CHAPTER 5

LINEAR DISCRIMINANT FUNCTIONS

5.1 INTRODUCTION

We assumed in Chapter 3 that the forms for the underlying *probability densities* were known, and that we will use the training samples to estimate the values of their parameters. In this chapter we shall instead assume we know the proper forms for the *discriminant functions*, and use the samples to estimate the values of parameters of the classifier. We shall examine various procedures for determining discriminant functions, some of which are statistical and some of which are not. None of them, however, requires knowledge of the forms of underlying probability distributions, and in this limited sense they can be said to be nonparametric.

Throughout this chapter we shall be concerned with discriminant functions that are either linear in the components of **x**, or linear in some given set of functions of **x**. Linear discriminant functions have a variety of pleasant analytical properties. As we have seen in Chapter 2, they can be optimal if the underlying distributions are cooperative, such as Gaussians having equal covariance, as might be obtained through an intelligent choice of feature detectors. Even when they are not optimal, we might be willing to sacrifice some performance in order to gain the advantage of their simplicity. Linear discriminant functions are relatively easy to compute and in the absence of information suggesting otherwise, linear classifiers are attractive candidates for initial, trial classifiers. They also illustrate a number of very important principles that will be used more fully in neural networks (Chapter 6).

The problem of finding a linear discriminant function will be formulated as a problem of minimizing a criterion function. The obvious criterion function for classification purposes is the *sample risk*, or *training error*—the average loss incurred in classifying the set of training samples. We must emphasize right away, however, that despite the attractiveness of this criterion, it is fraught with problems. While our goal will be to classify novel test patterns, a small training error does not guarantee a small test error—a fascinating and subtle problem that will command our attention in Chapter 9. As we shall see here, it is difficult to derive the minimum-risk linear discriminant anyway, and for that reason we investigate several related criterion functions that are analytically more tractable.

TRAINING ERROR

Much of our attention will be devoted to studying the convergence properties and computational complexities of various gradient descent procedures for minimizing criterion functions. The similarities between many of the procedures sometimes makes it difficult to keep the differences between them clear, and for this reason we have included a summary of the principal results in Table 5.1 at the end of Section 5.10.

5.2 LINEAR DISCRIMINANT FUNCTIONS AND DECISION SURFACES

A discriminant function that is a linear combination of the components of **x** can be written as

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0, \qquad (1)$$

THRESHOLD WEIGHT

where **w** is the *weight vector* and w_0 the *bias* or *threshold weight*. As we saw in Chapter 2, for the general case there will be c such discriminant functions, one for each of c categories. We shall return to this case, but first consider a simpler situation, that is, when there are only two categories.

5.2.1 The Two-Category Case

For a discriminant function of the form of Eq. 1, a two-category classifier implements the following decision rule: Decide ω_1 if $g(\mathbf{x}) > 0$ and ω_2 if $g(\mathbf{x}) < 0$. Thus, **x** is assigned to ω_1 if the inner product $\mathbf{w}^t\mathbf{x}$ exceeds the threshold $-w_0$ and to ω_2 otherwise. If $g(\mathbf{x}) = 0$, **x** can ordinarily be assigned to either class, but in this chapter we shall leave the assignment undefined. Figure 5.1 shows a typical implementation, a clear example of the general structure of a pattern recognition system we saw in Chapter 2.

The equation $g(\mathbf{x}) = 0$ defines the decision surface that separates points assigned to ω_1 from points assigned to ω_2. When g(**x**) is linear, this decision surface is a

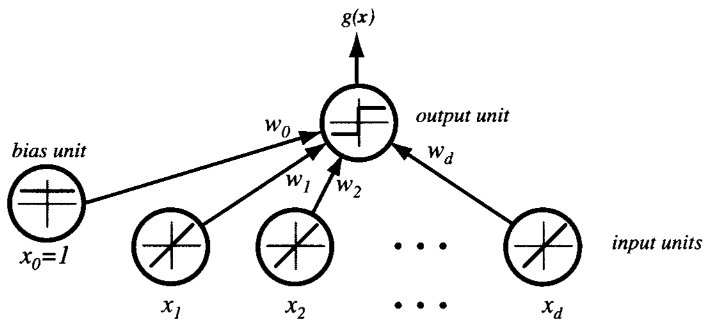


FIGURE 5.1. A simple linear classifier having d input units, each corresponding to the values of the components of an input vector. Each input feature value x_i is multiplied by its corresponding weight w_i; the effective input at the output unit is the sum all these products, $\sum w_i x_i$. We show in each unit its effective input-output function. Thus each of the d input units is linear, emitting exactly the value of its corresponding feature value. The single bias unit unit always emits the constant value 1.0. The single output unit emits a +1 if $\mathbf{w}^t\mathbf{x} + w_0 > 0$ or a −1 otherwise.

hyperplane. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are both on the decision surface, then

$$\mathbf{w}^t\mathbf{x}_1 + w_0 = \mathbf{w}^t\mathbf{x}_2 + w_0$$

or

$$\mathbf{w}^t(\mathbf{x}_1 - \mathbf{x}_2) = 0,$$

and this shows that $\mathbf{w}$ is normal to any vector lying in the hyperplane. In general, the hyperplane H divides the feature space into two half-spaces: decision region $\mathcal{R}_1$ for ω_1 and region $\mathcal{R}_2$ for ω_2. Because $g(\mathbf{x}) > 0$ if $\mathbf{x}$ is in $\mathcal{R}_1$, it follows that the normal vector $\mathbf{w}$ points into $\mathcal{R}_1$. It is sometimes said that any $\mathbf{x}$ in $\mathcal{R}_1$ is on the *positive* side of H, and any $\mathbf{x}$ in $\mathcal{R}_2$ is on the *negative* side.

The discriminant function g($\mathbf{x}$) gives an algebraic measure of the distance from $\mathbf{x}$ to the hyperplane. Perhaps the easiest way to see this is to express $\mathbf{x}$ as

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

where $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto H, and r is the desired algebraic distance—positive if $\mathbf{x}$ is on the positive side and negative if $\mathbf{x}$ is on the negative side. Then, because $g(\mathbf{x}_p) = 0$,

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = r\|\mathbf{w}\|,$$

or

$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}.$$

In particular, the distance from the origin to H is given by $w_0/\|\mathbf{w}\|$. If $w_0 > 0$, the origin is on the positive side of H, and if $w_0 < 0$, it is on the negative side. If $w_0 = 0$, then $g(\mathbf{x})$ has the homogeneous form $\mathbf{w}^t\mathbf{x}$, and the hyperplane passes through the origin. A geometric illustration of these algebraic results is given in Fig. 5.2.

To summarize, a linear discriminant function divides the feature space by a hyperplane decision surface. The orientation of the surface is determined by the normal

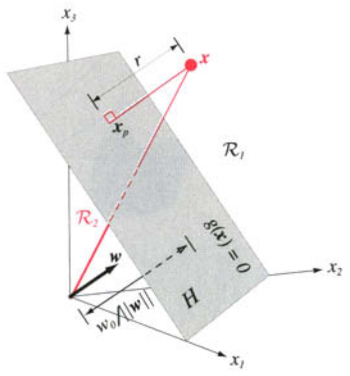


FIGURE 5.2. The linear decision boundary H, where $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = 0$, separates the feature space into two half-spaces $\mathcal{R}_1$ (where $g(\mathbf{x}) > 0$) and $\mathcal{R}_2$ (where $g(\mathbf{x}) < 0$).

vector **w**, and the location of the surface is determined by the bias w_0. The discriminant function $g(\mathbf{x})$ is proportional to the signed distance from **x** to the hyperplane, with $g(\mathbf{x}) > 0$ when **x** is on the positive side, and $g(\mathbf{x}) < 0$ when **x** is on the negative side.

5.2.2 The Multicategory Case

There is more than one way to devise multicategory classifiers employing linear discriminant functions. For example, we might reduce the problem to c two-class problems, where the ith problem is solved by a linear discriminant function that separates points assigned to ω_i from those not assigned to ω_i. A more extravagant approach would be to use $c(c-1)/2$ linear discriminants, one for every pair of classes. As illustrated in Fig. 5.3, both of these approaches can lead to regions in which the classification is undefined. We shall avoid this problem by adopting the approach taken in Chapter 2, defining c linear discriminant functions

$$g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x} + w_{i0} \qquad i = 1, \ldots, c, \tag{2}$$

LINEAR MACHINE

and assigning **x** to ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$; in case of ties, the classification is left undefined. The resulting classifier is called a *linear machine*. A linear machine

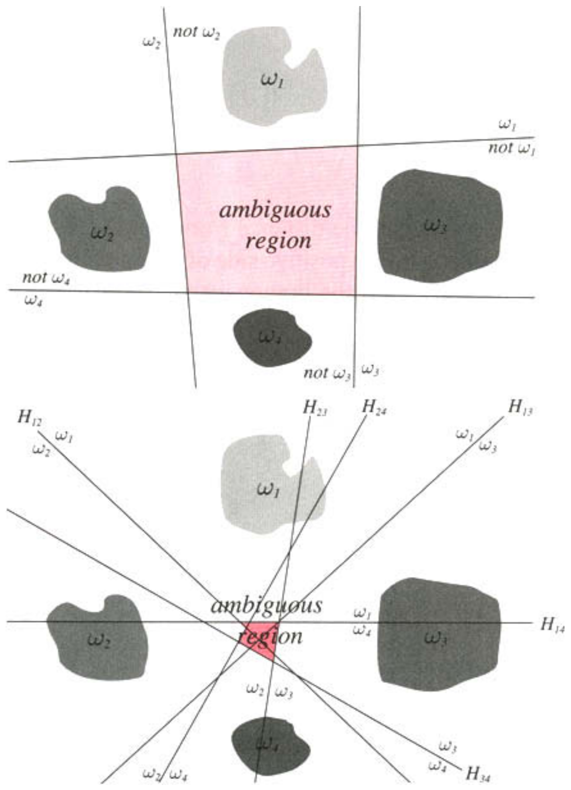


FIGURE 5.3. Linear decision boundaries for a four-class problem. The top figure shows ω_i/not ω_i dichotomies while the bottom figure shows ω_i/ω_j dichotomies and the corresponding decision boundaries H_{ij}. The pink regions have ambiguous category assignments.

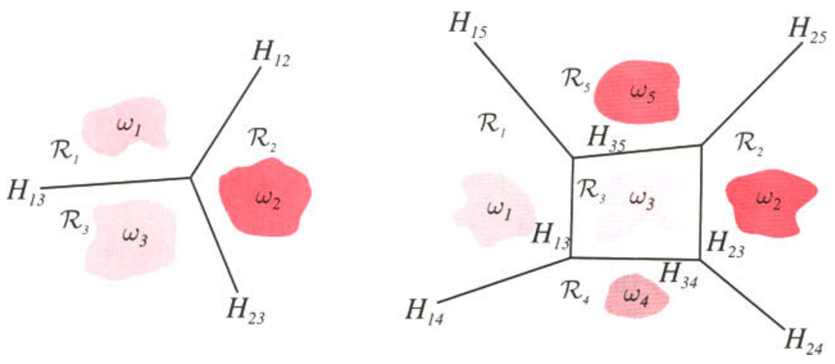


FIGURE 5.4. Decision boundaries produced by a linear machine for a three-class problem and a five-class problem.

divides the feature space into c decision regions, with $g_i(\mathbf{x})$ being the largest discriminant if $\mathbf{x}$ is in region $\mathcal{R}_i$. If $\mathcal{R}_i$ and $\mathcal{R}_j$ are contiguous, the boundary between them is a portion of the hyperplane H_{ij} defined by

$$g_i(\mathbf{x}) = g_j(\mathbf{x})$$

or

$$(\mathbf{w}_i - \mathbf{w}_j)^t\mathbf{x} + (w_{i0} - w_{j0}) = 0.$$

It follows at once that $\mathbf{w}_i - \mathbf{w}_j$ is normal to H_{ij}, and the signed distance from $\mathbf{x}$ to H_{ij} is given by $(g_i(\mathbf{x}) - g_j(\mathbf{x}))/\|\mathbf{w}_i - \mathbf{w}_j\|$. Thus, with the linear machine it is not the weight vectors themselves but their *differences* that are important. While there are $c(c-1)/2$ pairs of regions, they need not all be contiguous, and the total number of hyperplane segments appearing in the decision surfaces is often fewer than $c(c-1)/2$, as shown in Fig. 5.4.

It is easy to show that the decision regions for a linear machine are convex, and this restriction surely limits the flexibility and accuracy of the classifier (Problems 2 and 3). In particular, every decision region is singly connected, and this tends to make the linear machine most suitable for problems for which the conditional densities $p(\mathbf{x}|\omega_i)$ are unimodal. Nevertheless, we must be careful: There are multimodal distributions for which linear discriminants give excellent results and unimodal distributions for which they give poor classification results (Problem 1).

5.3 GENERALIZED LINEAR DISCRIMINANT FUNCTIONS

The linear discriminant function $g(\mathbf{x})$ can be written as

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i, \tag{3}$$

where the coefficients w_i are the components of the weight vector **w**. By adding additional terms involving the products of pairs of components of **x**, we obtain the *quadratic discriminant function*

QUADRATIC DISCRIMINANT

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j. \tag{4}$$

Because $x_i x_j = x_j x_i$, we can assume that $w_{ij} = w_{ji}$ with no loss in generality. Thus, the quadratic discriminant function has an additional $d(d+1)/2$ coefficients at its disposal with which to produce more complicated separating surfaces. The separating surface defined by $g(\mathbf{x}) = 0$ is a second-degree or *hyperquadric* surface. If the symmetric matrix $\mathbf{W} = [w_{ij}]$ is nonsingular the linear terms in g(**x**) can be eliminated by translating the axes. The basic character of the separating surface can be described in terms of the scaled matrix $\overline{\mathbf{W}} = \mathbf{W}/(\mathbf{w}^t\mathbf{W}^{-1}\mathbf{w} - 4w_0)$. If $\overline{\mathbf{W}}$ is a positive multiple of the identity matrix, the separating surface is a *hypersphere*. If $\overline{\mathbf{W}}$ is positive definite, the separating surfaces is a *hyperellipsoid*. If some of the eigenvalues of $\overline{\mathbf{W}}$ are positive and others are negative, the surface is one of the variety of types of *hyperhyperboloids* (Problem 12). As we observed in Chapter 2, these are the kinds of separating surfaces that arise in the general multivariate Gaussian case.

POLYNOMIAL DISCRIMINANT

By continuing to add terms such as $w_{ijk}x_i x_j x_k$, we can obtain the class of *polynomial discriminant functions*. These can be thought of as truncated series expansions of some arbitrary g(**x**), and this in turn suggests the *generalized linear discriminant function*

$$g(\mathbf{x}) = \sum_{i=1}^{\hat{d}} a_i y_i(\mathbf{x}) \tag{5}$$

or

$$g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}, \tag{6}$$

PHI FUNCTION

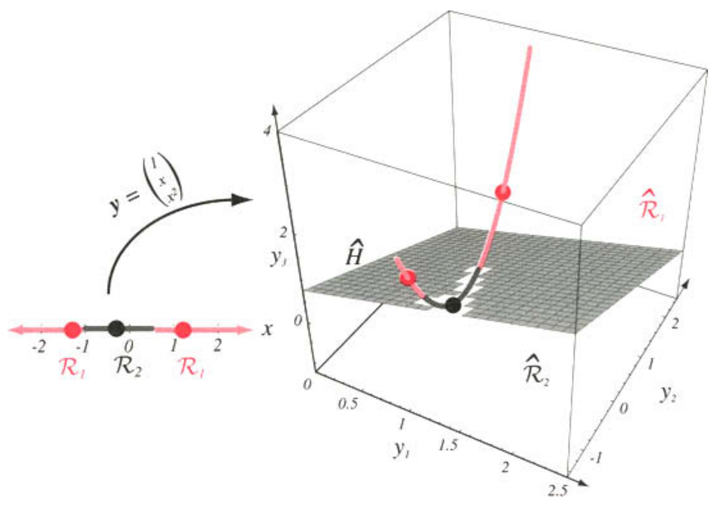
where **a** is now a $\hat{d}$-dimensional weight vector and where the $\hat{d}$ functions $y_i(\mathbf{x})$—sometimes called φ functions—can be arbitrary functions of **x**. Such functions might be computed by a feature detecting subsystem. By selecting these functions judiciously and letting $\hat{d}$ be sufficiently large, one can approximate any desired discriminant function by such an expansion. The resulting discriminant function is not linear in **x**, but it *is* linear in **y**. The $\hat{d}$ functions $y_i(\mathbf{x})$ merely map points in d-dimensional **x**-space to points in $\hat{d}$-dimensional **y**-space. The homogeneous discriminant $\mathbf{a}^t\mathbf{y}$ separates points in this transformed space by a hyperplane passing through the origin. Thus, the mapping from **x** to **y** reduces the problem to one of finding a homogeneous linear discriminant function.

Some of the advantages and disadvantages of this approach can be clarified by considering a simple example. Let the quadratic discriminant function be

$$g(x) = a_1 + a_2 x + a_3 x^2, \tag{7}$$

so that the three-dimensional vector **y** is given by

$$\mathbf{y} = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix}. \tag{8}$$



$\mathbf{y} = \begin{pmatrix}1\\x\\x^2\end{pmatrix}$
x
-2 -1 0 1 2
$\mathcal{R}_1$ $\mathcal{R}_2$ $\mathcal{R}_1$
$\hat{H}$
$\hat{\mathcal{R}}_1$
$\hat{\mathcal{R}}_2$
y_3
y_2
y_1
4
2
0
0.5
1
1.5
2
2.5
-1

FIGURE 5.5. The mapping $\mathbf{y} = (1, x, x^2)^t$ takes a line and transforms it to a parabola in three dimensions. A plane splits the resulting $\mathbf{y}$-space into regions corresponding to two categories, and this in turn gives a nonsimply connected decision region in the one-dimensional x-space.

The mapping from x to $\mathbf{y}$ is illustrated in Fig. 5.5. The data remain inherently one-dimensional, because varying x causes $\mathbf{y}$ to trace out a curve in three dimensions. Thus, one thing to notice immediately is that if x is governed by a probability law $p(x)$, the induced density $\hat{p}(\mathbf{y})$ will be degenerate, being zero everywhere except on the curve, where it is infinite. This is a common problem whenever $\hat{d} > d$, and the mapping takes points from a lower-dimensional space to a higher-dimensional space.

The plane $\hat{H}$ defined by $\mathbf{a}^t\mathbf{y} = 0$ divides the $\mathbf{y}$-space into two decision regions $\hat{\mathcal{R}}_1$ and $\hat{\mathcal{R}}_2$. Figure 5.5 shows the separating plane corresponding to $\mathbf{a} = (-1, 1, 2)^t$, the decision regions $\hat{\mathcal{R}}_1$ and $\hat{\mathcal{R}}_2$, and their corresponding decision regions $\mathcal{R}_1$ and $\mathcal{R}_2$ in the original x-space. The quadratic discriminant function $g(x) = -1 + x + 2x^2$ is positive if $x < -1$ or if $x > 0.5$, and hence $\mathcal{R}_1$ is multiply connected. Thus although the decision regions in $\mathbf{y}$-space are convex, this is by no means the case in x-space. More generally speaking, even with relatively simple functions $y_i(\mathbf{x})$, decision surfaces induced in an $\mathbf{x}$-space can be fairly complex.

Unfortunately, the curse of dimensionality often makes it hard to capitalize on this flexibility in practice. A complete quadratic discriminant function involves $\hat{d} = (d + 1)(d + 2)/2$ terms. If d is modestly large, say $d = 50$, this requires the computation of a great many terms; inclusion of cubic and in general kth-order components in the polynomial leads to $O(d^k)$ terms. Furthermore, the $\hat{d}$ components of the weight vector $\mathbf{a}$ must be determined from training samples. If we think of $\hat{d}$ as specifying the number of degrees of freedom for the discriminant function, it is natural to require that the number of samples be not less than the number of degrees of freedom (cf. Chapter 9). Clearly, a general series expansion of $g(\mathbf{x})$ can easily lead to completely unrealistic requirements for computation and data. We shall see in Section 5.11 that this drawback can be accommodated by imposing a constraint of large margins, or bands between the training patterns, however. In this case, we are not technically speaking fitting all the free parameters; instead, we are relying on the assumption that the mapping to a high-dimensional space does not impose any spurious structure or relationships among the training points. Alternatively,

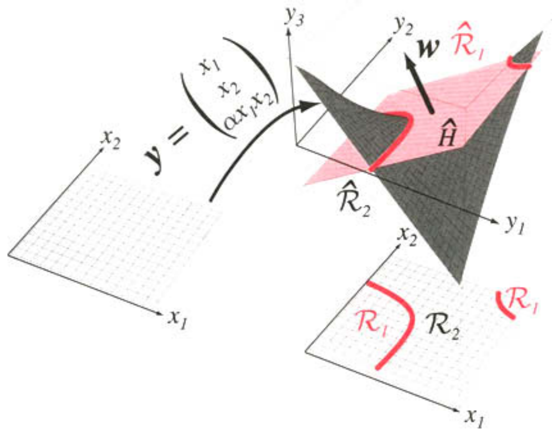


FIGURE 5.6. The two-dimensional input space **x** is mapped through a polynomial function f to **y**. Here the mapping is $y_1 = x_1$, $y_2 = x_2$ and $y_3 \propto x_1 x_2$. A linear discriminant in this transformed space is a hyperplane, which cuts the surface. Points to the positive side of the hyperplane $\hat{H}$ correspond to category ω_1, and those beneath it correspond to category ω_2. Here, in terms of the **x** space, $\mathcal{R}_1$ is a not simply connected.

multilayer neural networks approach this problem by employing multiple copies of a single nonlinear function of the input features, as we shall see in Chapter 6.

While it may be hard to realize the potential benefits of a generalized linear discriminant function, we can at least exploit the convenience of being able to write $g(\mathbf{x})$ in the homogeneous form $\mathbf{a}^t\mathbf{y}$. In the particular case of the linear discriminant function we have

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i = \sum_{i=0}^{d} w_i x_i \tag{9}$$

where we set $x_0 = 1$. Thus we can write

$$\mathbf{y} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} 1 \\ \\ \mathbf{x} \\ \\ \end{bmatrix}, \tag{10}$$

AUGMENTED VECTOR

and **y** is sometimes called an *augmented feature vector*. Likewise, an *augmented weight vector* can be written as:

$$\mathbf{a} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} w_0 \\ \\ \mathbf{w} \\ \\ \end{bmatrix}. \tag{11}$$

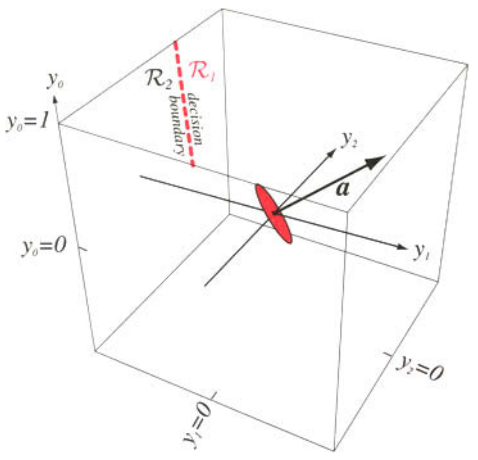


FIGURE 5.7. A three-dimensional augmented feature space **y** and augmented weight vector **a** (at the origin). The set of points for which $\mathbf{a}^t\mathbf{y} = 0$ is a plane (or more generally, a hyperplane) perpendicular to **a** and passing through the origin of **y**-space, as indicated by the red disk. Such a plane need not pass through the origin of the two-dimensional feature space of the problem, as illustrated by the dashed decision boundary shown at the top of the box. Thus there exists an augmented weight vector **a** that will lead to any straight decision line in **x**-space.

This mapping from d-dimensional **x**-space to $(d + 1)$-dimensional **y**-space is mathematically trivial but nonetheless quite convenient. The addition of a constant component to **x** preserves all distance relationships among samples. The resulting **y** vectors all lie in a d-dimensional subspace, which is the **x**-space itself. The hyperplane decision surface $\hat{H}$ defined by $\mathbf{a}^t\mathbf{y} = 0$ passes through the origin in **y**-space, even though the corresponding hyperplane H can be in any position in **x**-space. The distance from **y** to $\hat{H}$ is given by $|\mathbf{a}^t\mathbf{y}|/\|\mathbf{a}\|$, or $|g(\mathbf{x})|/\|\mathbf{a}\|$. Because $\|\mathbf{a}\| \geq \|\mathbf{w}\|$, this distance is less than, or at most equal to, the distance from **x** to H. By using this mapping we reduce the problem of finding a weight vector **w** and a threshold weight w_0 to the problem of finding a single weight vector **a** (Fig. 5.7).

5.4 THE TWO-CATEGORY LINEARLY SEPARABLE CASE

Suppose now that we have a set of n samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$, some labeled ω_1 and some labeled ω_2. We want to use these samples to determine the weights **a** in a linear discriminant function $g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$. Suppose we have reason to believe that there exists a solution for which the probability of error is very low. Then a reasonable approach is to look for a weight vector that classifies all of the samples correctly. If such a weight vector exists, the samples are said to be *linearly separable*.

LINEARLY SEPARABLE

A sample $\mathbf{y}_i$ is classified correctly if $\mathbf{a}^t\mathbf{y}_i > 0$ and $\mathbf{y}_i$ is labeled ω_1 or if $\mathbf{a}^t\mathbf{y}_i < 0$ and $\mathbf{y}_i$ is labeled ω_2. This suggests a "normalization" that simplifies the treatment of the two-category case, namely, the replacement of all samples labeled ω_2 by their negatives. With this "normalization" we can forget the labels and look for a weight vector **a** such that $\mathbf{a}^t\mathbf{y}_i > 0$ for *all* of the samples. Such a weight vector is called a *separating vector* or more generally a *solution vector*.

