10 *Linear Discrimination*

In linear discrimination, we assume that instances of a class are linearly separable from instances of other classes. This is a discriminant-based approach that estimates the parameters of the linear discriminant directly from a given labeled sample.

10.1 Introduction

WE REMEMBER from the previous chapters that in classification we define a set of discriminant functions $g_j(\boldsymbol{x}), j = 1, \ldots, K$, and then we

$$\text{choose } C_i \text{ if } g_i(\boldsymbol{x}) = \max_{j=1}^{K} g_j(\boldsymbol{x})$$

Previously, when we discussed methods for classification, we first estimated the prior probabilities, $\hat{P}(C_i)$, and the class likelihoods, $\hat{p}(\boldsymbol{x}|C_i)$, then used Bayes' rule to calculate the posterior densities. We then defined the discriminant functions in terms of the posterior, for example,

$$g_i(\boldsymbol{x}) = \log \hat{P}(C_i|\boldsymbol{x})$$

LIKELIHOOD-BASED CLASSIFICATION

This is called *likelihood-based classification*, and we have previously discussed the parametric (chapter 5), semiparametric (chapter 7), and nonparametric (chapter 8) approaches to estimating the class likelihoods, $p(\boldsymbol{x}|C_i)$.

DISCRIMINANT-BASED CLASSIFICATION

We are now going to discuss *discriminant-based classification* where we assume a model directly for the discriminant, bypassing the estimation of likelihoods or posteriors. The discriminant-based approach, as we also saw for the case of decision trees in chapter 9, makes an assumption on the form of the discriminant between the classes and makes no assumption about, or requires no knowledge of the densities—for example,

whether they are Gaussian, or whether the inputs are correlated, and so forth.

We define a model for the discriminant

$$g_i(\boldsymbol{x}|\Phi_i)$$

explicitly parameterized with the set of parameters Φ_i, as opposed to a likelihood-based scheme that has implicit parameters in defining the likelihood densities. This is a different inductive bias: Instead of making an assumption on the form of the class densities, we make an assumption on the form of the boundaries separating classes.

Learning is the optimization of the model parameters $\boldsymbol{\Phi}_i$ to maximize the quality of the separation, that is, the classification accuracy on a given labeled training set. This differs from the likelihood-based methods that search for the parameters that maximize sample likelihoods, separately for each class.

In the discriminant-based approach, we do not care about correctly estimating the densities inside class regions; all we care about is the correct estimation of the *boundaries* between the class regions. Those who advocate the discriminant-based approach (e.g., Vapnik 1995) state that estimating the class densities is a harder problem than estimating the class discriminants, and it does not make sense to solve a hard problem to solve an easier problem. This is of course true only when the discriminant can be approximated by a simple function.

In this chapter, we concern ourselves with the simplest case where the discriminant functions are linear in $\boldsymbol{x}$:

$$g_i(\boldsymbol{x}|\boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} = \sum_{j=1}^{d} w_{ij}x_j + w_{i0} \tag{10.1}$$

LINEAR DISCRIMINANT

The *linear discriminant* is used frequently mainly due to its simplicity: Both the space and time complexities are $\mathcal{O}(d)$. The linear model is easy to understand: the final output is a weighted sum of the input attributes x_j. The magnitude of the weight w_j shows the importance of x_j and its sign indicates if the effect is positive or negative. Most functions are additive in that the output is the sum of the effects of several attributes where the weights may be positive (enforcing) or negative (inhibiting). For example, when a customer applies for credit, financial institutions calculate the applicant's credit score that is generally written as a sum of the effects of various attributes; for example, yearly income has a positive effect (higher incomes increase the score).

In many applications, the linear discriminant is also quite accurate. We know, for example, that when classes are Gaussian with a shared covariance matrix, the optimal discriminant is linear. The linear discriminant, however, can be used even when this assumption does not hold, and the model parameters can be calculated without making any assumptions on the class densities. We should always use the linear discriminant before trying a more complicated model to make sure that the additional complexity is justified.

As always, we formulate the problem of finding a linear discriminant function as a search for the parameter values that minimize an error function. In particular, we concentrate on *gradient* methods for optimizing a criterion function.

10.2 Generalizing the Linear Model

QUADRATIC DISCRIMINANT

When a linear model is not flexible enough, we can use the *quadratic discriminant* function and increase complexity

$$g_i(\boldsymbol{x}|\mathbf{W}_i, \boldsymbol{w}_i, w_{i0}) = \boldsymbol{x}^T\mathbf{W}_i\boldsymbol{x} + \boldsymbol{w}_i\boldsymbol{x} + w_{i0} \tag{10.2}$$

but this approach is $O(d^2)$ and we again have the bias/variance dilemma: The quadratic model, though is more general, requires much larger training sets and may overfit on small samples.

HIGHER-ORDER TERMS
PRODUCT TERMS

An equivalent way is to preprocess the input by adding *higher-order terms*, also called *product terms*. For example, with two inputs x_1 and x_2, we can define new variables

$$z_1 = x_1, z_2 = x_2, z_3 = x_1^2, z_4 = x_2^2, z_5 = x_1x_2$$

and take $\boldsymbol{z} = [z_1, z_2, z_3, z_4, z_5]^T$ as the input. The linear function defined in the five-dimensional $\boldsymbol{z}$ space corresponds to a nonlinear function in the two-dimensional $\boldsymbol{x}$ space. Instead of defining a nonlinear function (discriminant or regression) in the original space, what we do is to define a suitable nonlinear transformation to a new space where the function can be written in a linear form.

We write the discriminant as

$$g_i(\boldsymbol{x}) = \sum_{j=1}^{k} w_j\phi_{ij}(\boldsymbol{x}) \tag{10.3}$$

BASIS FUNCTION

where $\phi_{ij}(\boldsymbol{x})$ are *basis functions*. Higher-order terms are only one set of possible basis functions; other examples are

- $\sin(x_1)$
- $\exp(-(x_1 - m)^2/c)$
- $\exp(-\|\boldsymbol{x} - \boldsymbol{m}\|^2/c)$
- $\log(x_2)$
- $1(x_1 > c)$
- $1(ax_1 + bx_2 > c)$

where m, a, b, c are scalars, $\boldsymbol{m}$ is a d-dimensional vector, and $1(b)$ returns 1 if b is true and returns 0 otherwise. The idea of writing a nonlinear function as a linear sum of nonlinear basis functions is an old idea and was originally called *potential functions* (Aizerman, Braverman, and Rozonoer 1964). Multilayer perceptrons (chapter 11) and radial basis functions (chapter 12) have the advantage that the parameters of the basis functions can be fine-tuned to the data during learning. In chapter 13, we discuss support vector machines that use kernel functions built from such basis functions.

POTENTIAL FUNCTION

10.3 Geometry of the Linear Discriminant

10.3.1 Two Classes

Let us start with the simpler case of two classes. In such a case, one discriminant function is sufficient:

$$\begin{aligned} g(\boldsymbol{x}) &= g_1(\boldsymbol{x}) - g_2(\boldsymbol{x}) \\ &= (\boldsymbol{w}_1^T\boldsymbol{x} + w_{10}) - (\boldsymbol{w}_2^T\boldsymbol{x} + w_{20}) \\ &= (\boldsymbol{w}_1 - \boldsymbol{w}_2)^T\boldsymbol{x} + (w_{10} - w_{20}) \\ &= \boldsymbol{w}^T\boldsymbol{x} + w_0 \end{aligned}$$

and we

$$\text{choose} \begin{cases} C_1 & \text{if } g(\boldsymbol{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}$$

WEIGHT VECTOR
THRESHOLD

This defines a hyperplane where $\boldsymbol{w}$ is the *weight vector* and w_0 is the *threshold*. This latter name comes from the fact that the decision rule can be rewritten as follows: Choose C_1 if $\boldsymbol{w}^T\boldsymbol{x} > -w_0$, and choose C_2

