speed of computation does depend upon C: as we increase the value of C, the speed of computation slows down considerably.

TABLE 11.2. *Summary of support vector machine (SVM) application to data sets for binary classification. Listed are the sample size (n), number of variables (r), and number of classes (K). Also listed for each data set is the 10-fold cross-validation (CV/10) misclassification rates corresponding to the best choice of (C, γ) for the SVM. The data sets are listed in increasing order of LDA misclassification rates (see Table 8.5).*

Data Set	n	r	K	SVM–CV/10
Breast cancer (logs)	569	30	2	0.0158
Spambase	4601	57	2	0.0691
Ionosphere	351	33	2	0.0427
Sonar	208	60	2	0.1010
BUPA liver disorders	345	6	2	0.2522

Also worth noting is that for fixed γ, increasing C reduces the number of support vectors and the apparent error rate. We cannot make similar general statements about fixed C and increasing γ; however, for fixed C, we generally see that the number of support vectors tends to increase (but not always) with increasing γ.

The nonlinear SVM is clearly a better classifier for this example than is LDA or QDA, whose leave-one-out CV misclassification rate is around 11% for LDA and 17% for QDA, but the amount of computational work involved in the grid search for the SVM solution is much greater and, hence, a lot more expensive.

11.3.8 Binary Classification Examples

We apply the SVM algorithm to the binary classification examples of Section 8.4: the log-transformed breast cancer data, the ionosphere data, the BUPA liver disorders data, the sonar data, and the spambase data. Except for spambase, computations for these examples were very fast.

In Table 11.2, we list the minimum 10-fold CV misclassification rate for each data set. Comparing these results to those of LDA (see Table 8.5, where we used leave-one-out CV), we see that SVM produces remarkable decreases in misclassification rates: the breast cancer rate decreased from 11.3% to 1.58%, the spambase rate decreased from 11.3% to 6.91%, the ionosphere rate decreased from 13.7% to 4.27%, the sonar rate decreased from 24.5% to 10.1%, and the BUPA liver disorders rate decreased from 30.1% to 25.22%.

11.3.9 SVM as a Regularization Method

The SVM classifier can also be regarded as the solution to a particular regularization problem. Let $f \in \mathcal{H}_K$, the reproducing kernel Hilbert space

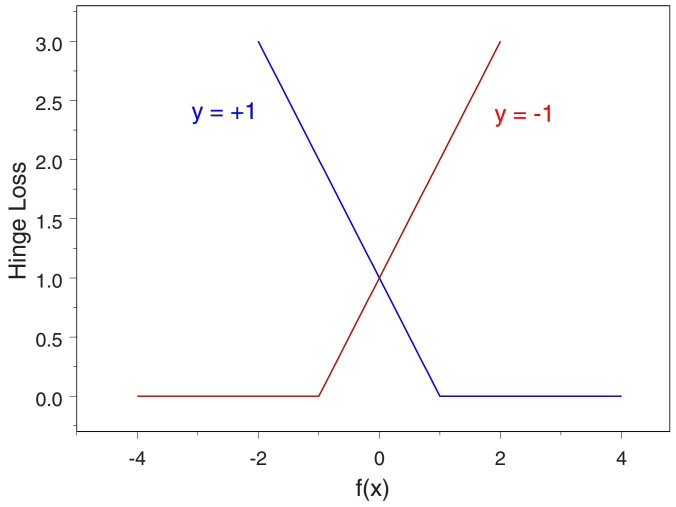

FIGURE 11.4. *Hinge loss function* $(1-yf(\mathbf{x}))_+$ *for* $y=-1$ *and* $y=+1$.

(rkhs) associated with the kernel K, with $\| f \|_{\mathcal{H}_K}{}^2$ the squared-norm of f in $\mathcal{H}_K$.

Consider the classification error, $y_i - f(\mathbf{x}_i)$, where $y_i \in \{-1,+1\}$. Then,

$$|y_i - f(\mathbf{x}_i)| = |y_i(1-y_if(\mathbf{x}_i))| = |1-y_if(\mathbf{x}_i)| = (1-y_if(\mathbf{x}_i))_+, \quad (11.67)$$

$i = 1,2,\ldots,n$, where $(x)_+ = \max\{x,0\}$. The quantity $(1-y_if(\mathbf{x}_i))_+$, which could be zero if all $\mathbf{x}_i$ are correctly classified, is called the *hinge loss function* and is displayed in Figure 11.4. The hinge loss plays a vital role in SVM methodology; indeed, it has been shown to be Bayes consistent for classification in the sense that minimizing the loss function yields the Bayes rule (Lin, 2002). The hinge loss is also related to the misclassification loss function $I_{[y_iC(\mathbf{x}_i)\leq 0]} = I_{[y_if(\mathbf{x}_i)\leq 0]}$. When $f(\mathbf{x}_i) = \pm 1$, the hinge loss is twice the misclassification loss; otherwise, the ratio of the two losses depends upon the sign of $y_if(\mathbf{x}_i)$.

We wish to find a function $f \in \mathcal{H}_K$ to minimize a penalized version of the hinge loss. Specifically, we wish to find $f \in \mathcal{H}_K$ to

$$\textit{minimize} \quad \frac{1}{n}\sum_{i=1}^{n}(1-y_if(\mathbf{x}_i))_+ + \lambda \| f \|_{\mathcal{H}_K}{}^2, \quad (11.68)$$

where $\lambda > 0$. In (11.69), the first term, $n^{-1}\sum_{i=1}^{n}(1-y_if(\mathbf{x}_i))_+$, measures the distance of the data from separability, and the second term, $\lambda \| f \|_{\mathcal{H}_K}^2$, penalizes overfitting. The tuning parameter λ balances the trade-off between estimating f (the first term) and how well f can be approximated (the second term). After the minimizing f has been found, the SVM classifier is $C(\mathbf{x}) = \text{sign}\{f(\mathbf{x})\}$, $\mathbf{x} \in \mathcal{R}^r$.

The optimizing criterion (11.68) is nondifferentiable due to the shape of the hinge-loss function. Fortunately, we can rewrite the problem in a slightly different form and thereby solve it.

We start from the fact that every $f \in \mathcal{H}$ can be written uniquely as the sum of two terms:

$$f(\cdot) = f^{\|}(\cdot) + f^{\perp}(\cdot) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \cdot) + f^{\perp}(\cdot), \tag{11.69}$$

where $f^{\|} \in \mathcal{H}_K$ is the projection of f onto the subspace $\mathcal{H}_K$ of $\mathcal{H}$ and $f^{\perp}$ is in the subspace perpendicular to $\mathcal{H}_K$; that is, $\langle f^{\perp}(\cdot), K(\mathbf{x}_i, \cdot)\rangle_{\mathcal{H}} = 0$, $i = 1, 2, \ldots, n$. We can write $f(\mathbf{x}_i)$ via the reproducing property as follows:

$$f(\mathbf{x}_i) = \langle f(\cdot), K(\mathbf{x}_i, \cdot)\rangle = \langle f^{\|}(\cdot), K(\mathbf{x}_i, \cdot)\rangle + \langle f^{\perp}(\cdot), K(\mathbf{x}_i, \cdot)\rangle. \tag{11.70}$$

Because the second term on the rhs is zero, then,

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}), \tag{11.71}$$

independent of $f^{\perp}$, where we used (11.69) and $\langle K(\mathbf{x}_i, \cdot), K(\mathbf{x}_j, \cdot)\rangle_{\mathcal{H}_K} = K(\mathbf{x}_i, \mathbf{x}_j)$. Now, from (11.69),

$$\begin{aligned} \| f \|^2_{\mathcal{H}_K} &= \| \sum_i \alpha_i K(\mathbf{x}_i, \cdot) + f^{\perp} \|^2_{\mathcal{H}_K} \\ &= \| \sum_i \alpha_i K(\mathbf{x}_i, \cdot) \|^2_{\mathcal{H}_K} + \| f^{\perp} \|^2_{\mathcal{H}_K} \\ &\geq \| \sum_i \alpha_i K(\mathbf{x}_i, \cdot) \|^2_{\mathcal{H}_K}, \end{aligned} \tag{11.72}$$

with equality iff $f^{\perp} = 0$, in which case any $f \in \mathcal{H}_K$ that minimizes (11.68) admits a representation of the form (11.71). This important result is known as the *representer theorem* (Kimeldorf and Wahba, 1971); it says that the minimizing f (which would live in an infinite-dimensional rkhs if, for example, the kernel is a Gaussian RBF) can be written as a linear combination of a reproducing kernel evaluated at each of the n data points.