SEPARATING VECTOR

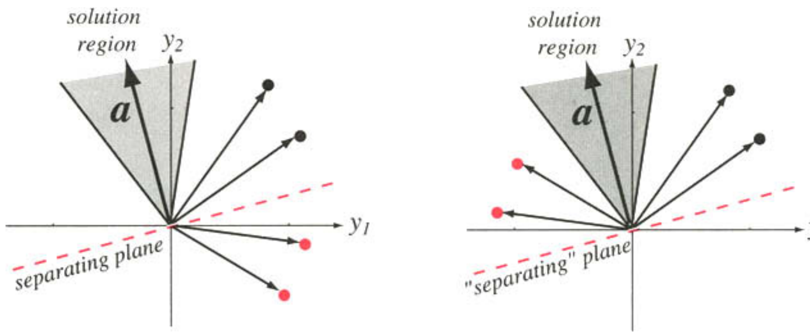


FIGURE 5.8. Four training samples (black for ω_1, red for ω_2) and the solution region in feature space. The figure on the left shows the raw data; the solution vectors leads to a plane that separates the patterns from the two categories. In the figure on the right, the red points have been "normalized"—that is, changed in sign. Now the solution vector leads to a plane that places all "normalized" points on the same side.

5.4.1 Geometry and Terminology

WEIGHT SPACE

The weight vector $\mathbf{a}$ can be thought of as specifying a point in *weight space*. Each sample $\mathbf{y}_i$ places a constraint on the possible location of a solution vector. The equation $\mathbf{a}^t\mathbf{y}_i = 0$ defines a hyperplane through the origin of weight space having $\mathbf{y}_i$ as a normal vector. The solution vector—if it exists—must be on the positive side of every hyperplane. Thus, a solution vector must lie in the intersection of n half-spaces; indeed any vector in this region is a solution vector. The corresponding region is called the *solution region*, and it should not be confused with the decision region in feature space corresponding to any particular category. A two-dimensional example illustrating the solution region for both the normalized and the unnormalized case is shown in Fig. 5.8.

SOLUTION REGION

From this discussion, it should be clear that the solution vector—again, if it exists—is not unique. There are several ways to impose additional requirements to constrain the solution vector. One possibility is to seek a unit-length weight vector that maximizes the minimum distance from the samples to the separating plane. Another possibility is to seek the minimum-length weight vector satisfying $\mathbf{a}^t\mathbf{y}_i \geq b$ for all i, where b is a positive constant called the *margin*. As shown in Fig. 5.9, the solution region resulting form the intersections of the halfspaces for which $\mathbf{a}^t\mathbf{y}_i \geq b > 0$ lies within the previous solution region, being insulated from the old boundaries by the distance $b/\|\mathbf{y}_i\|$.

MARGIN

The motivation behind these attempts to find a solution vector closer to the "middle" of the solution region is the natural belief that the resulting solution is more likely to classify new test samples correctly. In most of the cases we shall treat, however, we shall be satisfied with any solution strictly within the solution region. Our chief concern will be to see that any iterative procedure used does not converge to a limit point on the boundary. This problem can always be avoided by the introduction of a margin, that is, by requiring that $\mathbf{a}^t\mathbf{y}_i \geq b > 0$ for all i.

5.4.2 Gradient Descent Procedures

The approach we shall take to finding a solution to the set of linear inequalities $\mathbf{a}^t\mathbf{y}_i > 0$ will be to define a criterion function $J(\mathbf{a})$ that is minimized if $\mathbf{a}$ is a solution vector. This reduces our problem to one of minimizing a scalar function—a

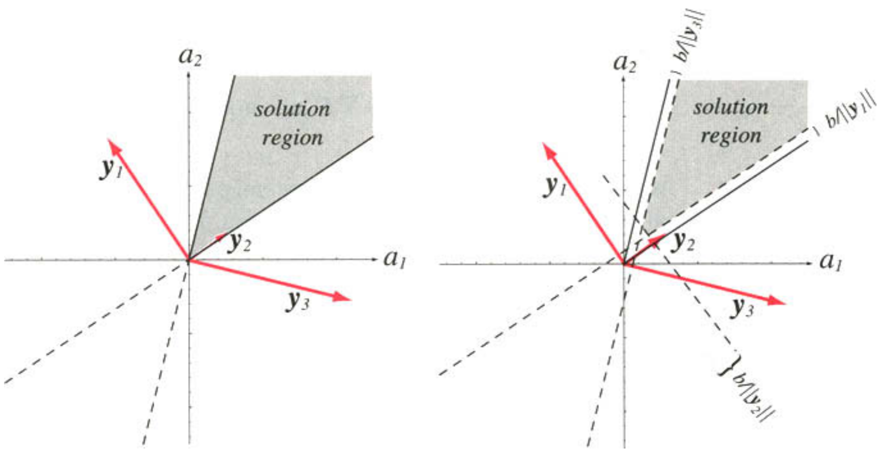


FIGURE 5.9. The effect of the margin on the solution region. At the left is the case of no margin ($b = 0$) equivalent to a case such as shown at the left in Fig. 5.8. At the right is the case $b > 0$, shrinking the solution region by margins $b/\|\mathbf{y}_i\|$.

problem that can often be solved by a gradient descent procedure. Basic gradient descent is very simple. We start with some arbitrarily chosen weight vector $\mathbf{a}(1)$ and compute the gradient vector $\nabla J(\mathbf{a}(1))$. The next value $\mathbf{a}(2)$ is obtained by moving some distance from $\mathbf{a}(1)$ in the direction of steepest descent, i.e., along the negative of the gradient. In general, $\mathbf{a}(k+1)$ is obtained from $\mathbf{a}(k)$ by the equation

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \eta(k)\nabla J(\mathbf{a}(k)), \tag{12}$$

LEARNING RATE

where η is a positive scale factor or *learning rate* that sets the step size. We hope that such a sequence of weight vectors will converge to a solution minimizing $J(\mathbf{a})$. In algorithmic form we have:

■ **Algorithm 1. (Basic Gradient Descent)**

```
1 begin initialize $\mathbf{a}$, threshold $\theta$, $\eta(\cdot)$, $k \leftarrow 0$
2         do $k \leftarrow k + 1$
3            $\mathbf{a} \leftarrow \mathbf{a} - \eta(k)\nabla J(\mathbf{a})$
4         until $|\eta(k)\nabla J(\mathbf{a})| < \theta$
5   return $\mathbf{a}$
6 end
```

The many problems associated with gradient descent procedures are well known. Fortunately, we shall be constructing the functions we want to minimize, and shall be able to avoid the most serious of these problems. One that will confront us repeatedly, however, is the choice of the learning rate $\eta(k)$. If $\eta(k)$ is too small, convergence is needlessly slow, whereas if $\eta(k)$ is too large, the correction process will overshoot and can even diverge (Section 5.6.1).

We now consider a principled method for setting the learning rate. Suppose that the criterion function can be well-approximated by the second-order expansion around a value $\mathbf{a}(k)$ as

$$J(\mathbf{a}) \simeq J(\mathbf{a}(k)) + \nabla J^t(\mathbf{a} - \mathbf{a}(k)) + \frac{1}{2}(\mathbf{a} - \mathbf{a}(k))^t\mathbf{H}\,(\mathbf{a} - \mathbf{a}(k)), \tag{13}$$

HESSIAN MATRIX

where $\mathbf{H}$ is the *Hessian matrix* of second partial derivatives $\partial^2 J/\partial a_i \partial a_j$ evaluated at $\mathbf{a}(k)$. Then, substituting $\mathbf{a}(k+1)$ from Eq. 12 into Eq. 13 we find

$$J(\mathbf{a}(k+1)) \simeq J(\mathbf{a}(k)) - \eta(k)\|\nabla J\|^2 + \frac{1}{2}\eta^2(k)\nabla J^t\mathbf{H}\nabla J.$$

From this it follows (Problem 13) that $J(\mathbf{a}(k+1))$ can be minimized by the choice

$$\eta(k) = \frac{\|\nabla J\|^2}{\nabla J^t\mathbf{H}\nabla J}, \tag{14}$$

where $\mathbf{H}$ depends on $\mathbf{a}$, and thus indirectly on k. This then is the optimal choice of $\eta(k)$ given the assumptions mentioned. Note that if the criterion function $J(\mathbf{a})$ is quadratic throughout the region of interest, then $\mathbf{H}$ is constant and η is a constant independent of k.

NEWTON'S ALGORITHM

An alternative approach, obtained by ignoring Eq. 12 and choosing $\mathbf{a}(k+1)$ to minimize the second-order expansion, is *Newton's algorithm* where line 3 in Algorithm 1 is replaced by

$$\mathbf{a}(k+1) = \mathbf{a}(k) - \mathbf{H}^{-1}\nabla J, \tag{15}$$

leading to the following algorithm:

■ **Algorithm 2. (Newton Descent)**

```
1 begin initialize  a, threshold $\theta$
2         do
3             $\mathbf{a} \leftarrow \mathbf{a} - \mathbf{H}^{-1}\nabla J(\mathbf{a})$
4         until $|\mathbf{H}^{-1}\nabla J(\mathbf{a})| < \theta$
5     return a
6 end
```

(We should point out that Newton's algorithm works for the quadratic error functions we have been considering, but not in the nonquadratic error functions of multilayer neural networks we shall meet in Chapter 6.) Simple gradient descent and Newton's algorithm are compared in Fig. 5.10.

Generally speaking, Newton's algorithm will usually give a greater improvement *per step* than the simple gradient descent algorithm, even with the optimal value of $\eta(k)$. However, Newton's algorithm is not applicable if the Hessian matrix $\mathbf{H}$ is singular. Furthermore, even when $\mathbf{H}$ is nonsingular, the $O(d^3)$ time required for matrix inversion on each iteration can easily offset the descent advantage. In fact, it often takes less time to set $\eta(k)$ to a constant η that is smaller than necessary and make a few more corrections than it is to compute the optimal $\eta(k)$ at each step (Computer exercise 1).

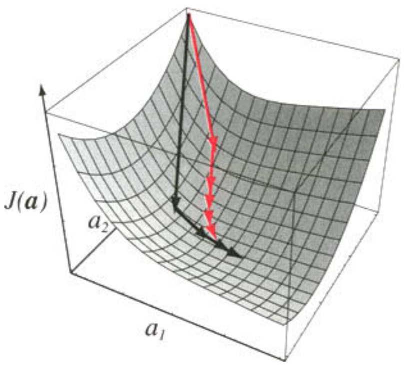


FIGURE 5.10. The sequence of weight vectors given by a simple gradient descent method (red) and by Newton's (second order) algorithm (black). Newton's method typically leads to greater improvement per step, even when using optimal learning rates for both methods. However the added computational burden of inverting the Hessian matrix used in Newton's method is not always justified, and simple gradient descent may suffice.

5.5 MINIMIZING THE PERCEPTRON CRITERION FUNCTION

5.5.1 The Perceptron Criterion Function

Consider now the problem of constructing a criterion function for solving the linear inequalities $\mathbf{a}^t\mathbf{y}_i > 0$. The most obvious choice is to let $J(\mathbf{a};\ \mathbf{y}_1, \ldots, \mathbf{y}_n)$ be the number of samples misclassified by $\mathbf{a}$. However, because this function is piecewise constant, it is obviously a poor candidate for a gradient search. A better choice is the *Perceptron criterion function*

$$J_p(\mathbf{a}) = \sum_{\mathbf{y}\in\mathcal{Y}} (-\mathbf{a}^t\mathbf{y}), \tag{16}$$

where $\mathcal{Y}(\mathbf{a})$ is the set of samples *misclassified* by $\mathbf{a}$. (If no samples are misclassified, $\mathcal{Y}$ is empty and we define J_p to be zero.) Because $\mathbf{a}^t\mathbf{y} \leq 0$ if $\mathbf{y}$ is misclassified, $J_p(\mathbf{a})$ is never negative, being zero only if $\mathbf{a}$ is a solution vector or if $\mathbf{a}$ is on the decision boundary. Geometrically, $J_p(\mathbf{a})$ is proportional to the sum of the distances from the misclassified samples to the decision boundary. Figure 5.11 illustrates J_p for a simple two-dimensional example.

Because the jth component of the gradient of J_p is $\partial J_p/\partial a_j$, we see from Eq. 16 that

$$\nabla J_p = \sum_{\mathbf{y}\in\mathcal{Y}} (-\mathbf{y}), \tag{17}$$

and hence the update rule becomes

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}, \tag{18}$$

where $\mathcal{Y}_k$ is the set of samples misclassified by $\mathbf{a}(k)$. Thus the Perceptron algorithm is as follows:

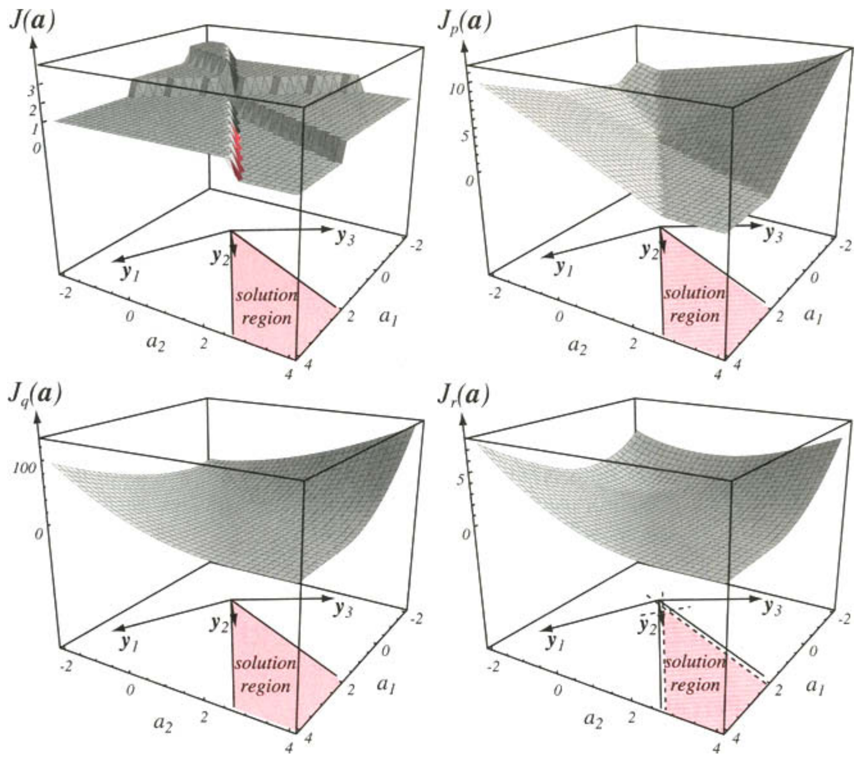


FIGURE 5.11. Four learning criteria as a function of weights in a linear classifier. At the upper left is the total number of patterns misclassified, which is piecewise constant and hence unacceptable for gradient descent procedures. At the upper right is the Perceptron criterion (Eq. 16), which is piecewise linear and acceptable for gradient descent. The lower left is squared error (Eq. 32), which has nice analytic properties and is useful even when the patterns are not linearly separable. The lower right is the square error with margin (Eq. 33). A designer may adjust the margin b in order to force the solution vector to lie toward the middle of the $b = 0$ solution region in hopes of improving generalization of the resulting classifier.

■ **Algorithm 3. (Batch Perceptron)**

```
1 begin initialize  a, $\eta(\cdot)$, criterion $\theta$, $k \leftarrow 0$
2         do $k \leftarrow k + 1$
3              $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)\sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}$
4         until $|\eta(k)\sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}| < \theta$
5     return a
6 end
```

Thus, the batch Perceptron algorithm for finding a solution vector can be stated very simply: The next weight vector is obtained by adding some multiple of the sum of the misclassified samples to the present weight vector. We use the term "batch" to refer to the fact that (in general) a large group of samples is used when computing

BATCH TRAINING

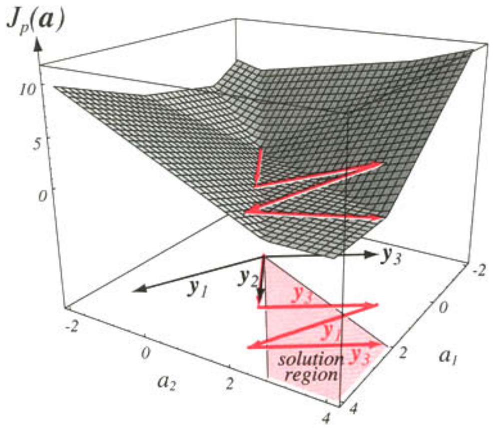


FIGURE 5.12. The Perceptron criterion, $J_p(\mathbf{a})$, is plotted as a function of the weights a_1 and a_2 for a three-pattern problem. The weight vector begins at $\mathbf{0}$, and the algorithm sequentially adds to it vectors equal to the "normalized" misclassified patterns themselves. In the example shown, this sequence is $\mathbf{y}_1 + \mathbf{y}_2 + \mathbf{y}_3$, $\mathbf{y}_2$, $\mathbf{y}_3$, $\mathbf{y}_1$, $\mathbf{y}_3$, at which time the vector lies in the solution region and iteration terminates. Note that the second update (by $\mathbf{y}_3$) takes the candidate vector *farther* from the solution region than after the first update (cf. Theorem 5.1).

each weight update. (We shall soon see alternative methods based on single samples.) Figure 5.12 shows how this algorithm yields a solution vector for a simple two-dimensional example with $\mathbf{a}(1) = \mathbf{0}$, and $\eta(k) = 1$. We shall now show that it will yield a solution for any linearly separable problem.

5.5.2 Convergence Proof for Single-Sample Correction

We begin our examination of convergence properties of the Perceptron algorithm with a variant that is easier to analyze. Rather than testing $\mathbf{a}(k)$ on all of the samples and basing our correction of the set $\mathcal{Y}_k$ of misclassified training samples, we shall consider the samples in a sequence and shall modify the weight vector whenever it misclassifies a *single* sample. For the purposes of the convergence proof, the detailed nature of the sequence is unimportant as long as every sample appears in the sequence infinitely often. The simplest way to assure this is to repeat the samples cyclically, though from a practical point of view random selection is often to be preferred (Section 5.8.5). Clearly neither the batch nor this single-sample version of the Perceptron algorithm are on-line because we must store and potentially revisit all of the training patterns.

FIXED INCREMENT

Two further simplifications help to clarify the exposition. First, we shall temporarily restrict our attention to the case in which $\eta(k)$ is constant—the so-called *fixed-increment* case. It is clear from Eq. 18 that if $\eta(t)$ is constant, it merely serves to scale the samples; thus, in the fixed-increment case we can take $\eta(t) = 1$ with no loss in generality. The second simplification merely involves notation. When the samples are considered sequentially, some will be misclassified. Because we shall only change the weight vector when there is an error, we really need only pay attention to the misclassified samples. Thus we denote the sequence of samples using superscripts—that is, by $\mathbf{y}^1$, $\mathbf{y}^2, \ldots, \mathbf{y}^k, \ldots$, where each $\mathbf{y}^k$ is one of the n samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$ and where each $\mathbf{y}^k$ is misclassified. For example, if the samples $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$ are considered cyclically and if the marked samples

$$\overset{\downarrow}{\mathbf{y}_1},\ \mathbf{y}_2,\ \overset{\downarrow}{\mathbf{y}_3},\ \overset{\downarrow}{\mathbf{y}_1},\ \overset{\downarrow}{\mathbf{y}_2},\ \mathbf{y}_3,\ \mathbf{y}_1,\ \overset{\downarrow}{\mathbf{y}_2},\ \ldots \tag{19}$$

FIXED-INCREMENT RULE

are misclassified, then the sequence $\mathbf{y}^1,\ \mathbf{y}^2,\ \mathbf{y}^3,\ \mathbf{y}^4,\ \mathbf{y}^5, \ldots$ denotes the sequence $\mathbf{y}_1,\ \mathbf{y}_3,\ \mathbf{y}_1,\ \mathbf{y}_2,\ \mathbf{y}_2, \ldots$. With this understanding, the *fixed-increment rule* for generating a sequence of weight vectors can be written as

$$\begin{aligned} &\mathbf{a}(1) && \text{arbitrary} \\ &\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k && k \geq 1 \end{aligned} \tag{20}$$

where $\mathbf{a}^t(k)\mathbf{y}^k \leq 0$ for all k. If we let n denote the total number of patterns, the algorithm is as follows:

Algorithm 4. (Fixed-Increment Single-Sample Perceptron)

```
1 begin initialize a, $k \leftarrow 0$
2         do $k \leftarrow (k+1) \bmod n$
3             if $\mathbf{y}^k$ is misclassified by a then $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{y}^k$
4         until all patterns properly classified
5     return a
6 end
```

The fixed-increment Perceptron rule is the simplest of many algorithms that have been proposed for solving systems of linear inequalities. Geometrically, its interpretation in weight space is particularly clear. Because $\mathbf{a}(k)$ misclassifies $\mathbf{y}^k$, $\mathbf{a}(k)$ is not on the positive side of the $\mathbf{y}^k$ hyperplane $\mathbf{a}^t\mathbf{y}^k = 0$. The addition of $\mathbf{y}^k$ to $\mathbf{a}(k)$ moves the weight vector directly toward and perhaps across this hyperplane. Whether the hyperplane is crossed or not, the new inner product $\mathbf{a}^t(k+1)\mathbf{y}^k$ is larger than the old inner product $\mathbf{a}^t(k)\mathbf{y}^k$ by the amount $\|\mathbf{y}^k\|^2$, and the correction is hence moving the weight vector in a good direction (Fig. 5.13).

Clearly this algorithm can only terminate if the samples are linearly separable; we now prove that indeed it terminates so long as the samples are linearly separable.

■ **Theorem 5.1. (Perceptron Convergence)** If training samples are linearly separable, then the sequence of weight vectors given by Algorithm 4 will terminate at a solution vector.

Proof. In seeking a proof, it is natural to try to show that each correction brings the weight vector closer to the solution region. That is, one might try to show that if $\hat{\mathbf{a}}$ is any solution vector, then $\|\mathbf{a}(k+1) - \hat{\mathbf{a}}\|$ is smaller than $\|\mathbf{a}(k) - \hat{\mathbf{a}}\|$. While this turns out not to be true in general (cf. steps 6 and 7 in Fig. 5.13), we shall see that it is true for solution vectors that are sufficiently long.

Let $\hat{\mathbf{a}}$ be any solution vector, so that $\hat{\mathbf{a}}^t\mathbf{y}_i$ is strictly positive for all i, and let α be a positive scale factor. From Eq. 20 we have

$$\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}} = (\mathbf{a}(k) - \alpha\hat{\mathbf{a}}) + \mathbf{y}^k$$

and hence

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 = \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 + 2(\mathbf{a}(k) - \alpha\hat{\mathbf{a}})^t\mathbf{y}^k + \|\mathbf{y}^k\|^2.$$

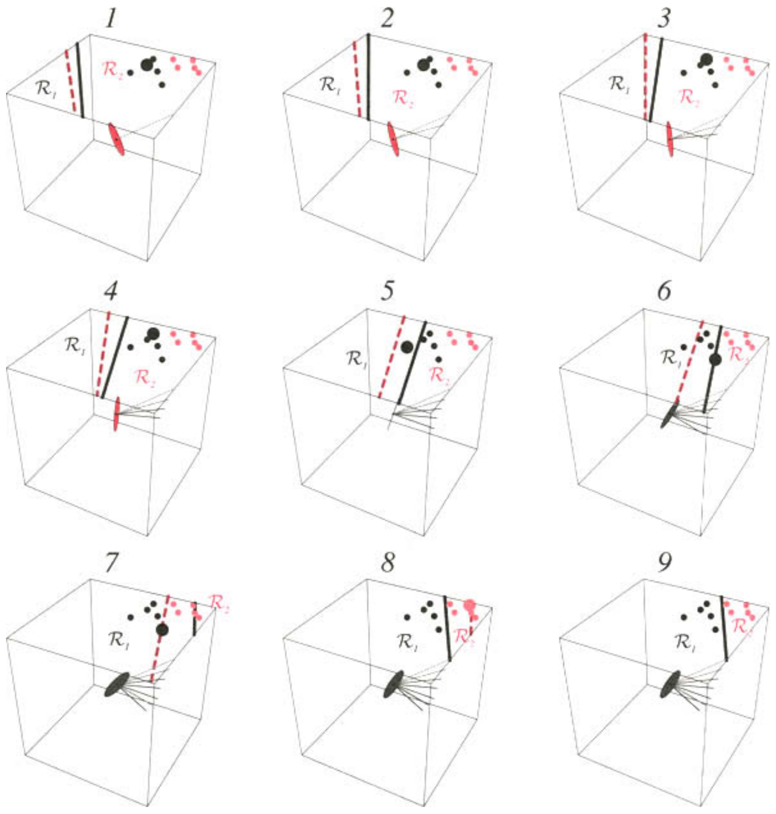


FIGURE 5.13. Samples from two categories, ω_1 (black) and ω_2 (red) are shown in augmented feature space, along with an augmented weight vector **a**. At each step in a fixed-increment rule, one of the misclassified patterns, $\mathbf{y}^k$, is shown by the large dot. A correction $\Delta\mathbf{a}$ (proportional to the pattern vector $\mathbf{y}^k$) is added to the weight vector—toward an ω_1 point or away from an ω_2 point. This changes the decision boundary from the dashed position (from the previous update) to the solid position. The sequence of resulting **a** vectors is shown, where later values are shown darker. In this example, by step 9 a solution vector has been found and the categories are successfully separated by the decision boundary shown.

