4 Linear Models for Classification

In the previous chapter, we explored a class of regression models having particularly simple analytical and computational properties. We now discuss an analogous class of models for solving classification problems. The goal in classification is to take an input vector $\mathbf{x}$ and to assign it to one of K discrete classes $\mathcal{C}_k$ where $k = 1, \ldots, K$. In the most common scenario, the classes are taken to be disjoint, so that each input is assigned to one and only one class. The input space is thereby divided into *decision regions* whose boundaries are called *decision boundaries* or *decision surfaces*. In this chapter, we consider linear models for classification, by which we mean that the decision surfaces are linear functions of the input vector $\mathbf{x}$ and hence are defined by $(D-1)$-dimensional hyperplanes within the D-dimensional input space. Data sets whose classes can be separated exactly by linear decision surfaces are said to be *linearly separable*.

For regression problems, the target variable $\mathbf{t}$ was simply the vector of real numbers whose values we wish to predict. In the case of classification, there are various ways of using target values to represent class labels. For probabilistic models, the most convenient, in the case of two-class problems, is the binary representation in which there is a single target variable $t \in \{0, 1\}$ such that $t = 1$ represents class $\mathcal{C}_1$ and $t = 0$ represents class $\mathcal{C}_2$. We can interpret the value of t as the probability that the class is $\mathcal{C}_1$, with the values of probability taking only the extreme values of 0 and 1. For $K > 2$ classes, it is convenient to use a 1-of-K coding scheme in which $\mathbf{t}$ is a vector of length K such that if the class is $\mathcal{C}_j$, then all elements t_k of $\mathbf{t}$ are zero except element t_j, which takes the value 1. For instance, if we have $K = 5$ classes, then a pattern from class 2 would be given the target vector

$$\mathbf{t} = (0, 1, 0, 0, 0)^{\mathrm{T}}. \tag{4.1}$$

Again, we can interpret the value of t_k as the probability that the class is $\mathcal{C}_k$. For nonprobabilistic models, alternative choices of target variable representation will sometimes prove convenient.

In Chapter 1, we identified three distinct approaches to the classification problem. The simplest involves constructing a *discriminant function* that directly assigns each vector $\mathbf{x}$ to a specific class. A more powerful approach, however, models the conditional probability distribution $p(\mathcal{C}_k|\mathbf{x})$ in an inference stage, and then subsequently uses this distribution to make optimal decisions. By separating inference and decision, we gain numerous benefits, as discussed in Section 1.5.4. There are two different approaches to determining the conditional probabilities $p(\mathcal{C}_k|\mathbf{x})$. One technique is to model them directly, for example by representing them as parametric models and then optimizing the parameters using a training set. Alternatively, we can adopt a generative approach in which we model the class-conditional densities given by $p(\mathbf{x}|\mathcal{C}_k)$, together with the prior probabilities $p(\mathcal{C}_k)$ for the classes, and then we compute the required posterior probabilities using Bayes' theorem

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}. \tag{4.2}$$

We shall discuss examples of all three approaches in this chapter.

In the linear regression models considered in Chapter 3, the model prediction $y(\mathbf{x}, \mathbf{w})$ was given by a linear function of the parameters $\mathbf{w}$. In the simplest case, the model is also linear in the input variables and therefore takes the form $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$, so that y is a real number. For classification problems, however, we wish to predict discrete class labels, or more generally posterior probabilities that lie in the range $(0, 1)$. To achieve this, we consider a generalization of this model in which we transform the linear function of $\mathbf{w}$ using a nonlinear function $f(\,\cdot\,)$ so that

$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0\right). \tag{4.3}$$

In the machine learning literature $f(\,\cdot\,)$ is known as an *activation function*, whereas its inverse is called a *link function* in the statistics literature. The decision surfaces correspond to $y(\mathbf{x}) = \text{constant}$, so that $\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 = \text{constant}$ and hence the decision surfaces are linear functions of $\mathbf{x}$, even if the function $f(\cdot)$ is nonlinear. For this reason, the class of models described by (4.3) are called *generalized linear models*

(McCullagh and Nelder, 1989). Note, however, that in contrast to the models used for regression, they are no longer linear in the parameters due to the presence of the nonlinear function $f(\cdot)$. This will lead to more complex analytical and computational properties than for linear regression models. Nevertheless, these models are still relatively simple compared to the more general nonlinear models that will be studied in subsequent chapters.

The algorithms discussed in this chapter will be equally applicable if we first make a fixed nonlinear transformation of the input variables using a vector of basis functions $\boldsymbol{\phi}(\mathbf{x})$ as we did for regression models in Chapter 3. We begin by considering classification directly in the original input space $\mathbf{x}$, while in Section 4.3 we shall find it convenient to switch to a notation involving basis functions for consistency with later chapters.

4.1. Discriminant Functions

A discriminant is a function that takes an input vector $\mathbf{x}$ and assigns it to one of K classes, denoted $\mathcal{C}_k$. In this chapter, we shall restrict attention to *linear discriminants*, namely those for which the decision surfaces are hyperplanes. To simplify the discussion, we consider first the case of two classes and then investigate the extension to $K > 2$ classes.

4.1.1 Two classes

The simplest representation of a linear discriminant function is obtained by taking a linear function of the input vector so that

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 \tag{4.4}$$

where $\mathbf{w}$ is called a *weight vector*, and w_0 is a *bias* (not to be confused with bias in the statistical sense). The negative of the bias is sometimes called a *threshold*. An input vector $\mathbf{x}$ is assigned to class $\mathcal{C}_1$ if $y(\mathbf{x}) \geqslant 0$ and to class $\mathcal{C}_2$ otherwise. The corresponding decision boundary is therefore defined by the relation $y(\mathbf{x}) = 0$, which corresponds to a $(D-1)$-dimensional hyperplane within the D-dimensional input space. Consider two points $\mathbf{x}_{\mathrm{A}}$ and $\mathbf{x}_{\mathrm{B}}$ both of which lie on the decision surface. Because $y(\mathbf{x}_{\mathrm{A}}) = y(\mathbf{x}_{\mathrm{B}}) = 0$, we have $\mathbf{w}^{\mathrm{T}}(\mathbf{x}_{\mathrm{A}} - \mathbf{x}_{\mathrm{B}}) = 0$ and hence the vector $\mathbf{w}$ is orthogonal to every vector lying within the decision surface, and so $\mathbf{w}$ determines the orientation of the decision surface. Similarly, if $\mathbf{x}$ is a point on the decision surface, then $y(\mathbf{x}) = 0$, and so the normal distance from the origin to the decision surface is given by

$$\frac{\mathbf{w}^{\mathrm{T}}\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}. \tag{4.5}$$

We therefore see that the bias parameter w_0 determines the location of the decision surface. These properties are illustrated for the case of $D = 2$ in Figure 4.1.