From (11.72), we have that $\| f \|^2_{\mathcal{H}_K} = \sum_i \sum_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) = \| \boldsymbol{\beta} \|^2$, where $\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i \boldsymbol{\Phi}(\mathbf{x}_i)$. If the space $\mathcal{H}_K$ consists of linear functions of the form $f(\mathbf{x}) = \beta_0 + \boldsymbol{\Phi}(\mathbf{x})^{\tau}\boldsymbol{\beta}$ with $\| f \|^2_{\mathcal{H}_K} = \| \boldsymbol{\beta} \|^2$, then the problem of finding f in (11.68) is equivalent to one of finding β_0 and $\boldsymbol{\beta}$ to

$$\textit{minimize} \quad \frac{1}{n} \sum_{i=1}^{n} (1 - y_i(\beta_0 + \boldsymbol{\Phi}(\mathbf{x}_i)^{\tau}\boldsymbol{\beta}))_+ + \lambda \| \boldsymbol{\beta} \|^2 . \tag{11.73}$$

Then, (11.68), which is nondifferentiable due to the hinge loss function, can be reformulated in terms of solving the 1-norm soft-margin optimization problem (11.34)–(11.35).

11.4 Multiclass Support Vector Machines

Often, data are derived from more than two classes. In the multiclass situation, $\mathbf{X} \in \Re^r$ is a random r-vector chosen for classification purposes and $Y \in \{1, 2, \ldots, K\}$ is a class label, where K is the number of classes. Because SVM classifiers are formulated for only two classes, we need to know if (and how) the SVM methodology can be extended to distinguish between $K > 2$ classes. There have been several attempts to define such a multiclass SVM strategy.

11.4.1 Multiclass SVM as a Series of Binary Problems

The standard SVM strategy for a multiclass classification problem (over K classes) has been to reduce it to a series of binary problems. There are different approaches to this strategy:

One-versus-rest: Divide the K-class problem into K binary classification subproblems of the type "kth class" vs. "not kth class," $k = 1, 2, \ldots, K$. Corresponding to the kth subproblem, a classifier $\widehat{f}_k$ is constructed in which the kth class is coded as positive and the union of the other classes is coded as negative. A new $\mathbf{x}$ is then assigned to the class with the largest value of $\widehat{f}_k(\mathbf{x})$, $k = 1, 2, \ldots, K$, where $\widehat{f}_k(\mathbf{x})$ is the optimal SVM solution for the binary problem of the kth class versus the rest.

One-versus-one: Divide the K-class problem into $\binom{K}{2}$ comparisons of all pairs of classes. A classifier $\widehat{f}_{jk}$ is constructed by coding the jth class as positive and the kth class as negative, $j, k = 1, 2, \ldots, K$, $j \neq k$. Then, for a new $\mathbf{x}$, aggregate the votes for each class and assign $\mathbf{x}$ to the class having the most votes.

Even though these strategies are widely used in practice to resolve multiclass SVM classification problems, one has to be cautious about their use.

In Table 11.3, we report the CV/10 misclassification rates for one-versus-one multiclass SVM applied to the same data sets from Section 8.7. Also listed in Table 11.3 are the values of (C, γ) that yield the minimum misclassification rate for each data set. It is instructive to compare these rates with those in Table 8.7, where we used LDA and QDA. We see that for

TABLE 11.3. *Summary of support vector machine (SVM) "one-versus-one" classification results for data sets with more than two classes. Listed are the sample size (n), number of variables (r), and number of classes (K). Also listed for each data set is the 10-fold cross-validation (CV/10) misclassification rates corresponding to the best choice of (C, γ). The data sets are listed in increasing order of LDA misclassification rates (Table 8.7).*

Data Set	n	r	K	SVM–CV/10	C	γ
Wine	178	13	3	0.0169	10^6	8×10^{-8}
Iris	150	4	3	0.0200	100	0.002
Primate scapulae	105	7	5	0.0286	100	0.0002
Shuttle	43,500	8	7	0.0019	10	0.0001
Diabetes	145	5	3	0.0414	100	0.000009
Pendigits	10,992	16	10	0.0031	10	0.0001
E-coli	336	7	8	0.1280	10	1.0
Vehicle	846	18	4	0.1501	600	0.00005
Letter recognition	20,000	16	26	0.0183	50	0.04
Glass	214	9	6	0.0093	10	0.001
Yeast	1,484	8	10	0.3935	10	7.0

the shuttle, diabetes, pendigits, vehicle, letter recognition, glass, and yeast data sets, the SVM method performs better than does the LDA method; for the iris, primate scapulae, and e-coli data sets, the SVM and LDA methods perform about the same; and LDA performs better than does SVM for the wine data set. Thus, neither one-versus-one SVM nor LDA performs uniformly best for all of these data sets.

The one-versus-rest approach is popular for carrying out text categorization tasks, where each document may belong to more than one class. Although it enjoys the optimality property of the SVM method for each binary subproblem, it can yield a different classifier than the Bayes optimal classifier for the multiclass case. Furthermore, the classification success of the one-versus-rest approach depends upon the extent of the class-size imbalance of each subproblem and whether one class dominates all other classes when determining the most-probable class for each new $\mathbf{x}$.

The one-versus-one approach, which uses only those observations belonging to the classes involved in each pairwise comparison, suffers from the problem of having to use smaller samples to train each classifier, which may, in turn, increase the variance of the solution.

11.4.2 A True Multiclass SVM

To construct a true multiclass SVM classifier, we need to consider all K classes, $\Pi_1, \Pi_2, \ldots, \Pi_K$, simultaneously, and the classifier has to reduce to the binary SVM classifier if $K = 2$. Here we describe the construction due to Lee, Lin, and Wahba (2004).

Let $\mathbf{v}_1, \ldots, \mathbf{v}_K$ be a sequence of K-vectors, where $\mathbf{v}_k$ has a 1 in the kth position and whose elements sum to zero, $k = 1, 2, \ldots, K$; that is, let

$$\begin{aligned}
\mathbf{v}_1 &= \left(1, -\frac{1}{K-1}, \cdots, -\frac{1}{K-1}\right)^{\tau} \\
\mathbf{v}_2 &= \left(-\frac{1}{K-1}, 1, \cdots . -\frac{1}{K-1}\right)^{\tau} \\
&\vdots \\
\mathbf{v}_K &= \left(-\frac{1}{K-1}, -\frac{1}{K-1}, \cdots, 1\right)^{\tau}.
\end{aligned}$$

Note that if $K = 2$, then $\mathbf{v}_1 = (1, -1)^{\tau}$ and $\mathbf{v}_2 = (-1, 1)^{\tau}$. Every $\mathbf{x}_i$ can be labeled as one of these K vectors; that is, $\mathbf{x}_i$ has label $\mathbf{y}_i = \mathbf{v}_k$ if $\mathbf{x}_i \in \Pi_k$, $i = 1, 2, \ldots, n$, $k = 1, 2, \ldots, K$.