Because $\mathbf{y}^k$ was misclassified, $\mathbf{a}^t(k)\mathbf{y}^k \leq 0$ and thus

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\hat{\mathbf{a}}^t\mathbf{y}^k + \|\mathbf{y}^k\|^2.$$

Because $\hat{\mathbf{a}}^t\mathbf{y}^k$ is strictly positive, the second term will dominate the third if α is sufficiently large. In particular, if we let β be the maximum length of a pattern vector,

$$\beta^2 = \max_i \|\mathbf{y}_i\|^2, \tag{21}$$

and let γ be the smallest inner product of the solution vector with any pattern vector, that is,

$$\gamma = \min_i \left[\hat{\mathbf{a}}^t\mathbf{y}_i\right] > 0, \tag{22}$$

then we have the inequality

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - 2\alpha\gamma + \beta^2.$$

If we choose

$$\alpha = \frac{\beta^2}{\gamma}, \tag{23}$$

we obtain

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k) - \alpha\hat{\mathbf{a}}\|^2 - \beta^2.$$

Thus, the squared distance from $\mathbf{a}(k)$ to $\alpha\hat{\mathbf{a}}$ is reduced by at least β^2 at each correction, and after k corrections we obtain

$$\|\mathbf{a}(k+1) - \alpha\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(1) - \alpha\hat{\mathbf{a}}\|^2 - k\beta^2. \tag{24}$$

Because this squared distance cannot become negative, it follows that the sequence of corrections must terminate after no more than k_0 corrections, where

$$k_0 = \frac{\|\mathbf{a}(1) - \alpha\hat{\mathbf{a}}\|^2}{\beta^2}. \tag{25}$$

Because a correction occurs whenever a sample is misclassified and because each sample appears infinitely often in the sequence, it follows that when corrections cease the resulting weight vector must classify all of the samples correctly.

The number k_0 gives us a bound on the number of corrections. If $\mathbf{a}(1) = \mathbf{0}$, we get the following particularly simple expression for k_0:

$$k_0 = \frac{\alpha^2\|\hat{\mathbf{a}}\|^2}{\beta^2} = \frac{\beta^2\|\hat{\mathbf{a}}\|^2}{\gamma^2} = \frac{\max\limits_i \|\mathbf{y}_i\|^2\|\hat{\mathbf{a}}\|^2}{\min\limits_i[\mathbf{y}_i^t\hat{\mathbf{a}}]^2}. \tag{26}$$

The denominator in Eq. 26 shows that the difficulty of the problem is essentially determined by the samples most nearly orthogonal to the solution vector. Unfortunately, it provides no help when we face an unsolved problem, because the bound is expressed in terms of a solution vector that is unknown. In general, it is clear that linearly separable problems can be made arbitrarily difficult to solve by making the samples almost coplanar (Computer exercise 2). Nevertheless, if the training samples are linearly separable, the fixed-increment rule will yield a solution after a finite number of corrections.

5.5.3 Some Direct Generalizations

The fixed increment rule can be generalized to provide a variety of related algorithms. We shall briefly consider two variants of particular interest. The first variant introduces a *variable increment* $\eta(k)$ and a margin b, and it calls for a correction whenever $\mathbf{a}^t(k)\mathbf{y}^k$ fails to exceed the margin. The update is given by

VARIABLE INCREMENT

$$\begin{array}{ll} \mathbf{a}(1) & \text{arbitrary} \\ \mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{y}^k & k \geq 1, \end{array} \tag{27}$$

where now $\mathbf{a}^t(k)\mathbf{y}^k \le b$ for all k. Thus for n patterns, our algorithm is:

Algorithm 5. (Variable-Increment Perceptron with Margin)

```
1 begin initialize a, threshold $\theta$, margin $b$, $\eta(\cdot)$, $k \leftarrow 0$
2         do $k \leftarrow (k+1) \bmod n$
3            if $\mathbf{a}^t\mathbf{y}^k \le b$ then $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)\mathbf{y}^k$
4         until $\mathbf{a}^t\mathbf{y}^k > b$ for all $k$
5     return a
6 end
```

It can be shown that if the samples are linearly separable and if

$$\eta(k) \ge 0, \tag{28}$$

$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta(k) = \infty \tag{29}$$

and

$$\lim_{m\to\infty} \frac{\sum_{k=1}^{m} \eta^2(k)}{\left(\sum_{k=1}^{m} \eta(k)\right)^2} = 0, \tag{30}$$

then $\mathbf{a}(k)$ converges to a solution vector $\mathbf{a}$ satisfying $\mathbf{a}^t\mathbf{y}_i > b$ for all i (Problem 19). In particular, these conditions on $\eta(k)$ are satisfied if $\eta(k)$ is a positive constant or if it decreases as $1/k$.

Another variant of interest is our original gradient descent algorithm for J_p,

$$\begin{aligned} &\mathbf{a}(1) \qquad \text{arbitrary} \\ &\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \textstyle\sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}, \end{aligned} \tag{31}$$

where $\mathcal{Y}_k$ is the set of training samples misclassified by $\mathbf{a}(k)$. It is easy to see that this algorithm will also yield a solution once one recognizes that if $\hat{\mathbf{a}}$ is a solution vector for $\mathbf{y}_1, \ldots, \mathbf{y}_n$, then it correctly classifies the correction vector

$$\mathbf{y}^k = \sum_{\mathbf{y}\in\mathcal{Y}_k} \mathbf{y}.$$

In greater detail, then, the algorithm is as follows:

Algorithm 6. (Batch Variable Increment Perceptron)

```
1 begin initialize a, $\eta(\cdot)$, $k \leftarrow 0$
2         do $k \leftarrow (k+1) \bmod n$
3            $\mathcal{Y}_k = \{\}$
4            $j = 0$
5            do $j \leftarrow j + 1$
```

```
6             if $\mathbf{y}_j$ is misclassified then  Append $\mathbf{y}_j$ to $\mathcal{Y}_k$
7          until $j = n$
8          $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y} \in \mathcal{Y}_k} \mathbf{y}$
9       until $\mathcal{Y}_k = \{\}$
10   return a
11 end
```

The benefit of batch gradient descent is that the trajectory of the weight vector is smoothed, compared to that in corresponding single-sample algorithms (e.g., Algorithm 5), because at each update the full set of misclassified patterns is used—the local statistical variations in the misclassified patterns tend to cancel while the large-scale trend does not. Thus, if the samples are linearly separable, all of the possible correction vectors form a linearly separable set, and if $\eta(k)$ satisfies Eqs. 28–30, the sequence of weight vectors produced by the gradient descent algorithm for $J_p(\cdot)$ will always converge to a solution vector.

It is interesting to note that the conditions on $\eta(k)$ are satisfied if $\eta(k)$ is a positive constant, if it decreases as $1/k$, or even if it increases as k. Generally speaking, one would prefer to have $\eta(k)$ become smaller as time goes on. This is particularly true if there is reason to believe that the set of samples is not linearly separable, because it reduces the disruptive effects of a few "bad" samples. However, in the separable case it is a curious fact that one can allow $\eta(k)$ to become larger and still obtain a solution.

This observation brings out one of the differences between theory and practice. From a theoretical viewpoint, it is interesting that we can obtain a solution in a finite number of steps for any finite set of separable samples, for any initial weight vector $\mathbf{a}(1)$, for any nonnegative margin b, and for any scale factor $\eta(k)$ satisfying Eqs. 28–30. From a practical viewpoint, we want to make wise choices for these quantities. Consider the margin b, for example. If b is much smaller than $\eta(k)\|\mathbf{y}^k\|^2$, the amount by which a correction increases $\mathbf{a}^t(k)\mathbf{y}^k$, it is clear that it will have little if any effect. If it is much larger than $\eta(k)\|\mathbf{y}^k\|^2$, many corrections will be needed to satisfy the conditions $\mathbf{a}^t(k)\mathbf{y}^k > b$. A value close to $\eta(k)\|\mathbf{y}^k\|^2$ is often a useful compromise. In addition to these choices for $\eta(k)$ and b, the scaling of the components of $\mathbf{y}^k$ can also have a great effect on the results. The possession of a convergence theorem does not remove the need for thought in applying these techniques.

WINNOW ALGORITHM

A close descendant of the Perceptron algorithm is the Winnow algorithm, which has applicability to separable training data. The key difference is that while the weight vector returned by the Perceptron algorithm has components a_i ($i = 0, \ldots, d$), in Winnow they are scaled according to $2\sinh[a_i]$. In one version, the balanced Winnow algorithm, there are separate "positive" and "negative" weight vectors, $\mathbf{a}^+$ and $\mathbf{a}^-$, each associated with one of the two categories to be learned. Corrections on the positive weight are made if and only if a training pattern in ω_1 is misclassified; conversely, corrections on the negative weight are made if and only if a training pattern in ω_2 is misclassified.

■ **Algorithm 7. (Balanced Winnow)**

```
1 begin initialize $\mathbf{a}^+, \mathbf{a}^-, \eta(\cdot), k \leftarrow 0, \alpha > 1$
2        if Sgn$[\mathbf{a}^{+t}\mathbf{y}_k - \mathbf{a}^{-t}\mathbf{y}_k] \neq z_k$ (pattern misclassified)
3        then if $z_k = +1$ then $a_i^+ \leftarrow \alpha^{+y_i} a_i^+$; $a_i^- \leftarrow \alpha^{-y_i} a_i^-$ for all $i$
```

```
4           if $z_k = -1$ then $a_i^+ \leftarrow \alpha^{-y_i} a_i^+$; $a_i^- \leftarrow \alpha^{+y_i} a_i^-$ for all $i$
5     return $\mathbf{a}^+, \mathbf{a}^-$
6  end
```

There are two main benefits of such a version of the Winnow algorithm. The first is that during training each of the two constituent weight vectors moves in a uniform direction and this means that for separable data the "gap," determined by these two vectors, can never increase in size. This leads to a convergence proof that, while somewhat more complicated, is nevertheless more general than the Perceptron convergence theorem (cf. Bibliography). The second benefit is that convergence is generally faster than in a Perceptron, because for proper setting of learning rate, each constituent weight does not overshoot its final value. This benefit is especially pronounced whenever a large number of irrelevant or redundant features are present (Computer exercise 6).

5.6 RELAXATION PROCEDURES

We have seen how a linear classifier is trained through the minimization of the Perceptron criterion of Eq. 16. We can generalize this approach, in so-called "relaxation procedures," to include a broader class of criterion functions and methods for minimizing them.

5.6.1 The Descent Algorithm

The criterion function $J_p(\cdot)$ is by no means the only function we can construct that is minimized when **a** is a solution vector. A close but distinct relative is

$$J_q(\mathbf{a}) = \sum_{\mathbf{y}\in\mathcal{Y}} (\mathbf{a}^t\mathbf{y})^2, \tag{32}$$

where $\mathcal{Y}(\mathbf{a})$ again denotes the set of training samples misclassified by **a**. Both J_p and J_q focus attention on the misclassified samples. The chief difference is that the gradient of J_q is continuous, whereas the gradient of J_p is not. Thus, J_q presents a smoother surface to search (Fig. 5.11). Unfortunately, J_q is so smooth near the boundary of the solution region that the sequence of weight vectors can converge to a point on the boundary. It is particularly embarrassing to spend some time following the gradient merely to reach the boundary point $\mathbf{a} = \mathbf{0}$. Another problem with J_q is that its value can be dominated by the longest sample vectors. Both of these problems are avoided by the criterion function

$$J_r(\mathbf{a}) = \frac{1}{2}\sum_{\mathbf{y}\in\mathcal{Y}} \frac{(\mathbf{a}^t\mathbf{y} - b)^2}{\|\mathbf{y}\|^2}, \tag{33}$$

where now $\mathcal{Y}(\mathbf{a})$ is the set of samples for which $\mathbf{a}^t\mathbf{y} \leq b$. (If $\mathcal{Y}(\mathbf{a})$ is empty, we define J_r to be zero.) Thus, $J_r(\mathbf{a})$ is never negative, and is zero if and only if $\mathbf{a}^t\mathbf{y} \geq b$ for all of the training samples. The gradient of J_r is given by

$$\nabla J_r = \sum_{\mathbf{y}\in\mathcal{Y}} \frac{\mathbf{a}^t\mathbf{y} - b}{\|\mathbf{y}\|^2}\mathbf{y},$$

and the update rule is

$$\begin{aligned} &\mathbf{a}(1) \qquad \text{arbitrary} \\ &\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}} \frac{b-\mathbf{a}^t\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}. \end{aligned} \tag{34}$$

Thus the relaxation algorithm becomes:

■ **Algorithm 8. (Batch Relaxation with Margin)**

```
 1 begin initialize $\mathbf{a}, \eta(\cdot), b, k \leftarrow 0$
 2         do $k \leftarrow (k+1) \bmod n$
 3             $\mathcal{Y}_k = \{\}$
 4             $j \leftarrow 0$
 5             do $j \leftarrow j+1$
 6                 if $\mathbf{a}^t\mathbf{y}^j \le b$ then Append $\mathbf{y}^j$ to $\mathcal{Y}_k$
 7             until $j = n$
 8             $\mathbf{a} \leftarrow \mathbf{a} + \eta(k) \sum_{\mathbf{y}\in\mathcal{Y}} \frac{b-\mathbf{a}^t\mathbf{y}}{\|\mathbf{y}\|^2}\mathbf{y}$
 9         until $\mathcal{Y}_k = \{\}$
10     return a
11 end
```

As before, we find it easier to prove convergence when the samples are considered one at a time rather than jointly—that is, single-sample rather than batch. We also limit our attention to the fixed-increment case, $\eta(k) = \eta$. Thus, we are again led to consider a sequence $\mathbf{y}^1, \mathbf{y}^2, \ldots$ formed from those samples that call for the weight vector to be corrected. The single-sample correction rule analogous to Eq. 34 is

$$\begin{aligned} &\mathbf{a}(1) \qquad \text{arbitrary} \\ &\mathbf{a}(k+1) = \mathbf{a}(k) + \eta \frac{b-\mathbf{a}^t(k)\mathbf{y}^k}{\|\mathbf{y}^k\|^2}\mathbf{y}^k, \end{aligned} \tag{35}$$

where $\mathbf{a}^t(k)\mathbf{y}^k \le b$ for all k. The algorithm is as follows:

■ **Algorithm 9. (Single-Sample Relaxation with Margin)**

```
1 begin initialize $\mathbf{a}, \eta(\cdot), k \leftarrow 0$
2         do $k \leftarrow (k+1) \bmod n$
3             if $\mathbf{a}^t\mathbf{y}^k \le b$ then $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)\frac{b-\mathbf{a}^t\mathbf{y}^k}{\|\mathbf{y}^k\|^2}\mathbf{y}^k$
4         until $\mathbf{a}^t\mathbf{y}^k > b$ for all $\mathbf{y}^k$
5     return a
6 end
```

This algorithm is known as the *single-sample relaxation rule with margin*, and it has a simple geometrical interpretation. The quantity

$$r(k) = \frac{b - \mathbf{a}^t(k)\mathbf{y}^k}{\|\mathbf{y}^k\|} \tag{36}$$

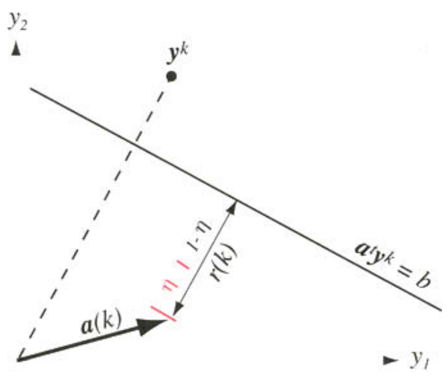


FIGURE 5.14. In each step of a basic relaxation algorithm, the weight vector is moved a proportion η of the way toward the hyperplane defined by $\mathbf{a}^t\mathbf{y}^k = b$.

is the distance from $\mathbf{a}(k)$ to the hyperplane $\mathbf{a}^t\mathbf{y}^k = b$. Because $\mathbf{y}^k/\|\mathbf{y}^k\|$ is the unit normal vector for the hyperplane, Eq. 35 calls for $\mathbf{a}(k)$ to be moved a certain fraction η of the distance from $\mathbf{a}(k)$ to the hyperplane. If $\eta = 1$, $\mathbf{a}(k)$ is moved exactly to the hyperplane, so that the "tension" created by the inequality $\mathbf{a}^t(k)\mathbf{y}^k \leq b$ is "relaxed" (Fig. 5.14). From Eq. 35, after a correction, we obtain

$$\mathbf{a}^t(k+1)\mathbf{y}^k - b = (1-\eta)(\mathbf{a}^t(k)\mathbf{y}^k - b). \tag{37}$$

UNDER-RELAXATION

OVER-RELAXATION

If $\eta < 1$, then $\mathbf{a}^t(k+1)\mathbf{y}^k$ is still less than b, while if $\eta > 1$, then $\mathbf{a}^t(k+1)\mathbf{y}^k$ is greater than b. These conditions are referred to as *underrelaxation* and *overrelaxation*, respectively. In general, we shall restrict η to the range $0 < \eta < 2$ (Figs. 5.14 and 5.15).

5.6.2 Convergence Proof

When the relaxation rule is applied to a set of linearly separable samples, the number of corrections may or may not be finite. If it is finite, then of course we have obtained a solution vector. If it is not finite, we shall see that $\mathbf{a}(k)$ converges to a limit vector on the boundary of the solution region. Because the region in which $\mathbf{a}^t\mathbf{y} \geq b$ is contained in a larger region where $\mathbf{a}^t\mathbf{y} > 0$ if $b > 0$, this implies that $\mathbf{a}(k)$ will enter

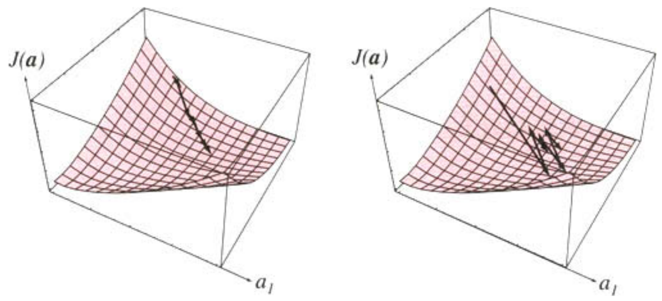


FIGURE 5.15. At the left, underrelaxation ($\eta < 1$) leads to needlessly slow descent, or even failure to converge. Overrelaxation ($1 < \eta < 2$, shown at the right) describes overshooting; nevertheless, convergence will ultimately be achieved.

this larger region at least once, eventually remaining there for all k greater than some finite k_0.

The proof depends upon the fact that if $\hat{\mathbf{a}}$ is *any* vector in the solution region—that is, any vector satisfying $\hat{\mathbf{a}}^t\mathbf{y}_i > b$ for all i—then at each step, $\mathbf{a}(k)$ gets closer to $\hat{\mathbf{a}}$. This fact follows at once from Eq. 35, because

$$\|\mathbf{a}(k+1)-\hat{\mathbf{a}}\|^2 = \|\mathbf{a}(k)-\hat{\mathbf{a}}\|^2 - 2\eta\frac{(b-\mathbf{a}^t(k)\mathbf{y}^k)}{\|\mathbf{y}^k\|^2}(\hat{\mathbf{a}}-\mathbf{a}(k))^t\mathbf{y}^k + \eta^2\frac{(b-\mathbf{a}^t(k)\mathbf{y}^k)^2}{\|\mathbf{y}^k\|^2} \qquad (38)$$

and

$$(\hat{\mathbf{a}}-\mathbf{a}(k))^t\mathbf{y}^k > b-\mathbf{a}^t(k)\mathbf{y}^k \geq 0, \qquad (39)$$

so that

$$\|\mathbf{a}(k+1)-\hat{\mathbf{a}}\|^2 \leq \|\mathbf{a}(k)-\hat{\mathbf{a}}\|^2 - \eta(2-\eta)\frac{(b-\mathbf{a}^t(k)\mathbf{y}^k)^2}{\|\mathbf{y}^k\|^2}. \qquad (40)$$

Because we restrict η to the range $0 < \eta < 2$, it follows that $\|\mathbf{a}(k+1)-\hat{\mathbf{a}}\| \leq \|\mathbf{a}(k)-\hat{\mathbf{a}}\|$. Thus, the vectors in the sequence $\mathbf{a}(1), \mathbf{a}(2), \ldots$ get closer and closer to $\hat{\mathbf{a}}$, and in the limit as k goes to infinity the distance $\|\mathbf{a}(k)-\hat{\mathbf{a}}\|$ approaches some limiting distance $r(\hat{\mathbf{a}})$. This means that as k goes to infinity, $\mathbf{a}(k)$ is confined to the surface of a hypersphere with center $\hat{\mathbf{a}}$ and radius $r(\hat{\mathbf{a}})$. Because this is true for any $\hat{\mathbf{a}}$ in the solution region, the limiting $\mathbf{a}(k)$ is confined to the intersection of the hyperspheres centered about all of the possible solution vectors.

We now show that the common intersection of these hyperspheres is a single point on the boundary of the solution region. Suppose first that there are at least two points $\mathbf{a}'$ and $\mathbf{a}''$ on the common intersection. Then $\|\mathbf{a}'-\hat{\mathbf{a}}\| = \|\mathbf{a}''-\hat{\mathbf{a}}\|$ for every $\hat{\mathbf{a}}$ in the solution region. But this implies that the solution region is contained in the $(\hat{d}-1)$-dimensional hyperplane of points equidistant from $\mathbf{a}'$ to $\mathbf{a}''$, whereas we know that the solution region is $\hat{d}$-dimensional. (Stated formally, if $\hat{\mathbf{a}}^t\mathbf{y}_i > 0$ for $i = 1, \ldots, n$, then for any $\hat{d}$-dimensional vector $\mathbf{v}$, we have $(\hat{\mathbf{a}}+\epsilon\mathbf{v})^t\mathbf{y} > 0$ for $i = 1, \ldots, n$ if ϵ is sufficiently small.) Thus, $\mathbf{a}(k)$ converges to a single point $\mathbf{a}$. This point is certainly not inside the solution region, because then the sequence would be finite. It is not outside either, because each correction causes the weight vector to move η times its distance from the boundary plane, thereby preventing the vector from being bounded away from the boundary forever. Hence the limit point must be on the boundary.

5.7 NONSEPARABLE BEHAVIOR

ERROR-CORRECTING PROCEDURE

The Perceptron and relaxation procedures give us a number of simple methods for finding a separating vector when the samples are linearly separable. All of these methods are called *error-correcting procedures* because they call for a modification of the weight vector when and only when an error is encountered. Their success on separable problems is largely due to this relentless search for an error-free solution. In practice, one would only consider the use of these methods if there was reason to believe that the error rate for the optimal linear discriminant function is low.

Of course, even if a separating vector is found for the training samples, it does not follow that the resulting classifier will perform well on independent test data. A moment's reflection will show that *any* set of fewer than $2\hat{d}$ samples is likely to be linearly separable—a matter we shall return to in Chapter 9. Thus, one should use several times that many design samples to overdetermine the classifier, thereby ensuring that the performance on training and test data will be similar. Unfortunately, sufficiently large design sets are almost certainly *not* linearly separable. This makes it important to know how the error-correction procedures will behave when the samples are nonseparable.

Because no weight vector can correctly classify every sample in a nonseparable set (by definition), it is clear that the corrections in an error-correction procedure can never cease. Each algorithm produces an infinite sequence of weight vectors, any member of which may or may not yield a useful "solution." The exact nonseparable behavior of these rules has been studied thoroughly in a few special cases. It is known, for example, that the length of the weight vectors produced by the fixed-increment rule are bounded. Empirical rules for terminating the correction procedure are often based on this tendency for the length of the weight vector to fluctuate near some limiting value. From a theoretical viewpoint, if the components of the samples are integer-valued, the fixed-increment procedure yields a finite-state process. If the correction process is terminated at some arbitrary point, the weight vector may or may not be in a good state. By averaging the weight vectors produced by the correction rule, one can reduce the risk of obtaining a bad solution by accidentally choosing an unfortunate termination time.

A number of similar heuristic modifications to the error-correction rules have been suggested and studied empirically. The goal of these modifications is to obtain acceptable performance on nonseparable problems while preserving the ability to find a separating vector on separable problems. A common suggestion is the use of a variable increment $\eta(k)$, with $\eta(k)$ approaching zero as k approaches infinity. The rate at which $\eta(k)$ approaches zero is quite important. If it is too slow, the results will still be sensitive to those training samples that render the set nonseparable. If it is too fast, the weight vector may converge prematurely with less than optimal results. One way to choose $\eta(k)$ is to make it a function of recent performance, decreasing it as performance improves. Another way is to program $\eta(k)$ by a choice such as $\eta(k) = \eta(1)/k$. When we examine stochastic approximation techniques, we shall see that this latter choice is the theoretical solution to an analogous problem. Before we take up this topic, however, we shall consider an approach that sacrifices the ability to obtain a separating vector for good compromise performance on both separable and nonseparable problems.

5.8 MINIMUM SQUARED-ERROR PROCEDURES

The criterion functions we have considered thus far have focused their attention on the misclassified samples. We shall now consider a criterion function that involves *all* of the samples. Where previously we have sought a weight vector $\mathbf{a}$ making all of the inner products $\mathbf{a}^t\mathbf{y}_i$ positive, now we shall try to make $\mathbf{a}^t\mathbf{y}_i = b_i$, where the b_i are some arbitrarily specified positive constants. Thus, we have replaced the problem of finding the solution to a set of linear inequalities with the more stringent but better understood problem of finding the solution to a set of linear equations.