Furthermore, we note that the value of $y(\mathbf{x})$ gives a signed measure of the perpendicular distance r of the point $\mathbf{x}$ from the decision surface. To see this, consider

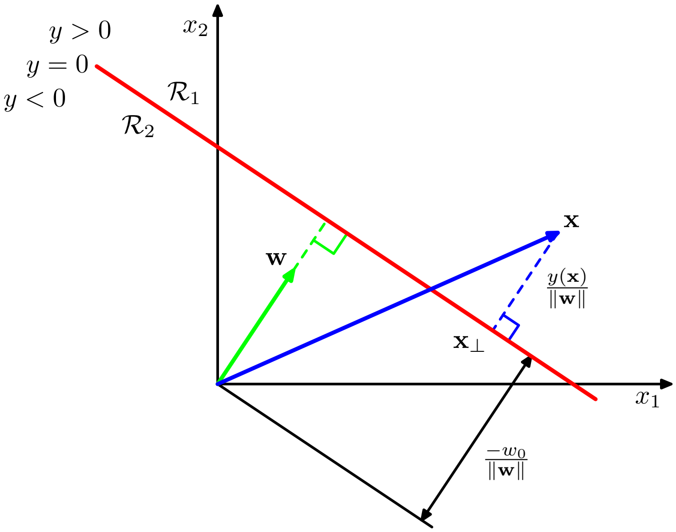

Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.

an arbitrary point $\mathbf{x}$ and let $\mathbf{x}_\perp$ be its orthogonal projection onto the decision surface, so that

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}. \tag{4.6}$$

Multiplying both sides of this result by $\mathbf{w}^{\mathrm{T}}$ and adding w_0, and making use of $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0$ and $y(\mathbf{x}_\perp) = \mathbf{w}^{\mathrm{T}}\mathbf{x}_\perp + w_0 = 0$, we have

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}. \tag{4.7}$$

This result is illustrated in Figure 4.1.

As with the linear regression models in Chapter 3, it is sometimes convenient to use a more compact notation in which we introduce an additional dummy 'input' value $x_0 = 1$ and then define $\widetilde{\mathbf{w}} = (w_0, \mathbf{w})$ and $\widetilde{\mathbf{x}} = (x_0, \mathbf{x})$ so that

$$y(\mathbf{x}) = \widetilde{\mathbf{w}}^{\mathrm{T}}\widetilde{\mathbf{x}}. \tag{4.8}$$

In this case, the decision surfaces are D-dimensional hyperplanes passing through the origin of the $D+1$-dimensional expanded input space.

4.1.2 Multiple classes

Now consider the extension of linear discriminants to $K > 2$ classes. We might be tempted be to build a K-class discriminant by combining a number of two-class discriminant functions. However, this leads to some serious difficulties (Duda and Hart, 1973) as we now show.

Consider the use of $K-1$ classifiers each of which solves a two-class problem of separating points in a particular class $\mathcal{C}_k$ from points not in that class. This is known as a *one-versus-the-rest* classifier. The left-hand example in Figure 4.2 shows an

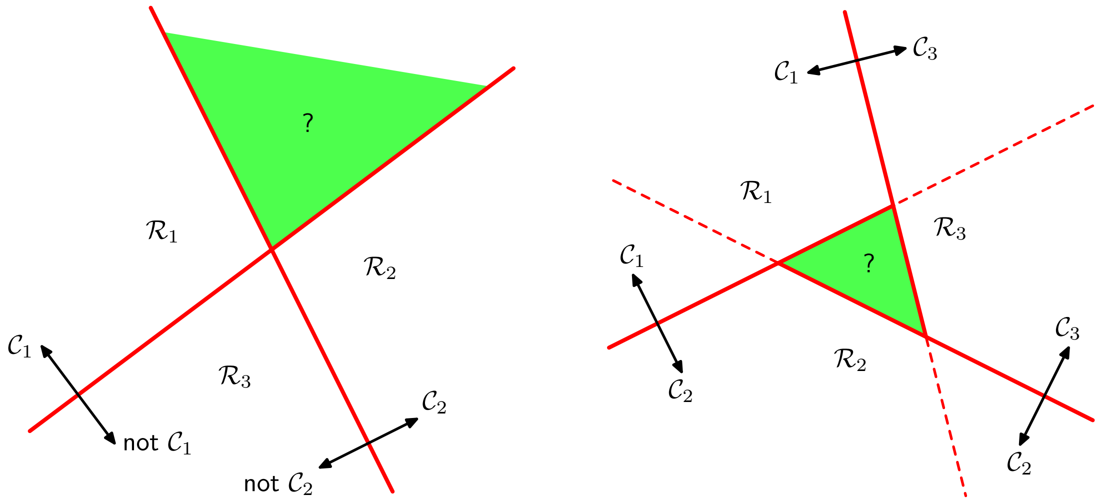

Figure 4.2 Attempting to construct a K class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class $\mathcal{C}_k$ from points not in class $\mathcal{C}_k$. On the right is an example involving three discriminant functions each of which is used to separate a pair of classes $\mathcal{C}_k$ and $\mathcal{C}_j$.

example involving three classes where this approach leads to regions of input space that are ambiguously classified.

An alternative is to introduce $K(K-1)/2$ binary discriminant functions, one for every possible pair of classes. This is known as a *one-versus-one* classifier. Each point is then classified according to a majority vote amongst the discriminant functions. However, this too runs into the problem of ambiguous regions, as illustrated in the right-hand diagram of Figure 4.2.

We can avoid these difficulties by considering a single K-class discriminant comprising K linear functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0} \tag{4.9}$$

and then assigning a point $\mathbf{x}$ to class $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$. The decision boundary between class $\mathcal{C}_k$ and class $\mathcal{C}_j$ is therefore given by $y_k(\mathbf{x}) = y_j(\mathbf{x})$ and hence corresponds to a $(D-1)$-dimensional hyperplane defined by

$$(\mathbf{w}_k - \mathbf{w}_j)^{\mathrm{T}}\mathbf{x} + (w_{k0} - w_{j0}) = 0. \tag{4.10}$$

This has the same form as the decision boundary for the two-class case discussed in Section 4.1.1, and so analogous geometrical properties apply.