Next, we generalize the separating function $f(\mathbf{x})$ to a K-vector of separating functions,

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_K(\mathbf{x}))^{\tau}, \tag{11.74}$$

where

$$f_k(\mathbf{x}) = \beta_{0k} + h_k(\mathbf{x}), \quad h_k \in \mathcal{H}_K, \quad k = 1, 2, \ldots, K. \tag{11.75}$$

In (11.75), $\mathcal{H}_K$ is a reproducing-kernel Hilbert space (rkhs) spanned by the $\{K(\mathbf{x}_i, \cdot), i = 1, 2, \ldots, n\}$. For example, in the linear case, $h_k(\mathbf{x}) = \mathbf{x}^{\tau}\boldsymbol{\beta}_k$, for some vector of coefficients $\boldsymbol{\beta}_k$. We also assume, for uniqueness, that

$$\sum_{k=1}^{K} f_k(\mathbf{x}) = 0. \tag{11.76}$$

Let $\mathbf{L}(\mathbf{y}_i)$ be a K-vector with 0 in the kth position if $\mathbf{x}_i \in \Pi_k$, and 1 in all other positions; this vector represents the equal costs of misclassifying $\mathbf{x}_i$ (and allows for an unequal misclassification cost structure if appropriate). If $K = 2$ and $\mathbf{x}_i \in \Pi_1$, then $\mathbf{L}(\mathbf{y}_i) = (0, 1)^{\tau}$, while if $\mathbf{x}_i \in \Pi_2$, then $\mathbf{L}(\mathbf{y}_i) = (1, 0)^{\tau}$.

The multiclass generalization of the optimization problem (11.68) is, therefore, to find functions $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_K(\mathbf{x}))^{\tau}$ satisfying (11.76) which

$$\textit{minimize} \quad I_{\lambda}(\mathbf{f}, \mathcal{Y}) = \frac{1}{n}\sum_{i=1}^{n}[\mathbf{L}(\mathbf{y}_i)]^{\tau}(\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)_+ + \frac{\lambda}{2}\sum_{k=1}^{K} \| h_k \|^2, \tag{11.77}$$

where $(\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)_+ = ((f_1(\mathbf{x}_i) - y_{i1})_+, \cdots, (f_K(\mathbf{x}_i) - y_{iK})_+)^{\tau}$ and $\mathcal{Y} = (\mathbf{y}_1, \cdots, \mathbf{y}_n)$ is a $(K \times n)$-matrix.

By setting $K = 2$, we can see that (11.77) is a generalization of (11.68). If $\mathbf{x}_i \in \Pi_1$, then $\mathbf{y}_i = \mathbf{v}_1 = (1, -1)^\tau$, and

$$\begin{aligned} [\mathbf{L}(\mathbf{y}_i)]^\tau (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)_+ &= (0,1)((f_1(\mathbf{x}_i) - 1)_+, (f_2(\mathbf{x}_i) + 1)_+)^\tau \\ &= (f_2(\mathbf{x}_i) + 1)_+ \\ &= (1 - f_1(\mathbf{x}_i))_+, \end{aligned} \tag{11.78}$$

while if $\mathbf{x}_i \in \Pi_2$, then $\mathbf{y}_i = \mathbf{v}_2 = (-1, 1)$, and

$$[\mathbf{L}(\mathbf{y}_i)]^\tau (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)_+ = (f_1(\mathbf{x}_i) + 1)_+. \tag{11.79}$$

So, the first term (with f) in (11.68) is identical to the first term (with f_1) in (11.77) when $K = 2$. If we set $K = 2$ in the second term of (11.77), we have that

$$\sum_{k=1}^{2} \| h_k \|^2 = \| h_1 \|^2 + \| -h_1 \|^2 = 2 \| h_1 \|^2, \tag{11.80}$$

so that the second terms of (11.68) and (11.77) are identical.

The function $h_k \in \mathcal{H}_K$ can be decomposed into two parts:

$$h_k(\cdot) = \sum_{\ell=1}^{n} \beta_{\ell k} K(\mathbf{x}_\ell, \cdot) + h_k^\perp(\cdot), \tag{11.81}$$

where the $\{\beta_{\ell k}\}$ are constants and $h_k^\perp(\cdot)$ is an element in the rkhs orthogonal to $\mathcal{H}_K$. Substituting (11.76) into (11.77), then using (11.81), and rearranging terms, we have that

$$f_K(\cdot) = -\sum_{k=1}^{K-1} \beta_{0k} - \sum_{k=1}^{K-1} \sum_{i=1}^{n} \beta_{ik} K(\mathbf{x}_i, \cdot) - \sum_{k=1}^{K-1} h_k^\perp(\cdot). \tag{11.82}$$

Because $K(\cdot, \cdot)$ is a reproducing kernel,

$$\langle h_k, K(\mathbf{x}_i, \cdot) \rangle = h_k(\mathbf{x}_i), \quad i = 1, 2, \ldots, n, \tag{11.83}$$

and so,

$$\begin{aligned} f_k(\mathbf{x}_i) &= \beta_{0k} + h_k(\mathbf{x}_i) \\ &= \beta_{0k} + \langle h_k, K(\mathbf{x}_i, \cdot) \rangle \\ &= \beta_{0k} + \langle \sum_{\ell=1}^{n} \beta_{\ell k} K(\mathbf{x}_\ell, \cdot) + h_k^\perp(\cdot), K(\mathbf{x}_i, \cdot) \rangle \\ &= \beta_{0k} + \sum_{\ell=1}^{n} \beta_{\ell k} K(\mathbf{x}_\ell, \mathbf{x}_i). \end{aligned} \tag{11.84}$$

Note that, for $k = 1, 2, \ldots, K-1$,

$$\begin{aligned} \| h_k(\cdot) \|^2 &= \| \sum_{\ell=1}^{n} \beta_{\ell k} K(\mathbf{x}_\ell, \cdot) + h_k^{\perp}(\cdot) \|^2 \\ &= \sum_{\ell=1}^{n} \sum_{i=1}^{n} \beta_{\ell k} \beta_{ik} K(\mathbf{x}_\ell, \mathbf{x}_i) + \| h_k^{\perp}(\cdot) \|^2, \end{aligned} \tag{11.85}$$

and, for $k = K$,

$$\| h_K(\cdot) \|^2 = \| \sum_{k=1}^{K-1} \sum_{i=1}^{n} \beta_{ik} K(\mathbf{x}_i, \cdot) \|^2 + \| \sum_{k=1}^{K-1} h_k^{\perp}(\cdot) \|^2 . \tag{11.86}$$

Thus, to minimize (11.86), we set $h_k^{\perp}(\cdot) = 0$ for all k.

From (11.84), the zero-sum constraint (11.76) becomes

$$\bar{\beta}_0 + \sum_{\ell=1}^{n} \bar{\beta}_\ell K(\mathbf{x}_\ell, \cdot) = 0, \tag{11.87}$$

where $\bar{\beta}_0 = K^{-1} \sum_{k=1}^{K} \beta_{0k}$ and $\bar{\beta}_i = K^{-1} \sum_{k=1}^{K} \beta_{ik}$. At the n data points, $\{\mathbf{x}_i, i = 1, 2, \ldots, n\}$, (11.87) in matrix notation is given by

$$\left(\sum_{k=1}^{K} \beta_{0k} \right) \mathbf{1}_n + \mathbf{K} \left(\sum_{k=1}^{K} \boldsymbol{\beta}_{\cdot k} \right) = \mathbf{0}, \tag{11.88}$$

where $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))$ is an $(n \times n)$ Gram matrix and $\boldsymbol{\beta}_{\cdot k} = (\beta_{1k}, \cdots, \beta_{nk})^\tau$. Let $\beta_{0k}^* = \beta_{0k} - \bar{\beta}_0$ and $\beta_{ik}^* = \beta_{ik} - \bar{\beta}_i$. Using (11.87), we see that the centered version of (11.84) is $f_k^*(\mathbf{x}_i) = \beta_{0k}^* + \sum_{\ell=1}^{n} \beta_{\ell k}^* K(\mathbf{x}_\ell, \mathbf{x}_i) = f_k(\mathbf{x}_i)$. Then,

$$\sum_{k=1}^{K} \| h_k^*(\cdot) \|^2 = \sum_{k=1}^{K} \boldsymbol{\beta}_{\cdot k}^\tau \mathbf{K} \boldsymbol{\beta}_{\cdot k} - K \bar{\boldsymbol{\beta}}^\tau \mathbf{K} \bar{\boldsymbol{\beta}} \leq \sum_{k=1}^{K} \boldsymbol{\beta}_{\cdot k}^\tau \mathbf{K} \boldsymbol{\beta}_{\cdot k} = \sum_{k=1}^{K} \| h_k(\cdot) \|^2, \tag{11.89}$$