5.8.1 Minimum Squared-Error and the Pseudoinverse

The treatment of simultaneous linear equations is simplified by introducing matrix notation. Let $\mathbf{Y}$ be the n-by-$\hat{d}$ matrix ($\hat{d} = d + 1$) whose ith row is the vector $\mathbf{y}_i^t$, and let $\mathbf{b}$ be the column vector $\mathbf{b} = (b_1, \ldots, b_n)^t$. Then our problem is to find a weight vector $\mathbf{a}$ satisfying

$$\begin{pmatrix} y_{10} & y_{11} & \cdots & y_{1d} \\ y_{20} & y_{21} & \cdots & y_{2d} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ y_{n0} & y_{n1} & \cdots & y_{nd} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ \vdots \\ b_n \end{pmatrix} \quad \text{or} \quad \mathbf{Ya} = \mathbf{b}. \tag{41}$$

If $\mathbf{Y}$ were nonsingular, we could write $\mathbf{a} = \mathbf{Y}^{-1}\mathbf{b}$ and obtain a formal solution at once. However, $\mathbf{Y}$ is rectangular, usually with more rows than columns. When there are more equations than unknowns, $\mathbf{a}$ is overdetermined, and ordinarily no exact solution exists. However, we can seek a weight vector $\mathbf{a}$ that minimizes some function of the error between $\mathbf{Ya}$ and $\mathbf{b}$. If we define the error vector $\mathbf{e}$ by

$$\mathbf{e} = \mathbf{Ya} - \mathbf{b} \tag{42}$$

then one approach is to try to minimize the squared length of the error vector. This is equivalent to minimizing the sum-of-squared-error criterion function

$$J_s(\mathbf{a}) = \|\mathbf{Ya} - \mathbf{b}\|^2 = \sum_{i=1}^{n} (\mathbf{a}^t\mathbf{y}_i - b_i)^2. \tag{43}$$

The problem of minimizing the sum of squared error is a classical one. It can be solved by a gradient search procedure, as we shall see below. A simple closed-form solution can also be found by forming the gradient

$$\nabla J_s = \sum_{i=1}^{n} 2(\mathbf{a}^t\mathbf{y}_i - b_i)\mathbf{y}_i = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b}) \tag{44}$$

and setting it equal to zero. This yields the necessary condition

$$\mathbf{Y}^t\mathbf{Ya} = \mathbf{Y}^t\mathbf{b}, \tag{45}$$

and in this way we have converted the problem of solving $\mathbf{Ya} = \mathbf{b}$ to that of solving $\mathbf{Y}^t\mathbf{Ya} = \mathbf{Y}^t\mathbf{b}$. This celebrated equation has the great advantage that the $\hat{d}$-by-$\hat{d}$ matrix $\mathbf{Y}^t\mathbf{Y}$ is square and often nonsingular. If it is nonsingular, we can solve for $\mathbf{a}$ uniquely as

$$\begin{aligned} \mathbf{a} &= (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t\mathbf{b} \\ &= \mathbf{Y}^\dagger\mathbf{b}, \end{aligned} \tag{46}$$

where the $\hat{d}$-by-n matrix

$$\mathbf{Y}^\dagger \equiv (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t \tag{47}$$

PSEUDOINVERSE

is called the *pseudoinverse* of **Y**. Note that if **Y** is square and nonsingular, the pseudoinverse coincides with the regular inverse. Note also that $\mathbf{Y}^\dagger\mathbf{Y} = \mathbf{I}$, but $\mathbf{Y}\mathbf{Y}^\dagger \neq \mathbf{I}$ in general. However, a minimum-squared-error (MSE) solution always exists. In particular, if $\mathbf{Y}^\dagger$ is defined more generally by

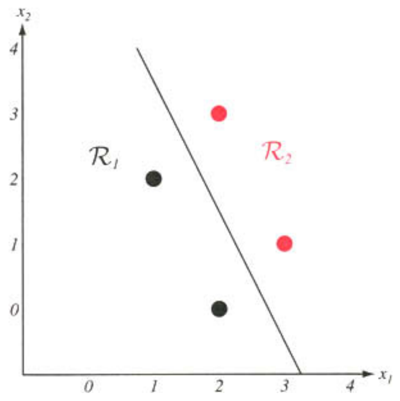
$$\mathbf{Y}^\dagger \equiv \lim_{\epsilon\to 0}(\mathbf{Y}^t\mathbf{Y} + \epsilon\mathbf{I})^{-1}\mathbf{Y}^t, \tag{48}$$

it can be shown that this limit always exists, and that $\mathbf{a} = \mathbf{Y}^\dagger\mathbf{b}$ is an MSE solution to $\mathbf{Ya} = \mathbf{b}$.

The MSE solution depends on the margin vector **b**, and we shall see that different choices for **b** give the solution different properties. If **b** is fixed arbitrarily, there is no reason to believe that the MSE solution yields a separating vector in the linearly separable case. However, it is reasonable to hope that by minimizing the squared-error criterion function we might obtain a useful discriminant function in both the separable and the nonseparable cases. Below we examine two properties of the solution that support this hope.

EXAMPLE 1 Constructing a Linear Classifier by Matrix Pseudoinverse

Suppose we have the following two-dimensional points for two categories: ω_1: $(1,2)^t$ and $(2,0)^t$, and ω_2: $(3,1)^t$ and $(2,3)^t$, as shown in black and red, respectively, in the accompanying figure.



Four training points and the decision boundary $\mathbf{a}^t\begin{pmatrix}1\\x_1\\x_2\end{pmatrix} = 0$, where **a** was found by means of a pseudoinverse technique.

Our matrix **Y** is therefore

$$\mathbf{Y} = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 0 \\ -1 & -3 & -1 \\ -1 & -2 & -3 \end{pmatrix}$$

and after a few simple calculations we find that its pseudoinverse is

$$\mathbf{Y}^{\dagger} = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t = \begin{pmatrix} 5/4 & 13/12 & 3/4 & 7/12 \\ -1/2 & -1/6 & -1/2 & -1/6 \\ 0 & -1/3 & 0 & -1/3 \end{pmatrix}$$

We arbitrarily let all the margins be equal—that is, $\mathbf{b} = (1, 1, 1, 1)^t$. Our solution is $\mathbf{a} = \mathbf{Y}^{\dagger}\mathbf{b} = (11/3, -4/3, -2/3)^t$, and leads to the decision boundary shown in the figure. Other choices for **b** would typically lead to different decision boundaries, of course.

5.8.2 Relation to Fisher's Linear Discriminant

In this section we shall show that with the proper choice of the vector **b**, the MSE discriminant function $\mathbf{a}^t\mathbf{y}$ is directly related to Fisher's linear discriminant (Chapter 3 Section 3.8.2). To do this, we must return to the use of linear rather than generalized linear discriminant functions. We assume that we have a set of n d-dimensional samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$, n_1 of which are in the subset $\mathcal{D}_1$ labeled ω_1, and n_2 of which are in the subset $\mathcal{D}_2$ labeled ω_2. Furthermore, we assume that a sample $\mathbf{y}_i$ is formed from $\mathbf{x}_i$ by adding a threshold component $x_0 = 1$ to make an *augmented pattern vector.* Further, if the sample is labeled ω_2, then the entire pattern vector is multiplied by -1—the "normalization" we saw in Section 5.4.1. With no loss in generality, we can assume that the first n_1 samples are labeled ω_1 and the second n_2 are labeled ω_2. Then the matrix **Y** can be partitioned as follows:

AUGMENTED PATTERN VECTOR

$$\mathbf{Y} = \begin{bmatrix} \mathbf{1}_1 & \mathbf{X}_1 \\ -\mathbf{1}_2 & -\mathbf{X}_2 \end{bmatrix},$$

where $\mathbf{1}_i$ is a column vector of n_i ones, and $\mathbf{X}_i$ is an n_i-by-d matrix whose rows are the samples labeled ω_i. We partition **a** and **b** correspondingly, with

$$\mathbf{a} = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}$$

and with

$$\mathbf{b} = \begin{bmatrix} \dfrac{n}{n_1}\mathbf{1}_1 \\ \dfrac{n}{n_2}\mathbf{1}_2 \end{bmatrix}.$$

We shall now show that this special choice for **b** links the MSE solution to Fisher's linear discriminant.

We begin by writing Eq. 45 for **a** in terms of the partitioned matrices:

$$\begin{bmatrix} \mathbf{1}_1^t & -\mathbf{1}_2^t \\ \mathbf{X}_1^t & -\mathbf{X}_2^t \end{bmatrix} \begin{bmatrix} \mathbf{1}_1 & \mathbf{X}_1 \\ -\mathbf{1}_2 & -\mathbf{X}_2 \end{bmatrix} \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{1}_1^t & -\mathbf{1}_2^t \\ \mathbf{X}_1^t & -\mathbf{X}_2^t \end{bmatrix} \begin{bmatrix} \dfrac{n}{n_1}\mathbf{1}_1 \\ \dfrac{n}{n_2}\mathbf{1}_2 \end{bmatrix}. \quad (49)$$

By defining the sample means $\mathbf{m}_i$ and the pooled sample scatter matrix $\mathbf{S}_W$ as

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x} \qquad i = 1, 2 \tag{50}$$

and

$$\mathbf{S}_W = \sum_{i=1}^{2} \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t, \tag{51}$$

we can multiply the matrices of Eq. 49 and obtain

$$\begin{bmatrix} n & (n_1\mathbf{m}_1 + n_2\mathbf{m}_2)^t \\ (n_1\mathbf{m}_1 + n_2\mathbf{m}_2) & \mathbf{S}_W + n_1\mathbf{m}_1\mathbf{m}_1^t + n_2\mathbf{m}_2\mathbf{m}_2^t \end{bmatrix} \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} 0 \\ n(\mathbf{m}_1 - \mathbf{m}_2) \end{bmatrix}.$$

This can be viewed as a pair of equations, the first of which can be solved for w_0 in terms of $\mathbf{w}$:

$$w_0 = -\mathbf{m}^t\mathbf{w}, \tag{52}$$

where $\mathbf{m}$ is the mean of all of the samples. Substituting this in the second equation and performing a few algebraic manipulations, we obtain

$$\left[\frac{1}{n}\mathbf{S}_W + \frac{n_1 n_2}{n^2}(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t\right]\mathbf{w} = \mathbf{m}_1 - \mathbf{m}_2. \tag{53}$$

Because the vector $(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t\mathbf{w}$ is in the direction of $\mathbf{m}_1 - \mathbf{m}_2$ for any value of $\mathbf{w}$, we can write

$$\frac{n_1 n_2}{n^2}(\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t\mathbf{w} = (1 - \alpha)(\mathbf{m}_1 - \mathbf{m}_2),$$

where α is some scalar. Then Eq. 53 yields

$$\mathbf{w} = \alpha n \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2), \tag{54}$$

which, except for an unimportant scale factor, is identical to the solution for Fisher's linear discriminant. In addition, we obtain the threshold weight w_0 and the following decision rule: Decide ω_1 if $\mathbf{w}^t(\mathbf{x} - \mathbf{m}) > 0$; otherwise decide ω_2.

5.8.3 Asymptotic Approximation to an Optimal Discriminant

Another property of the MSE solution that recommends its use is that if $\mathbf{b} = \mathbf{1}_n$, it approaches a minimum mean-squared-error approximation to the Bayes discriminant function

$$g_0(\mathbf{x}) = P(\omega_1|\mathbf{x}) - P(\omega_2|\mathbf{x}) \tag{55}$$

in the limit as the number of samples approaches infinity. To demonstrate this fact, we must assume that the samples are drawn independently, identically distributed (i.i.d.) according to the probability law

$$p(\mathbf{x}) = p(\mathbf{x}|\omega_1)P(\omega_1) + p(\mathbf{x}|\omega_2)P(\omega_2). \tag{56}$$

In terms of the augmented vector $\mathbf{y}$, the MSE solution yields the series expansion $g(\mathbf{x}) = \mathbf{a}^t\mathbf{y}$, where $\mathbf{y} = \mathbf{y}(\mathbf{x})$. If we define the mean-squared approximation error by

$$\epsilon^2 = \int [\mathbf{a}^t\mathbf{y} - g_0(\mathbf{x})]^2 p(\mathbf{x})\, d\mathbf{x}, \tag{57}$$

then our goal is to show that ϵ^2 is minimized by the solution $\mathbf{a} = \mathbf{Y}^\dagger \mathbf{1}_n$.

The proof is simplified if we preserve the distinction between category ω_1 and category ω_2 samples. In terms of the unnormalized data, the criterion function J_s becomes

$$\begin{aligned} J_s(\mathbf{a}) &= \sum_{\mathbf{y}\in\mathcal{Y}_1} (\mathbf{a}^t\mathbf{y} - 1)^2 + \sum_{\mathbf{y}\in\mathcal{Y}_2} (\mathbf{a}^t\mathbf{y} + 1)^2 \\ &= n\left[\frac{n_1}{n}\frac{1}{n_1}\sum_{\mathbf{y}\in\mathcal{Y}_1} (\mathbf{a}^t\mathbf{y} - 1)^2 + \frac{n_2}{n}\frac{1}{n_2}\sum_{\mathbf{y}\in\mathcal{Y}_2} (\mathbf{a}^t\mathbf{y} + 1)^2\right]. \end{aligned} \tag{58}$$

Thus, by the law of large numbers, as n approaches infinity $(1/n)J_s(\mathbf{a})$ approaches

$$\bar{J}(\mathbf{a}) = P(\omega_1)\mathcal{E}_1[(\mathbf{a}^t\mathbf{y} - 1)^2] + P(\omega_2)\mathcal{E}_2[(\mathbf{a}^t\mathbf{y} + 1)^2], \tag{59}$$

with probability one, where

$$\mathcal{E}_1[(\mathbf{a}^t\mathbf{y} - 1)^2] = \int (\mathbf{a}^t\mathbf{y} - 1)^2 p(\mathbf{x}|\omega_1)\, d\mathbf{x}$$

and

$$\mathcal{E}_2[(\mathbf{a}^t\mathbf{y} + 1)^2] = \int (\mathbf{a}^t\mathbf{y} + 1)^2 p(\mathbf{x}|\omega_2)\, d\mathbf{x}.$$

Now, if we recognize from Eq. 55 that

$$g_0(\mathbf{x}) = \frac{p(\mathbf{x}, \omega_1) - p(\mathbf{x}, \omega_2)}{p(\mathbf{x})},$$

we see that

$$\begin{aligned} \bar{J}(\mathbf{a}) &= \int (\mathbf{a}^t\mathbf{y} - 1)^2 p(\mathbf{x}, \omega_1)\, d\mathbf{x} + \int (\mathbf{a}^t\mathbf{y} + 1)^2 p(\mathbf{x}, \omega_2)\, d\mathbf{x} \\ &= \int (\mathbf{a}^t\mathbf{y})^2 p(\mathbf{x})\, d\mathbf{x} - 2\int \mathbf{a}^t\mathbf{y}\, g_0(\mathbf{x}) p(\mathbf{x})\, d\mathbf{x} + 1 \\ &= \underbrace{\int [\mathbf{a}^t\mathbf{y} - g_0(\mathbf{x})]^2 p(\mathbf{x})\, d\mathbf{x}}_{\epsilon^2} + \underbrace{\left[1 - \int g_0^2(\mathbf{x}) p(\mathbf{x})\, d\mathbf{x}\right]}_{\text{independent of } \mathbf{a}}. \end{aligned} \tag{60}$$

The second term in this sum is independent of the weight vector $\mathbf{a}$. Hence, the $\mathbf{a}$ that minimizes J_s also minimizes ϵ^2—the mean-squared-error between $\mathbf{a}^t\mathbf{y}$ and $g(\mathbf{x})$ (Fig. 5.16). In Chapter 6 we shall see that analogous properties also hold for many multilayer neural networks.

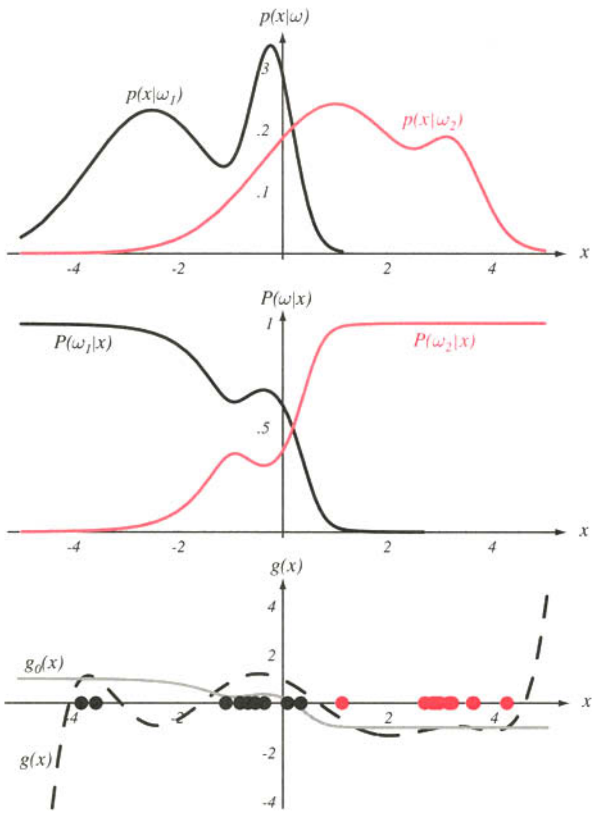


FIGURE 5.16. The top figure shows two class-conditional densities, and the middle figure the posteriors, assuming equal priors. Minimizing the MSE error also minimizes the mean-squared-error between $\mathbf{a}^t\mathbf{y}$ and the discriminant function $g(\mathbf{x})$ (here a seventh-order polynomial) measured over the data distribution, as shown at the bottom. Note that the resulting $g(x)$ best approximates $g_0(x)$ in the regions where the data points lie.

This result gives considerable insight into the MSE procedure. By approximating $g_0(\mathbf{x})$, the discriminant function $\mathbf{a}^t\mathbf{y}$ gives direct information about the posterior probabilities $P(\omega_1|\mathbf{x}) = (1 + g_0)/2$ and $P(\omega_2|\mathbf{x}) = (1 - g_0)/2$. The quality of the approximation depends on the functions $y_i(\mathbf{x})$ and the number of terms in the expansion $\mathbf{a}^t\mathbf{y}$. Unfortunately, the mean-square-error criterion places emphasis on points where $p(\mathbf{x})$ is larger, rather than on points near the decision surface $g_0(\mathbf{x}) = 0$. Thus, the discriminant function that "best" approximates the Bayes discriminant does not necessarily minimize the probability of error. Despite this property, the MSE solution has interesting properties and has received considerable attention in the literature. We shall encounter the mean-square approximation of $g_0(\mathbf{x})$ again when we consider stochastic approximation methods and multilayer neural networks.

5.8.4 The Widrow-Hoff or LMS Procedure

We remarked earlier that $J_s(\mathbf{a}) = \|\mathbf{Ya} - \mathbf{b}\|^2$ could be minimized by a gradient descent procedure. Such an approach has two advantages over merely computing the pseudoinverse: (1) It avoids the problems that arise when $\mathbf{Y}^t\mathbf{Y}$ is singular, and (2) it avoids the need for working with large matrices. In addition, the computation involved is effectively a feedback scheme which automatically copes with some of the computational problems due to roundoff or truncation. Because $\nabla J_s = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b})$, the obvious update rule is

$$\begin{aligned} &\mathbf{a}(1) \quad \text{arbitrary} \\ &\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{Y}^t(\mathbf{Ya}(k) - \mathbf{b}). \end{aligned}$$

In Problem 26 you are asked to show that if $\eta(k) = \eta(1)/k$, where $\eta(1)$ is any positive constant, then this rule generates a sequence of weight vectors that converges to a limiting vector $\mathbf{a}$ satisfying

$$\mathbf{Y}^t(\mathbf{Ya(k)} - \mathbf{b}) = 0.$$

Thus, the descent algorithm always yields a solution regardless of whether or not $\mathbf{Y}^t\mathbf{Y}$ is singular.

While the $\hat{d}$-by-$\hat{d}$ matrix $\mathbf{Y}^t\mathbf{Y}$ is usually smaller than the $\hat{d}$-by-n matrix $\mathbf{Y}^\dagger$, the storage requirements can be reduced still further by considering the samples sequentially and using the *Widrow-Hoff* or *LMS rule* (least-mean-squared):

LMS RULE

$$\left.\begin{aligned} &\mathbf{a}(1) \quad \text{arbitrary} \\ &\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)(b(k) - \mathbf{a}^t(k)\mathbf{y}^k)\mathbf{y}^k, \end{aligned}\right\} \tag{61}$$

or in algorithm form:

■ **Algorithm 10.** **(LMS)**

```
1 begin initialize a, b, threshold $\theta$, $\eta(\cdot)$, $k \leftarrow 0$
2         do $k \leftarrow (k+1) \bmod n$
3             $\mathbf{a} \leftarrow \mathbf{a} + \eta(k)(b_k - \mathbf{a}^t\mathbf{y}^k)\mathbf{y}^k$
4         until $|\eta(k)(b_k - \mathbf{a}^t\mathbf{y}^k)\mathbf{y}^k| < \theta$
5     return a
6 end
```

At first glance this descent algorithm appears to be essentially the same as the relaxation rule. The primary difference is that the relaxation rule is an error-correction rule, so that $\mathbf{a}^t(k)\mathbf{y}^k$ does not equal b_k, and thus the corrections never cease. Therefore, $\eta(k)$ must decrease with k to obtain convergence, the choice $\eta(k) = \eta(1)/k$ being common. Exact analysis of the behavior of the Widrow-Hoff rule in the deterministic case is rather complicated, and merely indicates that the sequence of weight vectors tends to converge to the desired solution. Instead of pursuing this topic further, we shall turn to a very similar rule that arises from a stochastic descent procedure. We note, however, that the solution need not give a separating vector, even if one exists, as shown in Fig. 5.17 (Computer exercise 10).

5.8.5 Stochastic Approximation Methods

All of the iterative descent procedures we have considered thus far have been described in deterministic terms. We are given a particular set of samples, and we generate a particular sequence of weight vectors. In this section we digress briefly to consider an MSE procedure in which the samples are drawn randomly, resulting in

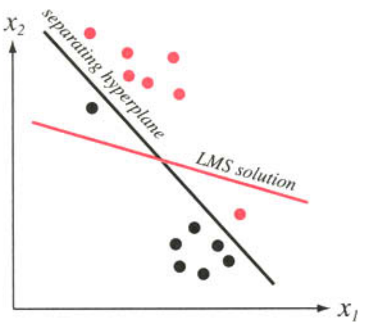


FIGURE 5.17. The LMS algorithm need not converge to a separating hyperplane, even if one exists. Because the LMS solution minimizes the sum of the squares of the distances of the training points to the hyperplane, for this example the plane is rotated clockwise compared to a separating hyperplane.

a random sequence of weight vectors. We will return in Chapter 7 to the theory of stochastic approximation though here some of the main ideas will be presented without proof.

Suppose that samples are drawn independently by selecting a state of nature with probability $P(\omega_i)$ and then selecting an $\mathbf{x}$ according to the probability law $p(\mathbf{x}|\omega_i)$. For each $\mathbf{x}$ we let θ be its *label*, with $\theta = +1$ if $\mathbf{x}$ is labeled ω_1 and $\theta = -1$ if $\mathbf{x}$ is labeled ω_2. Then the data consist of an infinite sequence of independent pairs $(\mathbf{x}, \theta_1)$, $(\mathbf{x}_2, \theta_2), \ldots, (\mathbf{x}_k, \theta_k), \ldots$. Even though the label variable θ is binary-valued, it can be thought of as a noisy version of the Bayes discriminant function $g_0(\mathbf{x})$. This follows from the observation that

$$P(\theta = 1|\mathbf{x}) = P(\omega_1|\mathbf{x}),$$

and

$$P(\theta = -1|\mathbf{x}) = P(\omega_2|\mathbf{x}),$$

so that the conditional mean of θ is given by

$$\mathcal{E}_{\theta|\mathbf{x}}[\theta] = \sum_{\theta} \theta P(\theta|\mathbf{x}) = P(\omega_1|\mathbf{x}) - P(\omega_2|\mathbf{x}) = g_0(\mathbf{x}). \tag{62}$$

Suppose that we wish to approximate $g_0(\mathbf{x})$ by the finite series expansion

$$g(\mathbf{x}) = \mathbf{a}^t\mathbf{y} = \sum_{i=1}^{\hat{d}} a_i y_i(\mathbf{x}),$$

where both the basis functions $y_i(\mathbf{x})$ and the number of terms $\hat{d}$ are known. Then we can seek a weight vector $\hat{\mathbf{a}}$ that minimizes the mean-squared approximation error

$$\epsilon^2 = \mathcal{E}[(\mathbf{a}^t\mathbf{y} - g_0(\mathbf{x}))^2]. \tag{63}$$

Minimization of ϵ^2 would appear to require knowledge of Bayes discriminant $g_0(\mathbf{x})$. However, as one might have guessed from the analogous situation in Section 5.8.3, it can be shown that the weight vector $\hat{\mathbf{a}}$ that minimizes ϵ^2 also minimizes the criterion function

$$J_m(\mathbf{a}) = \mathcal{E}[(\mathbf{a}^t\mathbf{y} - \theta)^2]. \tag{64}$$

This should also be plausible from the fact that θ is essentially a noisy version of $g_0(\mathbf{x})$. Because the gradient is

$$\nabla J_m = 2\mathcal{E}[(\mathbf{a}^t\mathbf{y} - \theta)\mathbf{y}], \tag{65}$$

we can obtain the closed-form solution

$$\hat{\mathbf{a}} = \mathcal{E}[\mathbf{y}\mathbf{y}^t]^{-1}\mathcal{E}[\theta\mathbf{y}]. \tag{66}$$

Thus, one way to use the samples is to estimate $\mathcal{E}[\mathbf{y}\mathbf{y}^t]$ and $\mathcal{E}[\theta\mathbf{y}]$, and use Eq. 66 to obtain the MSE-optimal linear discriminant. An alternative is to minimize $J_m(\mathbf{a})$ by a gradient descent procedure. Suppose that in place of the true gradient we substitute the noisy version $2(\mathbf{a}^t\mathbf{y}_k - \theta_k)\mathbf{y}_k$. This leads to the update rule

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(\theta_k - \mathbf{a}^t(k)\mathbf{y}_k)\mathbf{y}_k, \tag{67}$$

which is basically just the Widrow-Hoff rule. It can be shown (Problem 23) that if $\mathcal{E}[\mathbf{y}\mathbf{y}^t]$ is nonsingular and if the coefficients $\eta(k)$ satisfy

$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta(k) = +\infty \tag{68}$$

and

$$\lim_{m\to\infty} \sum_{k=1}^{m} \eta^2(k) < \infty \tag{69}$$

then $\mathbf{a}(k)$ converges to $\hat{\mathbf{a}}$ in mean square:

$$\lim_{k\to\infty} \mathcal{E}[\|\mathbf{a}(k) - \hat{\mathbf{a}}\|^2] = 0. \tag{70}$$

The reasons we need these conditions on $\eta(k)$ are simple. The first condition keeps the weight vector from converging so fast that a systematic error will remain forever uncorrected. The second condition ensures that random fluctuations are eventually suppressed. Both conditions are satisfied by the conventional choice $\eta(k) = 1/k$. Unfortunately, this kind of programmed decrease of $\eta(k)$, independent of the problem at hand, often leads to very slow convergence.