The decision regions of such a discriminant are always singly connected and convex. To see this, consider two points $\mathbf{x}_{\mathrm{A}}$ and $\mathbf{x}_{\mathrm{B}}$ both of which lie inside decision region $\mathcal{R}_k$, as illustrated in Figure 4.3. Any point $\widehat{\mathbf{x}}$ that lies on the line connecting $\mathbf{x}_{\mathrm{A}}$ and $\mathbf{x}_{\mathrm{B}}$ can be expressed in the form

$$\widehat{\mathbf{x}} = \lambda\mathbf{x}_{\mathrm{A}} + (1-\lambda)\mathbf{x}_{\mathrm{B}} \tag{4.11}$$

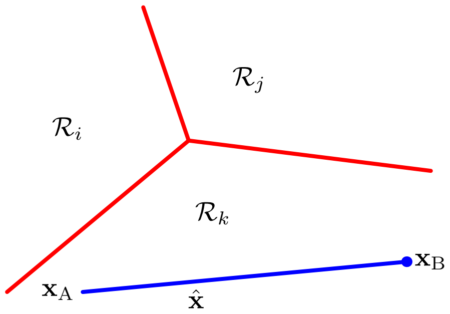

Figure 4.3 Illustration of the decision regions for a multiclass linear discriminant, with the decision boundaries shown in red. If two points $\mathbf{x}_\mathrm{A}$ and $\mathbf{x}_\mathrm{B}$ both lie inside the same decision region $\mathcal{R}_k$, then any point $\widehat{\mathbf{x}}$ that lies on the line connecting these two points must also lie in $\mathcal{R}_k$, and hence the decision region must be singly connected and convex.

where $0 \leqslant \lambda \leqslant 1$. From the linearity of the discriminant functions, it follows that

$$y_k(\widehat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_\mathrm{A}) + (1-\lambda) y_k(\mathbf{x}_\mathrm{B}). \tag{4.12}$$

Because both $\mathbf{x}_\mathrm{A}$ and $\mathbf{x}_\mathrm{B}$ lie inside $\mathcal{R}_k$, it follows that $y_k(\mathbf{x}_\mathrm{A}) > y_j(\mathbf{x}_\mathrm{A})$, and $y_k(\mathbf{x}_\mathrm{B}) > y_j(\mathbf{x}_\mathrm{B})$, for all $j \neq k$, and hence $y_k(\widehat{\mathbf{x}}) > y_j(\widehat{\mathbf{x}})$, and so $\widehat{\mathbf{x}}$ also lies inside $\mathcal{R}_k$. Thus $\mathcal{R}_k$ is singly connected and convex.

Note that for two classes, we can either employ the formalism discussed here, based on two discriminant functions $y_1(\mathbf{x})$ and $y_2(\mathbf{x})$, or else use the simpler but equivalent formulation described in Section 4.1.1 based on a single discriminant function $y(\mathbf{x})$.

We now explore three approaches to learning the parameters of linear discriminant functions, based on least squares, Fisher's linear discriminant, and the perceptron algorithm.

4.1.3 Least squares for classification

In Chapter 3, we considered models that were linear functions of the parameters, and we saw that the minimization of a sum-of-squares error function led to a simple closed-form solution for the parameter values. It is therefore tempting to see if we can apply the same formalism to classification problems. Consider a general classification problem with K classes, with a 1-of-K binary coding scheme for the target vector $\mathbf{t}$. One justification for using least squares in such a context is that it approximates the conditional expectation $\mathbb{E}[\mathbf{t}|\mathbf{x}]$ of the target values given the input vector. For the binary coding scheme, this conditional expectation is given by the vector of posterior class probabilities. Unfortunately, however, these probabilities are typically approximated rather poorly, indeed the approximations can have values outside the range $(0, 1)$, due to the limited flexibility of a linear model as we shall see shortly.

Each class $\mathcal{C}_k$ is described by its own linear model so that

$$y_k(\mathbf{x}) = \mathbf{w}_k^\mathrm{T}\mathbf{x} + w_{k0} \tag{4.13}$$

where $k = 1, \ldots, K$. We can conveniently group these together using vector notation so that

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^\mathrm{T}\widetilde{\mathbf{x}} \tag{4.14}$$

where $\widetilde{\mathbf{W}}$ is a matrix whose $k^{\rm th}$ column comprises the $D+1$-dimensional vector $\widetilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^{\rm T})^{\rm T}$ and $\widetilde{\mathbf{x}}$ is the corresponding augmented input vector $(1, \mathbf{x}^{\rm T})^{\rm T}$ with a dummy input $x_0 = 1$. This representation was discussed in detail in Section 3.1. A new input $\mathbf{x}$ is then assigned to the class for which the output $y_k = \widetilde{\mathbf{w}}_k^{\rm T}\widetilde{\mathbf{x}}$ is largest.

We now determine the parameter matrix $\widetilde{\mathbf{W}}$ by minimizing a sum-of-squares error function, as we did for regression in Chapter 3. Consider a training data set $\{\mathbf{x}_n, \mathbf{t}_n\}$ where $n = 1, \ldots, N$, and define a matrix $\mathbf{T}$ whose $n^{\rm th}$ row is the vector $\mathbf{t}_n^{\rm T}$, together with a matrix $\widetilde{\mathbf{X}}$ whose $n^{\rm th}$ row is $\widetilde{\mathbf{x}}_n^{\rm T}$. The sum-of-squares error function can then be written as

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\rm T}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}. \tag{4.15}$$

Setting the derivative with respect to $\widetilde{\mathbf{W}}$ to zero, and rearranging, we then obtain the solution for $\widetilde{\mathbf{W}}$ in the form

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\rm T}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\rm T}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T} \tag{4.16}$$

where $\widetilde{\mathbf{X}}^{\dagger}$ is the pseudo-inverse of the matrix $\widetilde{\mathbf{X}}$, as discussed in Section 3.1.1. We then obtain the discriminant function in the form

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\rm T}\widetilde{\mathbf{x}} = \mathbf{T}^{\rm T}\left(\widetilde{\mathbf{X}}^{\dagger}\right)^{\rm T}\widetilde{\mathbf{x}}. \tag{4.17}$$

An interesting property of least-squares solutions with multiple target variables is that if every target vector in the training set satisfies some linear constraint

$$\mathbf{a}^{\rm T}\mathbf{t}_n + b = 0 \tag{4.18}$$

Exercise 4.2

for some constants $\mathbf{a}$ and b, then the model prediction for any value of $\mathbf{x}$ will satisfy the same constraint so that

$$\mathbf{a}^{\rm T}\mathbf{y}(\mathbf{x}) + b = 0. \tag{4.19}$$

Thus if we use a 1-of-K coding scheme for K classes, then the predictions made by the model will have the property that the elements of $\mathbf{y}(\mathbf{x})$ will sum to 1 for any value of $\mathbf{x}$. However, this summation constraint alone is not sufficient to allow the model outputs to be interpreted as probabilities because they are not constrained to lie within the interval $(0, 1)$.

