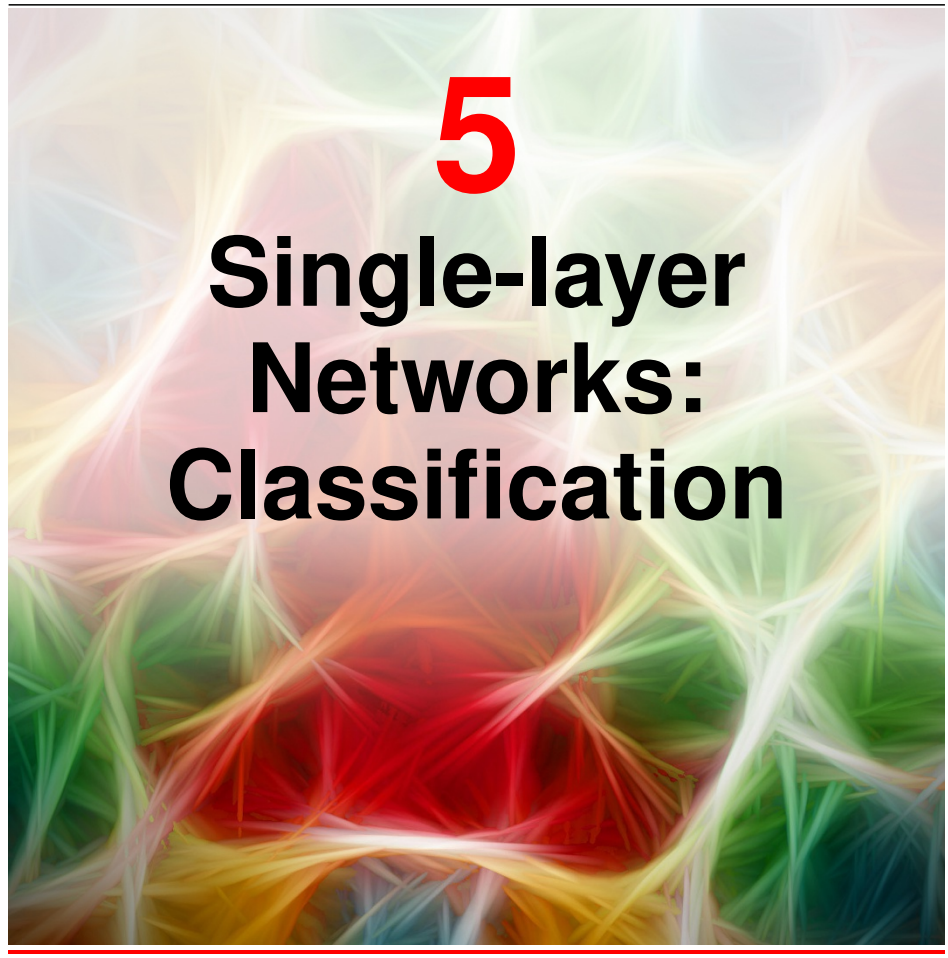

5 Single-layer Networks: Classification

In the previous chapter, we explored a class of regression models in which the output variables were linear functions of the model parameters and which can therefore be expressed as simple neural networks having a single layer of weight and bias parameters. We turn now to a discussion of classification problems, and in this chapter, we will focus on an analogous class of models that again can be expressed as single-layer neural networks. These will allow us to introduce many of the key concepts of classification before dealing with more general deep neural networks in later chapters.

The goal in classification is to take an input vector $\mathbf{x} \in \mathbb{R}^D$ and 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

C. M. Bishop, H. Bishop, *Deep Learning*, https://doi.org/10.1007/978-3-031-45468-4_5

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*. Linear classification models can be applied to data sets that are not linearly separable, although not all inputs will be correctly classified.

We can broadly identify three distinct approaches to solving classification problems. 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 distributions $p(\mathcal{C}_k|\mathbf{x})$ in an *inference* stage and subsequently uses these distributions to make optimal *decisions*. Separating inference and decision brings numerous benefits. 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. This will be called a *discriminative probabilistic model*. Alternatively, we can model the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$, together with the prior probabilities $p(\mathcal{C}_k)$ for the classes, and then compute the required posterior probabilities using Bayes' theorem:

Section 5.2.4

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}. \tag{5.1}$$

This will be called a *generative probabilistic model* because it offers the opportunity to generate samples from each of the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$. In this chapter, we will discuss examples of all three approaches: discriminant functions, generative probabilistic models, and discriminative probabilistic models.

5.1. 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 will restrict attention to *linear discriminants*, namely those for which the decision surfaces are hyperplanes. To simplify the discussion, we consider first two classes and then investigate the extension to $K > 2$ classes.

5.1.1 Two classes

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

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 \tag{5.2}$$

where $\mathbf{w}$ is called a *weight vector*, and w_0 is a *bias* (not to be confused with bias in the statistical sense). 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

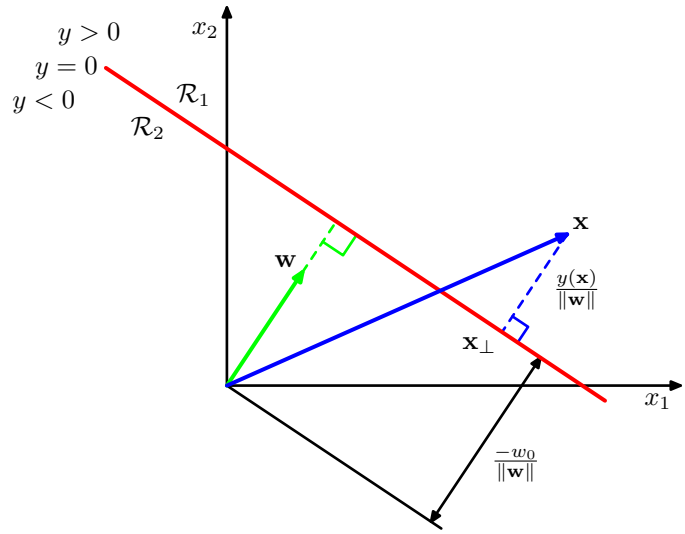

Figure 5.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter w_0. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $y(\mathbf{x})/\|\mathbf{w}\|$.

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

$$\frac{\mathbf{w}^\mathrm{T}\mathbf{x}}{\|\mathbf{w}\|} = -\frac{w_0}{\|\mathbf{w}\|}. \tag{5.3}$$

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 5.1.

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

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}. \tag{5.4}$$

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

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}. \tag{5.5}$$

This result is illustrated in Figure 5.1.