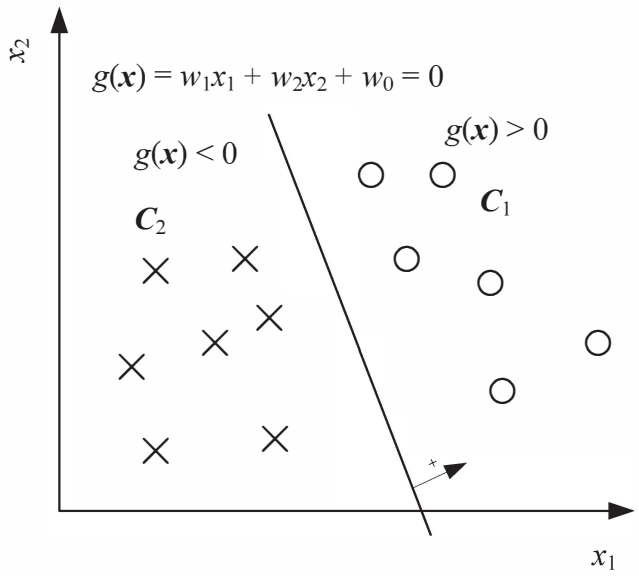

Figure 10.1 In the two-dimensional case, the linear discriminant is a line that separates the examples from two classes.

otherwise. The hyperplane divides the input space into two half-spaces: the decision region $\mathcal{R}_1$ for C_1 and $\mathcal{R}_2$ for C_2. Any $\boldsymbol{x}$ in $\mathcal{R}_1$ is on the *positive* side of the hyperplane and any $\boldsymbol{x}$ in $\mathcal{R}_2$ is on its *negative* side. When $\boldsymbol{x}$ is $\mathbf{0}$, $g(\boldsymbol{x}) = w_0$ and we see that if $w_0 > 0$, the origin is on the positive side of the hyperplane, and if $w_0 < 0$, the origin is on the negative side, and if $w_0 = 0$, the hyperplane passes through the origin (see figure 10.1).

Take two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ both on the decision surface; that is, $g(\boldsymbol{x}_1) = g(\boldsymbol{x}_2) = 0$, then

$$\begin{aligned} \boldsymbol{w}^T\boldsymbol{x}_1 + w_0 &= \boldsymbol{w}^T\boldsymbol{x}_2 + w_0 \\ \boldsymbol{w}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) &= 0 \end{aligned}$$

and we see that $\boldsymbol{w}$ is normal to any vector lying on the hyperplane. Let us rewrite $\boldsymbol{x}$ as (Duda, Hart, and Stork 2001)

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

where $\boldsymbol{x}_p$ is the normal projection of $\boldsymbol{x}$ onto the hyperplane and r gives us the distance from $\boldsymbol{x}$ to the hyperplane, negative if $\boldsymbol{x}$ is on the negative

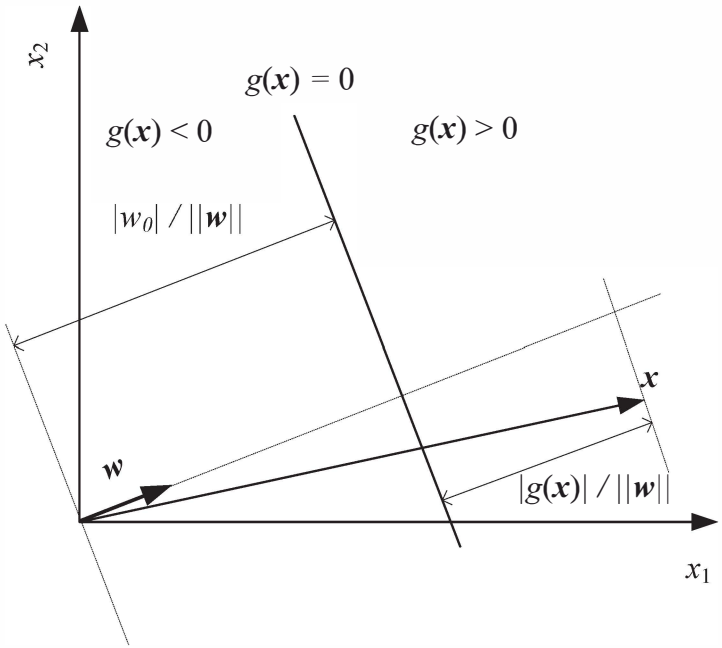

Figure 10.2 The geometric interpretation of the linear discriminant.

side, and positive if $\boldsymbol{x}$ is on the positive side (see figure 10.2). Calculating $g(\boldsymbol{x})$ and noting that $g(\boldsymbol{x}_p) = 0$, we have

$$r = \frac{g(\boldsymbol{x})}{\|\boldsymbol{w}\|} \tag{10.4}$$

We see then that the distance to origin is

$$r_0 = \frac{w_0}{\|\boldsymbol{w}\|} \tag{10.5}$$

Thus w_0 determines the location of the hyperplane with respect to the origin, and $\boldsymbol{w}$ determines its orientation.

10.3.2 Multiple Classes

When there are $K > 2$ classes, there are K discriminant functions. When they are linear, we have

$$g_i(\boldsymbol{x}|\boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} \tag{10.6}$$

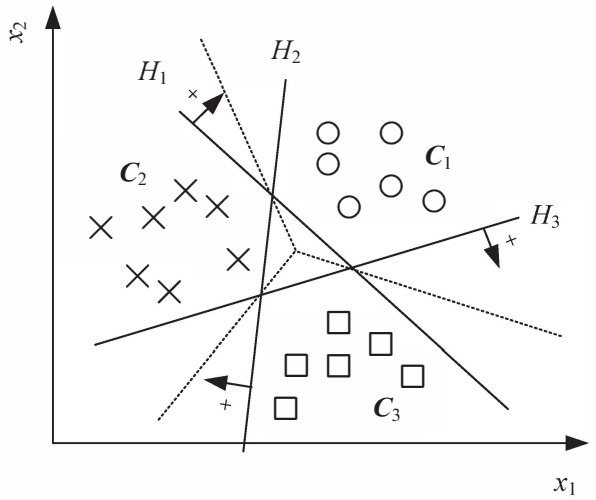

Figure 10.3 In linear classification, each hyperplane H_i separates the examples of C_i from the examples of all other classes. Thus for it to work, the classes should be linearly separable. Dotted lines are the induced boundaries of the linear classifier.

We are going to talk about learning later on but for now, we assume that the parameters, $\boldsymbol{w}_i, w_{i0}$, are computed so as to have

$$g_i(\boldsymbol{x}|\boldsymbol{w}_i, w_{i0}) = \begin{cases} > 0 & \text{if } \boldsymbol{x} \in C_i \\ \leq 0 & \text{otherwise} \end{cases} \tag{10.7}$$

for all $\boldsymbol{x}$ in the training set. Using such discriminant functions corresponds to assuming that all classes are *linearly separable*; that is, for each class C_i, there exists a hyperplane H_i such that all $\boldsymbol{x} \in C_i$ lie on its positive side and all $\boldsymbol{x} \in C_j, j \neq i$ lie on its negative side (see figure 10.3).

LINEARLY SEPARABLE CLASSES

During testing, given $\boldsymbol{x}$, ideally, we should have only one $g_j(\boldsymbol{x}), j = 1, \ldots, K$ greater than 0 and all others should be less than 0, but this is not always the case: The positive half-spaces of the hyperplanes may overlap, or, we may have a case where all $g_j(\boldsymbol{x}) < 0$. These may be taken as *reject* cases, but the usual approach is to assign $\boldsymbol{x}$ to the class having the highest discriminant:

$$\text{Choose } C_i \text{ if } g_i(\boldsymbol{x}) = \max_{j=1}^{K} g_j(\boldsymbol{x}) \tag{10.8}$$

Remembering that $|g_i(\boldsymbol{x})|/||\boldsymbol{w}_i||$ is the distance from the input point to the hyperplane, assuming that all $\boldsymbol{w}_i$ have similar length, this assigns the

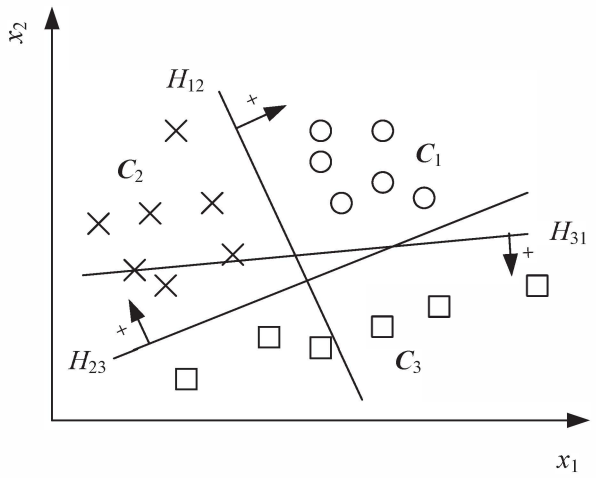

Figure 10.4 In pairwise linear separation, there is a separate hyperplane for each pair of classes. For an input to be assigned to C_1, it should be on the positive side of H_{12} and H_{13} (which is the negative side of H_{31}); we do not care about the value of H_{23}. In this case, C_1 is not linearly separable from other classes but is pairwise linearly separable.