where $\bar{\boldsymbol{\beta}} = (\bar{\beta}_1, \cdots, \bar{\beta}_n)^\tau$; if $\mathbf{K}\bar{\boldsymbol{\beta}} = \mathbf{0}$, the inequality becomes an equality and so $\sum_{k=1}^{K} \beta_{0k} = 0$. Thus,

$$0 = K^2 \bar{\boldsymbol{\beta}}^\tau \mathbf{K} \bar{\boldsymbol{\beta}} = \| \sum_{i=1}^{n} (\sum_{k=1}^{K} \beta_{ik}) K(\mathbf{x}_i, \cdot) \|^2 = \| \sum_{k=1}^{K} \sum_{i=1}^{n} \beta_{ik} K(\mathbf{x}_i, \cdot) \|^2, \tag{11.90}$$

whence, $\sum_{k=1}^{K} \sum_{i=1}^{n} \beta_{ik} K(\mathbf{x}_i, \mathbf{x}) = 0$, for all $\mathbf{x}$. Thus,

$$\sum_{k=1}^{K} \left\{ \beta_{0k} + \sum_{i=1}^{n} \beta_{ik} K(\mathbf{x}_i, \mathbf{x}) \right\} = 0, \tag{11.91}$$

for every $\mathbf{x}$. So, minimizing (11.77) under the zero-sum constraint (11.76) only at the n data points is equivalent to minimizing (11.77) under the same constraint for every $\mathbf{x}$.

We next construct a Lagrangian formulation of the optimization problem (11.77) using the following notation. Let $\boldsymbol{\xi}_i = (\xi_{i1}, \cdots, \xi_{iK})^\tau$ be a K-vector of slack variables corresponding to $(f(\mathbf{x}_i) - y_i)_+$, $i = 1, 2, \ldots, n$, and let $(\boldsymbol{\xi}_{\cdot 1}, \cdots, \boldsymbol{\xi}_{\cdot K}) = (\boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_n)^\tau$ be the $(n \times K)$-matrix whose kth column is $\boldsymbol{\xi}_{\cdot k}$ and whose ith row is $\boldsymbol{\xi}_i$. Let $(\mathbf{L}_1, \cdots, \mathbf{L}_K) = (\mathbf{L}(\mathbf{y}_1), \cdots, \mathbf{L}(\mathbf{y}_n))^\tau$ be the $(n \times K)$-matrix whose kth column is $\mathbf{L}_k$ and whose ith row is $\mathbf{L}(\mathbf{y}_i) = (L_{i1}, \cdots, L_{iK})$. Let $(\mathbf{y}_{\cdot 1}, \cdots, \mathbf{y}_{\cdot K}) = (\mathbf{y}_1, \cdots, \mathbf{y}_n)^\tau$ denote the $(n \times K)$-matrix whose kth column is $\mathbf{y}_{\cdot k}$ and whose ith row is $\mathbf{y}_i$.

The primal problem is to find $\{\beta_{0k}\}$, $\{\boldsymbol{\beta}_{\cdot k}\}$, and $\{\boldsymbol{\xi}_{\cdot k}\}$ to

$$\textit{minimize} \quad \sum_{k=1}^{K} \mathbf{L}_k^\tau \boldsymbol{\xi}_{\cdot k} + \frac{n\lambda}{2} \sum_{k=1}^{K} \boldsymbol{\beta}_{\cdot k}^\tau \mathbf{K} \boldsymbol{\beta}_{\cdot k} \tag{11.92}$$

subject to

$$\begin{aligned} \beta_{0k}\mathbf{1}_n + \mathbf{K}\boldsymbol{\beta}_{\cdot k} - \mathbf{y}_{\cdot k} &\leq \boldsymbol{\xi}_{\cdot k}, \quad k = 1, 2, \ldots, K, & (11.93)\\ \boldsymbol{\xi}_{\cdot k} &\geq \mathbf{0}, \quad k = 1, 2, \ldots, K, & (11.94)\\ (\sum_{k=1}^{K} \beta_{0k})\mathbf{1}_n + \mathbf{K}(\sum_{k=1}^{K} \boldsymbol{\beta}_{\cdot k}) &= \mathbf{0}. & (11.95) \end{aligned}$$

Form the primal functional $F_P = F_P(\{\beta_{0k}\}, \{\boldsymbol{\beta}_{\cdot k}\}, \{\boldsymbol{\xi}_{\cdot k}\})$, where

$$\begin{aligned} F_P = {} & \sum_{k=1}^{K} \mathbf{L}_k^\tau \boldsymbol{\xi}_{\cdot k} + \frac{n\lambda}{2} \sum_{k=1}^{K} \boldsymbol{\beta}_{\cdot k}^\tau \mathbf{K} \boldsymbol{\beta}_{\cdot k} \\ & + \sum_{k=1}^{K} \boldsymbol{\alpha}_{\cdot k}^\tau (\beta_{0k}\mathbf{1}_n + \mathbf{K}\boldsymbol{\beta}_{\cdot k} - \mathbf{y}_{\cdot k} - \boldsymbol{\xi}_{\cdot k}) \\ & - \sum_{k=1}^{K} \boldsymbol{\gamma}_k^\tau \boldsymbol{\xi}_{\cdot k} + \boldsymbol{\delta}^\tau \left((\sum_{k=1}^{K} \beta_{0k})\mathbf{1}_n + \mathbf{K}(\sum_{k=1}^{K} \boldsymbol{\beta}_{\cdot k}) \right). \end{aligned} \tag{11.96}$$

In (11.96), $\boldsymbol{\alpha}_{\cdot k} = (\alpha_{1k}, \cdots, \alpha_{nk})^\tau$ and $\boldsymbol{\gamma}_k$ are n-vectors of nonnegative Lagrange multipliers for the inequality constraints (11.93) and (11.94), respectively, and $\boldsymbol{\delta}$ is an n-vector of unconstrained Lagrange multipliers for the equality constraint (11.95).

Differentiating (11.96) with respect to β_{0k}, $\boldsymbol{\beta}_{\cdot k}$, and $\boldsymbol{\xi}_{\cdot k}$ yields

$$\begin{aligned} \frac{\partial F_P}{\partial \beta_{0k}} &= (\boldsymbol{\alpha}_{\cdot k} + \boldsymbol{\delta})^\tau \mathbf{1}_n, & (11.97)\\ \frac{\partial F_P}{\partial \boldsymbol{\beta}_{\cdot k}} &= n\lambda \mathbf{K}\boldsymbol{\beta}_{\cdot k} + \mathbf{K}\boldsymbol{\alpha}_{\cdot k} + \mathbf{K}\boldsymbol{\delta}, & (11.98) \end{aligned}$$

$$\begin{aligned}\frac{\partial F_P}{\partial \boldsymbol{\xi}_{\cdot k}} &= \mathbf{L}_k - \boldsymbol{\alpha}_{\cdot k} - \boldsymbol{\gamma}_k, &(11.99)\\ \boldsymbol{\alpha}_{\cdot k} &\geq \mathbf{0}, &(11.100)\\ \boldsymbol{\gamma}_k &\geq \mathbf{0}. &(11.101)\end{aligned}$$

The Karush–Kuhn–Tucker complementarity conditions are

$$\begin{aligned}\boldsymbol{\alpha}_{\cdot k}(\beta_{0k}\mathbf{1}_n + \mathbf{K}\boldsymbol{\beta}_{\cdot k} - \mathbf{y}_{\cdot k} - \boldsymbol{\xi}_{\cdot k})^\tau &= 0, \quad k = 1, 2, \ldots, K, &(11.102)\\ \boldsymbol{\gamma}_k \boldsymbol{\xi}_{\cdot k}^\tau &= 0, \quad k = 1, 2, \ldots, K, &(11.103)\end{aligned}$$

where, from (11.99), $\boldsymbol{\gamma}_k = \mathbf{L}_k - \boldsymbol{\alpha}_{\cdot k}$. Note that (11.102) and (11.103) are outer products of two column vectors, meaning that each of the n^2 elementwise products of those vectors are zero.