Of course, this is neither the only nor the best descent algorithm for minimizing J_m. For example, if we note that the matrix of second partial derivatives for J_m is given by

$$D = 2\mathcal{E}[\mathbf{y}\mathbf{y}^t],$$

we see that Newton's rule for minimizing J_m (Eq. 15) is

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathcal{E}[\mathbf{y}\mathbf{y}^t]^{-1}\mathcal{E}[(\theta - \mathbf{a}^t\mathbf{y})\mathbf{y}].$$

A stochastic analog of this rule is

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{R}_{k+1}(\theta_k - \mathbf{a}^t(k)\mathbf{y}_k)\mathbf{y}_k. \qquad (71)$$

with

$$\mathbf{R}_{k+1}^{-1} = \mathbf{R}_k^{-1} + \mathbf{y}_k\mathbf{y}_k^t, \qquad (72)$$

or, equivalently,*

$$\mathbf{R}_{k+1} = \mathbf{R}_k - \frac{\mathbf{R}_k\mathbf{y}_k(\mathbf{R}_k\mathbf{y}_k)^t}{1+\mathbf{y}_k^t\mathbf{R}_k\mathbf{y}_k}. \qquad (73)$$

This rule also produces a sequence of weight vectors that converges to the optimal solution in mean square. Its convergence is faster, but it requires more computation per step (Computer exercise 8).

These gradient procedures can be viewed as methods for minimizing a criterion function, or finding the zero of its gradient, in the presence of noise. In the statistical literature, functions such as J_m and ∇J_m that have the form $\mathcal{E}[f(\mathbf{a}, \mathbf{x})]$ are called *regression functions*, and the iterative algorithms are called *stochastic approximation procedures*. Two well-known ones are (1) the Kiefer-Wolfowitz procedure for minimizing a regression function, and (2) the Robbins-Monro procedure for finding a root of a regression function. Often the easiest way to obtain a convergence proof for a particular descent or approximation procedure is to show that it satisfies the convergence conditions for these more general procedures. Unfortunately, an exposition of these methods in their full generality would lead us rather far afield, and we must close this digression by referring the interested reader to the references in the Bibliography.

REGRESSION FUNCTION
STOCHASTIC APPROXIMATION

5.9 THE HO-KASHYAP PROCEDURES

The procedures we have considered thus far differ in several ways. The Perceptron and relaxation procedures find separating vectors if the samples are linearly separable, but do not converge on nonseparable problems. The MSE procedures yield a weight vector whether the samples are linearly separable or not, but there is no guarantee that this vector is a separating vector in the separable case (Fig. 5.17). If the margin vector $\mathbf{b}$ is chosen arbitrarily, all we can say is that the MSE procedures minimize $\|\mathbf{Ya} - \mathbf{b}\|^2$. Now if the training samples happen to be linearly separable, then there exists an $\hat{\mathbf{a}}$ and a $\hat{\mathbf{b}}$ such that

$$\mathbf{Y}\hat{\mathbf{a}} = \hat{\mathbf{b}} > 0,$$

where by $\hat{\mathbf{b}} > 0$, we mean that every component of $\hat{\mathbf{b}}$ is positive. Clearly, were we to take $\mathbf{b} = \hat{\mathbf{b}}$ and apply the MSE procedure, we would obtain a separating vector. Of course, we usually do not know $\hat{\mathbf{b}}$ beforehand. However, we shall now see how the MSE procedure can be modified to obtain both a separating vector $\mathbf{a}$ and a margin

*This recursive formula for computing R_k, which is roughly $(1/k)\mathcal{E}[\mathbf{yy}^t]^{-1}$, cannot be used if R_k is singular.

vector $\mathbf{b}$. The underlying idea comes from the observation that if the samples are separable, and if both $\mathbf{a}$ and $\mathbf{b}$ in the criterion function

$$J_s(\mathbf{a}, \mathbf{b}) = \|\mathbf{Ya} - \mathbf{b}\|^2 \tag{74}$$

are allowed to vary (subject to the constraint $\mathbf{b} > 0$), then the minimum value of J_s is zero, and the $\mathbf{a}$ that achieves that minimum is a separating vector.

5.9.1 The Descent Procedure

To minimize J_s in Eq. 74 we shall use a modified gradient descent procedure. The gradient of J_s with respect to $\mathbf{a}$ is given by

$$\nabla_{\mathbf{a}} J_s = 2\mathbf{Y}^t(\mathbf{Ya} - \mathbf{b}), \tag{75}$$

and the gradient of J_s with respect to $\mathbf{b}$ is given by

$$\nabla_{\mathbf{b}} J_s = -2(\mathbf{Ya} - \mathbf{b}). \tag{76}$$

For any value of $\mathbf{b}$, we can always take

$$\mathbf{a} = \mathbf{Y}^{\dagger}\mathbf{b}, \tag{77}$$

thereby obtaining $\nabla_{\mathbf{a}} J_s = \mathbf{0}$ and minimizing J_s with respect to $\mathbf{a}$ in one step. We are not so free to modify $\mathbf{b}$, however, because we must respect the constraint $\mathbf{b} > \mathbf{0}$, and we must avoid a descent procedure that converges to $\mathbf{b} = \mathbf{0}$. One way to prevent $\mathbf{b}$ from converging to zero is to start with $\mathbf{b} > \mathbf{0}$ and to refuse to reduce any of its components. We can do this and still try to follow the negative gradient if we first set all positive components of $\nabla_{\mathbf{b}} J_s$ to zero. Thus, if we let $|\mathbf{v}|$ denote the vector whose components are the magnitudes of the corresponding components of $\mathbf{v}$, we are led to consider an update rule for the margin of the form

$$\mathbf{b}(k+1) = \mathbf{b}(k) - \eta\frac{1}{2}[\nabla_{\mathbf{b}} J_s - |\nabla_{\mathbf{b}} J_s|]. \tag{78}$$

Using Eqs. 76 and 77, and being a bit more specific, we obtain the *Ho-Kashyap rule* for minimizing $J_s(\mathbf{a}, \mathbf{b})$:

$$\begin{aligned} &\mathbf{b}(1) > \mathbf{0} \qquad \text{but otherwise arbitrary} \\ &\mathbf{b}(k+1) = \mathbf{b}(k) + 2\eta(k)\mathbf{e}^{+}(k), \end{aligned} \tag{79}$$

where $\mathbf{e}(k)$ is the error vector

$$\mathbf{e}(k) = \mathbf{Ya}(k) - \mathbf{b}(k), \tag{80}$$

$\mathbf{e}^{+}(k)$ is the positive part of the error vector

$$\mathbf{e}^{+}(k) = \frac{1}{2}(\mathbf{e}(k) + |\mathbf{e}(k)|), \tag{81}$$

and

$$\mathbf{a}(k) = \mathbf{Y}^{\dagger}\mathbf{b}(k), \qquad k = 1, 2, \ldots. \tag{82}$$

Thus if we let b_{min} be a small convergence criterion and Abs[**e**] denote the positive part of **e**, our algorithm is as follows:

■ **Algorithm 11. (Ho-Kashyap)**

```
1 begin initialize a, b, $\eta(\cdot) < 1$, threshold $b_{min}$, $k_{max}$
2         do $k \leftarrow (k+1) \bmod n$
3            $\mathbf{e} \leftarrow \mathbf{Ya} - \mathbf{b}$
4            $\mathbf{e}^+ \leftarrow 1/2(\mathbf{e} + \text{Abs}[\mathbf{e}])$
5            $\mathbf{b} \leftarrow \mathbf{b} + 2\eta(k)\mathbf{e}^+$
6            $\mathbf{a} \leftarrow \mathbf{Y}^\dagger \mathbf{b}$
7            if Abs[e] $\le b_{min}$ then return a, b and exit
8         until $k = k_{max}$
9    print "NO SOLUTION FOUND"
10 end
```

Because the weight vector $\mathbf{a}(k)$ is completely determined by the margin vector $\mathbf{b}(k)$, this is basically an algorithm for producing a sequence of margin vectors. The initial vector $\mathbf{b}(1)$ is positive to begin with, and if $\eta > 0$, all subsequent vectors $\mathbf{b}(k)$ are positive. We might worry that if none of the components of $\mathbf{e}(k)$ is positive, so that $\mathbf{b}(k)$ stops changing, we might fail to find a solution. However, we shall see that in that case either $\mathbf{e}(k) = \mathbf{0}$ and we have a solution, or $\mathbf{e}(k) \le 0$ and we have proof that the samples are not linearly separable.

5.9.2 Convergence Proof

We shall now show that if the samples are linearly separable, and if $0 < \eta < 1$, then the Ho-Kashyap algorithm will yield a solution vector in a finite number of steps. To make the algorithm terminate, we should add a terminating condition stating that corrections cease once a solution vector is obtained or some large criterion number of iterations have occurred. However, it is mathematically more convenient to let the corrections continue and show that the error vector $\mathbf{e}(k)$ either becomes zero for some finite k, or converges to zero as k goes to infinity.

It is clear that either $\mathbf{e}(k) = \mathbf{0}$ for some k—say k_0—or there are no zero vectors in the sequence $\mathbf{e}(1), \mathbf{e}(2), \ldots$. In the first case, once a zero vector is obtained, no further changes occur to $\mathbf{a}(k)$, $\mathbf{b}(k)$, or $\mathbf{e}(k)$, and $\mathbf{Ya}(k) = \mathbf{b}(k) > \mathbf{0}$ for all $k \ge k_0$. Thus, if we happen to obtain a zero error vector, the algorithm automatically terminates with a solution vector.

Suppose now that $\mathbf{e}(k)$ is never zero for finite k. To see that $\mathbf{e}(k)$ must nevertheless converge to zero, we begin by asking whether or not we might possibly obtain an $\mathbf{e}(k)$ with no positive components. This would be most unfortunate, because we would have $\mathbf{Ya}(k) \le \mathbf{b}(k)$, and since $\mathbf{e}^+(k)$ would be zero, we would obtain no further changes in $\mathbf{a}(k)$, $\mathbf{b}(k)$, or $\mathbf{e}(k)$. Fortunately, this can never happen if the samples are linearly separable. A proof is simple and is based on the fact that if $\mathbf{Y}^t\mathbf{Ya}(k) = \mathbf{Y}^t\mathbf{b}$, then $\mathbf{Y}^t\mathbf{e}(k) = \mathbf{0}$. But if the samples are linearly separable, there exists an $\hat{\mathbf{a}}$ and a $\hat{\mathbf{b}} > \mathbf{0}$ such that

$$\mathbf{Y}\hat{\mathbf{a}} = \hat{\mathbf{b}}.$$

Thus,

$$\mathbf{e}^t(k)\mathbf{Y}\hat{\mathbf{a}} = 0 = \mathbf{e}^t(k)\hat{\mathbf{b}},$$

and because all the components of $\hat{\mathbf{b}}$ are positive, either $\mathbf{e}(k) = \mathbf{0}$ or at least one of the components of $\mathbf{e}(k)$ must be positive. Because we have excluded the case $\mathbf{e}(k) = \mathbf{0}$, it follows that $\mathbf{e}^+(k)$ cannot be zero for finite k.

The proof that the error vector always converges to zero exploits the fact that the matrix $\mathbf{YY}^\dagger$ is symmetric, positive semidefinite, and satisfies

$$(\mathbf{YY}^\dagger)^t(\mathbf{YY}^\dagger) = \mathbf{YY}^\dagger. \qquad (83)$$

Although these results are true in general, for simplicity we demonstrate them only for the case where $\mathbf{Y}^t\mathbf{Y}$ is nonsingular. In this case $\mathbf{YY}^\dagger = \mathbf{Y}(\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t$, and the symmetry is evident. Because $\mathbf{Y}^t\mathbf{Y}$ is positive definite, so is $(\mathbf{Y}^t\mathbf{Y})^{-1}$; thus, $\mathbf{bY}(\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t\mathbf{b} \geq \mathbf{0}$ for any $\mathbf{b}$, and $\mathbf{YY}^\dagger$ is at least positive semidefinite. Finally, Eq. 83 follows from

$$(\mathbf{YY}^\dagger)^t(\mathbf{YY}^\dagger) = [\mathbf{Y}(\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t][\mathbf{Y}(\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t].$$

To see that $\mathbf{e}(k)$ must converge to zero, we eliminate $\mathbf{a}(k)$ between Eqs. 80 and 82 and obtain

$$\mathbf{e}(k) = (\mathbf{YY}^\dagger - \mathbf{I})\mathbf{b}(k).$$

Then, using a constant learning rate and Eq. 79 we obtain the recursion relation

$$\begin{aligned}\mathbf{e}(k+1) &= (\mathbf{YY}^\dagger - \mathbf{I})(\mathbf{b}(k) + 2\eta\mathbf{e}^+(k))\\ &= \mathbf{e}(k) + 2\eta(\mathbf{YY}^\dagger - \mathbf{I})\mathbf{e}^+(k), \end{aligned} \qquad (84)$$

so that

$$\frac{1}{4}\|\mathbf{e}(k+1)\|^2 = \frac{1}{4}\|\mathbf{e}(k)\|^2 + \eta\mathbf{e}^t(k)(\mathbf{YY}^\dagger - \mathbf{I})\mathbf{e}^+(k) + \|\eta(\mathbf{YY}^\dagger - \mathbf{I})\mathbf{e}^+(k)\|^2.$$

Both the second and the third terms simplify considerably. Because $\mathbf{e}^t(k)\mathbf{Y} = 0$, the second term becomes

$$\eta\mathbf{e}^t(k)(\mathbf{YY}^\dagger - \mathbf{I})\mathbf{e}^+(k) = -\eta\mathbf{e}^t(k)\mathbf{e}^{+t}(k) = -\eta\|\mathbf{e}^+(k)\|^2,$$

the nonzero components of $\mathbf{e}^+(k)$ being the positive components of $\mathbf{e}(k)$. Because $\mathbf{YY}^\dagger$ is symmetric and is equal to $(\mathbf{YY}^\dagger)^t(\mathbf{YY}^\dagger)$, the third term simplifies to

$$\begin{aligned}\|\eta(\mathbf{YY}^\dagger - \mathbf{I})\mathbf{e}^+(k)\|^2 &= \eta^2\mathbf{e}^{+t}(k)(\mathbf{YY}^\dagger - \mathbf{I})^t(\mathbf{YY}^\dagger - \mathbf{I})\mathbf{e}^+(k)\\ &= \eta^2\|\mathbf{e}^+(k)\|^2 - \eta^2\mathbf{e}^+(k)\mathbf{YY}^\dagger\mathbf{e}^+(k),\end{aligned}$$

and thus we have

$$\frac{1}{4}(\|\mathbf{e}(k)\|^2 - \|\mathbf{e}(k+1)\|^2) = \eta(1-\eta)\|\mathbf{e}^+(k)\|^2 + \eta^2\mathbf{e}^{+t}(k)\mathbf{YY}^\dagger\mathbf{e}^+(k). \qquad (85)$$

Because $\mathbf{e}^+(k)$ is nonzero by assumption and because $\mathbf{YY}^\dagger$ is a positive semidefinite, $\|\mathbf{e}(k)\|^2 > \|\mathbf{e}(k+1)\|^2$ if $0 < \eta < 1$. Thus the sequence $\|\mathbf{e}(1)\|^2$, $\|\mathbf{e}(2)\|^2, \ldots$ is monotonically decreasing and must converge to some limiting value $\|\mathbf{e}\|^2$. But for convergence to take place, $\mathbf{e}^+(k)$ must converge to zero, so that all the positive components of $\mathbf{e}(k)$ must converge to zero. Because $\mathbf{e}^t(k)\hat{\mathbf{b}} = 0$ for all k, it follows that all of the components of $\mathbf{e}(k)$ must converge to zero. Thus, if $0 < \eta < 1$ and if the samples are linearly separable, $\mathbf{a}(k)$ will converge to a solution vector as k goes to infinity.

If we test the signs of the components of $\mathbf{Ya}(k)$ at each step and terminate the algorithm when they are all positive, we will in fact obtain a separating vector in a finite number of steps. This follows from the fact that $\mathbf{Ya}(k) = \mathbf{b}(k) + \mathbf{e}(k)$, and that the components of $\mathbf{b}(k)$ never decrease. Thus, if b_{min} is the smallest component of $\mathbf{b}(1)$ and if $\mathbf{e}(k)$ converges to zero, then $\mathbf{e}(k)$ must enter the hypersphere $\|\mathbf{e}(k)\| = b_{min}$ after a finite number of steps, at which point $\mathbf{Ya}(k) > 0$. Although we ignored terminating conditions to simplify the proof, such a terminating condition would always be used in practice.

5.9.3 Nonseparable Behavior

If the convergence proof just given is examined to see how the assumption of separability was employed, it will be seen that it was needed twice. First, the fact that $\mathbf{e}^t(k)\hat{\mathbf{b}} = 0$ was used to show that either $\mathbf{e}(k) = \mathbf{0}$ for some finite k, or $\mathbf{e}^+(k)$ is never zero and corrections go on forever. Second, this same constraint was used to show that if $\mathbf{e}^+(k)$ converges to zero, $\mathbf{e}(k)$ must also converge to zero.

If the samples are not linearly separable, it no longer follows that if $\mathbf{e}^+(k)$ is zero then $\mathbf{e}(k)$ must be zero. Indeed, on a nonseparable problem one may well obtain a nonzero error vector having no positive components. If this occurs, the algorithm automatically terminates and we have proof that the samples are not separable.

What happens if the patterns are not separable, but $\mathbf{e}^+(k)$ is never zero? In this case it still follows that

$$\mathbf{e}(k+1) = \mathbf{e}(k) + 2\eta(\mathbf{YY}^\dagger - \mathbf{I})\mathbf{e}^+(k) \tag{86}$$

and

$$\frac{1}{4}(\|\mathbf{e}(k)\|^2 - \|\mathbf{e}(k+1)\|^2) = \eta(1-\eta)\|\mathbf{e}^+(k)\|^2 + \eta^2\mathbf{e}^{+t}(k)\mathbf{YY}^\dagger\mathbf{e}^+(k). \tag{87}$$

Thus, the sequence $\|\mathbf{e}(1)\|^2$, $\|\mathbf{e}(2)\|^2$, ... must still converge, though the limiting value $\|\mathbf{e}\|^2$ cannot be zero. Convergence requires that $\mathbf{e}^+(k) = \mathbf{0}$ for some finite k, or $\mathbf{e}^+(k)$ converges to zero while $\|\mathbf{e}(k)\|$ is bounded away from zero. Thus, the Ho-Kashyap algorithm provides us with a separating vector in the separable case, and with evidence of nonseparability in the nonseparable case. However, there is no bound on the number of steps needed to disclose nonseparability.

5.9.4 Some Related Procedures

If we write $\mathbf{Y}^\dagger = (\mathbf{Y}^t\mathbf{Y})^{-1}\mathbf{Y}^t$ and make use of the fact that $\mathbf{Y}^t\mathbf{e}(k) = 0$, we can modify the Ho-Kashyap rule as follows

$$\begin{aligned}
\mathbf{b}(1) &> 0 \quad \text{but otherwise arbitrary} \\
\mathbf{a}(1) &= \mathbf{Y}^{\dagger}\mathbf{b}(1) \\
\mathbf{b}(k+1) &= \mathbf{b}(k) + \eta(\mathbf{e}(k) + |\mathbf{e}(k)|) \\
\mathbf{a}(k+1) &= \mathbf{a}(k) + \eta\mathbf{Y}^{\dagger}|\mathbf{e}(k)|,
\end{aligned} \tag{88}$$

where, as usual,

$$\mathbf{e}(k) = \mathbf{Y}\mathbf{a}(k) - \mathbf{b}(k). \tag{89}$$

This then gives the algorithm for fixed learning rate:

■ **Algorithm 12. (Modified Ho-Kashyap)**

```
begin initialize a, b, $\eta < 1$, threshold $b_{min}$, $k_{max}$
    do $k \leftarrow (k+1)$ mod $n$
        $\mathbf{e} \leftarrow \mathbf{Ya} - \mathbf{b}$
        $\mathbf{e}^{+} \leftarrow 1/2(\mathbf{e} + \text{Abs}[\mathbf{e}])$
        $\mathbf{b} \leftarrow \mathbf{b} + 2\eta(k)(\mathbf{e} + \text{Abs}[\mathbf{e}])$
        $\mathbf{a} \leftarrow \mathbf{Y}^{\dagger}\mathbf{b}$
        if Abs[e] $\leq b_{min}$ then return a, b and exit
    until $k = k_{max}$
  print "NO SOLUTION FOUND"
end
```

This algorithm differs from the Perceptron and relaxation algorithms for solving linear inequalities in at least three ways: (1) It varies both the weight vector **a** and the margin vector **b**, (2) it provides evidence of nonseparability, but (3) it requires the computation of the pseudoinverse of **Y**. Even though this last computation need be done only once, it can be time consuming, and it requires special treatment if $\mathbf{Y}^t\mathbf{Y}$ is singular. An interesting alternative algorithm that resembles Eq. 88 but avoids the need for computing $\mathbf{Y}^{\dagger}$ is

$$\begin{aligned}
\mathbf{b}(1) > 0 &\quad \text{but otherwise arbitrary}, \\
\mathbf{a}(1) &\quad \text{arbitrary}, \\
\mathbf{b}(k+1) &= \mathbf{b}(k) + (\mathbf{e}(k) + |\mathbf{e}(k)|), \\
\mathbf{a}(k+1) &= \mathbf{a}(k) + \eta\mathbf{R}\mathbf{Y}^t|\mathbf{e}(k)|,
\end{aligned} \tag{90}$$

where **R** is an arbitrary, constant, positive-definite $\hat{d}$-by-$\hat{d}$ matrix. We shall show that if η is properly chosen, this algorithm also yields a solution vector in a finite number of steps, provided that a solution exists. Furthermore, if no solution exists, the vector $\mathbf{Y}^t|\mathbf{e}(k)|$ either vanishes, exposing the nonseparability, or converges to zero.

The proof is fairly straightforward. Whether the samples are linearly separable or not, Eqs. 89 and 90 show that

$$\begin{aligned}
\mathbf{e}(k+1) &= \mathbf{Y}\mathbf{a}(k+1) - \mathbf{b}(k+1) \\
&= (\eta\mathbf{Y}\mathbf{R}\mathbf{Y}^t - \mathbf{I})|\mathbf{e}(k)|.
\end{aligned}$$

We can find, then, that the squared magnitude is

$$\|\mathbf{e}(k+1)\|^2 = |\mathbf{e}(k)|^t(\eta^2\mathbf{YRY}^t\mathbf{YRY} - 2\eta\mathbf{YRY}^t + \mathbf{I})|\mathbf{e}(k)|,$$

and, furthermore,

$$\|\mathbf{e}(k)\|^2 - \|\mathbf{e}(k+1)\|^2 = (\mathbf{Y}^t|\mathbf{e}(k)|)^t\mathbf{A}(\mathbf{Y}^t|\mathbf{e}(k)|), \tag{91}$$

where

$$\mathbf{A} = 2\eta\mathbf{R} - \eta^2\mathbf{RY}^t\mathbf{R}. \tag{92}$$

Clearly, if η is positive but sufficiently small, $\mathbf{A}$ will be approximately $2\eta\mathbf{R}$ and hence positive definite. Thus, if $\mathbf{Y}^t|\mathbf{e}(k)| \neq 0$ we will have $\|\mathbf{e}(k)\|^2 > \|\mathbf{e}(k+1)\|^2$.

At this point we must distinguish between the separable and the nonseparable case. In the separable case there exists an $\hat{\mathbf{a}}$ and a $\hat{\mathbf{b}} > 0$ satisfying $\mathbf{Y}\hat{\mathbf{a}} = \hat{\mathbf{b}}$. Thus, if $|\mathbf{e}(k)| \neq 0$,

$$|\mathbf{e}(k)|^t\mathbf{Y}\hat{\mathbf{a}} = |\mathbf{e}(k)|^t\hat{\mathbf{b}} > 0,$$

so that $\mathbf{Y}^t|\mathbf{e}(k)|$ cannot be zero unless $\mathbf{e}(k)$ is zero. Thus, the sequence $\|\mathbf{e}(1)\|^2$, $\|\mathbf{e}(2)\|^2, \ldots$ is monotonically decreasing and must converge to some limiting value $\|\mathbf{e}\|^2$. But for convergence to take place, $\mathbf{Y}^t|\mathbf{e}(k)|$ must converge to zero, which implies that $|\mathbf{e}(k)|$ and hence $\mathbf{e}(k)$ must converge to zero. Because $\mathbf{e}(k)$ starts out positive and never decreases, it follows that $\mathbf{a}(k)$ must converge to a separating vector. Moreover, by the same argument used before, a solution must actually be obtained after a finite number of steps.