Section 2.3.7

The least-squares approach gives an exact closed-form solution for the discriminant function parameters. However, even as a discriminant function (where we use it to make decisions directly and dispense with any probabilistic interpretation) it suffers from some severe problems. We have already seen that least-squares solutions lack robustness to outliers, and this applies equally to the classification application, as illustrated in Figure 4.4. Here we see that the additional data points in the right-hand figure produce a significant change in the location of the decision boundary, even though these point would be correctly classified by the original decision boundary in the left-hand figure. The sum-of-squares error function penalizes predictions that are 'too correct' in that they lie a long way on the correct side of the decision

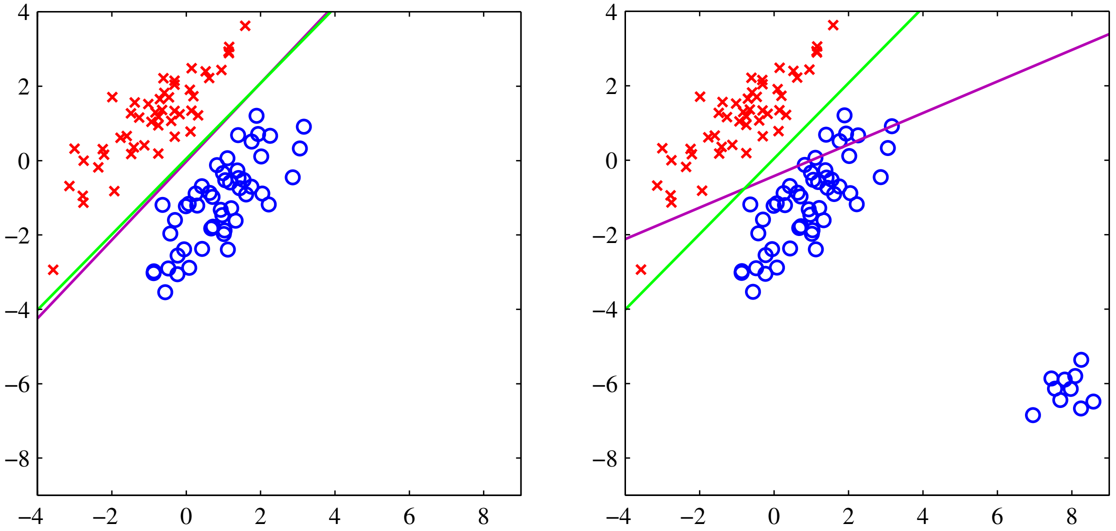

Figure 4.4 The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve), which is discussed later in Section 4.3.2. The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

boundary. In Section 7.1.2, we shall consider several alternative error functions for classification and we shall see that they do not suffer from this difficulty.

However, problems with least squares can be more severe than simply lack of robustness, as illustrated in Figure 4.5. This shows a synthetic data set drawn from three classes in a two-dimensional input space (x_1, x_2), having the property that linear decision boundaries can give excellent separation between the classes. Indeed, the technique of logistic regression, described later in this chapter, gives a satisfactory solution as seen in the right-hand plot. However, the least-squares solution gives poor results, with only a small region of the input space assigned to the green class.

The failure of least squares should not surprise us when we recall that it corresponds to maximum likelihood under the assumption of a Gaussian conditional distribution, whereas binary target vectors clearly have a distribution that is far from Gaussian. By adopting more appropriate probabilistic models, we shall obtain classification techniques with much better properties than least squares. For the moment, however, we continue to explore alternative nonprobabilistic methods for setting the parameters in the linear classification models.

4.1.4 Fisher's linear discriminant

One way to view a linear classification model is in terms of dimensionality reduction. Consider first the case of two classes, and suppose we take the D-

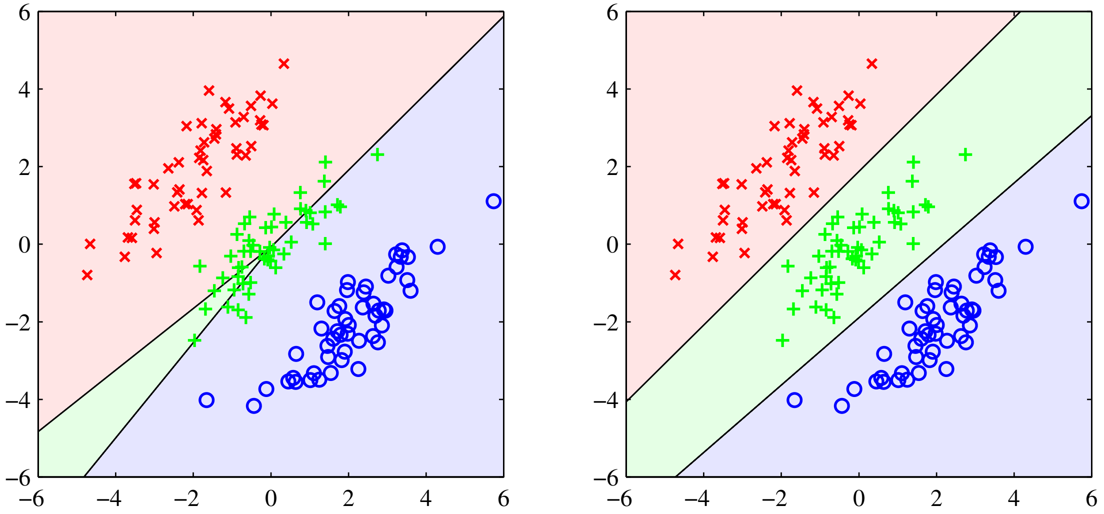

Figure 4.5 Example of a synthetic data set comprising three classes, with training data points denoted in red $(\times)$, green $(+)$, and blue $(\circ)$. Lines denote the decision boundaries, and the background colours denote the respective classes of the decision regions. On the left is the result of using a least-squares discriminant. We see that the region of input space assigned to the green class is too small and so most of the points from this class are misclassified. On the right is the result of using logistic regressions as described in Section 4.3.2 showing correct classification of the training data.

dimensional input vector $\mathbf{x}$ and project it down to one dimension using

$$y = \mathbf{w}^{\mathrm{T}}\mathbf{x}. \tag{4.20}$$

If we place a threshold on y and classify $y \geqslant -w_0$ as class $\mathcal{C}_1$, and otherwise class $\mathcal{C}_2$, then we obtain our standard linear classifier discussed in the previous section. In general, the projection onto one dimension leads to a considerable loss of information, and classes that are well separated in the original D-dimensional space may become strongly overlapping in one dimension. However, by adjusting the components of the weight vector $\mathbf{w}$, we can select a projection that maximizes the class separation. To begin with, consider a two-class problem in which there are N_1 points of class $\mathcal{C}_1$ and N_2 points of class $\mathcal{C}_2$, so that the mean vectors of the two classes are given by

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n\,\in\,\mathcal{C}_1}\mathbf{x}_n, \qquad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n\,\in\,\mathcal{C}_2}\mathbf{x}_n. \tag{4.21}$$