From (11.99) and (11.101), we have that $\mathbf{0} \leq \boldsymbol{\alpha}_{\cdot k} \leq \mathbf{L}_k$, $k = 1.2. \ldots, K$. Suppose, for some i, $0 < \alpha_{ik} < L_{ik}$; then, $\gamma_{ik} > 0$, and, from (11.103), $\xi_{ik} = 0$, whence, from (11.102), $y_{ik} = \beta_{0k} + \sum_{\ell=1}^{n} \beta_{\ell k} K(\mathbf{x}_\ell, \mathbf{x}_i)$.

Setting the derivatives equal to zero for $k = 1, 2, \ldots, K$ yields $\boldsymbol{\delta} = -\bar{\boldsymbol{\alpha}} = -K^{-1}\sum_{k=1}^{K} \boldsymbol{\alpha}_{\cdot k}$ from (11.97), whence, $(\boldsymbol{\alpha}_{\cdot k} - \bar{\boldsymbol{\alpha}})^\tau \mathbf{1}_n = 0$, and, from (11.98), $\boldsymbol{\beta}_{\cdot k} = -(n\lambda)^{-1}(\boldsymbol{\alpha}_{\cdot k} - \bar{\boldsymbol{\alpha}})$, assuming that $\mathbf{K}$ is positive-definite. If $\mathbf{K}$ is not positive-definite, then $\boldsymbol{\beta}_{\cdot k}$ is not uniquely determined. Because (11.97), (11.98), and (11.99) are each zero, we construct the dual functional F_D by using them to remove a number of the terms of F_P.

The resulting dual problem is to find $\{\boldsymbol{\alpha}_{\cdot k}\}$ to

$$\textit{minimize} \quad F_D = \frac{1}{2}\sum_{k=1}^{K}(\boldsymbol{\alpha}_{\cdot k} - \bar{\boldsymbol{\alpha}})^\tau \mathbf{K}(\boldsymbol{\alpha}_{\cdot k} - \bar{\boldsymbol{\alpha}}) + n\lambda \sum_{k=1}^{K} \boldsymbol{\alpha}_{\cdot k}^\tau \mathbf{y}_{\cdot k} \quad (11.104)$$

$$\begin{aligned}\textit{subject to} \qquad \mathbf{0} \leq \boldsymbol{\alpha}_{\cdot k} \leq \mathbf{L}_k, &\quad k = 1, 2, \ldots, K, &(11.105)\\ (\boldsymbol{\alpha}_{\cdot k} - \bar{\boldsymbol{\alpha}})^\tau \mathbf{1}_n = 0, &\quad k = 1, 2, \ldots, K. &(11.106)\end{aligned}$$

From the solution, $\{\widehat{\boldsymbol{\alpha}}_{\cdot k}\}$, to this quadratic programming problem, we set

$$\widehat{\boldsymbol{\beta}}_{\cdot k} = -(n\lambda)^{-1}(\widehat{\boldsymbol{\alpha}}_{\cdot k} - \widehat{\bar{\boldsymbol{\alpha}}}), \quad (11.107)$$

where $\widehat{\bar{\boldsymbol{\alpha}}} = K^{-1}\sum_{k=1}^{K} \widehat{\boldsymbol{\alpha}}_{\cdot k}$.

The multiclass classification solution for a new $\mathbf{x}$ is given by

$$C_k(\mathbf{x}) = \arg\max_k \{\widehat{f}_k(\mathbf{x})\}, \quad (11.108)$$

where

$$\widehat{f}_k(\mathbf{x}) = \widehat{\beta}_{0k} + \sum_{\ell=1}^{n} \widehat{\beta}_{\ell k} K(\mathbf{x}_\ell, \mathbf{x}), \quad k = 1, 2, \ldots, K. \quad (11.109)$$

Suppose the row vector $\widehat{\boldsymbol{\alpha}}_i = (\widehat{\alpha}_{i1}, \cdots, \widehat{\alpha}_{iK}) = \mathbf{0}$ for $(\mathbf{x}_i, \mathbf{y}_i)$; then, from (11.107), $\widehat{\boldsymbol{\beta}}_i = (\widehat{\beta}_{i1}, \cdots, \widehat{\beta}_{iK}) = \mathbf{0}$. It follows that the term $\widehat{\beta}_{ik} K(\mathbf{x}_i, \mathbf{x}) = 0$, $k = 1, 2, \ldots, K$. Thus, any term involving $(\mathbf{x}_i, \mathbf{y}_i)$ does not appear in (11.109); in other words, it does not matter whether $(\mathbf{x}_i, \mathbf{y}_i)$ is or is not included in the learning set $\mathcal{L}$ because it has no effect on the solution. This result leads us to a definition of support vectors: an observation $(\mathbf{x}_i, \mathbf{y}_i)$ is called a *support vector* if $\widehat{\boldsymbol{\beta}}_i = (\widehat{\beta}_{i1}, \cdots, \widehat{\beta}_{iK}) \neq \mathbf{0}$. As in the binary SVM solution, it is in our computational best interests for there to be relatively few support vectors for any given application.

The one issue remaining is the choice of tuning parameter λ (and any other parameters involved in the computation of the kernel). A *generalized approximate cross-validation* (GACV) method is derived in Lee, Lin, and Wahba (2004) based upon an approximation to the leave-one-out cross-validation technique used for penalized-likelihood methods. The basic idea behind GACV is the following. Write (11.77) as

$$I_\lambda(\mathbf{f}, \boldsymbol{\mathcal{Y}}) = n^{-1} \sum_{i=1}^{n} g(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_i)) + J_\lambda(\mathbf{f}), \tag{11.110}$$

where $g(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_i)) = [\mathbf{L}(\mathbf{y}_i)]^\tau (\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i)_+$ and $J_\lambda(\mathbf{f}) = (\lambda/2) \sum_{i=1}^{n} \| h_j \|^2$. Let $\mathbf{f}_\lambda = \arg\min_{\mathbf{f}} I_\lambda(\mathbf{f}, \mathcal{Y})$ and let $\mathbf{f}_\lambda^{(-i)}$ denote that $\mathbf{f}_\lambda$ that yields the minimum of $I_\lambda(\mathbf{f}, \mathcal{Y})$ by omitting the ith observation $(\mathbf{x}_i, \mathbf{y}_i)$ from the first term in (11.110). If we write

$$g(\mathbf{y}_i, \mathbf{f}_\lambda^{(-i)}(\mathbf{x}_i)) = g(\mathbf{y}_i, \mathbf{f}_\lambda(\mathbf{x}_i)) + [g(\mathbf{y}_i, \mathbf{f}_\lambda^{(-i)}(\mathbf{x}_i)) - g(\mathbf{y}_i, \mathbf{f}_\lambda(\mathbf{x}_i))], \tag{11.111}$$

then the λ that minimizes $n^{-1} \sum_{i=1}^{n} g(\mathbf{y}_i, \mathbf{f}_\lambda^{(-i)}(\mathbf{x}_i))$ is found by using a suitable approximation of $D(\lambda) = n^{-1} \sum_{i=1}^{n} [g(\mathbf{y}_i, \mathbf{f}_\lambda^{(-i)}(\mathbf{x}_i)) - g(\mathbf{y}_i, \mathbf{f}_\lambda(\mathbf{x}_i))]$, computed over a grid of values of λ.

This solution of the multiclass SVM problem has been found to be successful in simulations and in analyzing real data. Comparisons of various multiclass classification methods, such as multiclass SVM, "all-versus-rest," LDA, and QDA, over a number of data sets show that no one classification method appears to be superior for all situations studied; performance appears to depend upon the idiosyncracies of the data to be analyzed.

11.5 Support Vector Regression

The SVM was designed for classification. Can we extend (or generalize) the idea to regression? How would the main concepts used in SVM — convex optimization, optimal separating hyperplane, support vectors, margin, sparseness of the solution, slack variables, and the use of kernels — translate to the regression situation? It turns out that all of these concepts find their analogues in regression analysis and they add a different view to the topic than the views we saw in Chapter 5.