In the nonseparable case, $\mathbf{e}(k)$ can neither be zero nor converge to zero. It may happen that $\mathbf{Y}^t|\mathbf{e}(k)| = 0$ at some step, which would provide proof of nonseparability. However, it is also possible for the sequence of corrections to go on forever. In this case, it again follows that the sequence $\|\mathbf{e}(1)\|^2$, $\|\mathbf{e}(2)\|^2, \ldots$ must converge to a limiting value $\|\mathbf{e}\|^2 \neq 0$, and that $\mathbf{Y}^t|\mathbf{e}(k)|$ must converge to zero. Thus, we again obtain evidence of nonseparability in the nonseparable case.

Before closing this discussion, let us look briefly at the question of choosing η and $\mathbf{R}$. The simplest choice for $\mathbf{R}$ is the identity matrix, in which case $\mathbf{A} = 2\eta\mathbf{I} - \eta^2\mathbf{Y}^t\mathbf{Y}$. This matrix will be positive definite, thereby ensuring convergence, if $0 < \eta < 2/\lambda_{max}$, where λ_{max} is the largest eigenvalue of $\mathbf{Y}^t\mathbf{Y}$. Because the trace of $\mathbf{Y}^t\mathbf{Y}$ is both the sum of the eigenvalues of $\mathbf{Y}^t\mathbf{Y}$ and the sum of the squares of the elements of $\mathbf{Y}$, one can use the pessimistic bound $\hat{d}\lambda_{max} \leq \sum_i \|\mathbf{y}_i\|^2$ in selecting a value for η.

A more interesting approach is to change η at each step, selecting that value that maximizes $\|\mathbf{e}(k)\|^2 - \|\mathbf{e}(k+1)\|^2$. Equations 91 and 92 give

$$\|\mathbf{e}(k)\|^2 - \|\mathbf{e}(k+1)\|^2 = |\mathbf{e}(k)|^t\mathbf{Y}(2\eta\mathbf{R} - \eta^2\mathbf{RY}^t\mathbf{YR})\mathbf{Y}^t|\mathbf{e}(k)|. \tag{93}$$

By differentiating with respect to η, we obtain the optimal value

$$\eta(k) = \frac{|\mathbf{e}(k)|^t\mathbf{YRY}^t|\mathbf{e}(k)|}{|\mathbf{e}(k)|^t\mathbf{YRY}^t\mathbf{YRY}^t|\mathbf{e}(k)|}, \tag{94}$$

which, for $\mathbf{R} = \mathbf{I}$, simplifies to

$$\eta(k) = \frac{\|\mathbf{Y}^t|\mathbf{e}(k)|\,\|^2}{\|\mathbf{Y}\mathbf{Y}^t|\mathbf{e}(k)|\,\|^2}. \tag{95}$$

This same approach can also be used to select the matrix $\mathbf{R}$. By replacing $\mathbf{R}$ in Eq. 93 by the symmetric matrix $\mathbf{R} + \delta\mathbf{R}$ and neglecting second-order terms, we obtain

$$\delta(\|\mathbf{e}(k)\|^2 - \|\mathbf{e}(k+1)\|^2) = |\mathbf{e}(k)|\mathbf{Y}[\delta\mathbf{R}^t(\mathbf{I} - \eta\mathbf{Y}^t\mathbf{Y}\mathbf{R}) + (\mathbf{I} - \eta\mathbf{R}\mathbf{Y}^t\mathbf{Y})\delta\mathbf{R}]\mathbf{Y}^t|\mathbf{e}(k)|.$$

Thus, the decrease in the squared error vector is maximized by the choice

$$\mathbf{R} = \frac{1}{\eta}(\mathbf{Y}^t\mathbf{Y})^{-1} \tag{96}$$

and because $\eta\mathbf{R}\mathbf{Y}^t = \mathbf{Y}^\dagger$, the descent algorithm becomes virtually identical with the original Ho-Kashyap algorithm.

*5.10 LINEAR PROGRAMMING ALGORITHMS

The Perceptron, relaxation and Ho-Kashyap procedures are basically gradient descent procedures for solving simultaneous linear inequalities. Linear programming techniques are procedures for maximizing or minimizing linear functions subject to linear equality or inequality constraints. This at once suggests that one might be able to solve linear inequalities by using them as the constraints in a suitable linear programming problem. In this section we shall consider two of several ways that this can be done. The reader need have no knowledge of linear programming to understand these formulations, though such knowledge would certainly be useful in applying the techniques.

5.10.1 Linear Programming

OBJECTIVE FUNCTION

A classical linear programming problem can be stated as follows: Find a vector $\mathbf{u} = (u_1, \ldots, u_m)^t$ that minimizes the linear (scalar) *objective function*

$$z = \boldsymbol{\alpha}^t\mathbf{u} \tag{97}$$

subject to the constraint

$$\mathbf{A}\mathbf{u} \geq \boldsymbol{\beta}, \tag{98}$$

SIMPLEX ALGORITHM

where $\boldsymbol{\alpha}$ is an m-by-1 *cost vector*, $\boldsymbol{\beta}$ is an l-by-1 vector, and $\mathbf{A}$ is an l-by-m matrix. The *simplex algorithm* is the classical iterative procedure for solving this problem (Fig. 5.18). For technical reasons, it requires the imposition of one more constraint, namely, $\mathbf{u} \geq 0$.

If we think of $\mathbf{u}$ as being the weight vector $\mathbf{a}$, this constraint is unacceptable, because in most cases the solution vector will have both positive and negative components. However, suppose that we write

$$\mathbf{a} \equiv \mathbf{a}^+ - \mathbf{a}^-, \tag{99}$$

where

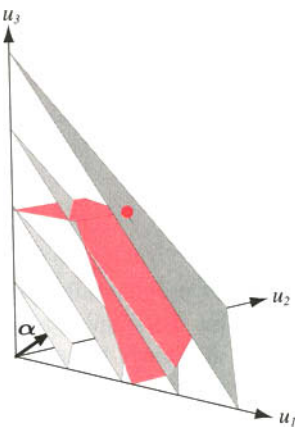


FIGURE 5.18. Surfaces of constant $z = \boldsymbol{\alpha}^t\mathbf{u}$ are shown in gray, while constraints of the form $\mathbf{Au} = \boldsymbol{\beta}$ are shown in red. The simplex algorithm finds an extremum of z given the constraints, that is, where the gray plane intersects the red at a single point.

$$\mathbf{a}^+ \equiv \frac{1}{2}(|\mathbf{a}| + \mathbf{a}) \tag{100}$$

and

$$\mathbf{a}^- \equiv \frac{1}{2}(|\mathbf{a}| - \mathbf{a}). \tag{101}$$

Then both $\mathbf{a}^+$ and $\mathbf{a}^-$ are nonnegative, and by identifying the components of $\mathbf{u}$ with the components of $\mathbf{a}^+$ and $\mathbf{a}^-$, for example, we can accept the constraint $\mathbf{u} \geq 0$.

5.10.2 The Linearly Separable Case

Suppose that we have a set of n samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$ and we want a weight vector $\mathbf{a}$ that satisfies $\mathbf{a}^t\mathbf{y}_i \geq b_i > 0$ for all i. How can we formulate this as a linear programming problem? One approach is to introduce what is called an *artificial variable* $\tau \geq 0$ by writing

$$\mathbf{a}^t\mathbf{y}_i + \tau \geq b_i.$$

If τ is sufficiently large, there is no problem in satisfying these constraints; for example, they are satisfied if $\mathbf{a} = \mathbf{0}$ and $\tau = \max_i b_i$.* However, this hardly solves our original problem. What we want is a solution with $\tau = 0$, which is the smallest value τ can have and still satisfy $\tau \geq 0$. Thus, we are led to consider the following problem: Minimize τ over all values of τ and $\mathbf{a}$ that satisfy the conditions $\mathbf{a}^t\mathbf{y}_i \geq b_i$ and $\tau \geq 0$. If the answer is zero, the samples are linearly separable, and we have a solution. If the answer is positive, there is no separating vector, but we have proof that the samples are nonseparable.

*In the terminology of linear programming, any solution satisfying the constraints is called a *feasible solution*. A feasible solution for which the number of nonzero variables does not exceed the number of constraints (not counting the simplex requirement for nonnegative variables) is called a *basic feasible solution*. Thus, the solution $\mathbf{a} = \mathbf{0}$ and $\tau = \max_i b_i$ is a basic feasible solution. Possession of such a solution simplifies the application of the simplex algorithm.

Formally, our problem is to find a vector $\mathbf{u}$ that minimizes the objective function $z = \boldsymbol{\alpha}^t\mathbf{u}$ subject to the constraints $\mathbf{A}\mathbf{u} \geq \boldsymbol{\beta}$ and $\mathbf{u} \geq 0$, where

$$\mathbf{A} = \begin{bmatrix} \mathbf{y}_1^t & -\mathbf{y}_1^t & 1 \\ \mathbf{y}_2^t & -\mathbf{y}_2^t & 1 \\ \vdots & \vdots & \vdots \\ \mathbf{y}_n^t & -\mathbf{y}_n^t & 1 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{a}^+ \\ \mathbf{a}^- \\ \tau \end{bmatrix}, \quad \boldsymbol{\alpha} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ 1 \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Thus, the linear programming problem involves $m = 2\hat{d} + 1$ variables and $l = n$ constraints, plus the simplex algorithm constraints $\mathbf{u} \geq 0$. The simplex algorithm will find the minimum value of the objective function $z = \boldsymbol{\alpha}^t\mathbf{u} = \tau$ in a finite number of steps, and it will exhibit a vector $\hat{\mathbf{u}}$ yielding that value. If the samples are linearly separable, the minimum value of τ will be zero, and a solution vector $\hat{\mathbf{a}}$ can be obtained from $\hat{\mathbf{u}}$. If the samples are not separable, the minimum value of τ will be positive. The resulting $\hat{\mathbf{u}}$ is usually not very useful as an approximate solution, but at least one obtains proof of nonseparability.

5.10.3 Minimizing the Perceptron Criterion Function

In the vast majority of pattern classification applications we cannot assume that the samples are linearly separable. In particular, when the patterns are not separable, one still wants to obtain a weight vector that classifies as many samples correctly as possible. Unfortunately, the number of errors is not a linear function of the components of the weight vector, and its minimization is not a linear programming problem. However, it turns out that the problem of minimizing the Perceptron criterion function can be posed as a problem in linear programming. Because minimization of this criterion function yields a separating vector in the separable case and a reasonable solution in the nonseparable case, this approach is quite attractive.

Recall from Section 5.5 that the basic Perceptron criterion function is given by

$$J_p(\mathbf{a}) = \sum_{\mathbf{y} \in \mathcal{Y}} (-\mathbf{a}^t\mathbf{y}), \tag{102}$$

where $\mathcal{Y}(\mathbf{a})$ is the set of training samples misclassified by $\mathbf{a}$. To avoid the useless solution $\mathbf{a} = \mathbf{0}$, we introduce a positive margin vector $\mathbf{b}$ and write

$$J_p'(\mathbf{a}) = \sum_{\mathbf{y} \in \mathcal{Y}'} (b_i - \mathbf{a}^t\mathbf{y}), \tag{103}$$

where $\mathbf{y}_i \in \mathcal{Y}'$ if $\mathbf{a}^t\mathbf{y}_i \leq b_i$. Clearly, J_p' is a piecewise-linear function of $\mathbf{a}$, not a linear function, and linear programming techniques are not immediately applicable. However, by introducing n artificial variables and their constraints, we can construct an equivalent linear objective function. Consider the problem of finding vectors $\mathbf{a}$ and $\boldsymbol{\tau}$ that minimize the linear function

$$z = \sum_{i=1}^{n} \tau_i$$

subject to the constraints

$$\tau_k \geq 0 \quad \text{and} \quad \tau_i \geq b_i - \mathbf{a}^t\mathbf{y}_i.$$

Of course for any fixed value of $\mathbf{a}$, the minimum value of z is exactly equal to $J'_p(\mathbf{a})$, since under these constraints the best we can do is to take $\tau_i = max[0, b_i - \mathbf{a}^t\mathbf{y}_i]$. If we minimize z over $\mathbf{t}$ *and* $\mathbf{a}$, we shall obtain the minimum possible value of $J'_p(\mathbf{a})$. Thus, we have converted the problem of minimizing $J'_p(\mathbf{a})$ to one of minimizing a linear function z subject to linear inequality constraints. Letting $\mathbf{u}_n$ denote an n-dimensional unit vector, we obtain the following problem with $m = 2\hat{d} + n$ variables and $l = n$ constraints: Minimize $\boldsymbol{\alpha}^t\mathbf{u}$ subject to $\mathbf{Au} \geq \boldsymbol{\beta}$ and $\mathbf{u} \geq 0$, where

$$\mathbf{A} = \begin{bmatrix} \mathbf{y}_1^t & -\mathbf{y}_1^t & 1 & 0 & \cdots & 0 \\ \mathbf{y}_2^t & -\mathbf{y}_2^t & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_n^t & -\mathbf{y}_n^t & 0 & 0 & \cdots & 1 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{a}^+ \\ \mathbf{a}^- \\ \boldsymbol{\tau} \end{bmatrix},$$

$$\boldsymbol{\alpha} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{1}_n \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

The choice $\mathbf{a} = \mathbf{0}$ and $\tau_i = b_i$ provides a basic feasible solution to start the simplex algorithm, and the simplex algorithm will provide an $\hat{\mathbf{a}}$ minimizing $J'_p(\mathbf{a})$ in a finite number of steps.

We have shown two ways to formulate the problem of finding a linear discriminant function as a problem in linear programming. There are other possible formulations, the ones involving the so-called *dual problem* being of particular interest from a computational standpoint. Generally speaking, methods such as the simplex method are merely sophisticated gradient descent methods for extremizing linear functions subject to linear constraints. The coding of a linear programming algorithm is usually more complicated than the coding of the simpler descent procedures we described earlier, and these descent procedures generalize naturally to multilayer neural networks. However, general purpose linear programming packages can often be used directly or modified appropriately with relatively little effort. When this can be done, one can secure the advantage of guaranteed convergence on both separable and nonseparable problems.

The various algorithms for finding linear discriminant functions presented in this chapter are summarized in Table 5.1. It is natural to ask which one is best, but none uniformly dominates or is uniformly dominated by all others. The choice depends upon such considerations as desired characteristics, ease of programming, the number of samples, and the dimensionality of the samples. If a linear discriminant function can yield a low error rate, any of these procedures, intelligently applied, can provide good performance.

*5.11 SUPPORT VECTOR MACHINES

We have seen how to train linear machines with margins. *Support vector machines* (SVMs) are motivated by many of the same considerations, but rely on preprocessing the data to represent patterns in a high dimension—typically much higher than the original feature space. With an appropriate nonlinear mapping $\varphi(\cdot)$ to a sufficiently high dimension, data from two categories can always be separated by a hyperplane

Table 5.1. Descent Procedures for Obtaining Linear Discriminant Functions

Name	Criterion	Algorithm	Conditions	Remarks
Fixed increment	$J_p = \sum_{\mathbf{a}^t\mathbf{y} \le 0} (-\mathbf{a}^t\mathbf{y})$	$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{y}^k$ $(\mathbf{a}^t(k)\mathbf{y}^k \le 0)$	—	Finite convergence if linearly separable to solution with $\mathbf{a}^t\mathbf{y} > 0$; $\mathbf{a}(k)$ always bounded.
Variable Increment	$J_p' = \sum_{\mathbf{a}^t\mathbf{y} \le b} -(\mathbf{a}^t\mathbf{y} - b)$	$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{y}^k$ $(\mathbf{a}^t(k)\mathbf{y}^k \le b)$	$\eta(k) \ge 0$ $\sum \eta(k) \to \infty$ $\frac{\sum \eta^2(k)}{(\sum \eta(k))^2} \to 0$	Converges if linearly separable to solution with $\mathbf{a}^t\mathbf{y} > b$; Finite convergence if $0 < \alpha \le \eta(k) \le \beta < \infty$.
Relaxation	$J_r = \frac{1}{2} \sum_{\mathbf{a}^t\mathbf{y} \le b} \frac{(\mathbf{a}^t\mathbf{y} - b)^2}{\|\mathbf{y}\|^2}$	$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta \frac{b - \mathbf{a}^t(k)\mathbf{y}^k}{\|\mathbf{y}^k\|^2} \mathbf{y}^k$ $(\mathbf{a}^t(k)\mathbf{y}^k \le b)$	$0 < \eta < 2$	Converges if linearly separable to solution with $\mathbf{a}^t\mathbf{y} \ge b$. If $b > 0$, finite convergence to solution with $\mathbf{a}^t\mathbf{y} > 0$.
Widrow-Hoff (LMS)	$J_s = \sum_i (\mathbf{a}^t\mathbf{y}_i - b_i)^2$	$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)(b_k - \mathbf{a}^t(k)\mathbf{y}^k)\mathbf{y}^k$	$\eta(k) > 0$ $\eta(k) \to 0$	Tends toward solution minimizing J_s.
Stochastic approximation	$J_m = \mathcal{E}\left[(\mathbf{a}^t\mathbf{y} - z)^2\right]$	$\mathbf{a}(k+1) =$ $\mathbf{a}(k) + \eta(k)(z_k - \mathbf{a}^t(k)\mathbf{y}^k)\mathbf{y}^k$	$\sum \eta(k) \to \infty$ $\sum \eta^2(k) \to L < \infty$	Involves an infinite number of randomly drawn samples; converges in mean square to a solution minimizing J_m and provides an MSE approximation to Bayes discriminant.
		$\mathbf{a}(k+1) =$ $\mathbf{a}(k) + \mathbf{R}(k)(z(k) - \mathbf{a}(k)^t\mathbf{y}^k)\mathbf{y}^k$	$\mathbf{R}^{-1}(k+1) = \mathbf{R}^{-1}(k) + \mathbf{y}_k\mathbf{y}_k^t$	
Pseudoinverse	$J_s = \|\mathbf{Y}\mathbf{a} - \mathbf{b}\|^2$	$\mathbf{a} = \mathbf{Y}^\dagger\mathbf{b}$	—	Classical MSE solution; special choices for $\mathbf{b}$ yield Fisher's linear discriminant and MSE approximation to Bayes discriminant.

Ho-Kashyap	$J_s = \|\mathbf{Ya} - \mathbf{b}\|^2$	$\mathbf{b}(k+1) = \mathbf{b}(k) + \eta(\mathbf{e}(k) + \|\mathbf{e}(k)\|)$ $\mathbf{e}(k) = \mathbf{Ya}(k) - \mathbf{b}(k)$ $\mathbf{a}(k) = \mathbf{Y}^\dagger \mathbf{b}(k)$	$0 < \eta < 1$ $\mathbf{b}(1) > 0$	$\mathbf{a}(k)$ is MSE solution for each $\mathbf{b}(k)$; finite convergence if linearly separable; if $\mathbf{e}(k) \leq 0$ but $\mathbf{e}(k) \neq 0$, the samples are nonseparable.
		$\mathbf{b}(k+1) = \mathbf{b}(k) + \eta(\mathbf{e}(k) + (\|\mathbf{e}(k)\|)$ $\mathbf{a}(k+1) = \mathbf{a}(k) + \eta \mathbf{RY}^t \|\mathbf{e}(k)\|$	$\eta(k) = \frac{\|\mathbf{e}(k)\|^t \mathbf{YRY}^t \|\mathbf{e}(k)\|}{\|\mathbf{e}(k)\|^t \mathbf{YRY}^t \mathbf{YRY}^t \|\mathbf{e}(k)\|}$ is optimum; $\mathbf{R}$ symmetric positive definite; $\mathbf{b}(1) > 0$	Finite convergence if linearly separable; if $\mathbf{Y}^t \|\mathbf{e}(k)\| = 0$, but $\mathbf{e}(k) \neq 0$, the samples are nonseparable.
Linear Programming	$\tau = \max\limits_{\mathbf{a}^t \mathbf{y}_i \leq b_i} [-(\mathbf{a}^t \mathbf{y}_i - b_i)]$	Simplex algorithm	$\mathbf{a}^t \mathbf{y}_i + \tau \geq b_i$ $\tau \geq 0$	Finite convergence in both separable and nonseparable cases; useful solution only if separable.
	$J'_p = \sum\limits_{i=1}^{n} \tau_i$ $= \sum\limits_{\mathbf{a}^t \mathbf{y}_i \leq b_i} -(\mathbf{a}^t \mathbf{y}_i - b_i)$	Simplex algorithm	$\mathbf{a}^t \mathbf{y}_i + \tau \geq b_i$ $\tau_i \geq 0$	Finite convergence in both separable and nonseparable cases; useful solution only if separable.

(Problem 29). Here we assume each pattern $\mathbf{x}_k$ has been transformed to $\mathbf{y}_k = \varphi(\mathbf{x}_k)$; we return to the choice of $\varphi(\cdot)$ below. For each of the n patterns, $k = 1, 2, \ldots, n$, we let $z_k = \pm 1$, according to whether pattern k is in ω_1 or ω_2. A linear discriminant in an augmented $\mathbf{y}$ space is

$$g(\mathbf{y}) = \mathbf{a}^t\mathbf{y}, \tag{104}$$

where both the weight vector and the transformed pattern vector are augmented (by $a_0 = w_0$ and $y_0 = 1$, respectively). Thus a separating hyperplane ensures

$$z_k g(\mathbf{y}_k) \geq 1, \qquad k = 1, \ldots, n, \tag{105}$$

much as was shown in Fig. 5.8.

In Section 5.9, the margin was any positive distance from the decision hyperplane. The goal in training a Support Vector Machine is to find the separating hyperplane with the *largest* margin; we expect that the larger the margin, the better generalization of the classifier. As illustrated in Fig. 5.2, the distance from any hyperplane to a (transformed) pattern $\mathbf{y}$ is $|g(\mathbf{y})|/||\mathbf{a}||$, and assuming that a positive margin b exists, Eq. 105 implies

$$\frac{z_k g(\mathbf{y}_k)}{||\mathbf{a}||} \geq b, \qquad k = 1, \ldots, n; \tag{106}$$

the goal is to find the weight vector $\mathbf{a}$ that maximizes b. Of course, the solution vector can be scaled arbitrarily and still preserve the hyperplane, and thus to ensure uniqueness we impose the constraint $b\ ||\mathbf{a}|| = 1$; that is, we demand the solution to Eqs. 104 and 105 also minimize $||\mathbf{a}||^2$.

SUPPORT VECTOR

The *support vectors* are the (transformed) training patterns for which Eq. 105 represents an equality—that is, the support vectors are (equally) close to the hyperplane (Fig. 5.19). The support vectors are the training samples that define the optimal separating hyperplane and are the most difficult patterns to classify. Informally speaking, they are the patterns most informative for the classification task.

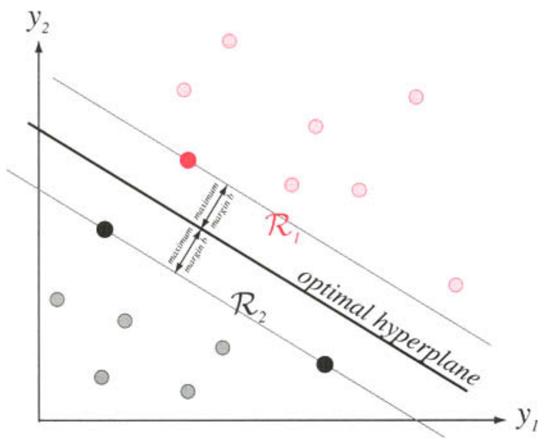


FIGURE 5.19. Training a support vector machine consists of finding the optimal hyperplane, that is, the one with the maximum distance from the nearest training patterns. The support vectors are those (nearest) patterns, a distance b from the hyperplane. The three support vectors are shown as solid dots.

A conceptually simple method of training a support vector machine is based on a small modification to the familiar Perceptron training rule (Algorithm 4). Recall that the Perceptron learning rule required merely updating the weight vector by an amount proportional to *any* randomly chosen misclassified pattern. A support vector machine can, instead, be trained by choosing the current *worst*-classified pattern. During most of the training period, such a pattern is one on the wrong side of the current decision boundary, farthest from that boundary. At the end of the training period, such a pattern will be one of the support vectors (Problem 31).

Alas, in practice, finding the worst-case pattern is computationally expensive, since for each update we must search through the entire training set to find the worst-classified pattern. As such, this simple method is used only for small problems. Before we return to the more common method of training an SVM having many patterns, we consider the error of such a classifier.

If N_s denotes the total number of support vectors, then for n training patterns the expected value of the generalization error rate is bounded, according to

$$\mathcal{E}_n[error] \leq \frac{\mathcal{E}_n[N_s]}{n}, \tag{107}$$

where the expectation is over all training sets of size n drawn from the (stationary) distributions describing the categories. This bound is independent of the dimensionality of the space of transformed vectors, determined by $\varphi(\cdot)$. We will return to this topic in Chapter 9, but for now we can understand this informally by means of the *leave-one-out bound.* Suppose we have n points in the training set, we train an SVM on $n-1$ of them, and we test on the single remaining point. If that remaining point happens to be a support vector for the full n sample case, then there will be an error; otherwise, there will not. Note that if we can find a transformation $\varphi(\cdot)$ that well separates the data—so the expected number of support vectors is small—then Eq. 107 shows that the expected error rate will be lower.

LEAVE-ONE-OUT BOUND

5.11.1 SVM Training

We now turn to the problem of training an SVM. The first step is, of course, to choose the nonlinear φ-functions that map the input to a higher-dimensional space. Often this choice will be informed by the designer's knowledge of the problem domain. In the absence of such information, one might choose to use polynomials, Gaussians, or yet other basis functions. The dimensionality of the mapped space can be arbitrarily high (though in practice it may be limited by computational resources).