The simplest measure of the separation of the classes, when projected onto $\mathbf{w}$, is the separation of the projected class means. This suggests that we might choose $\mathbf{w}$ so as to maximize

$$m_2 - m_1 = \mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1) \tag{4.22}$$

where

$$m_k = \mathbf{w}^{\mathrm{T}}\mathbf{m}_k \tag{4.23}$$

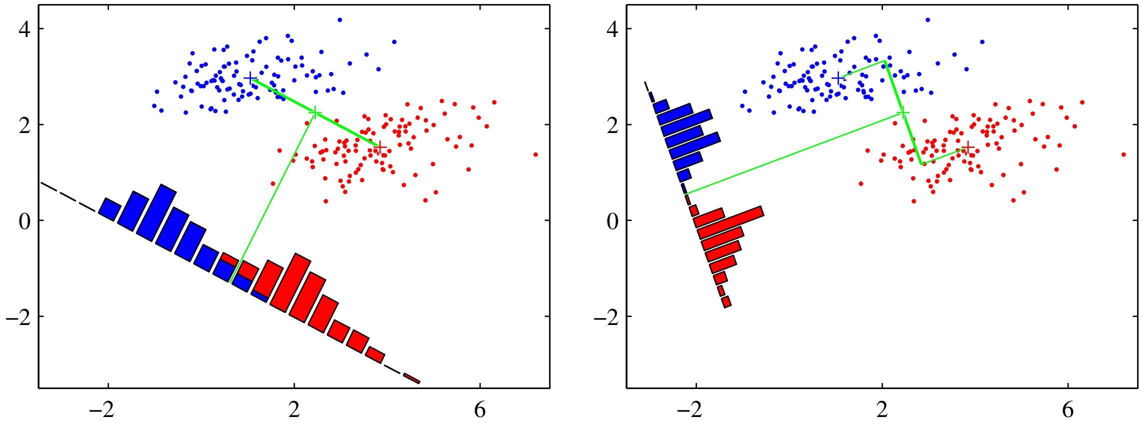

Figure 4.6 The left plot shows samples from two classes (depicted in red and blue) along with the histograms resulting from projection onto the line joining the class means. Note that there is considerable class overlap in the projected space. The right plot shows the corresponding projection based on the Fisher linear discriminant, showing the greatly improved class separation.

Appendix E

Exercise 4.4

is the mean of the projected data from class $\mathcal{C}_k$. However, this expression can be made arbitrarily large simply by increasing the magnitude of $\mathbf{w}$. To solve this problem, we could constrain $\mathbf{w}$ to have unit length, so that $\sum_i w_i^2 = 1$. Using a Lagrange multiplier to perform the constrained maximization, we then find that $\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$. There is still a problem with this approach, however, as illustrated in Figure 4.6. This shows two classes that are well separated in the original two-dimensional space (x_1, x_2) but that have considerable overlap when projected onto the line joining their means. This difficulty arises from the strongly nondiagonal covariances of the class distributions. The idea proposed by Fisher is to maximize a function that will give a large separation between the projected class means while also giving a small variance within each class, thereby minimizing the class overlap.

The projection formula (4.20) transforms the set of labelled data points in $\mathbf{x}$ into a labelled set in the one-dimensional space y. The within-class variance of the transformed data from class $\mathcal{C}_k$ is therefore given by

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 \tag{4.24}$$

where $y_n = \mathbf{w}^{\mathrm{T}}\mathbf{x}_n$. We can define the total within-class variance for the whole data set to be simply $s_1^2 + s_2^2$. The Fisher criterion is defined to be the ratio of the between-class variance to the within-class variance and is given by

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}. \tag{4.25}$$

Exercise 4.5

We can make the dependence on $\mathbf{w}$ explicit by using (4.20), (4.23), and (4.24) to rewrite the Fisher criterion in the form

$$J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}} \tag{4.26}$$

where $\mathbf{S}_{\mathrm{B}}$ is the *between-class* covariance matrix and is given by

$$\mathbf{S}_{\mathrm{B}} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}} \tag{4.27}$$

and $\mathbf{S}_{\mathrm{W}}$ is the total *within-class* covariance matrix, given by

$$\mathbf{S}_{\mathrm{W}} = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^{\mathrm{T}} + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^{\mathrm{T}}. \tag{4.28}$$

Differentiating (4.26) with respect to $\mathbf{w}$, we find that $J(\mathbf{w})$ is maximized when

$$(\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w})\mathbf{S}_{\mathrm{W}}\mathbf{w} = (\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w})\mathbf{S}_{\mathrm{B}}\mathbf{w}. \tag{4.29}$$

From (4.27), we see that $\mathbf{S}_{\mathrm{B}}\mathbf{w}$ is always in the direction of $(\mathbf{m}_2 - \mathbf{m}_1)$. Furthermore, we do not care about the magnitude of $\mathbf{w}$, only its direction, and so we can drop the scalar factors $(\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w})$ and $(\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w})$. Multiplying both sides of (4.29) by $\mathbf{S}_{\mathrm{W}}^{-1}$ we then obtain

$$\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1). \tag{4.30}$$

Note that if the within-class covariance is isotropic, so that $\mathbf{S}_{\mathrm{W}}$ is proportional to the unit matrix, we find that $\mathbf{w}$ is proportional to the difference of the class means, as discussed above.

The result (4.30) is known as *Fisher's linear discriminant*, although strictly it is not a discriminant but rather a specific choice of direction for projection of the data down to one dimension. However, the projected data can subsequently be used to construct a discriminant, by choosing a threshold y_0 so that we classify a new point as belonging to $\mathcal{C}_1$ if $y(\mathbf{x}) \geqslant y_0$ and classify it as belonging to $\mathcal{C}_2$ otherwise. For example, we can model the class-conditional densities $p(y|\mathcal{C}_k)$ using Gaussian distributions and then use the techniques of Section 1.2.4 to find the parameters of the Gaussian distributions by maximum likelihood. Having found Gaussian approximations to the projected classes, the formalism of Section 1.5.1 then gives an expression for the optimal threshold. Some justification for the Gaussian assumption comes from the central limit theorem by noting that $y = \mathbf{w}^{\mathrm{T}}\mathbf{x}$ is the sum of a set of random variables.

4.1.5 Relation to least squares

The least-squares approach to the determination of a linear discriminant was based on the goal of making the model predictions as close as possible to a set of target values. By contrast, the Fisher criterion was derived by requiring maximum class separation in the output space. It is interesting to see the relationship between these two approaches. In particular, we shall show that, for the two-class problem, the Fisher criterion can be obtained as a special case of least squares.

So far we have considered 1-of-K coding for the target values. If, however, we adopt a slightly different target coding scheme, then the least-squares solution for the weights becomes equivalent to the Fisher solution (Duda and Hart, 1973). In particular, we shall take the targets for class $\mathcal{C}_1$ to be N/N_1, where N_1 is the number of patterns in class $\mathcal{C}_1$, and N is the total number of patterns. This target value approximates the reciprocal of the prior probability for class $\mathcal{C}_1$. For class $\mathcal{C}_2$, we shall take the targets to be $-N/N_2$, where N_2 is the number of patterns in class $\mathcal{C}_2$.