LINEAR CLASSIFIER

point to the class (among all $g_j(\boldsymbol{x}) > 0$) to whose hyperplane the point is most distant. This is called a *linear classifier*, and geometrically it divides the feature space into K convex decision regions $\mathcal{R}_i$ (see figure 10.3).

10.4 Pairwise Separation

PAIRWISE SEPARATION

If the classes are not linearly separable, one approach is to divide it into a set of linear problems. One possibility is *pairwise separation* of classes (Duda, Hart, and Stork 2001). It uses $K(K-1)/2$ linear discriminants, $g_{ij}(\boldsymbol{x})$, one for every pair of distinct classes (see figure 10.4):

$$g_{ij}(\boldsymbol{x}|\boldsymbol{w}_{ij}, w_{ij0}) = \boldsymbol{w}_{ij}^T\boldsymbol{x} + w_{ij0}$$

The parameters $\boldsymbol{w}_{ij}, j \neq i$ are computed during training so as to have

$$g_{ij}(\boldsymbol{x}) = \begin{cases} > 0 & \text{if } \boldsymbol{x} \in C_i \\ \leq 0 & \text{if } \boldsymbol{x} \in C_j \\ \text{don't care} & \text{otherwise} \end{cases} \quad i, j = 1, \ldots, K \text{ and } i \neq j \tag{10.9}$$

that is, if $\boldsymbol{x}^t \in C_k$ where $k \neq i, k \neq j$, then $\boldsymbol{x}^t$ is not used during training of $g_{ij}(\boldsymbol{x})$.

During testing, we

$$\text{choose } C_i \text{ if } \forall j \neq i, g_{ij}(\boldsymbol{x}) > 0$$

In many cases, this may not be true for any i and if we do not want to reject such cases, we can relax the conjunction by using a summation and choosing the maximum of

$$g_i(\boldsymbol{x}) = \sum_{j \neq i} g_{ij}(\boldsymbol{x}) \tag{10.10}$$

Even if the classes are not linearly separable, if the classes are pairwise linearly separable—which is much more likely—pairwise separation can be used, leading to nonlinear separation of classes (see figure 10.4). This is another example of breaking down a complex (e.g., nonlinear) problem, into a set of simpler (e.g., linear) problems. We have already seen decision trees (chapter 9) that use this idea, and we will see more examples of this in chapter 17 on combining multiple models, for example, error-correcting output codes, and mixture of experts, where the number of linear models is less than $\mathcal{O}(K^2)$.

10.5 Parametric Discrimination Revisited

In chapter 5, we saw that if the class densities, $p(\boldsymbol{x}|C_i)$, are Gaussian and share a common covariance matrix, the discriminant function is linear

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} \tag{10.11}$$

where the parameters can be analytically calculated as

$$\begin{aligned} \boldsymbol{w}_i &= \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i \\ w_{i0} &= -\frac{1}{2}\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \log P(C_i) \end{aligned} \tag{10.12}$$

Given a dataset, we first calculate the estimates for $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}$ and then plug the estimates, $\boldsymbol{m}_i$, $\mathbf{S}$, in equation 10.12 and calculate the parameters of the linear discriminant.

Let us again see the special case where there are two classes. We define $y \equiv P(C_1|\boldsymbol{x})$ and $P(C_2|\boldsymbol{x}) = 1 - y$. Then in classification, we

$$\text{choose } C_1 \text{ if } \begin{cases} y > 0.5 \\ \frac{y}{1-y} > 1 \\ \log \frac{y}{1-y} > 0 \end{cases} \quad \text{and } C_2 \text{ otherwise}$$

LOGIT
LOG ODDS

$\log y/(1-y)$ is known as the *logit* transformation or *log odds* of y. In the case of two normal classes sharing a common covariance matrix, the log odds is linear:

$$\begin{aligned}\text{logit}(P(C_1|\boldsymbol{x})) &= \log\frac{P(C_1|\boldsymbol{x})}{1-P(C_1|\boldsymbol{x})} = \log\frac{P(C_1|\boldsymbol{x})}{P(C_2|\boldsymbol{x})}\\ &= \log\frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} + \log\frac{P(C_1)}{P(C_2)}\\ &= \log\frac{(2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2}\exp[-(1/2)(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)]}{(2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2}\exp[-(1/2)(\boldsymbol{x}-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2)]} + \log\frac{P(C_1)}{P(C_2)}\\ &= \boldsymbol{w}^T\boldsymbol{x} + w_0 \qquad (10.13)\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{w} &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\\ w_0 &= -\frac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) + \log\frac{P(C_1)}{P(C_2)} \qquad (10.14)\end{aligned}$$

The inverse of logit

$$\log\frac{P(C_1|\boldsymbol{x})}{1-P(C_1|\boldsymbol{x})} = \boldsymbol{w}^T\boldsymbol{x} + w_0$$

LOGISTIC
SIGMOID

is the *logistic* function, also called the *sigmoid* function (see figure 10.5):

$$P(C_1|\boldsymbol{x}) = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x} + w_0) = \frac{1}{1+\exp[-(\boldsymbol{w}^T\boldsymbol{x}+w_0)]} \qquad (10.15)$$

During training, we estimate $\boldsymbol{m}_1, \boldsymbol{m}_2, \mathbf{S}$ and plug these estimates in equation 10.14 to calculate the discriminant parameters. During testing, given $\boldsymbol{x}$, we can either

1. calculate $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$ and choose C_1 if $g(\boldsymbol{x}) > 0$, or
2. calculate $y = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x} + w_0)$ and choose C_1 if $y > 0.5$,

because $\text{sigmoid}(0) = 0.5$. In this latter case, sigmoid transforms the discriminant value to a posterior probability. This is valid when there are two classes and one discriminant; we see in section 10.7 how we can estimate posterior probabilities for $K > 2$.

10.6 Gradient Descent

In likelihood-based classification, the parameters were the sufficient statistics of $p(\boldsymbol{x}|C_i)$ and $P(C_i)$, and the method we used to estimate the parameters is maximum likelihood. In the discriminant-based approach,

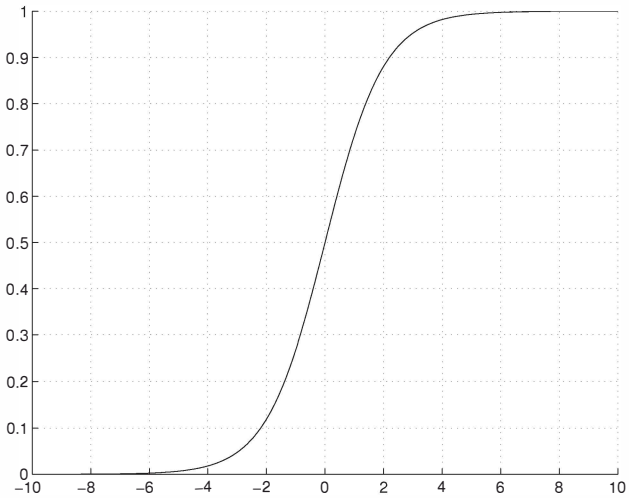

Figure 10.5 The logistic, or sigmoid, function.