Section 4.1.1

As with linear regression models, 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

$$y(\mathbf{x}) = \widetilde{\mathbf{w}}^\mathrm{T}\widetilde{\mathbf{x}}. \tag{5.6}$$

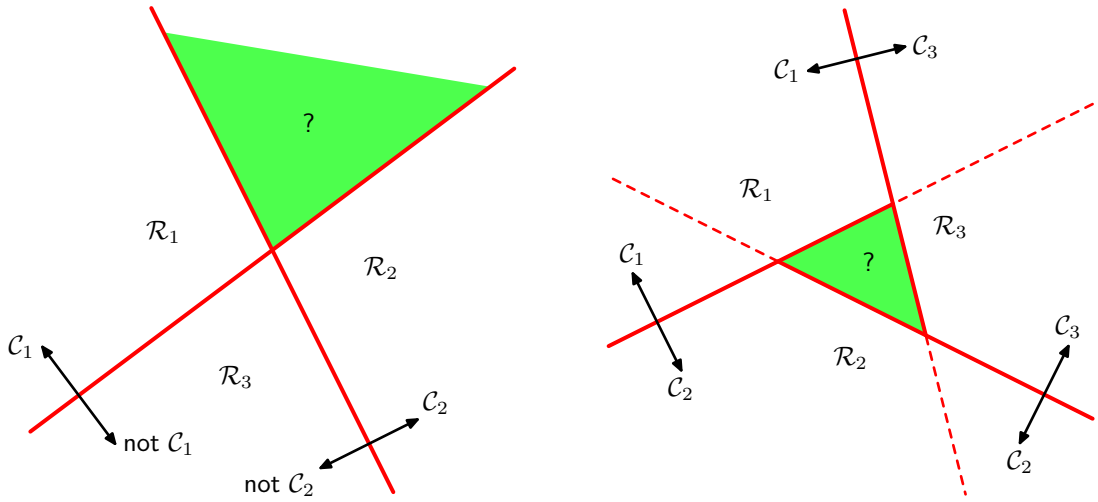

Figure 5.2 Attempting to construct a K-class discriminant from a set of two-class discriminant functions leads to ambiguous regions, as shown in green. On the left is an example with two discriminant functions designed to distinguish points in class $\mathcal{C}_k$ from points not in class $\mathcal{C}_k$. 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 $\mathcal{C}_j$.

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

5.1.2 Multiple classes

Now consider the extension of linear discriminant functions to $K > 2$ classes. We might be tempted 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 a model with $K-1$ 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 5.2 shows an 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 5.2.

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

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0} \tag{5.7}$$

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

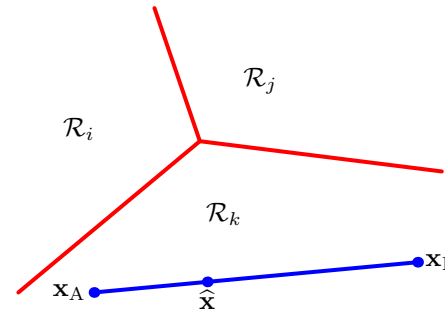

Figure 5.3 Illustration of the decision regions for a multi-class linear discriminant, with the decision boundaries shown in red. If two points $\mathbf{x}_\mathrm{A}$ and $\mathbf{x}_\mathrm{B}$ both lie inside the same decision region $\mathcal{R}_k$, then any point $\widehat{\mathbf{x}}$ that lies 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.

hence corresponds to a $(D-1)$-dimensional hyperplane defined by

$$(\mathbf{w}_k - \mathbf{w}_j)^\mathrm{T}\mathbf{x} + (w_{k0} - w_{j0}) = 0. \tag{5.8}$$

This has the same form as the decision boundary for the two-class case discussed in Section 5.1.1, 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 5.3. 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

$$\widehat{\mathbf{x}} = \lambda\mathbf{x}_\mathrm{A} + (1-\lambda)\mathbf{x}_\mathrm{B} \tag{5.9}$$

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

$$y_k(\widehat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_\mathrm{A}) + (1-\lambda)y_k(\mathbf{x}_\mathrm{B}). \tag{5.10}$$

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 that $y_k(\mathbf{x}_\mathrm{B}) > y_j(\mathbf{x}_\mathrm{B})$, 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 essentially equivalent formulation based on a single discriminant function $y(\mathbf{x})$.

Section 5.1.1

5.1.3 1-of-K coding

For regression problems, the target variable $\mathbf{t}$ was simply the vector of real numbers whose values we wish to predict. In classification, there are various ways of using target values to represent class labels. For two-class problems, the most convenient 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 probability values taking only the extreme values of 0 and 1. For $K > 2$ classes, it is convenient to use a 1-of-K coding scheme, also known as the one-hot encoding 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 data point from class 2 would be given the target vector

$$\mathbf{t} = (0, 1, 0, 0, 0)^{\mathrm{T}}. \tag{5.11}$$

Again, we can interpret the value of t_k as the probability that the class is $\mathcal{C}_k$ in which the probabilities take only the values 0 and 1.

5.1.4 Least squares for classification

Section 4.1.3

With linear regression models, the minimization of a sum-of-squares error function leads to a simple closed-form solution for the parameter values. It is therefore tempting to see if we can apply the same least-squares formalism to classification problems. Consider a general classification problem with K classes and 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 a binary coding scheme, this conditional expectation is given by the vector of posterior class probabilities. Unfortunately, these probabilities are typically approximated rather poorly, and indeed the approximations can have values outside the range $(0, 1)$. However, it is instructional to explore these simple models and to understand how these limitations arise.

Exercise 5.1

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

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0} \tag{5.12}$$

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

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}} \tag{5.13}$$

where $\widetilde{\mathbf{W}}$ is a matrix whose kth column comprises the $(D + 1)$-dimensional vector $\widetilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^{\mathrm{T}})^{\mathrm{T}}$ and $\widetilde{\mathbf{x}}$ is the corresponding augmented input vector $(1, \mathbf{x}^{\mathrm{T}})^{\mathrm{T}}$ with a dummy input $x_0 = 1$. A new input $\mathbf{x}$ is then assigned to the class for which the output $y_k = \widetilde{\mathbf{w}}_k^{\mathrm{T}}\widetilde{\mathbf{x}}$ is largest.

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

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}. \tag{5.14}$$

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

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T} \tag{5.15}$$

Section 4.1.3

where $\widetilde{\mathbf{X}}^{\dagger}$ is the pseudo-inverse of the matrix $\widetilde{\mathbf{X}}$. We then obtain the discriminant

function in the form

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^{\mathrm{T}}\widetilde{\mathbf{x}} = \mathbf{T}^{\mathrm{T}}\left(\widetilde{\mathbf{X}}^{\dagger}\right)^{\mathrm{T}}\widetilde{\mathbf{x}}. \tag{5.16}$$

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

$$\mathbf{a}^{\mathrm{T}}\mathbf{t}_n + b = 0 \tag{5.17}$$

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

Exercise 5.3

$$\mathbf{a}^{\mathrm{T}}\mathbf{y}(\mathbf{x}) + b = 0. \tag{5.18}$$

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 $(0, 1)$.