The sum-of-squares error function can be written

$$E = \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + w_0 - t_n\right)^2. \tag{4.31}$$

Setting the derivatives of E with respect to w_0 and $\mathbf{w}$ to zero, we obtain respectively

$$\sum_{n=1}^{N}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + w_0 - t_n\right) = 0 \tag{4.32}$$

$$\sum_{n=1}^{N}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + w_0 - t_n\right)\mathbf{x}_n = 0. \tag{4.33}$$

From (4.32), and making use of our choice of target coding scheme for the t_n, we obtain an expression for the bias in the form

$$w_0 = -\mathbf{w}^{\mathrm{T}}\mathbf{m} \tag{4.34}$$

where we have used

$$\sum_{n=1}^{N} t_n = N_1\frac{N}{N_1} - N_2\frac{N}{N_2} = 0 \tag{4.35}$$

and where $\mathbf{m}$ is the mean of the total data set and is given by

$$\mathbf{m} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n = \frac{1}{N}(N_1\mathbf{m}_1 + N_2\mathbf{m}_2). \tag{4.36}$$

Exercise 4.6

After some straightforward algebra, and again making use of the choice of t_n, the second equation (4.33) becomes

$$\left(\mathbf{S}_{\mathrm{W}} + \frac{N_1N_2}{N}\mathbf{S}_{\mathrm{B}}\right)\mathbf{w} = N(\mathbf{m}_1 - \mathbf{m}_2) \tag{4.37}$$

where $\mathbf{S}_{\mathrm{W}}$ is defined by (4.28), $\mathbf{S}_{\mathrm{B}}$ is defined by (4.27), and we have substituted for the bias using (4.34). Using (4.27), we note that $\mathbf{S}_{\mathrm{B}}\mathbf{w}$ is always in the direction of $(\mathbf{m}_2 - \mathbf{m}_1)$. Thus we can write

$$\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1) \tag{4.38}$$

where we have ignored irrelevant scale factors. Thus the weight vector coincides with that found from the Fisher criterion. In addition, we have also found an expression for the bias value w_0 given by (4.34). This tells us that a new vector $\mathbf{x}$ should be classified as belonging to class $\mathcal{C}_1$ if $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}(\mathbf{x} - \mathbf{m}) > 0$ and class $\mathcal{C}_2$ otherwise.

4.1.6 Fisher's discriminant for multiple classes

We now consider the generalization of the Fisher discriminant to $K > 2$ classes, and we shall assume that the dimensionality D of the input space is greater than the number K of classes. Next, we introduce $D' > 1$ linear 'features' $y_k = \mathbf{w}_k^{\mathrm{T}}\mathbf{x}$, where $k = 1, \ldots, D'$. These feature values can conveniently be grouped together to form a vector $\mathbf{y}$. Similarly, the weight vectors $\{\mathbf{w}_k\}$ can be considered to be the columns of a matrix $\mathbf{W}$, so that

$$\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x}. \tag{4.39}$$

Note that again we are not including any bias parameters in the definition of $\mathbf{y}$. The generalization of the within-class covariance matrix to the case of K classes follows from (4.28) to give

$$\mathbf{S}_{\mathrm{W}} = \sum_{k=1}^{K} \mathbf{S}_k \tag{4.40}$$

where

$$\mathbf{S}_k = \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^{\mathrm{T}} \tag{4.41}$$

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \tag{4.42}$$

and N_k is the number of patterns in class $\mathcal{C}_k$. In order to find a generalization of the between-class covariance matrix, we follow Duda and Hart (1973) and consider first the total covariance matrix

$$\mathbf{S}_{\mathrm{T}} = \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{m})(\mathbf{x}_n - \mathbf{m})^{\mathrm{T}} \tag{4.43}$$

where $\mathbf{m}$ is the mean of the total data set

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \mathbf{m}_k \tag{4.44}$$

and $N = \sum_k N_k$ is the total number of data points. The total covariance matrix can be decomposed into the sum of the within-class covariance matrix, given by (4.40) and (4.41), plus an additional matrix $\mathbf{S}_{\mathrm{B}}$, which we identify as a measure of the between-class covariance

$$\mathbf{S}_{\mathrm{T}} = \mathbf{S}_{\mathrm{W}} + \mathbf{S}_{\mathrm{B}} \tag{4.45}$$

where

$$\mathbf{S}_{\mathrm{B}} = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^{\mathrm{T}}. \tag{4.46}$$

These covariance matrices have been defined in the original $\mathbf{x}$-space. We can now define similar matrices in the projected D'-dimensional $\mathbf{y}$-space

$$\mathbf{s}_{\mathrm{W}} = \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (\mathbf{y}_n - \boldsymbol{\mu}_k)(\mathbf{y}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \tag{4.47}$$

and

$$\mathbf{s}_{\mathrm{B}} = \sum_{k=1}^{K} N_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^{\mathrm{T}} \tag{4.48}$$

where

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{y}_n, \qquad \boldsymbol{\mu} = \frac{1}{N} \sum_{k=1}^{K} N_k \boldsymbol{\mu}_k. \tag{4.49}$$

Again we wish to construct a scalar that is large when the between-class covariance is large and when the within-class covariance is small. There are now many possible choices of criterion (Fukunaga, 1990). One example is given by

$$J(\mathbf{W}) = \mathrm{Tr}\left\{\mathbf{s}_{\mathrm{W}}^{-1}\mathbf{s}_{\mathrm{B}}\right\}. \tag{4.50}$$

This criterion can then be rewritten as an explicit function of the projection matrix $\mathbf{W}$ in the form

$$J(\mathbf{w}) = \mathrm{Tr}\left\{(\mathbf{W}\mathbf{S}_{\mathrm{W}}\mathbf{W}^{\mathrm{T}})^{-1}(\mathbf{W}\mathbf{S}_{\mathrm{B}}\mathbf{W}^{\mathrm{T}})\right\}. \tag{4.51}$$

Maximization of such criteria is straightforward, though somewhat involved, and is discussed at length in Fukunaga (1990). The weight values are determined by those eigenvectors of $\mathbf{S}_{\mathrm{W}}^{-1}\mathbf{S}_{\mathrm{B}}$ that correspond to the D' largest eigenvalues.

There is one important result that is common to all such criteria, which is worth emphasizing. We first note from (4.46) that $\mathbf{S}_{\mathrm{B}}$ is composed of the sum of K matrices, each of which is an outer product of two vectors and therefore of rank 1. In addition, only $(K-1)$ of these matrices are independent as a result of the constraint (4.44). Thus, $\mathbf{S}_{\mathrm{B}}$ has rank at most equal to $(K-1)$ and so there are at most $(K-1)$ nonzero eigenvalues. This shows that the projection onto the $(K-1)$-dimensional subspace spanned by the eigenvectors of $\mathbf{S}_{\mathrm{B}}$ does not alter the value of $J(\mathbf{w})$, and so we are therefore unable to find more than $(K-1)$ linear 'features' by this means (Fukunaga, 1990).