11.5.1 ϵ-Insensitive Loss Functions

In SVM classification, the margin is used to determine the amount of separation between two nonoverlapping classes of points: the bigger the margin, the more confident we are that the optimal separating hyperplane is a superior classifier. In regression, we are not interested in separating points but in providing a function of the input vectors that would track the points closely. Thus, a regression analogue for the margin would entail forming a "band" or "tube" around the true regression function that contains most of the points. Points not contained within the tube would be described through slack variables. In formulating these ideas, we first need to define an appropriate loss function.

We define a loss function that ignores errors associated with points falling within a certain distance (e.g., $\epsilon > 0$) of the true linear regression function,

$$\mu(\mathbf{x}) = \beta_0 + \mathbf{x}^\tau \boldsymbol{\beta}. \tag{11.112}$$

In other words, if the point $(\mathbf{x}, y)$ is such that $|y - \mu(\mathbf{x})| \leq \epsilon$, then the loss is taken to be zero; if, on the other hand, $|y - \mu(\mathbf{x})| > \epsilon$, then we take the loss to be $|y - \mu(\mathbf{x})| - \epsilon$.

With this strategy in mind, we can define the following two types of loss function:

- $L_1^\epsilon(y, \mu(\mathbf{x})) = \max\{0, |y - \mu(\mathbf{x})| - \epsilon\}$,

- $L_2^\epsilon(y, \mu(\mathbf{x})) = \max\{0, (y - \mu(\mathbf{x}))^2 - \epsilon\}$.

The first loss function, L_1^ϵ, is called the *linear ϵ-insensitive loss function*, and the second, L_2^ϵ, is the *quadratic ϵ-insensitive loss function*. The two loss functions, linear (red curve) and quadratic (blue curve), are graphed in Figure 11.5. We see that the linear loss function ignores all errors falling within $\pm\epsilon$ of the true regression function $\mu(\mathbf{x})$ while dampening in a linear fashion errors that fall outside those limits.

11.5.2 Optimization for Linear ϵ-Insensitive Loss

We define slack variables ξ_i and ξ_j' in the following way. If the point $(\mathbf{x}_i, y_i)$ lies above the ϵ-tube, then $\xi_i' = y_i - \mu(\mathbf{x}_i) - \epsilon \geq 0$, whereas if the point $(\mathbf{x}_j, y_j)$ lies below the ϵ-tube, then $\xi_j = \mu(\mathbf{x}_j) - \epsilon - y_j \geq 0$. For points that fall outside the ϵ-tube, the values of the slack variables depend

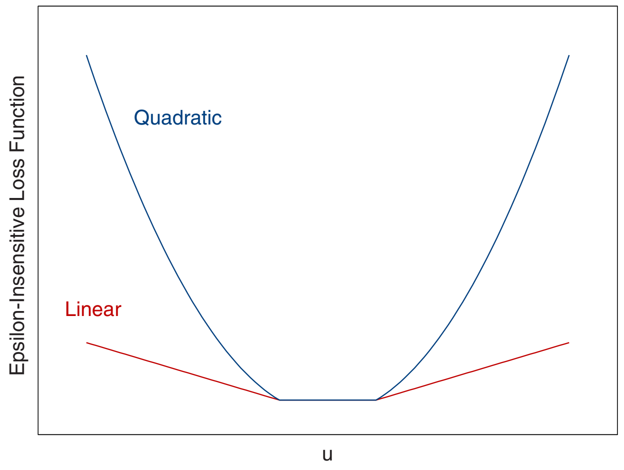

FIGURE 11.5. *The linear ϵ-insensitive loss function (red curve) and the quadratic ϵ-insensitive loss function (blue curve) for support vector regression. Plotted are $L_i(u) = \max\{0, |u|^i - \epsilon\}$ vs. u, $i = 1, 2$, where $u = y - \mu(\mathbf{x})$. For the linear loss function, the "flat" part of the curve has width 2ϵ.*

upon the shape of the loss function; for points inside the ϵ-tube, the slack variables have value zero.

For linear ϵ-insensitive loss, the primal optimization problem is to find β_0, $\boldsymbol{\beta}$, $\boldsymbol{\xi}' = (\xi'_1, \cdots, \xi'_n)^\tau$, and $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_n)^\tau$ to

$$\textit{minimize} \quad \frac{1}{2} \parallel \boldsymbol{\beta} \parallel^2 + C \sum_{i=1}^{n} (\xi_i + \xi'_i) \tag{11.113}$$

$$\begin{aligned} \textit{subject to} \qquad & y_i - (\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) \leq \epsilon + \xi'_i, \\ & (\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - y_i \leq \epsilon + \xi_i, \\ & \xi'_i \geq 0, \;\; \xi_i \geq 0, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{11.114}$$

The constant $C > 0$ exists to balance the flatness of the function μ against our tolerance of deviations larger than ϵ. Notice that because ϵ is found only in the constraints, the solution to this optimization problem has to incorporate a band around the regression function.

Form the primal Lagrangian,

$$\begin{aligned} F_P \;=\; & \frac{1}{2} \parallel \boldsymbol{\beta} \parallel^2 + C \sum_{i=1}^{n} (\xi_i + \xi'_i) - \sum_i a_i \{y_i - (\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - \epsilon - \xi'_i\} \\ & - \sum_i b_i \{(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) - y_i - \epsilon - \xi_i\} \\ & - \sum_i c_i \xi'_i - \sum_i d_i \xi_i, \end{aligned} \tag{11.115}$$

where a_i, b_i, c_i, and d_i, $i = 1, 2, \ldots, n$, are the Lagrange multipliers. This, in turn, implies that a_i, b_i, c_i, d_i, $i = 1, 2, \ldots, n$, are all nonnegative. The derivatives are

$$\frac{\partial F_P}{\partial \beta_0} = \sum_i a_i - \sum_i b_i \tag{11.116}$$

$$\frac{\partial F_P}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} + \sum_i a_i \mathbf{x}_i - \sum_i b_i \mathbf{x}_i \tag{11.117}$$

$$\frac{\partial F_P}{\partial \xi_i} = C + b_i - d_i \tag{11.118}$$

$$\frac{\partial F_P}{\partial \xi_i'} = C + a_i - c_i \tag{11.119}$$

Setting these derivatives equal to zero for a stationary solution yields:

$$\boldsymbol{\beta}^* = \sum_i (b_i - a_i)\mathbf{x}_i, \tag{11.120}$$

$$\sum_i (b_i - a_i) = 0, \tag{11.121}$$

$$C + b_i - d_i = 0, \quad C + a_i - c_i = 0, \quad i = 1, 2, \ldots, n. \tag{11.122}$$

The expression (11.120) is known as the *support vector expansion* because $\boldsymbol{\beta}^*$ can be written as a linear combination of the input vectors $\{\mathbf{x}_i\}$. Setting $\boldsymbol{\beta} = \boldsymbol{\beta}^*$ in the true regression equation (11.112) gives us

$$\mu^*(\mathbf{x}) = \beta_0 + \sum_{i=1}^{n} (b_i - a_i)(\mathbf{x}^\tau \mathbf{x}_i). \tag{11.123}$$

Substituting $\boldsymbol{\beta}^*$ into the primal Lagrangian and using (11.120) and (11.121) gives us the dual problem: find $\mathbf{a} = (a_1, \cdots, a_n)^\tau$, $\mathbf{b} = (b_1, \cdots, b_n)^\tau$ to

$$\begin{aligned} \textit{maximize} \quad F_D &= (\mathbf{b} - \mathbf{a})^\tau \mathbf{y} - \epsilon(\mathbf{b} + \mathbf{a})^\tau \mathbf{1}_n \\ &\quad - \frac{1}{2}(\mathbf{b} - \mathbf{a})^\tau \mathbf{K}(\mathbf{b} - \mathbf{a}) \end{aligned} \tag{11.124}$$

$$\textit{subject to} \quad \mathbf{0} \leq \mathbf{a}, \mathbf{b} \leq C\mathbf{1}_n, \quad (\mathbf{b} - \mathbf{a})^\tau \mathbf{1}_n = 0, \tag{11.125}$$

where $\mathbf{K} = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle)$ for linear SVM. The Karush–Kuhn–Tucker complementarity conditions state that the products of the dual variables and the constraints are all zero:

$$a_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta} - y_i - \epsilon - \xi_i) = 0, \qquad i = 1, 2, \ldots, n, \tag{11.126}$$

$$b_i(y_i - \beta_0 - \mathbf{x}_i^\tau \boldsymbol{\beta} - \epsilon - \xi_i') = 0, \qquad i = 1, 2, \ldots, n, \tag{11.127}$$

$$\xi_i \xi_i' = 0, \quad a_i b_i = 0, \qquad i = 1, 2, \ldots, n, \tag{11.128}$$

$$(a_i - C)\xi_i = 0, \quad (b_i - C)\xi_i' = 0, \qquad i = 1, 2, \ldots, n. \tag{11.129}$$

In practice, the value of ϵ is usually taken to be around 0.1.