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 seen that the sum-of-squares error function can be viewed as the negative log likelihood under the assumption of a Gaussian noise distribution. If the true distribution of the data is markedly different from being Gaussian, then least squares can give poor results. In particular, least squares is very sensitive to the presence of *outliers*, which are data points located a long way from the bulk of the data. This is illustrated in Figure 5.4. 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 points would be correctly classified by the original decision boundary in the left-hand figure. The sum-of-squares error function gives too much weight to data points that are a long way from the decision boundary, even though they are correctly classified. Outliers can arise due to rare events or may simply be due to mistakes in the data set. Techniques that are sensitive to a very few data points are said to lack *robustness*. For comparison, Figure 5.4 also shows results from a technique called *logistic regression*, which is more robust to outliers.

Section 2.3.4

Section 5.4.3

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 can obtain classification techniques with much better properties than least squares, and which can also be generalized to give flexible nonlinear neural network models, as we will see in later chapters.

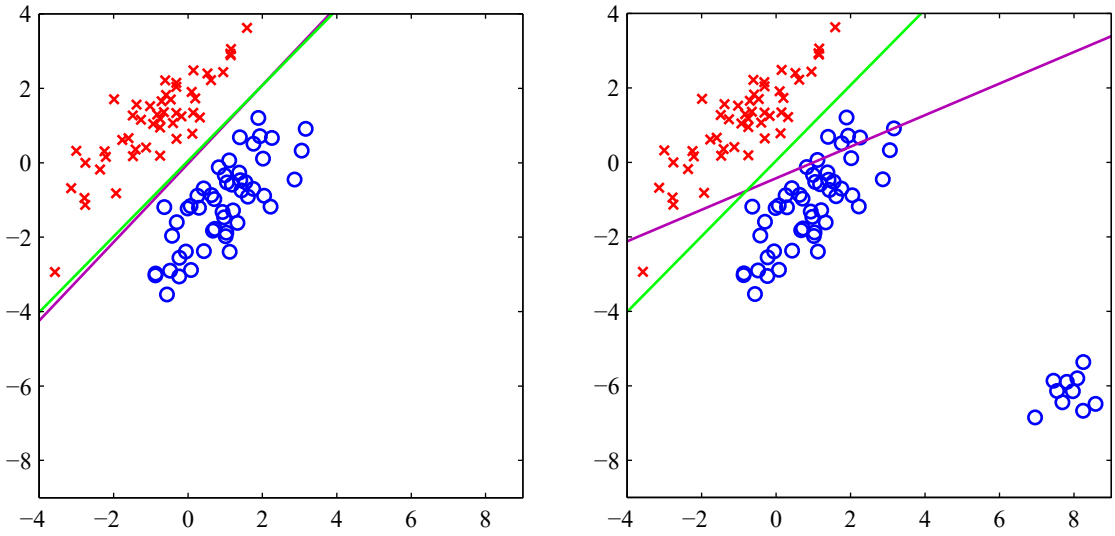

Figure 5.4 The left-hand plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundaries found by least squares (magenta curve) and by a logistic regression model (green curve). The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom right of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

5.2. Decision Theory

Section 4.2

When we discussed linear regression we saw how the process of making predictions in machine learning can be broken down into the two stages of inference and decision. We now explore this perspective in much greater depth specifically in the context of classifiers.

Suppose we have an input vector $\mathbf{x}$ together with a corresponding vector $\mathbf{t}$ of target variables, and our goal is to predict $\mathbf{t}$ given a new value for $\mathbf{x}$. For regression problems, $\mathbf{t}$ will comprise continuous variables and in general will be a vector as we may wish to predict several related quantities. For classification problems, $\mathbf{t}$ will represent class labels. Again, $\mathbf{t}$ will in general be a vector if we have more than two classes. The joint probability distribution $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of the uncertainty associated with these variables. Determining $p(\mathbf{x}, \mathbf{t})$ from a set of training data is an example of *inference* and is typically a very difficult problem whose solution forms the subject of much of this book. In a practical application, however, we must often make a specific prediction for the value of $\mathbf{t}$ or more generally take a specific action based on our understanding of the values $\mathbf{t}$ is likely to take, and this aspect is the subject of decision theory.

Consider, for example, our earlier medical diagnosis problem in which we have taken an image of a skin lesion on a patient, and we wish to determine whether the patient has cancer. In this case, the input vector $\mathbf{x}$ is the set of pixel intensities in

the image, and the output variable t will represent the absence of cancer, which we denote by the class $\mathcal{C}_1$, or the presence of cancer, which we denote by the class $\mathcal{C}_2$. We might, for instance, choose t to be a binary variable such that $t = 0$ corresponds to class $\mathcal{C}_1$ and $t = 1$ corresponds to class $\mathcal{C}_2$. We will see later that this choice of label values is particularly convenient when working with probabilities. The general inference problem then involves determining the joint distribution $p(\mathbf{x}, \mathcal{C}_k)$, or equivalently $p(\mathbf{x}, t)$, which gives us the most complete probabilistic description of the variables. Although this can be a very useful and informative quantity, ultimately, we must decide either to give treatment to the patient or not, and we would like this choice to be optimal according to some appropriate criterion (Duda and Hart, 1973). This is the *decision* step, and the aim of decision theory is that it should tell us how to make optimal decisions given the appropriate probabilities. We will see that the decision stage is generally very simple, even trivial, once we have solved the inference problem. Here we give an introduction to the key ideas of decision theory as required for the rest of the book. Further background, as well as more detailed accounts, can be found in Berger (1985) and Bather (2000).

Before giving a more detailed analysis, let us first consider informally how we might expect probabilities to play a role in making decisions. When we obtain the skin image $\mathbf{x}$ for a new patient, our goal is to decide which of the two classes to assign the image to. We are therefore interested in the probabilities of the two classes, given the image, which are given by $p(\mathcal{C}_k|\mathbf{x})$. Using Bayes' theorem, these probabilities can be expressed in the form

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}. \tag{5.19}$$

Note that any of the quantities appearing in Bayes' theorem can be obtained from the joint distribution $p(\mathbf{x}, \mathcal{C}_k)$ by either marginalizing or conditioning with respect to the appropriate variables. We can now interpret $p(\mathcal{C}_k)$ as the prior probability for the class $\mathcal{C}_k$ and $p(\mathcal{C}_k|\mathbf{x})$ as the corresponding posterior probability. Thus, $p(\mathcal{C}_1)$ represents the probability that a person has cancer, before the image is taken. Similarly, $p(\mathcal{C}_1|\mathbf{x})$ is the posterior probability, revised using Bayes' theorem in light of the information contained in the image. If our aim is to minimize the chance of assigning $\mathbf{x}$ to the wrong class, then intuitively we would choose the class having the higher posterior probability. We now show that this intuition is correct, and we also discuss more general criteria for making decisions.