the parameters are those of the discriminants, and they are optimized to minimize the classification error on the training set. When $\mathbf{w}$ denotes the set of parameters and $E(\mathbf{w}|\mathcal{X})$ is the error with parameters $\mathbf{w}$ on the given training set $\mathcal{X}$, we look for

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{w}|\mathcal{X})$$

In many cases, some of which we will see shortly, there is no analytical solution and we need to resort to iterative optimization methods, the most commonly employed being that of *gradient descent*. When $E(\mathbf{w})$ is a differentiable function of a vector of variables, we have the *gradient vector* composed of the partial derivatives

GRADIENT DESCENT
GRADIENT VECTOR

$$\nabla_w E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_d}\right]^T$$

and the *gradient descent* procedure to minimize E starts from a random $\mathbf{w}$, and at each step, updates $\mathbf{w}$, in the opposite direction of the gradient

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \forall i \tag{10.16}$$

$$w_i = w_i + \Delta w_i \tag{10.17}$$

where η is called the *stepsize*, or *learning factor*, and determines how much to move in that direction. Gradient ascent is used to maximize a

function and goes in the direction of the gradient. When we get to a minimum (or maximum), the derivative is 0 and the procedure terminates. This indicates that the procedure finds the nearest minimum that can be a local minimum, and there is no guarantee of finding the global minimum unless the function has only one minimum. The use of a good value for η is also critical; if it is too small, the convergence may be too slow, and a large value may cause oscillations and even divergence.

Throughout this book, we use gradient methods that are simple and quite effective. We keep in mind, however, that once a suitable model and an error function is defined, the optimization of the model parameters to minimize the error function can be done by using one of many possible techniques. There are second-order methods and conjugate gradient that converge faster, at the expense of more memory and computation. More costly methods like simulated annealing and genetic algorithms allow a more thorough search of the parameter space and do not depend as much on the initial point.

10.7 Logistic Discrimination

10.7.1 Two Classes

LOGISTIC DISCRIMINATION

In *logistic discrimination*, we do not model the class-conditional densities, $p(\boldsymbol{x}|C_i)$, but rather their ratio. Let us again start with two classes and assume that the log likelihood ratio is linear:

$$\log \frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} = \boldsymbol{w}^T\boldsymbol{x} + w_0^o \tag{10.18}$$

This indeed holds when the class-conditional densities are normal (equation 10.13). But logistic discrimination has a wider scope of applicability; for example, $\boldsymbol{x}$ may be composed of discrete attributes or may be a mixture of continuous and discrete attributes.

Using Bayes' rule, we have

$$\begin{aligned} \text{logit}(P(C_1|\boldsymbol{x})) &= \log \frac{P(C_1|\boldsymbol{x})}{1 - P(C_1|\boldsymbol{x})} \\ &= \log \frac{p(\boldsymbol{x}|C_1)}{p(\boldsymbol{x}|C_2)} + \log \frac{P(C_1)}{P(C_2)} \\ &= \boldsymbol{w}^T\boldsymbol{x} + w_0 \end{aligned} \tag{10.19}$$

where

$$w_0 = w_0^o + \log \frac{P(C_1)}{P(C_2)} \tag{10.20}$$

Rearranging terms, we get the sigmoid function again:

$$y = \hat{P}(C_1|\boldsymbol{x}) = \frac{1}{1 + \exp[-(\boldsymbol{w}^T\boldsymbol{x} + w_0)]} \tag{10.21}$$

as our estimator of $P(C_1|\boldsymbol{x})$.

Let us see how we can learn $\boldsymbol{w}$ and w_0. We are given a sample of two classes, $\mathcal{X} = \{\boldsymbol{x}^t, r^t\}$, where $r^t = 1$ if $\boldsymbol{x} \in C_1$ and $r^t = 0$ if $\boldsymbol{x} \in C_2$. We assume r^t, given $\boldsymbol{x}^t$, is Bernoulli with probability $y^t \equiv P(C_1|\boldsymbol{x}^t)$ as calculated in equation 10.21:

$$r^t|\boldsymbol{x}^t \sim \text{Bernoulli}(y^t)$$

Here, we see the difference from the likelihood-based methods where we modeled $p(\boldsymbol{x}|C_i)$; in the discriminant-based approach, we model directly $r|\boldsymbol{x}$. The sample likelihood is

$$l(\boldsymbol{w}, w_0|\mathcal{X}) = \prod_t (y^t)^{(r^t)}(1 - y^t)^{(1-r^t)} \tag{10.22}$$

We know that when we have a likelihood function to maximize, we can always turn it into an error function to be minimized as $E = -\log l$, and in our case, we have *cross-entropy*:

CROSS-ENTROPY

$$E(\boldsymbol{w}, w_0|\mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t)\log(1 - y^t) \tag{10.23}$$

Because of the nonlinearity of the sigmoid function, we cannot solve directly and we use gradient descent to minimize cross-entropy, equivalent to maximizing the likelihood or the log likelihood. If $y = \text{sigmoid}(a) = 1/(1 + \exp(-a))$, its derivative is given as

$$\frac{dy}{da} = y(1 - y)$$

and we get the following update equations:

$$\begin{aligned}
\Delta w_j &= -\eta \frac{\partial E}{\partial w_j} = \eta \sum_t \left(\frac{r^t}{y^t} - \frac{1 - r^t}{1 - y^t} \right) y^t(1 - y^t)x_j^t \\
&= \eta \sum_t (r^t - y^t)x_j^t, j = 1, \ldots, d \\
\Delta w_0 &= -\eta \frac{\partial E}{\partial w_0} = \eta \sum_t (r^t - y^t)
\end{aligned} \tag{10.24}$$

```
For j = 0, ..., d
    w_j ← rand(−0.01, 0.01)
Repeat
    For j = 0, ..., d
        Δw_j ← 0
    For t = 1, ..., N
        o ← 0
        For j = 0, ..., d
            o ← o + w_j x_j^t
        y ← sigmoid(o)
        For j = 0, ..., d
            Δw_j ← Δw_j + (r^t − y) x_j^t
    For j = 0, ..., d
        w_j ← w_j + η Δw_j
Until convergence
```

Figure 10.6 Logistic discrimination algorithm implementing gradient descent for the single output case with two classes. For w_0, we assume that there is an extra input x_0, which is always $+1$: $x_0^t \equiv +1, \forall t$.

It is best to initialize w_j with random values close to 0; generally they are drawn uniformly from the interval $[-0.01, 0.01]$. The reason for this is that if the initial w_j are large in magnitude, the weighted sum may also be large and may saturate the sigmoid. We see from figure 10.5 that if the initial weights are close to 0, the sum will stay in the middle region where the derivative is nonzero and an update can take place. If the weighted sum is large in magnitude (smaller than -5 or larger than $+5$), the derivative of the sigmoid will be almost 0 and weights will not be updated.

Pseudocode is given in figure 10.6. We see an example in figure 10.7 where the input is one-dimensional. Both the line $wx + w_0$ and its value after the sigmoid are shown as a function of learning iterations. We see that to get outputs of 0 and 1, the sigmoid hardens, which is achieved by increasing the magnitude of w, or $\|\boldsymbol{w}\|$ in the multivariate case.