4.1.7 The perceptron algorithm

Another example of a linear discriminant model is the perceptron of Rosenblatt (1962), which occupies an important place in the history of pattern recognition algorithms. It corresponds to a two-class model in which the input vector $\mathbf{x}$ is first transformed using a fixed nonlinear transformation to give a feature vector $\boldsymbol{\phi}(\mathbf{x})$, and this is then used to construct a generalized linear model of the form

$$y(\mathbf{x}) = f\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})\right) \tag{4.52}$$

where the nonlinear activation function $f(\cdot)$ is given by a step function of the form

$$f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0. \end{cases} \tag{4.53}$$

The vector $\boldsymbol{\phi}(\mathbf{x})$ will typically include a bias component $\phi_0(\mathbf{x}) = 1$. In earlier discussions of two-class classification problems, we have focussed on a target coding scheme in which $t \in \{0, 1\}$, which is appropriate in the context of probabilistic models. For the perceptron, however, it is more convenient to use target values $t = +1$ for class $\mathcal{C}_1$ and $t = -1$ for class $\mathcal{C}_2$, which matches the choice of activation function.

The algorithm used to determine the parameters $\mathbf{w}$ of the perceptron can most easily be motivated by error function minimization. A natural choice of error function would be the total number of misclassified patterns. However, this does not lead to a simple learning algorithm because the error is a piecewise constant function of $\mathbf{w}$, with discontinuities wherever a change in $\mathbf{w}$ causes the decision boundary to move across one of the data points. Methods based on changing $\mathbf{w}$ using the gradient of the error function cannot then be applied, because the gradient is zero almost everywhere.

We therefore consider an alternative error function known as the *perceptron criterion*. To derive this, we note that we are seeking a weight vector $\mathbf{w}$ such that patterns $\mathbf{x}_n$ in class $\mathcal{C}_1$ will have $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) > 0$, whereas patterns $\mathbf{x}_n$ in class $\mathcal{C}_2$ have $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) < 0$. Using the $t \in \{-1, +1\}$ target coding scheme it follows that we would like all patterns to satisfy $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)t_n > 0$. The perceptron criterion associates zero error with any pattern that is correctly classified, whereas for a misclassified pattern $\mathbf{x}_n$ it tries to minimize the quantity $-\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)t_n$. The perceptron criterion is therefore given by

$$E_{\mathrm{P}}(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n t_n \tag{4.54}$$

Frank Rosenblatt
1928–1969

Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms" published in 1962. Rosenblatt's work was criticized by Marvin Minksy, whose objections were published in the book "Perceptrons", co-authored with Seymour Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

where $\mathcal{M}$ denotes the set of all misclassified patterns. The contribution to the error associated with a particular misclassified pattern is a linear function of $\mathbf{w}$ in regions of $\mathbf{w}$ space where the pattern is misclassified and zero in regions where it is correctly classified. The total error function is therefore piecewise linear.

Section 3.1.3 We now apply the stochastic gradient descent algorithm to this error function. The change in the weight vector $\mathbf{w}$ is then given by

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \boldsymbol{\phi}_n t_n \tag{4.55}$$

where η is the learning rate parameter and τ is an integer that indexes the steps of the algorithm. Because the perceptron function $y(\mathbf{x}, \mathbf{w})$ is unchanged if we multiply $\mathbf{w}$ by a constant, we can set the learning rate parameter η equal to 1 without of generality. Note that, as the weight vector evolves during training, the set of patterns that are misclassified will change.

The perceptron learning algorithm has a simple interpretation, as follows. We cycle through the training patterns in turn, and for each pattern $\mathbf{x}_n$ we evaluate the perceptron function (4.52). If the pattern is correctly classified, then the weight vector remains unchanged, whereas if it is incorrectly classified, then for class $\mathcal{C}_1$ we add the vector $\boldsymbol{\phi}(\mathbf{x}_n)$ onto the current estimate of weight vector $\mathbf{w}$ while for class $\mathcal{C}_2$ we subtract the vector $\boldsymbol{\phi}(\mathbf{x}_n)$ from $\mathbf{w}$. The perceptron learning algorithm is illustrated in Figure 4.7.

If we consider the effect of a single update in the perceptron learning algorithm, we see that the contribution to the error from a misclassified pattern will be reduced because from (4.55) we have

$$-\mathbf{w}^{(\tau+1)\mathrm{T}} \boldsymbol{\phi}_n t_n = -\mathbf{w}^{(\tau)\mathrm{T}} \boldsymbol{\phi}_n t_n - (\boldsymbol{\phi}_n t_n)^{\mathrm{T}} \boldsymbol{\phi}_n t_n < -\mathbf{w}^{(\tau)\mathrm{T}} \boldsymbol{\phi}_n t_n \tag{4.56}$$

where we have set $\eta = 1$, and made use of $\|\boldsymbol{\phi}_n t_n\|^2 > 0$. Of course, this does not imply that the contribution to the error function from the other misclassified patterns will have been reduced. Furthermore, the change in weight vector may have caused some previously correctly classified patterns to become misclassified. Thus the perceptron learning rule is not guaranteed to reduce the total error function at each stage.

However, the *perceptron convergence theorem* states that if there exists an exact solution (in other words, if the training data set is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps. Proofs of this theorem can be found for example in Rosenblatt (1962), Block (1962), Nilsson (1965), Minsky and Papert (1969), Hertz *et al.* (1991), and Bishop (1995a). Note, however, that the number of steps required to achieve convergence could still be substantial, and in practice, until convergence is achieved, we will not be able to distinguish between a nonseparable problem and one that is simply slow to converge.

Even when the data set is linearly separable, there may be many solutions, and which one is found will depend on the initialization of the parameters and on the order of presentation of the data points. Furthermore, for data sets that are not linearly separable, the perceptron learning algorithm will never converge.