5.2.1 Misclassification rate

Suppose that our goal is simply to make as few misclassifications as possible. We need a rule that assigns each value of $\mathbf{x}$ to one of the available classes. Such a rule will divide the input space into regions $\mathcal{R}_k$ called *decision regions*, one for each class, such that all points in $\mathcal{R}_k$ are assigned to class $\mathcal{C}_k$. The boundaries between decision regions are called *decision boundaries* or *decision surfaces*. Note that each decision region need not be contiguous but could comprise some number of disjoint regions. To find the optimal decision rule, consider first the case of two classes, as in the cancer problem, for instance. A mistake occurs when an input vector belonging

to class $\mathcal{C}_1$ is assigned to class $\mathcal{C}_2$ or vice versa. The probability of this occurring is given by

$$\begin{aligned} p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2)\, \mathrm{d}\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1)\, \mathrm{d}\mathbf{x}. \end{aligned} \tag{5.20}$$

We are free to choose the decision rule that assigns each point $\mathbf{x}$ to one of the two classes. Clearly, to minimize $p(\text{mistake})$ we should arrange that each $\mathbf{x}$ is assigned to whichever class has the smaller value of the integrand in (5.20). Thus, if $p(\mathbf{x}, \mathcal{C}_1) > p(\mathbf{x}, \mathcal{C}_2)$ for a given value of $\mathbf{x}$, then we should assign that $\mathbf{x}$ to class $\mathcal{C}_1$. From the product rule of probability, we have $p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k|\mathbf{x})p(\mathbf{x})$. Because the factor $p(\mathbf{x})$ is common to both terms, we can restate this result as saying that the minimum probability of making a mistake is obtained if each value of $\mathbf{x}$ is assigned to the class for which the posterior probability $p(\mathcal{C}_k|\mathbf{x})$ is largest. This result is illustrated for two classes and a single input variable x in Figure 5.5.

For the more general case of K classes, it is slightly easier to maximize the probability of being correct, which is given by

$$\begin{aligned} p(\text{correct}) &= \sum_{k=1}^{K} p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) \\ &= \sum_{k=1}^{K} \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k)\, \mathrm{d}\mathbf{x}, \end{aligned} \tag{5.21}$$

which is maximized when the regions $\mathcal{R}_k$ are chosen such that each $\mathbf{x}$ is assigned to the class for which $p(\mathbf{x}, \mathcal{C}_k)$ is largest. Again, using the product rule $p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k|\mathbf{x})p(\mathbf{x})$, and noting that the factor of $p(\mathbf{x})$ is common to all terms, we see that each $\mathbf{x}$ should be assigned to the class having the largest posterior probability $p(\mathcal{C}_k|\mathbf{x})$.

5.2.2 Expected loss

For many applications, our objective will be more complex than simply minimizing the number of misclassifications. Let us consider again the medical diagnosis problem. We note that, if a patient who does not have cancer is incorrectly diagnosed as having cancer, the consequences may be that they experience some distress plus there is the need for further investigations. Conversely, if a patient with cancer is diagnosed as healthy, the result may be premature death due to lack of treatment. Thus, the consequences of these two types of mistake can be dramatically different. It would clearly be better to make fewer mistakes of the second kind, even if this was at the expense of making more mistakes of the first kind.

We can formalize such issues through the introduction of a *loss function*, also called a *cost function*, which is a single, overall measure of loss incurred in taking any of the available decisions or actions. Our goal is then to minimize the total loss incurred. Note that some authors consider instead a *utility function*, whose value