Once training is complete and we have the final $\boldsymbol{w}$ and w_0, during testing, given $\boldsymbol{x}^t$, we calculate $y^t = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x}^t + w_0)$ and we choose C_1 if $y^t > 0.5$ and choose C_2 otherwise. This implies that to minimize the number of misclassifications, we do not need to continue learning un-

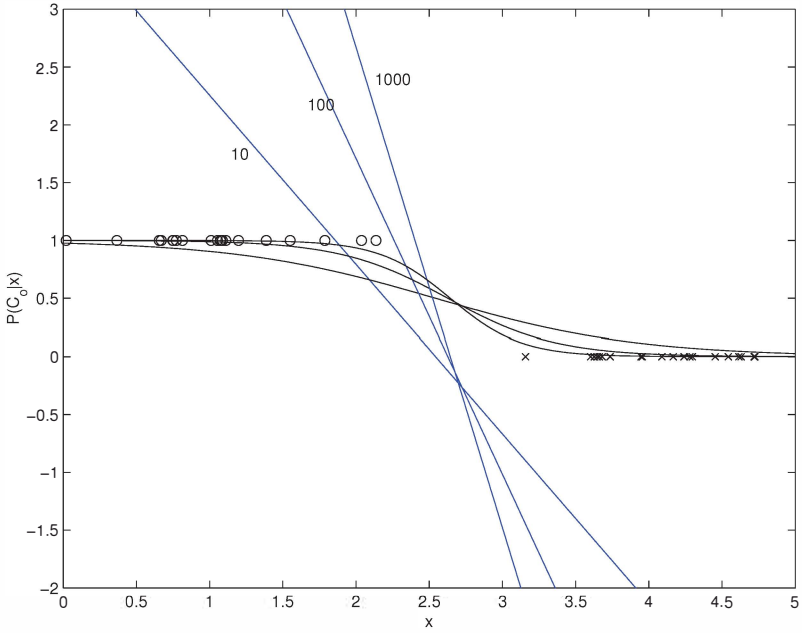

Figure 10.7 For a univariate two-class problem (shown with '$\circ$' and '$\times$'), the evolution of the line $wx + w_0$ and the sigmoid output after 10, 100, and 1,000 iterations over the sample.

til all y^t are 0 or 1, but only until y^t are less than or greater than 0.5, that is, on the correct side of the decision boundary. If we do continue training beyond this point, cross-entropy will continue decreasing ($|w_j|$ will continue increasing to harden the sigmoid), but the number of misclassifications will not decrease. Generally, we continue training until the number of misclassifications does not decrease (which will be 0 if the classes are linearly separable). Actually *stopping early* before we have 0 training error is a form of regularization. Because we start with weights almost 0 and they move away as training continues, stopping early corresponds to a model with more weights close to 0 and effectively fewer parameters.

EARLY STOPPING

Note that though we assumed the log ratio of the class densities are linear to derive the discriminant, we estimate directly the posterior and never explicitly estimate $p(\boldsymbol{x}|C_i)$ or $P(C_i)$.

10.7.2 Multiple Classes

Let us now generalize to $K > 2$ classes. We take one of the classes, for example, C_K, as the reference class and assume that

$$\log \frac{p(\mathbf{x}|C_i)}{p(\mathbf{x}|C_K)} = \mathbf{w}_i^T \mathbf{x} + w_{i0}^o \tag{10.25}$$

Then we have

$$\frac{P(C_i|\mathbf{x})}{P(C_K|\mathbf{x})} = \exp[\mathbf{w}_i^T \mathbf{x} + w_{i0}] \tag{10.26}$$

with $w_{i0} = w_{i0}^o + \log P(C_i)/P(C_K)$.

We see that

$$\begin{aligned} \sum_{i=1}^{K-1} \frac{P(C_i|\mathbf{x})}{P(C_K|\mathbf{x})} &= \frac{1 - P(C_K|\mathbf{x})}{P(C_K|\mathbf{x})} = \sum_{i=1}^{K-1} \exp[\mathbf{w}_i^T \mathbf{x} + w_{i0}] \\ \Rightarrow \quad P(C_K|\mathbf{x}) &= \frac{1}{1 + \sum_{i=1}^{K-1} \exp[\mathbf{w}_i^T \mathbf{x} + w_{i0}]} \end{aligned} \tag{10.27}$$

and also that

$$\begin{aligned} \frac{P(C_i|\mathbf{x})}{P(C_K|\mathbf{x})} &= \exp[\mathbf{w}_i^T \mathbf{x} + w_{i0}] \\ \Rightarrow \quad P(C_i|\mathbf{x}) &= \frac{\exp[\mathbf{w}_i^T \mathbf{x} + w_{i0}]}{1 + \sum_{j=1}^{K-1} \exp[\mathbf{w}_j^T \mathbf{x} + w_{j0}]}, \; i = 1, \ldots, K-1 \end{aligned} \tag{10.28}$$

To treat all classes uniformly, we can write

$$y_i = \hat{P}(C_i|\mathbf{x}) = \frac{\exp[\mathbf{w}_i^T \mathbf{x} + w_{i0}]}{\sum_{j=1}^{K} \exp[\mathbf{w}_j^T \mathbf{x} + w_{j0}]}, \; i = 1, \ldots, K \tag{10.29}$$

SOFTMAX

which is called the *softmax* function (Bridle 1990). If the weighted sum for one class is sufficiently larger than for the others, after it is boosted through exponentiation and normalization, its corresponding y_i will be close to 1 and the others will be close to 0. Thus it works like taking a maximum, except that it is differentiable; hence the name softmax. Softmax also guarantees that $\sum_i y_i = 1$.

Let us see how we can learn the parameters. In this case of $K > 2$ classes, each sample point is a multinomial trial with one draw; that is, $\mathbf{r}^t|\mathbf{x}^t \sim \text{Mult}_K(1, \mathbf{y}^t)$, where $y_i^t \equiv P(C_i|\mathbf{x}^t)$. The sample likelihood is

$$l(\{\mathbf{w}_i, w_{i0}\}_i|\mathcal{X}) = \prod_t \prod_i (y_i^t)^{r_i^t} \tag{10.30}$$

and the error function is again cross-entropy:

$$E(\{\mathbf{w}_i, w_{i0}\}_i|\mathcal{X}) = -\sum_t \sum_i r_i^t \log y_i^t \tag{10.31}$$

We again use gradient descent. If $y_i = \exp(a_i)/\sum_j \exp(a_j)$, we have

$$\frac{\partial y_i}{\partial a_j} = y_i(\delta_{ij} - y_j) \tag{10.32}$$

where δ_{ij} is the Kronecker delta, which is 1 if $i = j$ and 0 if $i \neq j$ (exercise 3). Given that $\sum_i r_i^t = 1$, we have the following update equations, for $j = 1, \ldots, K$

$$\begin{aligned}
\Delta \mathbf{w}_j &= \eta \sum_t \sum_i \frac{r_i^t}{y_i^t} y_i^t (\delta_{ij} - y_j^t)\mathbf{x}^t \\
&= \eta \sum_t \sum_i r_i^t (\delta_{ij} - y_j^t)\mathbf{x}^t \\
&= \eta \sum_t \left[\sum_i r_i^t \delta_{ij} - y_j^t \sum_i r_i^t \right] \mathbf{x}^t \\
&= \eta \sum_t (r_j^t - y_j^t)\mathbf{x}^t \\
\Delta w_{j0} &= \eta \sum_t (r_j^t - y_j^t) \qquad (10.33)
\end{aligned}$$

Note that because of the normalization in softmax, $\mathbf{w}_j$ and w_{j0} are affected not only by $\mathbf{x}^t \in C_j$ but also by $\mathbf{x}^t \in C_i, i \neq j$. The discriminants are updated so that the correct class has the highest weighted sum after softmax, and the other classes have their weighted sums as low as possible. Pseudocode is given in figure 10.8. For a two-dimensional example with three classes, the contour plot is given in figure 10.9, and the discriminants and the posterior probabilities in figure 10.10.

During testing, we calculate all $y_k, k = 1, \ldots, K$ and choose C_i if $y_i = \max_k y_k$. Again we do not need to continue training to minimize cross-entropy as much as possible; we train only until the correct class has the highest weighted sum, and therefore we can stop training earlier by checking the number of misclassifications.