We begin by recasting the problem of minimizing the magnitude of the weight vector constrained by the separation into an unconstrained problem by the method of Lagrange undetermined multipliers. Thus from Eq. 106 and our goal of minimizing $||\mathbf{a}||$, we construct the functional

$$L(\mathbf{a}, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{a}||^2 - \sum_{k=1}^{n} \alpha_k [z_k \mathbf{a}^t \mathbf{y}_k - 1] \tag{108}$$

and seek to minimize $L()$ with respect to the weight vector $\mathbf{a}$, and maximize it with respect to the undetermined multipliers $\alpha_k \geq 0$. The last term in Eq. 108 expresses the goal of classifying the points correctly. It can be shown using the so-called Kuhn-Tucker construction (Problem 33) that this optimization can be reformulated as maximizing

$$L(\boldsymbol{\alpha}) = \sum_{k=1}^{n} \alpha_i - \frac{1}{2} \sum_{k,j}^{n} \alpha_k \alpha_j z_k z_j \mathbf{y}_j^t \mathbf{y}_k, \tag{109}$$

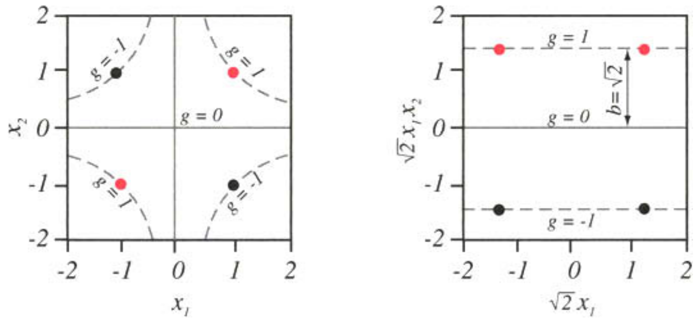
subject to the constraints

$$\sum_{k=1}^{n} z_k \alpha_k = 0 \qquad \alpha_k \geq 0, \qquad k = 1, \ldots, n, \tag{110}$$

given the training data. While these equations can be solved using quadratic programming, a number of alternate schemes have been devised (cf. Bibliography).

EXAMPLE 2 SVM for the XOR Problem

The exclusive-OR is the simplest problem that cannot be solved using a linear discriminant operating directly on the features. The points $k = 1, 3$ at $\mathbf{x} = (1, 1)^t$ and $(-1, -1)^t$ are in category ω_1 (red in the figure), while $k = 2, 4$ at $\mathbf{x} = (1, -1)^t$ and $(-1, 1)^t$ are in ω_2 (black in the figure). Following the approach of SVMs, we preprocess the features to map them to a higher dimension space where they can be linearly separated. While many φ-functions could be used, here we use the simplest expansion up to second order: 1, $\sqrt{2}x_1$, $\sqrt{2}x_2$, $\sqrt{2}x_1x_2$, x_1^2 and x_2^2, where the $\sqrt{2}$ is convenient for normalization.



The XOR problem in the original x_1x_2-space is shown at the left; the two red patterns are in category ω_1 and the two black ones in ω_2. These four training patterns $\mathbf{x}$ are mapped to a six-dimensional space by 1, $\sqrt{2}x_1$, $\sqrt{2}x_2$, $\sqrt{2}x_1x_2$, x_1^2 and x_2^2. In this space, the optimal hyperplane is found to be $g(x_1, x_2) = x_1x_2 = 0$ and the margin is $b = \sqrt{2}$. A two-dimensional projection of this space is shown at the right. The hyperplanes through the support vectors are $\sqrt{2}x_1x_2 = \pm 1$, and correspond to the hyperbolas $x_1x_2 = \pm 1$ in the original feature space, as shown.

We seek to maximize Eq. 109,

$$\sum_{k=1}^{4} \alpha_k - \frac{1}{2} \sum_{k,j}^{n} \alpha_k \alpha_j z_k z_j \mathbf{y}_j^t \mathbf{y}_k$$

subject to the constraints (Eq. 110)

$$\alpha_1 - \alpha_2 + \alpha_3 - \alpha_4 = 0$$
$$0 \leq \alpha_k \qquad k = 1, 2, 3, 4.$$

It is clear from the symmetry of the problem that $\alpha_1 = \alpha_3$ and that $\alpha_2 = \alpha_4$ at the solution. While we could use iterative gradient descent as described in Section 5.9, for this small problem we can use analytic techniques instead. The solution is $a_k^* = 1/8$, for $k = 1, 2, 3, 4$, and from the last term in Eq. 108 this implies that all four training patterns are support vectors—an unusual case due to the highly symmetric nature of the XOR problem.

The final discriminant function is $g(\mathbf{x}) = g(x_1, x_2) = x_1 x_2$, and the decision hyperplane is defined by $g = 0$, which properly classifies all training patterns. The margin is easily computed from the solution $||\mathbf{a}||$ and is found to be $b = 1/||\mathbf{a}|| = \sqrt{2}$. The figure at the right shows the margin projected into two dimensions of the five-dimensional transformed space. Problem 30 asks you to consider this margin as viewed in other two-dimensional projected subspaces.

An important benefit of the support vector machine approach is that the complexity of the resulting classifier is characterized by the number of support vectors rather than the dimensionality of the transformed space. As a result, SVMs tend to be less prone to problems of overfitting than some other methods.

5.12 MULTICATEGORY GENERALIZATIONS

There is no uniform way to extend all of the two-category procedures we have discussed to the multicategory case. In Section 5.2.2 we defined a multicategory classifier called a linear machine which classifies a pattern by computing c linear discriminant functions

$$g_i(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_{i0} \qquad i = 1, \ldots, c,$$

and assigning $\mathbf{x}$ to the category corresponding to the largest discriminant. This is a natural generalization for the multiclass case, particularly in view of the results of Chapter 2 for the multivariate normal problem. It can be extended simply to generalized linear discriminant functions by letting $\mathbf{y}(\mathbf{x})$ be a $\hat{d}$-dimensional vector of functions of $\mathbf{x}$, and by writing

$$g_i(\mathbf{x}) = \mathbf{a}_i^t\mathbf{y} \qquad i = 1, \ldots, c, \tag{111}$$

where again $\mathbf{x}$ is assigned to ω_i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$.

The generalization of our procedures from a two-category linear classifier to a multicategory linear machine is simplest in the linearly-separable case. Suppose that we have a set of labeled samples $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$, with n_1 in the subset $\mathcal{Y}_1$ labeled ω_1, n_2 in the subset $\mathcal{Y}_2$ labeled $\omega_2, \ldots$, and n_c in the subset $\mathcal{Y}_c$ labeled ω_c. We say that this set is linearly separable if there exists a linear machine that classifies all of them correctly. That is, if these samples are linearly separable, then there exists a set of weight vectors $\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_c$ such that if $\mathbf{y}_k \in \mathcal{Y}_i$, then

$$\hat{\mathbf{a}}_i^t\mathbf{y}_k > \hat{\mathbf{a}}_j^t\mathbf{y}_k \tag{112}$$

for all $j \neq i$.

5.12.1 Kesler's Construction

One of the pleasant things about this definition in Eq. 112 is that it is possible to manipulate these inequalities and reduce the multicategory problem to the two-category case. Suppose for the moment that $\mathbf{y}_k \in \mathcal{Y}_1$, so that Eq. 112 becomes

$$\hat{\mathbf{a}}_1^t\mathbf{y}_k - \hat{\mathbf{a}}_j^t\mathbf{y}_k > 0, \quad j = 2, \ldots, c. \tag{113}$$

This set of $c - 1$ inequalities can be thought of as requiring that the $c\hat{d}$-dimensional weight vector

$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_c \end{bmatrix}$$

correctly classifies all $c - 1$ of the $c\hat{d}$-dimensional sample sets

$$\boldsymbol{\eta}_{12} = \begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\eta}_{13} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \\ -\mathbf{y} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad \ldots, \quad \boldsymbol{\eta}_{1c} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ -\mathbf{y} \end{bmatrix}.$$

In other words, each $\boldsymbol{\eta}_{1j}$ corresponds to "normalizing" the patterns in ω_1 and ω_j. More generally, if $\mathbf{y} \in \mathcal{Y}_i$, we construct $(c-1)c\ \hat{d}$-dimensional training samples $\boldsymbol{\eta}_{ij}$ by partitioning $\boldsymbol{\eta}_{ij}$ into $c\ \hat{d}$-dimensional subvectors, with the ith subvector being $\mathbf{y}$, the jth being $-\mathbf{y}$, and all others being zero. Clearly, if $\hat{\boldsymbol{\alpha}}^t\boldsymbol{\eta}_{ij} > 0$ for $j \neq i$, then the linear machine corresponding to the components of $\hat{\boldsymbol{\alpha}}$ classifies $\mathbf{y}$ correctly.

This so-called Kesler construction multiplies the dimensionality of the data by c and the number of samples by $c - 1$, which does not make its direct use attractive. Its importance resides in the fact that it allows us to convert many multicategory error-correction procedures to two-category procedures for the purpose of obtaining a convergence proof.

5.12.2 Convergence of the Fixed-Increment Rule

We now use use Kesler's construction to prove convergence for a generalization of the fixed-increment rule for a linear machine. Suppose that we have a set of n linearly-separable samples $\mathbf{y}_1, \ldots, \mathbf{y}_n$, and we use them to form an infinite sequence in which every sample appears infinitely often. Let L_k denote the linear machine whose weight vectors are $\mathbf{a}_1(k), \ldots, \mathbf{a}_c(k)$. Starting with an arbitrary initial linear machine L_1, we want to use the sequence of samples to construct a sequence of linear machines that converges to a solution machine, one that classifies all of the samples correctly. We shall propose an error-correction rule in which weight changes are made if and only if the present linear machine misclassifies a sample. Let $\mathbf{y}^k$ denote the kth sample requiring correction, and suppose that $\mathbf{y}^k \in \mathcal{Y}_i$. Because $\mathbf{y}^k$ requires correction, there must be at least one $j \neq i$ for which

$$\mathbf{a}_i^t(k)\mathbf{y}^k \leq \mathbf{a}_j^t(k)\mathbf{y}^k. \tag{114}$$

Then the fixed-increment rule for correcting L_k is

$$\begin{aligned} \mathbf{a}_i(k+1) &= \mathbf{a}_i(k) + \mathbf{y}^k \\ \mathbf{a}_j(k+1) &= \mathbf{a}_j(k) - \mathbf{y}^k \\ \mathbf{a}_l(k+1) &= \mathbf{a}_l(k), \ l \neq i \text{ and } l \neq j. \end{aligned} \tag{115}$$

That is, the weight vector for the desired category is incremented by the pattern, the weight vector for the incorrectly chosen category is decremented, and all other weights are left unchanged (Problem 36, Computer exercise 12).

We shall now show that this rule must lead to a solution machine after a finite number of corrections. The proof is simple. For each linear machine L_k there corresponds a weight vector

$$\boldsymbol{\alpha}_k = \begin{bmatrix} \mathbf{a}_1(k) \\ \vdots \\ \mathbf{a}_c(k) \end{bmatrix}.$$

For each sample $\mathbf{y} \in \mathcal{Y}_i$ there are $c - 1$ samples $\boldsymbol{\eta}_{ij}$ formed as described in Section 5.12.1. In particular, corresponding to the vector $\mathbf{y}^k$ satisfying Eq. 114 there is a vector

$$\boldsymbol{\eta}_{ij}^k = \begin{array}{c} \left[\begin{array}{c} \cdot \\ \cdot \\ \cdot \\ \mathbf{y}^k \\ \cdot \\ \cdot \\ \cdot \\ -\mathbf{y}^k \\ \cdot \\ \cdot \\ \cdot \end{array}\right] \end{array} \begin{array}{c} \\ \\ \\ \leftarrow i \\ \\ \\ \\ \leftarrow j \\ \\ \\ \\ \end{array}$$

satisfying

$$\boldsymbol{\alpha}^t(k)\boldsymbol{\eta}_{ij}^k \leq 0.$$

Furthermore, the fixed-increment rule for correcting L_k is the fixed-increment rule for correcting $\boldsymbol{\alpha}(k)$, namely,

$$\boldsymbol{\alpha}(k+1) = \boldsymbol{\alpha}(k) + \boldsymbol{\eta}_{ij}^k.$$

Thus, we have obtained a complete correspondence between the multicategory case and the two-category case, in which the multicategory procedure produces a sequence of samples $\boldsymbol{\eta}^1, \boldsymbol{\eta}^2, \ldots, \boldsymbol{\eta}^k, \ldots$ and a sequence of weight vectors $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_k, \ldots$ By our results for the the two-category case, this latter sequence cannot be infinite, but must terminate in a solution vector. Hence, the sequence $L_1, L_2, \ldots, L_k, \ldots$ must terminate in a solution machine after a finite number of corrections.

This use of Kesler's construction to establish equivalences between multicategory and two-category procedures is a powerful theoretical tool. It can be used to extend all of our results for the Perceptron and relaxation procedures to the multicategory case, and it applies as well to the error-correction rules for the method of potential functions (Problem 38). Unfortunately, it is not as directly useful in generalizing the MSE or the linear programming approaches.

5.12.3 Generalizations for MSE Procedures

Perhaps the simplest way to obtain a natural generalization of the MSE procedures to the multiclass case is to consider the problem as a set of c two-class problems. The ith problem is to obtain a weight vector $\mathbf{a}_i$ that is minimum-squared-error solution to the equations

$$\begin{aligned} \mathbf{a}_i^t\mathbf{y} &= \ \ 1 && \text{for all } \mathbf{y} \in \mathcal{Y}_i \\ \mathbf{a}_i^t\mathbf{y} &= -1 && \text{for all } \mathbf{y} \notin \mathcal{Y}_i. \end{aligned}$$

In light of the results of Section 5.8.3, we see that if the number of samples is very large we will obtain a minimum mean-squared-error approximation to the Bayes discriminant function

$$P(\omega_i|\mathbf{x}) - P(\text{not } \omega_i|\mathbf{x}) = 2P(\omega_i|\mathbf{x}) - 1.$$

This observation has two immediate consequences. First, it suggests a modification in which we seek a weight vector $\mathbf{a}_i$ that is a minimum-squared-error solution to the equations

$$\begin{aligned} \mathbf{a}_i^t\mathbf{y} &= 1 && \text{for all } \mathbf{y} \in \mathcal{Y}_i \\ \mathbf{a}_i^t\mathbf{y} &= 0 && \text{for all } \mathbf{y} \notin \mathcal{Y}_i \end{aligned} \tag{116}$$

so that $\mathbf{a}^t\mathbf{y}$ will be a minimum mean-squared-error approximation to $P(\omega_i|\mathbf{x})$. Second, it justifies the use of the resulting discriminant functions in a linear machine, in which we assign $\mathbf{y}$ to ω_i if $\mathbf{a}_i^t\mathbf{y} > \mathbf{a}_j^t\mathbf{y}$ for all $j \neq i$.

The pseudoinverse solution to the multiclass MSE problem can be written in a form analogous to the form for the two-class case. Let $\mathbf{Y}$ be the n-by-$\hat{d}$ matrix of training samples, which we assume to be partitioned as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \vdots \\ \mathbf{Y}_c \end{bmatrix}, \tag{117}$$

with the samples labeled ω_i comprising the rows of $\mathbf{Y}_i$. Similarly, let $\mathbf{A}$ be the $\hat{d}$-by-c matrix of weight vectors

$$\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_c], \tag{118}$$

and let $\mathbf{B}$ be the n-by-c matrix

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_c \end{bmatrix}, \tag{119}$$

where all of the elements of $\mathbf{B}_i$ are zero except for those in the ith column, which are unity. Then the trace of the "squared" error matrix $(\mathbf{YA}-\mathbf{B})^t(\mathbf{YA}-\mathbf{B})$ is minimized by the solution*

$$\mathbf{A} = \mathbf{Y}^\dagger\mathbf{B}, \tag{120}$$

where, as usual, $\mathbf{Y}^\dagger$ is the pseudoinverse of $\mathbf{Y}$.

This result can be generalized in a theoretically interesting fashion. Let λ_{ij} be the loss incurred for deciding ω_i when the true state of nature is ω_j, and let the jth submatrix of $\mathbf{B}$ be given by

$$\mathbf{B}_j = -\begin{bmatrix} \lambda_{1j} & \lambda_{2j} & \cdots & \lambda_{cj} \\ \lambda_{1j} & \lambda_{2j} & \cdots & \lambda_{cj} \\ \vdots & & & \vdots \\ \lambda_{1j} & \lambda_{2j} & \cdots & \lambda_{cj} \end{bmatrix} \Bigg\updownarrow n_j \qquad j = 1, \ldots, c. \tag{121}$$

Then, as the number of samples approaches infinity, the solution $\mathbf{A} = \mathbf{Y}^\dagger\mathbf{B}$ yields discriminant functions $\mathbf{a}_i^t\mathbf{y}$ that provide a minimum-mean-square-error approximation to the Bayes discriminant function

$$g_{0i} = -\sum_{j=1}^{c} \lambda_{ij} P(\omega_i|\mathbf{x}). \tag{122}$$

The proof of this is a direct extension of the proof given in Section 5.8.3 (Problem 37).

SUMMARY

This chapter considers discriminant functions that are a linear function of a set of parameters, generally called weights. In all two-category cases, such discriminants lead to hyperplane decision boundaries, either in the feature space itself or in a space where the features have been mapped by a nonlinear function (general linear discriminants).

In broad overview, techniques such as the Perceptron algorithm adjust the parameters to increase the inner product with patterns in category ω_1 and decrease the inner product with those in ω_2. A very general approach is to form some criterion function and perform gradient descent. Different criterion functions have different strengths and weaknesses related to computation and convergence, none uniformly dominates the others. One can use linear algebra to solve for the weights (parameters) directly, by means of pseudoinverse matrices for small problems.

In support vector machines, the input is mapped by a nonlinear function to a high-dimensional space, and the optimal hyperplane found the one that has the largest margin. The support vectors are those (transformed) patterns that determine the margin; they are informally the hardest patterns to classify, and the most informative ones for designing the classifier. An upper bound on expected error rate of the classifier depends linearly upon the expected number of support vectors.

*If we let $\mathbf{b}_i$ denote the ith column of $\mathbf{B}$, the trace of $(\mathbf{YA}-\mathbf{B})^t(\mathbf{YA}-\mathbf{B})$ is equal to the sum of the squared lengths of the error vectors $\mathbf{Ya}_i - \mathbf{b}_i$. The solution $\mathbf{A} = \mathbf{Y}^\dagger\mathbf{B}$ not only minimizes this sum, but also minimizes each term in the sum.

For multicategory problems, the linear machines create decision boundaries consisting of sections of such hyperplanes. One can prove convergence of multicategory algorithms by first converting them to two-category algorithms and using the two-category proofs. The simplex algorithm finds the optimum of a linear function subject to (inequality) constraints, and it can be used for training linear classifiers.

Linear discriminants, while useful, are not sufficiently general for arbitrary challenging pattern recognition problems (such as those involving multimodal or nonconvex densities) unless an appropriate nonlinear mapping (φ function) can be found. In this chapter we have not considered any principled approaches to choosing such functions, but we turn to that topic in Chapter 6.

BIBLIOGRAPHICAL AND HISTORICAL REMARKS

Because linear discriminant functions are so amenable to analysis, far more papers have been written about them than the subject deserves. Historically, all of this work begins with the classic paper by Ronald A. Fisher [5]. The application of linear discriminant function to pattern classification was well described in reference [9], which posed the problem of optimal (minimum-risk) linear discriminant and proposed plausible gradient descent procedures to determine a solution from samples. Unfortunately, little can be said about such procedures without knowing the underlying distributions, and even then the situation is analytically complex. The design of multicategory classifiers using two-category procedures stems from reference [16]. Minsky and Papert's **Perceptrons** [13] was influential in pointing out the weaknesses of linear classifiers—weaknesses that were overcome by the methods we shall study in Chapter 6. The Winnow algorithms [10] in the error-free case [7, 11] and subsequent work in the general case have been useful in the computational learning community, as they allow one to derive convergence bounds.

While this work was statistically oriented, many of the pattern recognition papers that appeared in the late 1950s and early 1960s adopted other viewpoints. One viewpoint was that of neural networks, in which individual neurons were modeled as threshold elements, two-category linear machines—work that had its origins in the famous paper by McCulloch and Pitts [12].

As linear machines have been applied to larger and larger data sets in higher and higher dimensions, the computational burden of linear programming [1] has made this approach less popular. Stochastic approximation methods have been applied to this problem, as described in reference [21].

Early papers on the key ideas in Support Vector Machines are references [2] and [15]. A more extensive treatment, including complexity control, can be found in references [18] and [14]—material we shall visit in Chapter 9. A clear presentation of the method is [4], which provided the inspiration behind our Example 2. Guyon and Stork provide an overview of the relationships among linear classifiers, including Support Vector Machines [8]. The Kuhn-Tucker construction, used in the SVM training method described in the text and explored in Problem 33, is from reference [6] and is used in reference [17]. The fundamental result is that exactly one of the following three cases holds. (1) The original (primal) conditions have an optimal solution; in that case the dual cases do too, and their objective values are equal, or (2) the primal conditions are infeasible; in that case the dual is either unbounded or itself infeasible, or (3) the primal conditions are unbounded; in that case the dual is infeasible.

PROBLEMS

Section 5.2

1. Explore the applicability of linear discriminants for unimodal and multimodal problems in two dimensions through the following.
   - **(a)** Sketch two multimodal distributions for which a linear discriminant could give excellent or possibly even the optimal classification accurcy.
   - **(b)** Sketch two unimodal distributions for which even the best linear discriminant would give poor classification accuracy.
   - **(c)** Consider two circular Gaussian distributions $p(\mathbf{x}|\omega_i) \sim N(\boldsymbol{\mu}_i, a_i\mathbf{I})$ and $P(\omega_i)$ for $i = 1, 2$ where $\mathbf{I}$ is the identity matrix and the other parameters can take on arbitrary values. Without performing any explicit calculations, explain whether the optimal decision boundary for this two-category problem must be a line. If not, sketch an example where the optimal discriminant is not a line.

2. Consider a linear machine with discriminant functions $g_i(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_{i0}$, $i = 1, \ldots, c$. Show that the decision regions are convex by showing that if $\mathbf{x}_1 \in \mathcal{R}_i$ and $\mathbf{x}_2 \in \mathcal{R}_i$ then $\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in \mathcal{R}_i$ for $0 \leq \lambda \leq 1$.

3. Figure 5.3 illustrates the two most popular methods for designing a c-category classifier from linear boundary segments. Another method is to save the full $\binom{c}{2}$ linear ω_i/ω_j boundaries, and classify any point by taking a *vote* based on all these boundaries. Prove whether the resulting decision regions must be convex. If they need not be convex, construct a nonpathological example yielding at least one nonconvex decision region.

4. Consider the hyperplane used in discrimination.
   - **(a)** Show that the distance from the hyperplane $g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = 0$ to the point $\mathbf{x}_a$ is $|g(\mathbf{x}_a)|/\|\mathbf{w}\|$ by minimizing $\|\mathbf{x} - \mathbf{x}_a\|^2$ subject to the constraint $g(\mathbf{x}) = 0$.
   - **(b)** Show that the projection of $\mathbf{x}_a$ onto the hyperplane is given by

$$\mathbf{x}_p = \mathbf{x}_a - \frac{g(\mathbf{x}_a)}{\|\mathbf{w}\|^2}\mathbf{w}.$$

5. Consider the three-category linear machine with discriminant functions $g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x} + w_{i0}$, $i = 1, 2, 3$.
   - **(a)** For the special case where $\mathbf{x}$ is two-dimensional and the threshold weights w_{i0} are zero, sketch the weight vectors with their tails at the origin, the three lines joining their heads, and the decision boundaries.
   - **(b)** How does this sketch change when a constant vector $\mathbf{c}$ is added to each of the three weight vectors?

TOTAL LINEAR SEPARABILITY

6. In the multicategory case, a set of samples is said to be linearly separable if there exists a linear machine that can classify them all correctly. If any samples labeled ω_i can be separated from all others by a single hyperplane, we shall say the samples are *totally linearly separable*. Show that totally linearly separable samples must be linearly separable, but that the converse need not be true. (For the converse, you may wish to consider a case in which a linear machine like the one in Problem 5 separates the samples.)

PAIRWISE LINEAR SEPARABILITY

7. A set of samples is said to be *pairwise linearly separable* if there exist $c(c-1)/2$ hyperplanes H_{ij} such that H_{ij} separates samples labeled ω_i from samples ω_j. Show that a pairwise-linearly separable set of patterns may not be linearly separable.

8. Let $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be a finite set of linearly separable training samples, and let $\mathbf{a}$ be called a solution vector if $\mathbf{a}^t\mathbf{y}_i \geq b$ for all i. Show that the minimum-length solution vector is unique. (You may wish to consider the effect of averaging two solution vectors.)

CONVEX HULL

9. The *convex hull* of a set of vectors $\mathbf{x}_i$, $i = 1, \ldots, n$ is the set of all vectors of the form

$$\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i,$$

where the coefficients α_i are nonnegative and sum to one. Given two sets of vectors, show that either they are linearly separable or their convex hulls intersect. (To answer this, suppose that both statements are true, and consider the classification of a point in the intersection of the convex hulls.)

PIECEWISE LINEAR MACHINE

10. A classifier is said to be a *piecewise linear machine* if its discriminant functions have the form

$$g_i(\mathbf{x}) = \max_{j=1,\ldots,n_i} g_{ij}(\mathbf{x}),$$

where

$$g_{ij}(\mathbf{x}) = \mathbf{w}_{ij}^t\mathbf{x} + w_{ij0}, \qquad \begin{array}{l} i = 1, \ldots, c \\ j = 1, \ldots, n_i. \end{array}$$

(a) Indicate how a piecewise linear machine can be viewed in terms of a linear machine for classifying subclasses of patterns.

(b) Show that the decision regions of a piecewise linear machine can be nonconvex and even multiply connected.

(c) Sketch a plot of $g_{ij}(x)$ for a one-dimensional example in which $n_1 = 2$ and $n_2 = 1$ to illustrate your answer to part (b).

11. Let the d components of $\mathbf{x}$ be either 0 or 1. Suppose we assign $\mathbf{x}$ to ω_1 if the number of nonzero components of $\mathbf{x}$ is odd, and to ω_2 otherwise. (This is called the *d-bit parity problem.*)

(a) Show that this dichotomy is not linearly separable if $d > 1$.