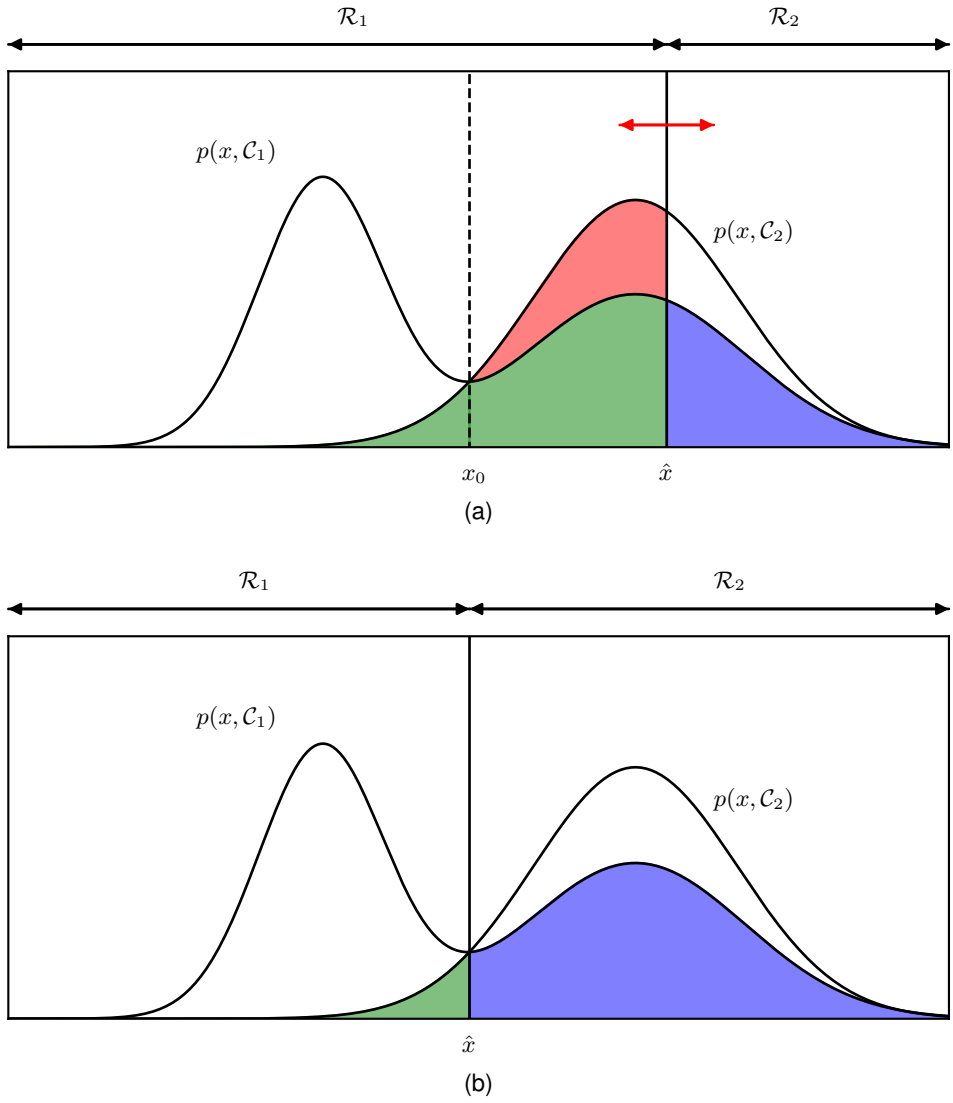

Figure 5.5 Schematic illustration of the joint probabilities $p(x, \mathcal{C}_k)$ for each of two classes plotted against x, together with the decision boundary $x = \widehat{x}$. Values of $x \geqslant \widehat{x}$ are classified as class $\mathcal{C}_2$ and hence belong to decision region $\mathcal{R}_2$, whereas points $x < \widehat{x}$ are classified as $\mathcal{C}_1$ and belong to $\mathcal{R}_1$. Errors arise from the blue, green, and red regions, so that for $x < \widehat{x}$, the errors are due to points from class $\mathcal{C}_2$ being misclassified as $\mathcal{C}_1$ (represented by the sum of the red and green regions). Conversely for points in the region $x \geqslant \widehat{x}$, the errors are due to points from class $\mathcal{C}_1$ being misclassified as $\mathcal{C}_2$ (represented by the blue region). By varying the location $\widehat{x}$ of the decision boundary, as indicated by the red double-headed arrow in (a), the combined areas of the blue and green regions remains constant, whereas the size of the red region varies. The optimal choice for $\widehat{x}$ is where the curves for $p(x, \mathcal{C}_1)$ and $p(x, \mathcal{C}_2)$ cross, as shown in (b) and corresponding to $\widehat{x} = x_0$, because in this case the red region disappears. This is equivalent to the minimum misclassification rate decision rule, which assigns each value of x to the class having the higher posterior probability $p(\mathcal{C}_k|x)$.

Figure 5.6 An example of a loss matrix with elements L_{kj} for the cancer treatment problem. The rows correspond to the true class, whereas the columns correspond to the assignment of class made by our decision criterion.

	normal	cancer
normal	0	1
cancer	100	0

they aim to maximize. These are equivalent concepts if we take the utility to be simply the negative of the loss. Throughout this text we will use the loss function convention. Suppose that, for a new value of $\mathbf{x}$, the true class is $\mathcal{C}_k$ and that we assign $\mathbf{x}$ to class $\mathcal{C}_j$ (where j may or may not be equal to k). In so doing, we incur some level of loss that we denote by L_{kj}, which we can view as the k, j element of a *loss matrix*. For instance, in our cancer example, we might have a loss matrix of the form shown in Figure 5.6. This particular loss matrix says that there is no loss incurred if the correct decision is made, there is a loss of 1 if a healthy patient is diagnosed as having cancer, whereas there is a loss of 100 if a patient having cancer is diagnosed as healthy.

The optimal solution is the one that minimizes the loss function. However, the loss function depends on the true class, which is unknown. For a given input vector $\mathbf{x}$, our uncertainty in the true class is expressed through the joint probability distribution $p(\mathbf{x}, \mathcal{C}_k)$, and so we seek instead to minimize the average loss, where the average is computed with respect to this distribution and is given by

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) \, \mathrm{d}\mathbf{x}. \tag{5.22}$$

Each $\mathbf{x}$ can be assigned independently to one of the decision regions $\mathcal{R}_j$. Our goal is to choose the regions $\mathcal{R}_j$ to minimize the expected loss (5.22), which implies that for each $\mathbf{x}$, we should minimize $\sum_k L_{kj} p(\mathbf{x}, \mathcal{C}_k)$. As before, we can use the product rule $p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k|\mathbf{x})p(\mathbf{x})$ to eliminate the common factor of $p(\mathbf{x})$. Thus, the decision rule that minimizes the expected loss assigns each new $\mathbf{x}$ to the class j for which the quantity

$$\sum_k L_{kj} p(\mathcal{C}_k|\mathbf{x}) \tag{5.23}$$

is a minimum. Once we have chosen values for the loss matrix elements L_{kj}, this is clearly trivial to do.

5.2.3 The reject option

We have seen that classification errors arise from the regions of input space where the largest of the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$ is significantly less than unity or equivalently where the joint distributions $p(\mathbf{x}, \mathcal{C}_k)$ have comparable values. These are the regions where we are relatively uncertain about class membership. In some applications, it will be appropriate to avoid making decisions on the difficult cases in anticipation of obtaining a lower error rate on those examples for which a classification decision is made. This is known as the *reject option*. For example, in our hypothetical cancer screening example, it may be appropriate to use an automatic

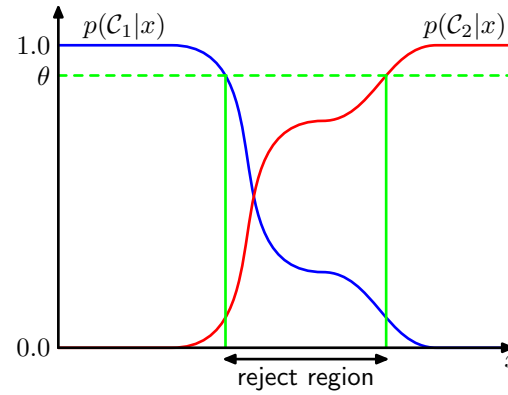

Figure 5.7 Illustration of the reject option. Inputs x such that the larger of the two posterior probabilities is less than or equal to some threshold θ will be rejected.

system to classify those images for which there is little doubt as to the correct class, while requesting a biopsy to classify the more ambiguous cases. We can achieve this by introducing a threshold θ and rejecting those inputs $\mathbf{x}$ for which the largest of the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$ is less than or equal to θ. This is illustrated for two classes and a single continuous input variable x in Figure 5.7. Note that setting $\theta = 1$ will ensure that all examples are rejected, whereas if there are K classes, then setting $\theta < 1/K$ will ensure that no examples are rejected. Thus, the fraction of examples that are rejected is controlled by the value of θ.

We can easily extend the reject criterion to minimize the expected loss, when a loss matrix is given, by taking account of the loss incurred when a reject decision is made.

Exercise 5.10

5.2.4 Inference and decision

We have broken the classification problem down into two separate stages, the *inference stage* in which we use training data to learn a model for $p(\mathcal{C}_k|\mathbf{x})$ and the subsequent *decision* stage in which we use these posterior probabilities to make optimal class assignments. An alternative possibility would be to solve both problems together and simply learn a function that maps inputs $\mathbf{x}$ directly into decisions. Such a function is called a *discriminant function*.