When data are normally distributed, the logistic discriminant has a comparable error rate to the parametric, normal-based linear discriminant (McLachlan 1992). Logistic discrimination can still be used when the class-conditional densities are nonnormal or when they are not unimodal, as long as classes are linearly separable.

```
For i = 1, ..., K
    For j = 0, ..., d
        w_ij ← rand(−0.01, 0.01)
Repeat
    For i = 1, ..., K
        For j = 0, ..., d
            Δw_ij ← 0
    For t = 1, ..., N
        For i = 1, ..., K
            o_i ← 0
            For j = 0, ..., d
                o_i ← o_i + w_ij x_j^t
        For i = 1, ..., K
            y_i ← exp(o_i) / Σ_k exp(o_k)
        For i = 1, ..., K
            For j = 0, ..., d
                Δw_ij ← Δw_ij + (r_i^t − y_i) x_j^t
    For i = 1, ..., K
        For j = 0, ..., d
            w_ij ← w_ij + η Δw_ij
Until convergence
```

Figure 10.8 Logistic discrimination algorithm implementing gradient descent for the case with $K > 2$ classes. For generality, we take $x_0^t \equiv 1, \forall t$.

The ratio of class-conditional densities is of course not restricted to be linear (Anderson 1982; McLachlan 1992). Assuming a quadratic discriminant, we have

$$\log \frac{p(\boldsymbol{x}|C_i)}{p(\boldsymbol{x}|C_K)} = \boldsymbol{x}^T \mathbf{W}_i \boldsymbol{x} + \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} \tag{10.34}$$

corresponding to and generalizing parametric discrimination with multivariate normal class-conditionals having different covariance matrices. When d is large, just as we can simplify (regularize) $\boldsymbol{\Sigma}_i$, we can equally do it on $\mathbf{W}_i$ by taking only its leading eigenvectors into account.

As discussed in section 10.2, any specified function of the basic variables can be included as x-variates. One can, for example, write the dis-

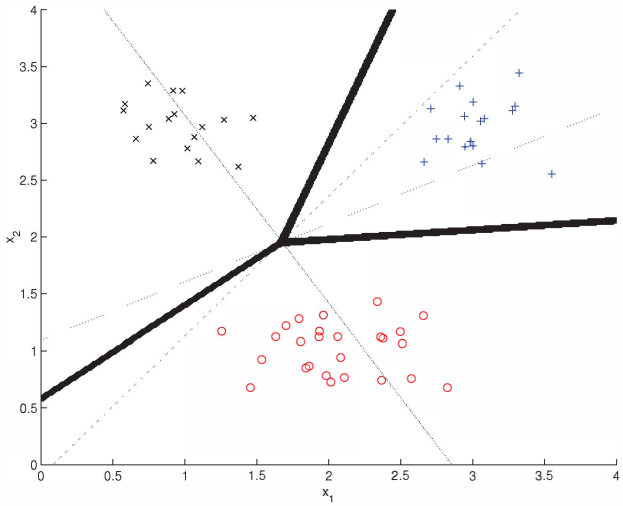

Figure 10.9 For a two-dimensional problem with three classes, the solution found by logistic discrimination. Thin lines are where $g_i(\boldsymbol{x}) = 0$, and the thick line is the boundary induced by the linear classifier choosing the maximum.

criminant as a linear sum of nonlinear basis functions

$$\log \frac{p(\boldsymbol{x}|C_i)}{p(\boldsymbol{x}|C_K)} = \boldsymbol{w}_i^T \boldsymbol{\phi}(\boldsymbol{x}) + w_{i0} \tag{10.35}$$

where $\boldsymbol{\phi}(\cdot)$ are the basis functions, which can be viewed as transformed variables. In neural network terminology, this is called a *multilayer perceptron* (chapter 11), and sigmoid is the most popular basis function. When a Gaussian basis function is used, the model is called *radial basis functions* (chapter 12). We can even use a completely nonparametric approach, for example, Parzen windows (chapter 8).

10.8 Discrimination by Regression

In regression, the probabilistic model is

$$r^t = y^t + \epsilon \tag{10.36}$$

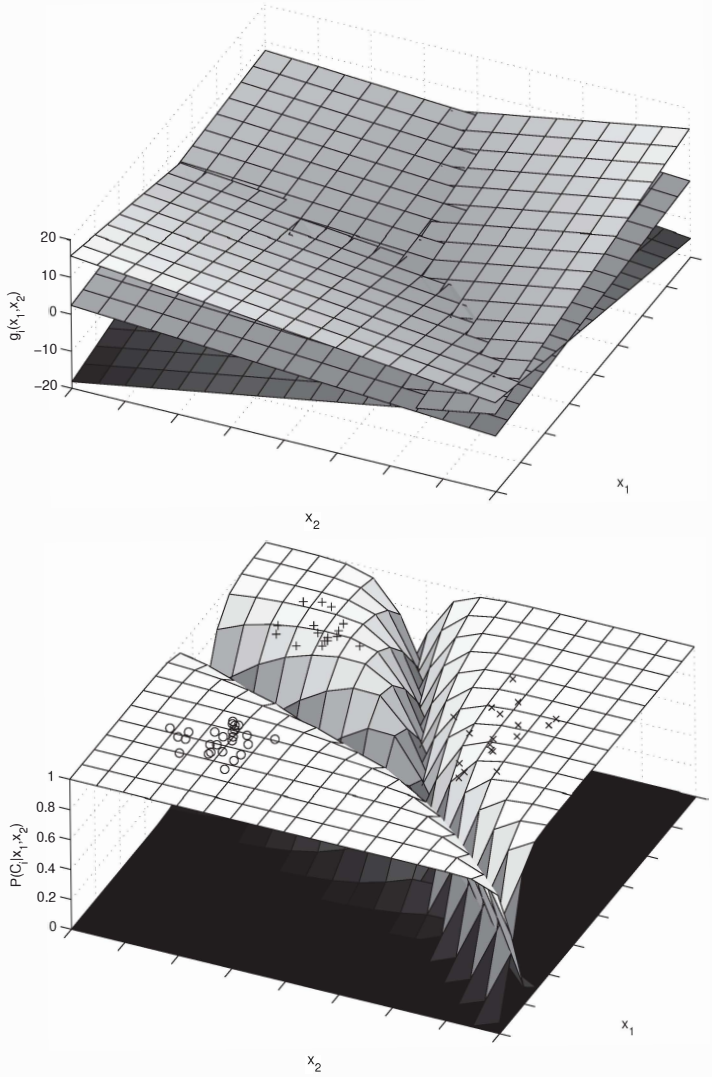

Figure 10.10 For the same example in figure 10.9, the linear discriminants (top), and the posterior probabilities after the softmax (bottom).

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. If $r^t \in \{0, 1\}$, y^t can be constrained to lie in this range using the sigmoid function. Assuming a linear model and two classes, we have

$$y^t = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x}^t + w_0) = \frac{1}{1 + \exp[-(\boldsymbol{w}^T \boldsymbol{x}^t + w_0)]} \tag{10.37}$$

Then the sample likelihood in regression, assuming $r|\boldsymbol{x} \sim \mathcal{N}(y, \sigma^2)$, is

$$l(\boldsymbol{w}, w_0|\mathcal{X}) = \prod_t \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(r^t - y^t)^2}{2\sigma^2}\right] \tag{10.38}$$

Maximizing the log likelihood is minimizing the sum of square errors:

$$E(\boldsymbol{w}, w_0|\mathcal{X}) = \frac{1}{2}\sum_t (r^t - y^t)^2 \tag{10.39}$$

Using gradient descent, we get

$$\begin{aligned} \Delta \boldsymbol{w} &= \eta \sum_t (r^t - y^t) y^t (1 - y^t) \boldsymbol{x}^t \\ \Delta w_0 &= \eta \sum_t (r^t - y^t) y^t (1 - y^t) \end{aligned} \tag{10.40}$$

This method can also be used when there are $K > 2$ classes. The probabilistic model is

$$\boldsymbol{r}^t = \boldsymbol{y}^t + \boldsymbol{\epsilon} \tag{10.41}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}_K(0, \sigma^2 \mathbf{I}_K)$. Assuming a linear model for each class, we have

$$y_i^t = \text{sigmoid}(\boldsymbol{w}_i^T \boldsymbol{x}^t + w_{i0}) = \frac{1}{1 + \exp[-(\boldsymbol{w}_i^T \boldsymbol{x}^t + w_{i0})]} \tag{10.42}$$

Then the sample likelihood is

$$l(\{\boldsymbol{w}_i, w_{i0}\}_i|\mathcal{X}) = \prod_t \frac{1}{(2\pi)^{K/2}|\Sigma|^{1/2}} \exp\left[-\frac{\|\boldsymbol{r}^t - \boldsymbol{y}^t\|^2}{2\sigma^2}\right] \tag{10.43}$$

and the error function is

$$E(\{\boldsymbol{w}_i, w_{i0}\}_i|\mathcal{X}) = \frac{1}{2}\sum_t \|\boldsymbol{r}^t - \boldsymbol{y}^t\|^2 = \frac{1}{2}\sum_t \sum_i (r_i^t - y_i^t)^2 \tag{10.44}$$

The update equations for $i = 1, \ldots, K$, are

$$\begin{aligned} \Delta \boldsymbol{w}_i &= \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t) \boldsymbol{x}^t \\ \Delta w_{i0} &= \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t) \end{aligned} \tag{10.45}$$

But note that in doing so, we do not make use of the information that only one of y_i needs to be 1 and all others are 0, or that $\sum_i y_i = 1$. The softmax function of equation 10.29 allows us to incorporate this extra information we have due to the outputs' estimating class posterior probabilities. Using sigmoid outputs in $K > 2$ case, we treat y_i as if they are independent functions.