The solution to this optimization problem produces a linear function of $\mathbf{x}$ accompanied by a band or tube of $\pm\epsilon$ around the function. Points that do not fall inside the tube are the *support vectors*.

11.5.3 Extensions

The optimization problem using quadratic ϵ-insensitive loss can be solved in a similar manner; see Exercise 11.3.

If we formulate this problem using nonlinear transformations of the input vectors, $\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$, to a feature space defined by the kernel $K(\mathbf{x}, \mathbf{y})$, then the stationary solution (11.120) is replaced by

$$\boldsymbol{\beta}^* = \sum_{i=1}^{n} (b_i - a_i)\mathbf{\Phi}(\mathbf{x}_i), \tag{11.130}$$

the inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^\tau \mathbf{x}_j$ in (11.120) is replaced by the more general kernel function,

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle = \mathbf{\Phi}(\mathbf{x}_i)^\tau \mathbf{\Phi}(\mathbf{x}_j), \tag{11.131}$$

the matrix $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))$ replaces the matrix $\mathbf{K}$ in (11.124), and the SVM regression function (11.122) becomes

$$\mu^*(\mathbf{x}) = \beta_0 + \sum_{i=1}^{n} (b_i - a_i) K(\mathbf{x}, \mathbf{x}_i); \tag{11.132}$$

see Exercise 11.4. Note that $\boldsymbol{\beta}^*$ in (11.130) does not have an explicit representation as it has in (11.120).

11.6 Optimization Algorithms for SVMs

When a data set is small, general-purpose linear programming (LP) or quadratic programming (QP) optimizers work quite well to solve SVM problems; QP optimizers can solve problems having about a thousand points, whereas LP optimizers can deal with hundreds of thousands of points. With large data sets, however, a more sophisticated approach is required.

The main problem when computing SVMs for very large data sets is that storing the entire kernel in main memory dramatically slows down computation. Alternative algorithms, constructed for the specific task of overcoming such computational inefficiencies, are now available in certain SVM software.

We give only brief descriptions of some of these algorithms. The simplest procedure for solving a convex optimization problem is that of gradient ascent:

Gradient Ascent: Start with an initial estimate of the $\boldsymbol{\alpha}$-coefficient vector and then successively update $\boldsymbol{\alpha}$ one α-coefficient at a time using the steepest ascent algorithm.

A problem with this approach is that the solution for $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^\tau$ has to satisfy the linear constraint $\boldsymbol{\alpha}^\tau \mathbf{y} = \sum_{i=1}^n \alpha_i y_i = 0$. Carrying out a non-trivial one-at-a-time update of each α-component (while holding the remaining αs constant at their current values) will violate this constraint, and the solution at each iteration will fall outside the feasible region. The minimum number of αs that can be changed at each iteration is two.

More complicated (but also more efficient) numerical techniques for large learning data sets are now available in many SVM software packages. Examples of such advanced techniques include "chunking," decomposition, and sequential minimal optimization. Each method builds upon certain common elements: (1) choose a subset of the learning set $\mathcal{L}$, (2) monitor closely the KKT optimality conditions to discover which points not in the subset violate the conditions, and (3) apply a suitable optimizing strategy. These strategies are

Chunking: Start with an arbitrary subset (called the "working set" or "chunk") of size 100–500 of the learning set $\mathcal{L}$; use a general LP or QP optimizer to train an SVM on that subset and keep only the support vectors; apply the resulting classifier to all the remaining data in $\mathcal{L}$ and sort the misclassified points by how badly they violate the KKT conditions; add to the support vectors found previously a predetermined number of those points that most violate the KKT conditions; iterate until all points satisfy the KKT conditions. The general optimizer and the point selection process make this algorithm slow and inefficient.

Decomposition: Similar to chunking, except that at each iteration, the size of the subset is always the same; adding new points to the subset means that an equal number of old points must be removed.

Sequential Minimal Optimization (SMO): An extreme version of the decomposition algorithm, whereby the subset consists of only two points at each iteration (see above comments related to the gradient ascent algorithm). These two αs are found at each iteration by using a heuristic argument and then updated so that the constraint $\boldsymbol{\alpha}^\tau \mathbf{y} = \sum_{i=1}^n \alpha_i y_i = 0$ is satisfied and the solution is found within the feasible region.

TABLE 11.4. *Some implementations of SVM.*

Package	Implementation
SVM^{light}	`http://svmlight.joachims.org/`
LIBSVM	`http://csie.ntu.edu.tw/~cjlin/libsvm/`
SVMTorch II	`http://www.idiap.ch/machine-learning.php`
SVMsequel	`http://www.isi.edu/~hdaume/SVMsequel/`
TinySVM	`http://chasen.org/~taku/TinySVM/`

A big advantage of SMO (Platt, 1999) is that the algorithm has an analytical solution and so does not need to refer to a general QP optimizer; it also does not need to store the entire kernel matrix in memory. Although more iterations are needed, SMO is much faster than the other algorithms. The SMO algorithm has been improved in many ways for use with massive data sets.

11.7 Software Packages

There are several software packages for computing SVMs. Many are available for downloading over the Internet. See Table 11.4 for a partial list. Most of these SVM packages use similar data-input formats and command lines.

The most popular SVM package is SVM^{light} by Thorsten Joachims; it is very fast and can carry out classification and regression using a variety of kernels and is used for text classification. It is often used as the basis for other SVM software packages.

The C++–based package LIBSVM by C.-C. Chang and C.-J. Lin, which carries out classification and regression, is based upon SMO and SVM^{light}, and has interfaces to MATLAB, python, perl, ruby, S-Plus (function `svm` in library `libsvm`), and R (function `svm` in library `e1071`); see Venables and Ripley (2002, pp. 344–346). SVMTorch II is an extremely fast C++ program for classification and regression that can handle more than 20,000 observations and more than 100 input variables. SVMsequel is a very fast program that handles classification problems, a variety of kernels (including string kernels), and enormous data sets. TinySVM, which supports C++, perl, ruby, python, and Java interfaces, is based upon SVM^{light}, carries out classification and regression, and can deal with very large data sets.

Bibliographical Notes

There are several excellent references on support vector machines. Our primary references include the books by Vapnik (1998, 2000), Cristianini and Shawe-Taylor (2000), Shawe-Taylor and Cristianini (2004, Chapter 7), Schölkopf and Smola (2002), and Hastie, Tibshirani, and Friedman (2001, Section 4.5 and Chapter 12) and the review articles by Burges (1998), Schölkopf and Smola (2003), and Moguerza and Munoz (2006). An excellent book on convex optimization is Boyd and Vandenberghe (2004).

Most of the theoretical work on kernel functions goes back to about the beginning of the 1900s. The idea of using kernel functions as inner products was introduced into machine learning by Aizerman, Braverman, and Rozoener (1964). Kernels were then put to work in SVM methodology by Boser, Guyon, and Vapnik (1992), who borrowed the "kernel" name from the theory of integral operators.

Our description of string kernels for text categorization is based upon Lodhi, Saunders, Shawe-Taylor, Cristianini, and Watkins (2002). See also Shawe-Taylor and Cristianini (2004, Chapter 11). For applications of SVM to text categorization, see the book by Joachims (2002) and Cristianini and Shawe-Taylor (2000, Section 8.1).

Exercises

11.1 (a) Show that the perpendicular distance of the point (h, k) to the line $f(x, y) = ax + by + c = 0$ is $\pm\ (ah + bk + c)/\sqrt{a^2 + b^2}$, where the sign chosen is that of c.