In fact, we can identify three distinct approaches to solving decision problems, all of which have been used in practical applications. These are, in decreasing order of complexity, as follows:

(a) First, solve the inference problem of determining the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ for each class $\mathcal{C}_k$ individually. Separately infer the prior class probabilities $p(\mathcal{C}_k)$. Then use Bayes' theorem in the form

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} \tag{5.24}$$

to find the posterior class probabilities $p(\mathcal{C}_k|\mathbf{x})$. As usual, the denominator in

Bayes' theorem can be found in terms of the quantities in the numerator, using

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k). \tag{5.25}$$

Equivalently, we can model the joint distribution $p(\mathbf{x}, \mathcal{C}_k)$ directly and then normalize to obtain the posterior probabilities. Having found the posterior probabilities, we use decision theory to determine the class membership for each new input $\mathbf{x}$. Approaches that explicitly or implicitly model the distribution of inputs as well as outputs are known as *generative models*, because by sampling from them, it is possible to generate synthetic data points in the input space.

(b) First, solve the inference problem of determining the posterior class probabilities $p(\mathcal{C}_k|\mathbf{x})$, and then subsequently use decision theory to assign each new $\mathbf{x}$ to one of the classes. Approaches that model the posterior probabilities directly are called *discriminative models*.

(c) Find a function $f(\mathbf{x})$, called a discriminant function, that maps each input $\mathbf{x}$ directly onto a class label. For instance, for two-class problems, $f(\cdot)$ might be binary valued and such that $f = 0$ represents class $\mathcal{C}_1$ and $f = 1$ represents class $\mathcal{C}_2$. In this case, probabilities play no role.

Let us consider the relative merits of these three alternatives. Approach (a) is the most demanding because it involves finding the joint distribution over both $\mathbf{x}$ and $\mathcal{C}_k$. For many applications, $\mathbf{x}$ will have high dimensionality, and consequently, we may need a large training set to be able to determine the class-conditional densities to reasonable accuracy. Note that the class priors $p(\mathcal{C}_k)$ can often be estimated simply from the fractions of the training set data points in each of the classes. One advantage of approach (a), however, is that it also allows the marginal density of data $p(\mathbf{x})$ to be determined from (5.25). This can be useful for detecting new data points that have low probability under the model and for which the predictions may be of low accuracy, which is known as *outlier detection* or *novelty detection* (Bishop, 1994; Tarassenko, 1995).

However, if we wish only to make classification decisions, then it can be wasteful of computational resources and excessively demanding of data to find the joint distribution $p(\mathbf{x}, \mathcal{C}_k)$ when in fact we really need only the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$, which can be obtained directly through approach (b). Indeed, the class-conditional densities may contain a significant amount of structure that has little effect on the posterior probabilities, as illustrated in Figure 5.8. There has been much interest in exploring the relative merits of generative and discriminative approaches to machine learning and in finding ways to combine them (Jebara, 2004; Lasserre, Bishop, and Minka, 2006).

An even simpler approach is (c) in which we use the training data to find a discriminant function $f(\mathbf{x})$ that maps each $\mathbf{x}$ directly onto a class label, thereby combining the inference and decision stages into a single learning problem. In the example of Figure 5.8, this would correspond to finding the value of x shown by

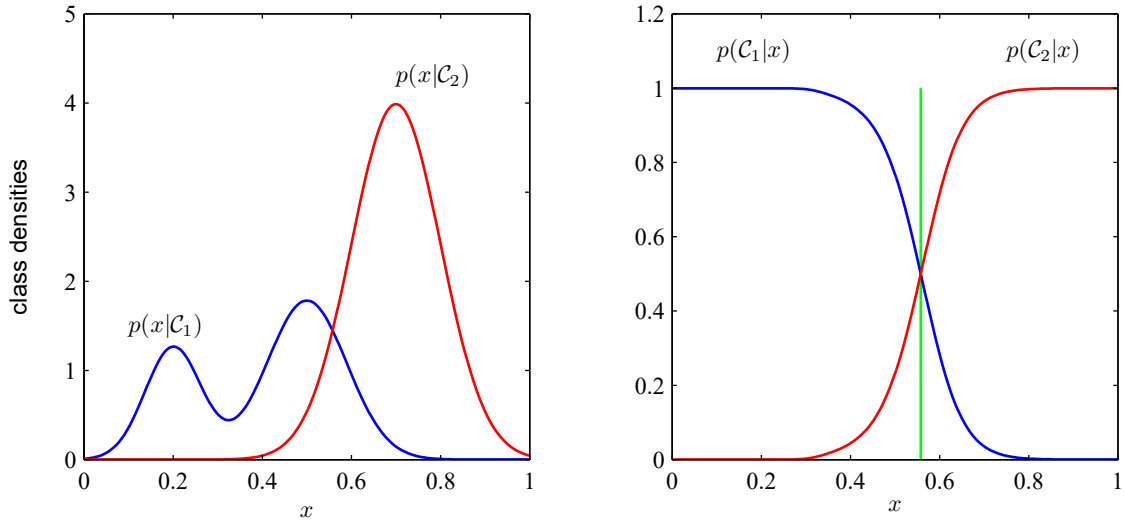

Figure 5.8 Example of the class-conditional densities for two classes having a single input variable x (left plot) together with the corresponding posterior probabilities (right plot). Note that the left-hand mode of the class-conditional density $p(\mathbf{x}|\mathcal{C}_1)$, shown in blue on the left plot, has no effect on the posterior probabilities. The vertical green line in the right plot shows the decision boundary in x that gives the minimum misclassification rate, assuming the prior class probabilities, $p(\mathcal{C}_1)$ and $p(\mathcal{C}_2)$, are equal.

the vertical green line, because this is the decision boundary giving the minimum probability of misclassification.

With option (c), however, we no longer have access to the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$. There are many powerful reasons for wanting to compute the posterior probabilities, even if we subsequently use them to make decisions. These include:

Minimizing risk. Consider a problem in which the elements of the loss matrix are subjected to revision from time to time (such as might occur in a financial application). If we know the posterior probabilities, we can trivially revise the minimum risk decision criterion by modifying (5.23) appropriately. If we have only a discriminant function, then any change to the loss matrix would require that we return to the training data and solve the inference problem afresh.

Reject option. Posterior probabilities allow us to determine a rejection criterion that will minimize the misclassification rate, or more generally the expected loss, for a given fraction of rejected data points.

Section 2.1.1

Compensating for class priors. Consider our cancer screening example again, and suppose that we have collected a large number of images from the general population for use as training data, which we use to build an automated screening system. Because cancer is rare amongst the general population, we might find that, say, only 1 in every 1,000 examples corresponds to the presence of cancer.