Note also that for a given class, if we use the regression approach, there will be updates until the right output is 1 and all others are 0. This is not in fact necessary because during testing, we are just going to choose the maximum anyway; it is enough to train only until the right output is larger than others, which is exactly what the softmax function does.

So this approach with multiple sigmoid outputs is more appropriate when the classes are *not* mutually exclusive and exhaustive. That is, for an $\boldsymbol{x}^t$, all r_i^t may be 0; namely, $\boldsymbol{x}^t$ does not belong to any of the classes, or more than one r_i^t may be 1, when classes overlap.

10.9 Learning to Rank

RANKING

Ranking is an application area of machine learning that is different from classification and regression, and is sort of between the two. Unlike classification and regression where there is an input $\boldsymbol{x}^t$ and a desired output r^t, in ranking we are asked to put two or more instances in the correct order (Liu 2011).

For example, let us say $\boldsymbol{x}^u$ and $\boldsymbol{x}^v$ represent two movies, and let us say that a user has enjoyed u more than v (in this case, we need to give higher rank to movies similar to u). This is labeled as $r^u \prec r^v$. What we learn is not a discriminant or a regression function but a *score function* $g(\boldsymbol{x}|\theta)$, and what is important are not the absolute values of $g(\boldsymbol{x}^u|\theta)$ and $g(\boldsymbol{x}^v|\theta)$, but that we need to give a higher score to $\boldsymbol{x}^u$ than $\boldsymbol{x}^v$; that is, $g(\boldsymbol{x}^u|\theta) > g(\boldsymbol{x}^v|\theta)$ should be satisfied for all such pairs of u and v.

As usual, we assume a certain model $g(\cdot)$ and we optimize its parameters θ so that all rank constraints are satisfied. Then, for example, to make a recommendation among the movies that the user has not yet seen, we choose the one with the highest score:

$$\text{Choose } u \text{ if } g(\boldsymbol{x}^u|\theta) = \max_t g(\boldsymbol{x}^t|\theta)$$

Sometimes, instead of only the topmost, we may want a list of the highest k.

We can note here the advantage and difference of a ranker. If users rate the movies they have seen as "enjoyed/not enjoyed," this will be a two-class classification problem and a classifier can be used, but taste is nuanced and a binary rating is very hard. On the other hand, if people rate their enjoyment of a movie on a scale of, say, 1 to 10, this will be a regression problem, but such absolute values are difficult to assign. It is more natural and easier for people to say that of the two movies they have watched, they like one more than the other, instead of a yes/no decision or a numeric value.

Ranking has many applications. In search engines, for example, given a query, we want to retrieve the most relevant documents. If we retrieve and display the current top ten candidates and then the user clicks the third one skipping the first two, we understand that the third should have been ranked higher than the first and the second. Such click logs are used to train rankers.

Sometimes reranking is used to improve the output of a ranker with additional information. For example, in speech recognition, an acoustic model can first be used to generate an ordered list of possible sentences, and then the N-best candidates can then be reranked using features from a language model; this can improve accuracy significantly (Shen and Joshi 2005).

A ranker can be trained in a number of different ways. For all (u, v) pairs where $\boldsymbol{r}^u \prec \boldsymbol{r}^v$ is defined, we have an error if $g(\boldsymbol{x}^v|\theta) > g(\boldsymbol{x}^u|\theta)$. Generally, we do not have a full ordering of all N^2 pairs but a subset, thereby defining a partial order. The sum of differences make up the error:

$$E(\boldsymbol{w}|\{\boldsymbol{r}^u, \boldsymbol{r}^v\}) = \sum_{\boldsymbol{r}^u \prec \boldsymbol{r}^v} \left[g(\boldsymbol{x}^v|\theta) - g(\boldsymbol{x}^u|\theta)\right]_+ \tag{10.46}$$

where a_+ is equal to a if $a \geq 0$ and 0 otherwise.

Let us use a linear model, as we do throughout this chapter:

$$g(\boldsymbol{x}|\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x} \tag{10.47}$$

Because we do not care about the absolute values, we do not need w_0. The error in equation 10.46 becomes

$$E(\boldsymbol{w}|\{\boldsymbol{r}^u, \boldsymbol{r}^v\}) = \sum_{\boldsymbol{r}^u \prec \boldsymbol{r}^v} \boldsymbol{w}^T(\boldsymbol{x}^v - \boldsymbol{x}^u)_+ \tag{10.48}$$

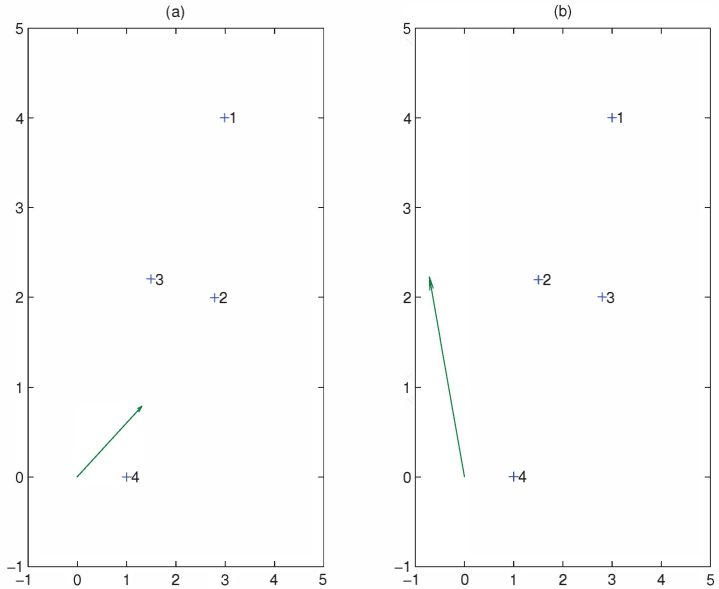

Figure 10.11 Sample ranking problems and solutions. Data points are indicated by '+' and the numbers next to them indicate the rank where 1 is the highest. We have a full ordering here. The arrow indicate the learned $\mathbf{w}$. In (a) and (b), we see two different ranking problems and the two corresponding solutions.

We can do an online update of $\mathbf{w}$ using gradient descent. For each $r^u \prec r^v$ where $g(\mathbf{x}^v|\theta) > g(\mathbf{x}^u|\theta)$, we do a small update:

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = -\eta(x_j^v - x_j^u), j = 1, \ldots, d \tag{10.49}$$

$\mathbf{w}$ is chosen so that when the instances are projected onto $\mathbf{w}$, the correct orderings are obtained. In figure 10.11, we see example data and the projection directions learned. We see there that a slight change in the ranks may cause a big change in $\mathbf{w}$.

For error functions and gradient descent approaches in ranking and their use in practice, see Burges et al. 2005 and Shin and Josh 2005. Sometimes for confident decision, when $r^u \prec r^v$, we require that the output be not only larger, but larger with a margin, for example, $g(\mathbf{x}^u|\theta) > 1 + g(\mathbf{x}^u|\theta)$; we will see an example of this when we talk about learning to rank using kernel machines in section 13.11.