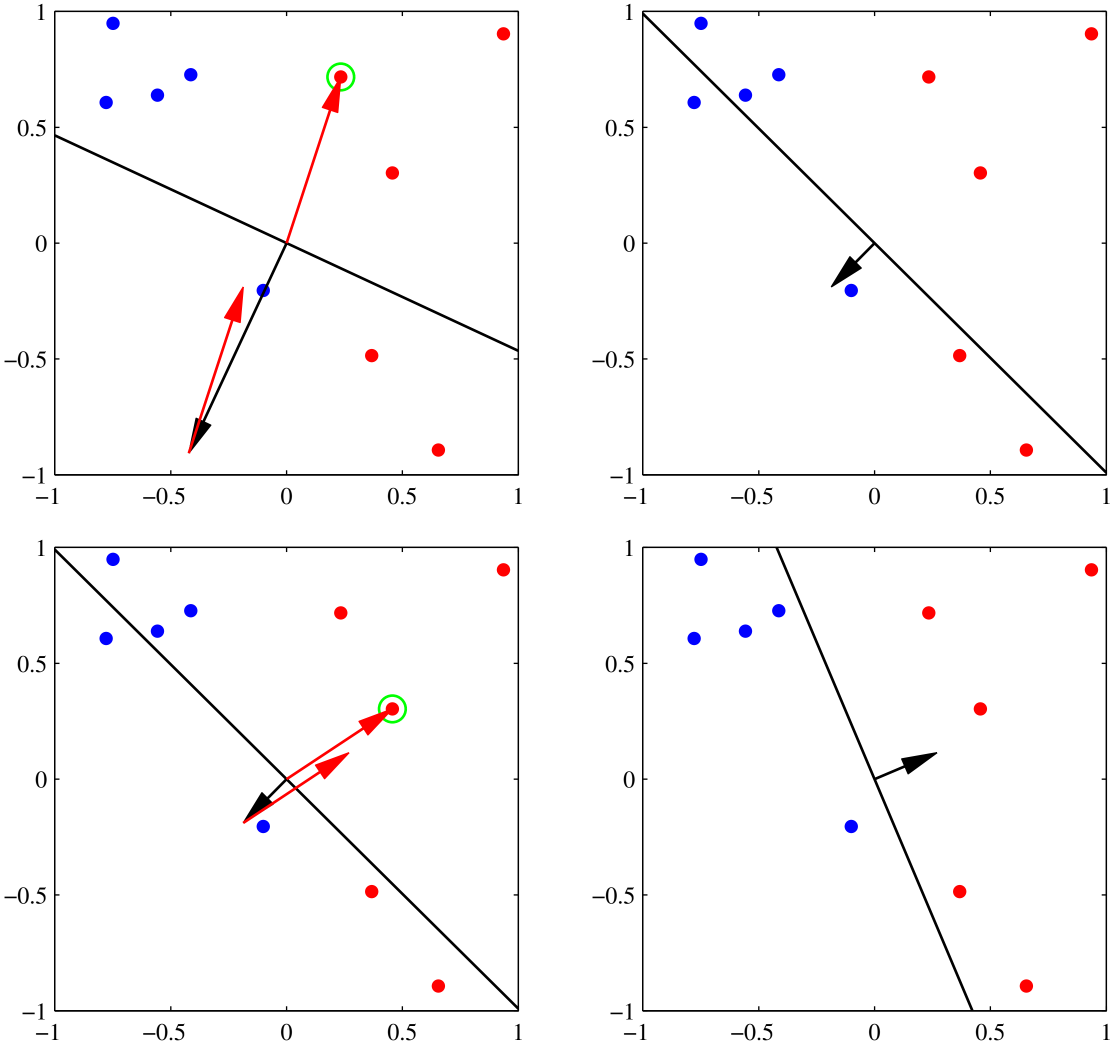

Figure 4.7 Illustration of the convergence of the perceptron learning algorithm, showing data points from two classes (red and blue) in a two-dimensional feature space (ϕ_1, ϕ_2). The top left plot shows the initial parameter vector $\mathbf{w}$ shown as a black arrow together with the corresponding decision boundary (black line), in which the arrow points towards the decision region which classified as belonging to the red class. The data point circled in green is misclassified and so its feature vector is added to the current weight vector, giving the new decision boundary shown in the top right plot. The bottom left plot shows the next misclassified point to be considered, indicated by the green circle, and its feature vector is again added to the weight vector giving the decision boundary shown in the bottom right plot for which all data points are correctly classified.

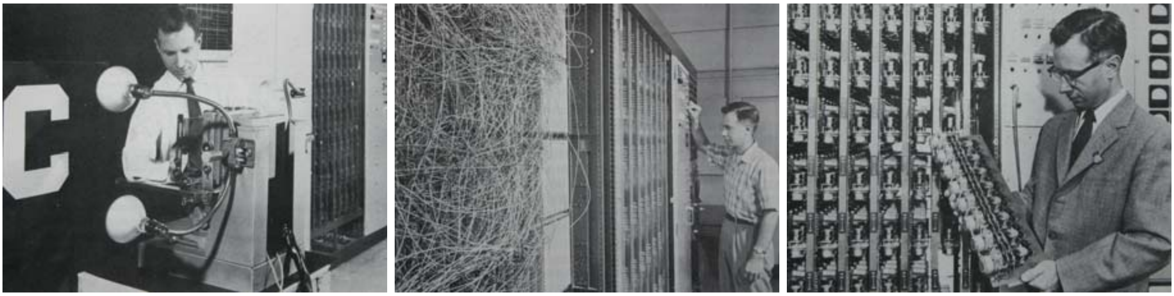

Figure 4.8 Illustration of the Mark 1 perceptron hardware. The photograph on the left shows how the inputs were obtained using a simple camera system in which an input scene, in this case a printed character, was illuminated by powerful lights, and an image focussed onto a 20×20 array of cadmium sulphide photocells, giving a primitive 400 pixel image. The perceptron also had a patch board, shown in the middle photograph, which allowed different configurations of input features to be tried. Often these were wired up at random to demonstrate the ability of the perceptron to learn without the need for precise wiring, in contrast to a modern digital computer. The photograph on the right shows one of the racks of adaptive weights. Each weight was implemented using a rotary variable resistor, also called a potentiometer, driven by an electric motor thereby allowing the value of the weight to be adjusted automatically by the learning algorithm.

Aside from difficulties with the learning algorithm, the perceptron does not provide probabilistic outputs, nor does it generalize readily to $K > 2$ classes. The most important limitation, however, arises from the fact that (in common with all of the models discussed in this chapter and the previous one) it is based on linear combinations of fixed basis functions. More detailed discussions of the limitations of perceptrons can be found in Minsky and Papert (1969) and Bishop (1995a).

Analogue hardware implementations of the perceptron were built by Rosenblatt, based on motor-driven variable resistors to implement the adaptive parameters w_j. These are illustrated in Figure 4.8. The inputs were obtained from a simple camera system based on an array of photo-sensors, while the basis functions ϕ could be chosen in a variety of ways, for example based on simple fixed functions of randomly chosen subsets of pixels from the input image. Typical applications involved learning to discriminate simple shapes or characters.

At the same time that the perceptron was being developed, a closely related system called the *adaline*, which is short for ‘adaptive linear element’, was being explored by Widrow and co-workers. The functional form of the model was the same as for the perceptron, but a different approach to training was adopted (Widrow and Hoff, 1960; Widrow and Lehr, 1990).

4.2. Probabilistic Generative Models

We turn next to a probabilistic view of classification and show how models with linear decision boundaries arise from simple assumptions about the distribution of the data. In Section 1.5.4, we discussed the distinction between the discriminative and the generative approaches to classification. Here we shall adopt a generative