If we used such a data set to train an adaptive model, we could run into severe difficulties due to the small proportion of those in the cancer class. For instance, a classifier that assigned every point to the normal class would achieve 99.9% accuracy, and it may be difficult to avoid this trivial solution. Also, even a large data set will contain very few examples of skin images corresponding to cancer, and so the learning algorithm will not be exposed to a broad range of examples of such images and hence is not likely to generalize well. A balanced data set with equal numbers of examples from each of the classes would allow us to find a more accurate model. However, we then have to compensate for the effects of our modifications to the training data. Suppose we have used such a modified data set and found models for the posterior probabilities. From Bayes' theorem (5.24), we see that the posterior probabilities are proportional to the prior probabilities, which we can interpret as the fractions of points in each class. We can therefore simply take the posterior probabilities obtained from our artificially balanced data set, divide by the class fractions in that data set, and then multiply by the class fractions in the population to which we wish to apply the model. Finally, we need to normalize to ensure that the new posterior probabilities sum to one. Note that this procedure cannot be applied if we have learned a discriminant function directly instead of determining posterior probabilities.

Combining models. For complex applications, we may wish to break the problem into a number of smaller sub-problems each of which can be tackled by a separate module. For example, in our hypothetical medical diagnosis problem, we may have information available from, say, blood tests as well as skin images. Rather than combine all of this heterogeneous information into one huge input space, it may be more effective to build one system to interpret the images and a different one to interpret the blood data. If each of the two models gives posterior probabilities for the classes, then we can combine the outputs systematically using the rules of probability. One simple way to do this is to assume that, for each class separately, the distributions of inputs for the images, denoted by $\mathbf{x}_{\mathrm{I}}$, and the blood data, denoted by $\mathbf{x}_{\mathrm{B}}$, are independent, so that

$$p(\mathbf{x}_{\mathrm{I}}, \mathbf{x}_{\mathrm{B}}|\mathcal{C}_k) = p(\mathbf{x}_{\mathrm{I}}|\mathcal{C}_k)p(\mathbf{x}_{\mathrm{B}}|\mathcal{C}_k). \tag{5.26}$$

Section 11.2

This is an example of a *conditional independence* property, because the independence holds when the distribution is conditioned on the class $\mathcal{C}_k$. The posterior probability, given both the image and blood data, is then given by

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}_{\mathrm{I}}, \mathbf{x}_{\mathrm{B}}) &\propto p(\mathbf{x}_{\mathrm{I}}, \mathbf{x}_{\mathrm{B}}|\mathcal{C}_k)p(\mathcal{C}_k) \\ &\propto p(\mathbf{x}_{\mathrm{I}}|\mathcal{C}_k)p(\mathbf{x}_{\mathrm{B}}|\mathcal{C}_k)p(\mathcal{C}_k) \\ &\propto \frac{p(\mathcal{C}_k|\mathbf{x}_{\mathrm{I}})p(\mathcal{C}_k|\mathbf{x}_{\mathrm{B}})}{p(\mathcal{C}_k)}. \end{aligned} \tag{5.27}$$

Thus, we need the class prior probabilities $p(\mathcal{C}_k)$, which we can easily estimate from the fractions of data points in each class, and then we need to normalize

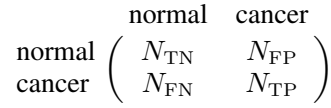

Figure 5.9 The confusion matrix for the cancer treatment problem, in which the rows correspond to the true class and the columns correspond to the assignment of class made by our decision criterion. The elements of the matrix show the numbers of true negatives, false positives, false negatives, and true positives.

the resulting posterior probabilities so they sum to one. The particular conditional independence assumption (5.26) is an example of a *naive Bayes model*. Note that the joint marginal distribution $p(\mathbf{x}_{\mathrm{I}}, \mathbf{x}_{\mathrm{B}})$ will typically not factorize under this model. We will see in later chapters how to construct models for combining data that do not require the conditional independence assumption (5.26). A further advantage of using models that output probabilities rather than decisions is that they can easily be made differentiable with respect to any adjustable parameters (such as the weight coefficients in the polynomial regression example), which allows them to be composed and trained jointly using gradient-based optimization methods.

Section 11.2.3

Chapter 7

5.2.5 Classifier accuracy

The simplest measure of performance for a classifier is the fraction of test set points that are correctly classified. However, we have seen that different types of error can have different consequences, as expressed through the loss matrix, and often we therefore do not simply wish to minimize the number of misclassifications. By changing the location of the decision boundary, we can make trade-offs between different kinds of error, for example with the goal of minimizing an expected loss. Because this is such an important concept, we will introduce some definitions and terminology so that the performance of a classifier can be better characterized.

Section 2.1.1

We will consider again our cancer screening example. For each person tested, there is a 'true label' of whether they have cancer or not, and there is also the prediction made by the classifier. If, for a particular person, the classifier predicts cancer and this is in fact the true label, then the prediction is called a *true positive*. However, if the person does not have cancer it is a *false positive*. Likewise, if the classifier predicts that a person does not have cancer and this is correct, then the prediction is called a *true negative*, otherwise it is a *false negative*. The false positives are also known as *type 1 errors* whereas the false negatives are called *type 2 errors*. If N is the total number of people taking the test, then N_{TP} is the number of true positives, N_{FP} is the number of false positives, N_{TN} is the number of true negatives, and N_{FN} is the number of false negatives, where

$$N = N_{\mathrm{TP}} + N_{\mathrm{FP}} + N_{\mathrm{TN}} + N_{\mathrm{FN}}. \tag{5.28}$$

This can be represented as a *confusion matrix* as shown in Figure 5.9. Accuracy, measured by the fraction of correct classifications, is then given by

$$\text{Accuracy} = \frac{N_{\mathrm{TP}} + N_{\mathrm{TN}}}{N_{\mathrm{TP}} + N_{\mathrm{FP}} + N_{\mathrm{TN}} + N_{\mathrm{FN}}}. \tag{5.29}$$

We can see that accuracy can be misleading if there are strongly imbalanced classes. In our cancer screening example, for instance, where only 1 person in $1{,}000$ has cancer, a naive classifier that simply decides that nobody has cancer will achieve 99.9% accuracy and yet is completely useless.