10.10 Notes

The linear discriminant, due to its simplicity, is the classifier most used in pattern recognition (Duda, Hart, and Stork 2001; McLachlan 1992). We discussed the case of Gaussian distributions with a common covariance matrix in chapter 4 and Fisher's linear discriminant in chapter 6, and in this chapter we discuss the logistic discriminant. In chapter 11, we discuss the perceptron that is the neural network implementation of the linear discriminant. In chapter 13, we discuss support vector machines, another type of linear discriminant.

Logistic discrimination is covered in more detail in Anderson 1982 and in McLachlan 1992. Logistic (sigmoid) is the inverse of logit, which is the *canonical link* in case of Bernoulli samples. Softmax is its generalization to multinomial samples. More information on such *generalized linear models* is given in McCullogh and Nelder 1989.

GENERALIZED LINEAR MODELS

Ranking has recently become a major application area of machine learning because of its use in search engines, information retrieval, and natural language processing. An extensive review of both important applications and machine learning algorithms is given in Liu 2011. The model we discussed here is a linear model; in section 13.11, we discuss how to learn a ranker using kernel machines where we get a nonlinear model with kernels that allow the integration of different measures of similarity.

Generalizing linear models by using nonlinear basis functions is a very old idea. We will discuss multilayer perceptrons (chapter 11) and radial basis functions (chapter 12) where the parameters of the basis functions can also be learned from data while learning the discriminant. Support vector machines (chapter 13) use kernel functions built from such basis functions.

10.11 Exercises

1. For each of the following basis functions, describe where it is nonzero:
   a. $\sin(x_1)$
   b. $\exp(-(x_1 - a)^2/c)$
   c. $\exp(-\|\boldsymbol{x} - \boldsymbol{a}\|^2/c)$
   d. $\log(x_2)$
   e. $1(x_1 > c)$

f. $1(ax_1 + bx_2 > c)$

2. For the two-dimensional case of figure 10.2, show equations 10.4 and 10.5.

3. Show that the derivative of the softmax, $y_i = \exp(a_i) / \sum_j \exp(a_j)$, is $\partial y_i / \partial a_j = y_i(\delta_{ij} - y_j)$, where δ_{ij} is 1 if $i = j$ and 0 otherwise.

4. With $K = 2$, show that using two softmax outputs is equal to using one sigmoid output.

   SOLUTION:

$$\begin{aligned} y_1 &= \frac{\exp o_1}{\exp o_1 + \exp o_2} = \frac{1}{1 + \exp(o_2 - o_1)} = \frac{1}{1 + \exp(-(o_1 - o_2))} \\ &= \text{sigmoid}(o_1 - o_2) \end{aligned}$$

   For example, if we have $o_1 = \boldsymbol{w}_1^T \boldsymbol{x}$, we have

$$y_1 = \frac{\exp \boldsymbol{w}_1^T \boldsymbol{x}}{\exp \boldsymbol{w}_1^T \boldsymbol{x} + \exp \boldsymbol{w}_2^T \boldsymbol{x}} = \text{sigmoid}(\boldsymbol{w}_1^T \boldsymbol{x} - \boldsymbol{w}_2^T \boldsymbol{x}) = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x})$$

   where $\boldsymbol{w} \equiv \boldsymbol{w}_1 - \boldsymbol{w}_2$ and $y_2 = 1 - y_1$.

5. How can we learn $\mathbf{W}_i$ in equation 10.34?

   SOLUTION: For example, if we have two inputs x_1 and x_2, we have

$$\begin{aligned} \log \frac{p(x_1, x_2|C_i)}{p(x_1, x_2|C_K)} &= \mathbf{W}_{i11} x_1^2 + \mathbf{W}_{i12} x_1 x_2 + \mathbf{W}_{i21} x_2 x_1 + \mathbf{W}_{i22} x_2^2 \\ &+ w_{i1} x_1 + w_{i2} x_2 + w_{i0} \end{aligned}$$

   Then we can use gradient descent and take derivative with respect to any $\mathbf{W}_{jkl}$ to calculate an update rule:

$$\Delta \mathbf{W}_{jkl} = \eta \sum_t (r_j^t - y_j^t) x_k^t x_l^t$$

6. In using quadratic (or higher-order) discriminants as in equation 10.34, how can we keep variance under control?

7. What is the implication of the use of a single η for all x_j in gradient descent?

   SOLUTION: Using a single η for all x_j implies doing updates in the same scale, which in turn implies that all x_j are in the same scale. If they are not, it is a good idea to normalize all x_j, for example, by z-normalization, before training. Note that we need to save the scaling parameters for all inputs, so that the same scaling can also be done later to the test instances.

8. In the univariate case for classification as in figure 10.7, what do w and w_0 correspond to?

   SOLUTION: The slope and the intercept of the line, which are then fed to the sigmoid.

9. Let us say for univariate x, $x \in (2,4)$ belong to C_1 and $x < 2$ or $x > 4$ belong to C_2. How can we separate the two classes using a linear discriminant?

   SOLUTION: We define an extra variable $z \equiv x^2$ and use the linear discriminant $w_2 z + w_1 x + w_0$ in the (z,x) space, which corresponds to a quadratic discriminant in the x space. For example, we can manually write

$$\text{Choose } \begin{cases} C_1 & \text{if } (x-3)^2 - 1 \leq 0 \\ C_2 & \text{otherwise} \end{cases}$$

   or rewrite it using a sigmoid (see figure 10.12):

$$\text{Choose } \begin{cases} C_1 & \text{if sigmoid}((x-3)^2 - 1) \leq 0.5 \\ C_2 & \text{otherwise} \end{cases}$$

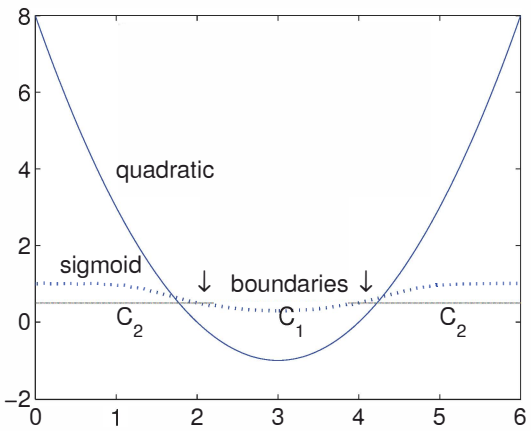

Figure 10.12 The quadratic discriminant, before and after the sigmoid. The boundaries are where the discriminant is 0 or where the sigmoid is 0.5.

   Or, we can use *two* linear discriminants in the x space, one separating at 2 and the other separating at 4, and then we can OR them. Such layered linear discriminants are discussed in chapter 11.

10. For the sample data in figure 10.11, define ranks such that a linear model would not be able to learn them. Explain how the model can be generalized so that they can be learned.

10.12 References

Aizerman, M. A., E. M. Braverman, and L. I. Rozonoer. 1964. "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning." *Automation and Remote Control* 25:821–837.

Anderson, J. A. 1982. "Logistic Discrimination." In *Handbook of Statistics*, Vol. 2, *Classification, Pattern Recognition and Reduction of Dimensionality*, ed. P. R. Krishnaiah and L. N. Kanal, 169–191. Amsterdam: North Holland.

Bridle, J. S. 1990. "Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition." In *Neurocomputing: Algorithms, Architectures and Applications*, ed. F. Fogelman-Soulie and J. Herault, 227–236. Berlin: Springer.

Burges, C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. "Learning to Rank using Gradient Descent." In *22nd International Conference on Machine Learning*, 89–96, New York: ACM Press.

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, 2nd ed. New York: Wiley.

Liu, T.-Y. 2011. *Learning to Rank for Information Retrieval*. Heidelberg: Springer.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. London: Chapman and Hall.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Shen, L., and A. K. Joshi. 2005. "Ranking and Reranking with Perceptron." *Machine Learning* 60:73–96.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. New York: Springer.