(b) Let $\mu(\mathbf{x}) = \beta_0 + \mathbf{x}^\tau \boldsymbol{\beta} = 0$ denote a hyperplane, where $\beta_0 \in \Re$ and $\boldsymbol{\beta} \in \Re^r$, and let $\mathbf{x}_k \in \Re^r$ be a point in the space. By minimizing $\| \mathbf{x} - \mathbf{x}_k \|^2$ subject to $\mu(\mathbf{x}) = 0$, show that the perpendicular distance from the point to the hyperplane is $|\mu(\mathbf{x}_k)| / \| \boldsymbol{\beta} \|$.

11.2 In the support vector regression problem using a quadratic ϵ-insensitive loss function, formulate and solve the resulting optimization problem.

11.3 The "2-norm soft margin" optimization problem for SVM classification: Consider the regularization problem of minimizing $\frac{1}{2} \| \boldsymbol{\beta} \|^2 + C \sum_{i=1}^{n} \xi_i^2$ subject to the constraints $y_i(\beta_0 + \mathbf{x}_i^\tau \boldsymbol{\beta}) \geq 1 - \xi_i$, and $\xi \geq 0$, for $i = 1, 2, \ldots, n$.

(a) Show that the same optimal solution to this problem is reached if we remove the constraints $\xi_i \geq 0$, $i = 1, 2, \ldots, n$, on the slack variables. (Hint: What is the effect on the objective functional if this constraint is violated?)

(b) Form the primal Lagrangian F_P, which will be a function of β_0, $\boldsymbol{\beta}$, $\boldsymbol{\xi}$, and the Lagrangian multipliers $\boldsymbol{\alpha}$. Differentiate F_P wrt β_0, $\boldsymbol{\beta}$, and $\boldsymbol{\xi}$, set the results equal to zero, and solve for a stationary solution.

(c) Substitute the results from (b) into the primal Lagrangian to obtain the dual objective functional F_D. Write out the dual problem (objective functional and constraints) in matrix notation. Maximize the dual wrt $\boldsymbol{\alpha}$. Use the Karush–Kuhn–Tucker complementary conditions $\alpha_i\{y_i(\beta_0+\mathbf{x}_i^\tau\boldsymbol{\beta})-(1-\xi_i)\}=0$ for $i=1,2,\ldots,n$.

(d) If $\boldsymbol{\alpha}^*$ is the solution to the dual problem, find $\widehat{\boldsymbol{\beta}}$ and its norm, which gives the width of the margin.

11.4 For the support vector regression problem in a feature space defined by a general kernel function K representing the inner product of pairs of nonlinearly transformed input vectors, formulate and solve the resulting optimization problem using (a) a linear ϵ-insensitive loss function and (b) a quadratic ϵ-insensitive loss function.

11.5 In the support vector regression problem, let $\epsilon=0$. Consider the quadratic (2-norm) primal optimization problem,

minimize $\lambda \parallel \boldsymbol{\beta} \parallel^2 + \sum_{i=1}^n \xi_i^2$

subject to $y_i - \mathbf{x}_i^\tau\boldsymbol{\beta} = \xi_i, \quad i=1,2,\ldots,n.$

Form the Lagrangian, differentiate wrt $\boldsymbol{\beta}$ and ξ_i, $i=1,2,\ldots,n$, and set the results equal to zero for a stationary solution. Substitute these values into the primal functional to get the dual problem. Use $\mathbf{K}$ to represent the Gram matrix with entries either $K_{ij}=\mathbf{x}_i^\tau\mathbf{x}_j$ or $K_{ij}=K(\mathbf{x}_i,\mathbf{x}_j)$. Differentiate the dual functional wrt the Lagrange multipliers $\boldsymbol{\alpha}$, and set the result equal to zero. Show that this solution is related to ridge regression (see Section 5.7.4).

11.6 Let $\mathbf{x},\mathbf{y}\in\Re^2$. Consider the polynomial kernel function, $K(\mathbf{x},\mathbf{y})=\langle\mathbf{x},\mathbf{y}\rangle^2$, so that $r=2$ and $d=2$. Find two different maps $\boldsymbol{\Phi}:\Re^2\rightarrow\mathcal{H}$ for $\mathcal{H}=\Re^3$.

11.7 Let $z\in\Re$ and define the $(2m+1)$-dimensional $\boldsymbol{\Phi}$-mapping,

$$\boldsymbol{\Phi}(z)=(2^{-1/2},\cos z,\cdots,\cos mz,\sin z,\cdots,\sin mz)^\tau.$$

Using this mapping, show that the kernel $K(x,y)=\langle\boldsymbol{\Phi}(x),\boldsymbol{\Phi}(y)\rangle$, $x,y\in\Re$, reduces to the *Dirichlet kernel* given by

$$K(x,y)=\frac{\sin((m+\frac{1}{2})\delta)}{2\sin(\delta/2)},$$

where $\delta=x-y$.

11.8 Show that the homogeneous polynomial kernel, $K(\mathbf{x},\mathbf{y})=\langle\mathbf{x},\mathbf{y}\rangle^d$, satisfies Mercer's condition (11.54).

11.9 If K_1 and K_2 are kernels and $c_1, c_2 \geq 0$ are real numbers, show that the following functions are kernels:

(*a*) $c_1 K_1(\mathbf{x}, \mathbf{y}) + c_2 K_2(\mathbf{x}, \mathbf{y})$;

(*b*) $K_1(\mathbf{x}, \mathbf{y}) K_2(\mathbf{x}, \mathbf{y})$;

(*c*) $\exp\{K_1(\mathbf{x}, \mathbf{y})\}$.

(Hint: In each case, you have to show that the function is nonnegative-definite.)

11.10 Prove that in finite-dimensional input space, a symmetric function $K(\mathbf{x}, \mathbf{y})$ is a kernel function iff $\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))$ is a nonnegative-definite matrix with nonnegative eigenvalues. (Hint: Use the symmetry and the spectral theorem for $\mathbf{K}$ to show that K is a kernel. Then, show that for a negative eigenvalue, the squared-norm of any point $\mathbf{z} \in \mathcal{H}$ is negative, which is impossible.)

11.11 Show that the functional $F_D(\boldsymbol{\alpha})$ in (11.40) is convex; i.e., show that, for $\theta \in (0, 1)$ and $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \Re^n$,

$$F_D(\theta\boldsymbol{\alpha} + (1-\theta)\boldsymbol{\beta}) \leq \theta F_D(\boldsymbol{\alpha}) + (1-\theta) F_D(\boldsymbol{\beta}).$$

11.12 Apply nonlinear-SVM to a binary classification data set of your choice. Make up a two-way table of values of (C, γ) and for each cell in that table compute the CV/10 misclassification rate. Find the pair (C, γ) with the smallest CV/10 misclassification rate. Compare this rate with results obtained using LDA and that using a classification tree.

12
Cluster Analysis

12.1 Introduction

Cluster analysis, which is the most well-known example of *unsupervised learning*, is a very popular tool for analyzing unstructured multivariate data. Within the data-mining community, cluster analysis is also known as *data segmentation*, and within the machine-learning community, it is also known as *class discovery*. The methodology consists of various algorithms each of which seeks to organize a given data set into homogeneous subgroups, or "clusters." There is no guarantee that more than one such group can be found; however, in any practical application, the underlying hypothesis is that the data form a heterogeneous set that should separate into natural groups familiar to the domain experts.

Clustering is a statistical tool for those who need to arrange large quantities of multivariate data into natural groups. For example, marketers use demographics and consumer profiles in an attempt to segment the marketplace into small, homogeneous groups so that promotional campaigns may be carried out more efficiently; biologists divide organisms into hierarchical orders in order to describe the notion of biological diversity; financial managers categorize corporations into different types based upon relevant financial characteristics; archaeologists group artifacts (e.g., broaches) found in

A.J. Izenman, *Modern Multivariate Statistical Techniques*,
doi: 10.1007/978-0-387-78189-1_12,
© Springer Science+Business Media, LLC 2008