(b) Show that this problem can be solved by a piecewise linear machine with $d + 1$ weight vectors $\mathbf{w}_{ij}$ (see Problem 10). (To answer this, you may wish to consider vectors of the form $\mathbf{w}_{ij} = \alpha_{ij}(1, 1, \ldots, 1)^t$.)

Section 5.3

12. Consider the quadratic discriminant function (Eq. 4)

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j,$$

and define the symmetric, nonsingular matrix $\mathbf{W} = [w_{ij}]$. Show that the basic character of the decision boundary can be described in terms of the scaled matrix $\overline{\mathbf{W}} = \mathbf{W}/(\mathbf{w}^t\mathbf{W}^{-1}\mathbf{w} - 4w_0)$ as follows:

(a) If $\overline{\mathbf{W}} \propto \mathbf{I}$ (the identity matrix), then the decision boundary is a hypersphere.

(b) If $\overline{\mathbf{W}}$ is positive definite, then the decision boundary is a hyperellipsoid.

(c) If some eigenvalues of $\overline{\mathbf{W}}$ are positive and some negative, then the decision boundary is a hyperhyperboloid.

(d) Suppose $\mathbf{w} = \begin{pmatrix} 5 \\ 2 \\ -3 \end{pmatrix}$ and $\mathbf{W} = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 5 & 1 \\ 0 & 1 & -3 \end{pmatrix}$. What is the character of the solution?

(e) Repeat part (d) for $\mathbf{w} = \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix}$ and $\mathbf{W} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 0 & 4 \\ 3 & 4 & -5 \end{pmatrix}$.

Section 5.4

13. Derive Eq. 14, where $J(\cdot)$ depends on the iteration step k.

14. Consider the sum-of-squared-error criterion function (Eq. 43),

$$J_s(\mathbf{a}) = \sum_{i=1}^{n} (\mathbf{a}^t\mathbf{y}_i - b_i)^2.$$

Let $b_i = b$ and consider the following six training points:

$$\begin{array}{llll} \omega_1: & (1,5)^t & (2,9)^t & (-5,-3)^t \\ \omega_2: & (2,-3)^t & (-1,-4)^t & (0,2)^t \end{array}$$

(a) Calculate the Hessian matrix for this problem.

(b) Assuming the quadratic criterion function calculate the optimal learning rate η.

Section 5.5

15. In the convergence proof for the Perceptron algorithm (Theorem 5.1) the scale factor α was taken to be β^2/γ.

(a) Using the notation of Section 5.5, show that if α is greater than $\beta^2/(2\gamma)$, the maximum number of corrections is given by

$$k_0 = \frac{\|\mathbf{a}_1 - \alpha\mathbf{a}\|^2}{2\alpha\gamma - \beta^2}.$$

(b) If $\mathbf{a}_1 = \mathbf{0}$, what value of α minimizes k_0?

16. Modify the convergence proof given in Section 5.5.2 (Theorem 5.1) to prove the convergence of the following correction procedure: starting with an arbitrary initial weight vector $\mathbf{a}_1$, correct $\mathbf{a}(k)$ according to

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \eta(k)\mathbf{y}^k,$$

if and only if $\mathbf{a}^t(k)\mathbf{y}^k$ fails to exceed the margin b, where $\eta(k)$ is bounded by $0 < \eta_a \le \eta(k) \le \eta_b < \infty$. What happens if b is negative?

17. Let $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be a finite set of linearly separable samples in d dimensions.

(a) Suggest an exhaustive procedure that will find a separating vector in a finite number of steps. (You might wish to consider weight vectors whose components are integer-valued.)

(b) What is the computational complexity of your procedure?

18. Consider the criterion function

$$J_q(\mathbf{a}) = \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{a})} (\mathbf{a}^t\mathbf{y} - b)^2$$

where $\mathcal{Y}(\mathbf{a})$ is the set of samples for which $\mathbf{a}^t\mathbf{y} \leq b$. Suppose that $\mathbf{y}_1$ is the only sample in $\mathcal{Y}(\mathbf{a}(k))$. Show that $\nabla J_q(\mathbf{a}(k)) = 2(\mathbf{a}^t(k)\mathbf{y}_1 - b)\mathbf{y}_1$ and that the matrix of second partial derivatives is given by $\mathbf{D} = 2\mathbf{y}_1\mathbf{y}_1^t$. Use this to show that when the optimal $\eta(k)$ is used in Eq. 12 the gradient descent algorithm yields

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \frac{b - \mathbf{a}^t\mathbf{y}_1}{\|\mathbf{y}_1\|^2}\mathbf{y}_1.$$

19. Given the conditions in Eqs. 28–30, show that $\mathbf{a}(k)$ in the variable increment descent rule indeed converges for $\mathbf{a}^t\mathbf{y}_i > b$ for all i.

Section 5.6

20. Sketch a figure to illustrate the proof in Section 5.6.2. Be sure to take a general case, and label all variables.

Section 5.8

21. Show that the scale factor α in the MSE solution corresponding to Fisher's linear discriminant (Section 5.8.2) is given by

$$\alpha = \left[1 + \frac{n_1 n_2}{n}(\mathbf{m}_1 - \mathbf{m}_2)^t \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)\right]^{-1}$$

22. Generalize the results of Section 5.8.3 to show that the vector $\mathbf{a}$ that minimizes the criterion function

$$J_s'(\mathbf{a}) = \sum_{\mathbf{y} \in \mathcal{Y}_1} (\mathbf{a}^t\mathbf{y} - (\lambda_{21} - \lambda_{11}))^2 + \sum_{\mathbf{y} \in \mathcal{Y}_2} (\mathbf{a}^t\mathbf{y} - (\lambda_{12} - \lambda_{22}))^2$$

provides asymptotically a minimum-mean-squared-error approximation to the Bayes discriminant function $(\lambda_{21} - \lambda_{11})P(\omega_1|\mathbf{x}) - (\lambda_{12} - \lambda_{22})P(\omega_2|\mathbf{x})$.

23. Using the notation in the text, show that if $\mathcal{E}[\mathbf{y}\mathbf{y}^t]$ is nonsingular, and if the rate coefficients $\eta(k)$ for iteration index k satisfy Eqs. 68 and 69, then the weight vector $\mathbf{a}(k)$ convergest to $\hat{\mathbf{a}}$ in mean square, that is, $\lim_{k\to\infty} \mathcal{E}[\|\mathbf{a}(k) - \hat{\mathbf{a}}\|^2] = 0$, as stated in Eq. 70.

24. Consider the criterion function $J_m(\mathbf{a}) = \mathcal{E}[(\mathbf{a}^t\mathbf{y}(\mathbf{x}) - z)^2]$ and the Bayes discriminant function $g_0(\mathbf{x})$.

(a) Show that

$$J_m = \mathcal{E}[(\mathbf{a}^t\mathbf{y} - g_0)^2] - 2\mathcal{E}[(\mathbf{a}^t\mathbf{y} - g_0)(z - g_0)] + \mathcal{E}[(z - g_0)^2].$$

(b) Use the fact that the conditional mean of z is $g_0(\mathbf{x})$ in showing that the $\hat{\mathbf{a}}$ that minimizes J_m also minimizes $\mathcal{E}[(\mathbf{a}^t\mathbf{y} - g_0)^2]$.

25. A scalar analog of the relation $\mathbf{R}_{k+1}^{-1} = \mathbf{R}_k^{-1} + \mathbf{y}_k\mathbf{y}_k^t$ used in stochastic approximation is $\eta^{-1}(k+1) = \eta^{-1}(k) + y_k^2$.

(a) Show that this has the closed-form solution

$$\eta(k) = \frac{\eta(1)}{1 + \eta(1)\sum_{i=1}^{k-1} y_i^2}.$$

(b) Assume that $\eta(1) > 0$ and $0 < a \leq y_i^2 \leq b < \infty$, and indicate why this sequence of coefficients will satisfy $\sum \eta(k) \to \infty$ and $\sum \eta(k)^2 \to L < \infty$.

26. Show for the Widrow-Hoff or LMS rule that if $\eta(k) = \eta(1)/k$, then the sequence of weight vectors converges to a limiting vector $\mathbf{a}$ satisfying $\mathbf{Y}^\dagger(\mathbf{Ya} - \mathbf{b}) = 0$ (Eq. 61).

Section 5.9

27. Consider the following six data points:

$$\begin{array}{lll} \omega_1{:}(1,2)^t & (2,-4)^t & (-3,-1)^t \\ \omega_2{:}(2,4)^t & (-1,-5)^t & (5,0)^t \end{array}$$

(a) Are they linearly separable?

(b) Using the approach in the text, assume $\mathbf{R} = \mathbf{I}$, the identity matrix, and calculate the optimal learning rate η by Eq. 95.

Section 5.10

28. The linear programming problem formulated in Section 5.10.2 involved minimizing a single artificial variable τ under the constraints $\mathbf{a}^t\mathbf{y}_i + \tau > b_i$ and $\tau \geq 0$. Show that the resulting weight vector minimizes the criterion function

$$J_\tau(\mathbf{a}) = \max_{\mathbf{a}^t\mathbf{y}_i \leq b_i} [b_i - \mathbf{a}^t\mathbf{y}_i].$$

Section 5.11

29. Discuss qualitatively why if samples from two categories are distinct (i.e., no feature point is labeled by both categories), there always exists a nonlinear mapping to a higher dimension that leaves the points linearly separable.

30. The figure in Example 2 shows the maximum margin for a Support Vector Machine applied to the exclusive-OR problem mapped to a five-dimensional space. That figure shows the training patterns and contours of the discriminant function, as projected in the two-dimensional subspace defined by the features $\sqrt{2}x_1$ and $\sqrt{2}x_1x_2$. Ignore the constant feature, and consider the other four features. For each of the $\binom{4}{2} - 1 = 5$ pairs of features other than the one shown in the Example, show the patterns and the lines corresponding to the discriminant $g = \pm 1$. Are the margins in your figures the same? Explain why or why not?

31. Write pseudocode to implement a simple learning method for support vector machines by modifying the Perceptron learning algorithm (Algorithm 4). Be sure to write a detailed step, including mathematical form, for describing the current worst-classified training pattern. Explain why at the end of the training period, weight updates will involve only the support vectors.

32. Consider a Support Vector Machine and the following training data from two categories:

$$\begin{array}{llll} \omega_1: & (1,1)^t & (2,2)^t & (2,0)^t \\ \omega_2: & (0,0)^t & (1,0)^t & (0,1)^t \end{array}$$

(a) Plot these six training points, and construct by inspection the weight vector for the optimal hyperplane, and the optimal margin.

(b) What are the support vectors?

(c) Construct the solution in the dual space by finding the Lagrange undetermined multipliers, α_i. Compare your result to that in part (a).

33. This problem asks you to follow the Kuhn-Tucker theorem to convert the constrained optimization problem in support vector machines into a dual, unconstrained one. For SVMs, the goal is to find the minimum-length weight vector **a** subject to the (classification) constraints

$$z_k \mathbf{a}^t \mathbf{y}_k \geq 1 \qquad k = 1, \ldots, n,$$

where $z_k = \pm 1$ indicates the target getegory of each of the n patterns $\mathbf{y}_k$. Note that **a** and **y** are augmented (by a_0 and $y_0 = 1$, respectively).

(a) Consider the unconstrained optimization associated with SVMs:

$$L(\mathbf{a}, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{a}||^2 - \sum_{k=1}^{n} \alpha_k [z_k \mathbf{a}^t \mathbf{y}_k - 1].$$

In the space determined by the components of **a**, and the n (scalar) undetermined multipliers α_k, the desired solution is a *saddle point*, rather than a global maximum or minimum. Explain.

(b) Next eliminate the dependency of this ("primal") functional upon **a** (i.e., reformulated the optimization in a dual form) by the following steps. Note that at the saddle point of the primal functional, we have

$$\frac{\partial L(\mathbf{a}^*, \boldsymbol{\alpha}^*)}{\partial \mathbf{a}} = \mathbf{0}.$$

Solve for the partial derivatives and conclude

$$\sum_{k=1}^{n} \alpha_k^* z_k = 0 \qquad \alpha_k^* \geq 0, \; k = 1, \ldots, n.$$

(c) Prove that at this inflection point, the optimal hyperplane is a linear combination of the training vectors:

$$\mathbf{a}^* = \sum_{k=1}^{n} \alpha_k^* z_k \mathbf{y}_k.$$

(d) According to the Kuhn-Tucker theorem (cf. Bibliography), an undetermined multiplier α_k^* is non-zero only if the corresponding sample $\mathbf{y}_k$ satisfies $z_k\mathbf{a}^t\mathbf{y}_k = 1$. Show that this can be expressed as

$$\alpha_k^*[z_k\mathbf{a}^{*t}\mathbf{y}_k - 1] = 0 \qquad k = 1, \ldots, n.$$

Recall that the samples where α_k^* are nonzero (i.e., $z_k\mathbf{a}^t\mathbf{y}_k = 1$), are the support vectors.

(e) Use the results from parts (b) and (c) to eliminate the weight vector in the functional, and thereby construct the dual functional

$$\tilde{L}(\mathbf{a}, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{a}||^2 - \sum_{k=1}^{n} \alpha_k z_k \mathbf{a}^t\mathbf{y}_k + \sum_{k=1}^{n} \alpha_k.$$

(f) Substitute the solution $\mathbf{a}^*$ from part (c) to find the dual functional

$$\tilde{L}(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{j,k=1}^{n} \alpha_j\alpha_k z_j z_k (\mathbf{y}_j^t\mathbf{y}_k) + \sum_{j=1}^{n} \alpha_j.$$

34. Repeat Example 2 using the same $\varphi(\cdot)$ but with the following four points:

$$\omega_1: (1,5)^t \quad (-2,-4)^t$$
$$\omega_2: (2,3)^t \quad (-1,5)^t.$$

Section 5.12

35. Suppose that for each two-dimensional training point $\mathbf{y}_i$ in category ω_1 there is a corresponding (symmetric) point in ω_2 at $-\mathbf{y}_i$.

(a) Prove that a separating hyperplane (should one exist) or LMS solution must go through the origin.

(b) Consider such a symmetric, six-point problem with the following points:

$$\omega_1: \ (1,2)^t \quad (2,-4)^t \quad \mathbf{y}$$
$$\omega_2: (-1,-2)^t \quad (-2,4)^t \quad -\mathbf{y}$$

Find the mathematical conditions on $\mathbf{y}$ such that the LMS solution for this problem *not* give a separating hyperplane.

(c) Generalize your result as follows. Suppose ω_1 consists of $\mathbf{y}_1$ and $\mathbf{y}_2$ (known) and the symmetric versions in ω_2. What is the condition on $\mathbf{y}_3$ such that the LMS solution does not separate the points.

36. Write pseudocode for a fixed increment multicategory algorithm based on Eq. 115. Discuss the strengths and weakness of such an implementation.

37. Generalize the discussion in Section 5.8.3 in order to prove that the solution derived from Eq. 120 provides a minimum-mean-square-error approximation to the Bayes discriminant function given in Eq. 122.

38. Use Kesler's construction to establish equivalences between multicategory and two-category procedures. Generalize the Perceptron and relaxation procedures to the multicategory case.

COMPUTER EXERCISES

Several of the exercises use the data in the following table.

	ω_1		ω_2		ω_3		ω_4	
sample	x_1	x_2	x_1	x_2	x_1	x_2	x_1	x_2
1	0.1	1.1	7.1	4.2	−3.0	−2.9	−2.0	−8.4
2	6.8	7.1	−1.4	−4.3	0.5	8.7	−8.9	0.2
3	−3.5	−4.1	4.5	0.0	2.9	2.1	−4.2	−7.7
4	2.0	2.7	6.3	1.6	−0.1	5.2	−8.5	−3.2
5	4.1	2.8	4.2	1.9	−4.0	2.2	−6.7	−4.0
6	3.1	5.0	1.4	−3.2	−1.3	3.7	−0.5	−9.2
7	−0.8	−1.3	2.4	−4.0	−3.4	6.2	−5.3	−6.7
8	0.9	1.2	2.5	−6.1	−4.1	3.4	−8.7	−6.4
9	5.0	6.4	8.4	3.7	−5.1	1.6	−7.1	−9.7
10	3.9	4.0	4.1	−2.2	1.9	5.1	−8.0	−6.3

Section 5.4

1. Consider basic gradient descent (Algorithm 1) and the Perceptron criterion (Eq. 16) applied to the data in the tables.

(a) Apply both to the two-dimensional data in order to discriminate categories ω_1 and ω_3. For the gradient descent, use $\eta(k) = 0.01$. Plot the criterion function as function of the iteration number.

(b) Estimate the total number of mathematical operations in the two algorithms.

(c) Plot the convergence time versus learning rate. What is the minimum learning rate that fails to lead to convergence?

2. Write a program to implement the batch Perceptron algorithm (Algorithm 3).

(a) Starting with $\mathbf{a} = \mathbf{0}$, apply your program to the training data from ω_1 and ω_2. Note the number of iterations required for convergence.

(b) Apply your program to ω_3 and ω_2. Again, note the number of iterations required for convergence.

(c) Explain the difference between the iterations required in the two cases.

POCKET ALGORITHM

3. The *Pocket algorithm* uses the criterion of longest sequence of correctly classified points, and can be used in conjunction with a number of basic learning algorithms. For instance, the Pocket version of the Perceptron algorithm works in a sort of ratchet scheme as follows. There are two sets of weights, one for the normal Perceptron algorithm, and a separate one (not directly used for training) which is kept "in your pocket." Both are randomly chosen at the start. The "pocket" weights are tested on the full data set to find the longest run of patterns properly classified. (At the beginning, this run will be short.) The Perceptron weights are trained as usual, but after every weight update (or after some finite number of such weight updates), the Perceptron weight is tested on data points, randomly selected, to determine the longest run of properly classified points.

If this length is greater than with the pocket weights, the Perceptron weights replace the pocket weights, and Perceptron training continues. In this way, the pocket weights continually improve, classifying longer and longer runs of randomly selected points.

(a) Write a pocket version of the single-sample Perceptron algorithm (Algorithm 4).

(b) Apply it to the data from ω_1 and ω_3. How often are the pocket weights updated?

4. Explore through simulation, several equations governing the rate of convergence of the Perceptron algorithm.

(a) Generate 25 three-dimensional points for each of two categories according to Gaussian distributions, $p(\mathbf{x}|\omega_i) \sim N(\boldsymbol{\mu}_i, \mathbf{I})$, and $\boldsymbol{\mu}_1 = \mathbf{0}$, $\boldsymbol{\mu}_2 = (4, 4, 4)^t$.

(b) Randomly choose an initial weight vector $\mathbf{a}$ with the constraint $\|\mathbf{a}\| = 0.1$ (i.e., $\mathbf{a}$ lies on a three-dimensional sphere of radius 0.1). Implement the single-sample Perceptron algorithm (Algorithm 4) using your initial weight vector and the data from part (a).

(c) Calculate β^2 according to Eq. 21.

(d) After training is complete, calculate γ according to Eq. 22 and verify that the value of k_0 from Eq. 25 is consistent with your simulation results.

5. Show that the first five points of categories ω_3 and ω_4 are not linearly separable. Construct by hand a nonlinear mapping of the feature space to make them linearly separable and train a Perceptron classifier on them. What is the classification error on the remaining (transformed) points in the table?

6. Consider a version of the Balanced Winnow training algorithm (Algorithm 7). Classification of test data is given by line 2. Compare the converge rate of Balanced Winnow with the fixed-increment, single-sample Perceptron (Algorithm 4) on a problem with large number of redundant features, as follows.

(a) Generate a training set of 2000 100-dimensional patterns (1000 from each of two categories) in which only the first ten features are informative, in the following way. For patterns in category ω_1, each of the first ten features are chosen randomly and uniformly from the range $+1 \leq x_i \leq 2$, for $i = 1, \ldots, 10$. Conversely, for patterns in ω_2, each of the first ten features are chosen randomly and uniformly from the range $-2 \leq x_i \leq -1$. All other features from both categories are chosen from the range $-2 \leq x_i \leq +2$.

(b) Construct by hand the obvious separating hyperplane.

(c) Adjust the learning rates so that your two algorithms have roughly the same convergence rate on the full training set when only the first ten features are considered. That is, assume each of the 2000 training patterns consists of just the first ten features.

(d) Now apply your two algorithms to 2000 50-dimensional patterns, in which the first ten features are informative and the remaining 40 are not. Plot the total number of errors versus iteration.

(e) Now apply your two algorithms to the full training set of 2000 100-dimensional patterns.

(f) Summarize your answers to parts (c)–(e).

Section 5.6

7. Consider relaxation methods as described in the text.

(a) Implement batch relaxation with margin (Algorithm 8), set $b = 0.1$ and $\mathbf{a}(1) = \mathbf{0}$ and apply it to the data in ω_1 and ω_3. Plot the criterion function as a function of the number of passes through the training set.

(b) Repeat for $b = 0.5$ and $\mathbf{a}(1) = \mathbf{0}$. Explain qualitatively any difference you find in the convergence rates.

(c) Modify your program to use single sample learning. Again, plot the criterion as a function of the number of passes through the training set.

(d) Discuss any differences, being sure to consider the learning rate.

Section 5.8

8. Write a single-sample relaxation algorithm using Eq. 72 to update the matrix $\mathbf{R}$, as described in the text. Apply your program to the ω_2 and ω_3 data in the table above.

Section 5.9

9. Implement the Ho-Kashyap algorithm (Algorithm 11) and apply to the data in categories ω_1 and ω_3. Repeat for categories ω_4 and ω_2.

Section 5.10

10. Write a program to implement the LMS algorithm (Algorithm 10) for two categories and two-dimensional data. Suppose ω_1 consists of 21 points: 10 copies of $(0, 0)^t$, 10 copies of $(0, 1)^t$ and a single point at $(1, -2)^t$. The other category, ω_2 consists of 10 copies of $(1, 0)^t$, 10 copies of $(1, 1)^t$ and a single point at $(0, 3)^t$. Find the LMS solution for the weight vector and the equation of the corresponding hyperplane. Does it separate the classes? Are the classes linearly separable? Explain.

Section 5.11

11. Write a program to implement the Support Vector Machine algorithm. Train an SVM classifer with data from ω_3 and ω_4 in the following way. Preprocess each training pattern to form a new vector having components $1, x_1, x_2, x_1^2, x_1x_2$, and x_2^2.

(a) Train your classifier with just the first patterns in ω_3 and ω_4 and find the separating hyperplane and the margin.

(b) Repeat part (a) using the first two points in the two categories (four points total). What is the equation of the separating hyperplane, the margin, and the support vectors?

(c) Repeat part (b) with the first three points in each category (six points total), the first four points, and so on, until the transformed patterns cannot be linearly separated in the transformed space.

Section 5.12

12. Write a program based on Kesler's construction to implement a multicategory generalization of basic LMS algorithm (Algorithm 10).

(a) Apply it to the data in all four categories in the table.

(b) Use your algorithm in a two-category mode to form ω_i/not ω_i boundaries for $i = 1, 2, 3, 4$. Find any regions whose categorization by your system is ambiguous. That is, give representative patterns $\mathbf{x}$ where your classifier cannot return a category label unambiguously.

BIBLIOGRAPHY

[1] Henry D. Block and Simon A. Levin. On the boundedness of an iterative procedure for solving a system of linear inequalities. *Proceedings of the American Mathematical Society*, 26:229–235, 1970.

[2] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In David Haussler, editor, *Proceedings of the 4th Workshop on Computational Learning Theory*, pages 144–152, ACM Press, San Mateo, CA, 1992.

[3] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[4] Vladimir Cherkassky and Filip Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley, New York, 1998.

[5] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 Part II:179–188, 1936.

[6] David Gale, Harold W. Kuhn, and Albert W. Tucker. Linear programming and the theory of games. In Tjalling C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 317–329. Wiley, New York, 1951.

[7] Adam J. Grove, Nicholas Littlestone, and Dale Schuurmans. General convergence results for linear discriminant updates. In *Proceedings of the COLT 97*, pages 171–183. ACM Press, 1997.

[8] Isabelle Guyon and David G. Stork. Linear discriminant and support vector classifiers. In Alex Smola, Peter Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in large margin classifiers*. MIT Press, Cambridge, MA, 1999.

[9] Wilbur H. Highleyman. Linear decision functions, with application to pattern recognition. *Proceedings of the IRE*, 50:1501–1514, 1962.

[10] Nicholas Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[11] Nicholas Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using Winnow. In Manfred K. Warmuth and Leslie G. Valiant, editors, *Proceedings of COLT 91*, pages 147–156, Morgan Kaufmann, San Mateo, CA, 1991.

[12] Warren S. McCulloch and Walter Pitts. A logical calculus of ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[13] Marvin L. Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, 1969.

[14] Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, Cambridge, MA, 1999.

[15] Bernhard Schölkopf, Christopher J. C. Burges, and Vladimir Vapnik. Extracting support data for a given task. In Usama M. Fayyad and Ramasamy Uthurasamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 252–257. AAAI Press, Menlo Park, CA, 1995.

[16] Fred W. Smith. Design of multicategory pattern classifiers with two-category classifier design procedures. *IEEE Transactions on Computers*, C-18(6):548–551, 1969.

[17] Gilbert Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, MA, 1986.

[18] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[19] Šarūnas Raudys. Evolution and generalization of a single neurone: I. Single-layer perceptron as seven statistical classifiers. *Neural Networks*, 11(2):283–296, 1998.

[20] Šarūnas Raudys. Evolution and generalization of a single neurone: II. Complexity of statistical classifiers and sample size considerations. *Neural Networks*, 11(2):297–313, 1998.

[21] Stephen S. Yau and John M. Schumpert. Design of pattern classifiers with the updating property using stochastic approximation techniques. *IEEE Transactions on Computers*, C-17(9):861–872, 1968.