Several other quantities can be defined in terms of these numbers, of which the most commonly encountered are

$$\text{Precision} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FP}}} \tag{5.30}$$

$$\text{Recall} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FN}}} \tag{5.31}$$

$$\text{False positive rate} = \frac{N_{\text{FP}}}{N_{\text{FP}} + N_{\text{TN}}} \tag{5.32}$$

$$\text{False discovery rate} = \frac{N_{\text{FP}}}{N_{\text{FP}} + N_{\text{TP}}} \tag{5.33}$$

In our cancer screening example, precision represents an estimate of the probability that a person who has a positive test does indeed have cancer, whereas recall is an estimate of the probability that a person who has cancer is correctly detected by the test. The false positive rate is an estimate of the probability that a person who is normal will be classified as having cancer, whereas the false discovery rate represents the fraction of those testing positive who do not in fact have cancer.

By altering the location of the decision boundary, we can change the trade-offs between the two kinds of errors. To understand this trade-off, we revisit Figure 5.5, but now we label the various regions as shown in Figure 5.10. We can relate the labelled regions to the various true and false rates as follows:

$$N_{\text{FP}}/N = E \tag{5.34}$$

$$N_{\text{TP}}/N = D + E \tag{5.35}$$

$$N_{\text{FN}}/N = B + C \tag{5.36}$$

$$N_{\text{TN}}/N = A + C \tag{5.37}$$

where we are implicitly considering the limit $N \to \infty$ so that we can relate number of observations to probabilities.

5.2.6 ROC curve

A probabilistic classifier will output a posterior probability, which can be converted to a decision by setting a threshold. As the value of the threshold is varied, we can reduce type 1 errors at the expense of increasing type 2 errors, or vice versa. To better understand this trade-off, it is useful to plot the *receiver operating characteristic* or *ROC curve* (Fawcett, 2006), a name that originates from procedures to measure the performance of radar receivers. This is a graph of true positive rate versus false positive rate, as shown in Figure 5.11.

As the decision boundary in Figure 5.10 is moved from $-\infty$ to ∞, the ROC curve is traced out and can then be generated by plotting the cumulative fraction of

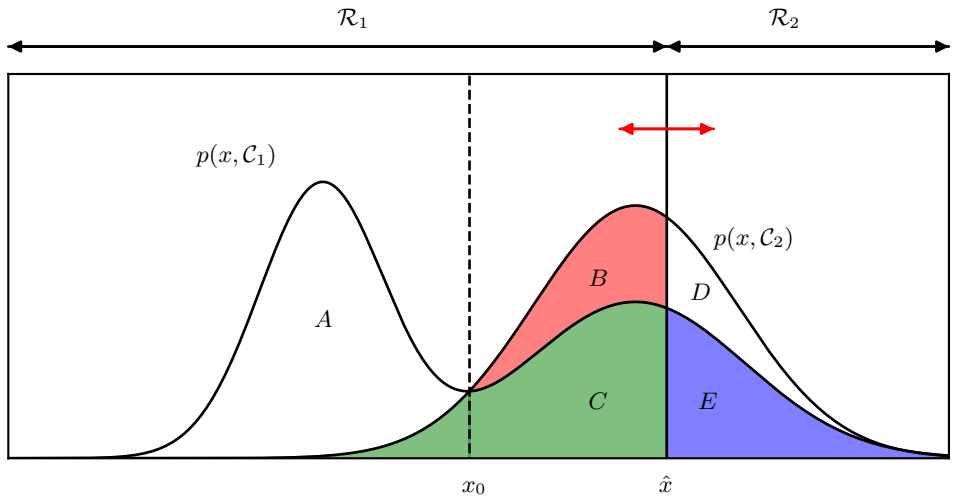

Figure 5.10 As in Figure 5.5, with the various regions labelled. In the cancer classification problem, region $\mathcal{R}_1$ is assigned to the normal class whereas region $\mathcal{R}_2$ is assigned to the cancer class.

correct detection of cancer on the y-axis versus the cumulative fraction of incorrect detection on the x-axis. Note that a specific confusion matrix represents one point along the ROC curve. The best possible classifier would be represented by a point at the top left corner of the ROC diagram. The bottom left corner represents a simple classifier that assigns every point to the normal class and therefore has no true positives but also no false positives. Similarly, the top right corner represents a classifier that assigns everything to the cancer class and therefore has no false negatives but also no true negatives. In Figure 5.11, the classifiers represented by the blue curve are better than those of the red curve for any choice of, say, false positive rate. It is also possible, however, for such curves to cross over, in which case the choice of which is better will depend on the choice of operating point.

As a baseline, we can consider a random classifier that simply assigns each data point to cancer with probability ρ and to normal with probability $1 - \rho$. As we vary the value of ρ it will trace out an ROC curve given by a diagonal straight line, as shown in Figure 5.11. Any classifier below the diagonal line performs worse than random guessing.

Sometimes it is useful to have a single number that characterises the whole ROC curve. One approach is to measure the area under the curve (AUC). A value of 0.5 for the AUC represents random guessing whereas a value of 1.0 represents a perfect classifier.

Another measure is the *F-score*, which is the geometric mean of precision and

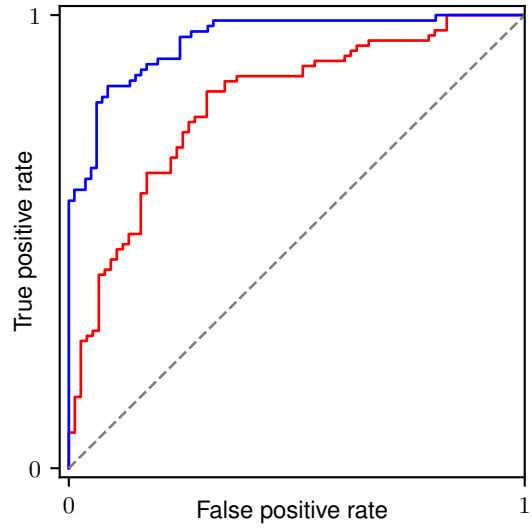

Figure 5.11 The receiver operator characteristic (ROC) curve is a plot of true positive rate against false positive rate, and it characterizes the trade-off between type 1 and type 2 errors in a classification problem. The upper blue curve represents a better classifier than the lower red curve. Here the dashed curve represents the performance of a simple random classifier.

recall, and is therefore defined by

$$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{5.38}$$

$$= \frac{2N_{\text{TP}}}{2N_{\text{TP}} + N_{\text{FP}} + N_{\text{FN}}}. \tag{5.39}$$

Of course, we can also combine the confusion matrix in Figure 5.9 with the loss matrix in Figure 5.6 to compute the expected loss by multiplying the elements pointwise and summing the resulting products.

Although the ROC curve can be extended to more than two classes, it rapidly becomes cumbersome as the number of classes increases.

5.3. Generative Classifiers

Section 5.2.4

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. We have already discussed the distinction between the discriminative and the generative approaches to classification. Here we will adopt a generative approach in which we model the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ as well as the class priors $p(\mathcal{C}_k)$ and then use these to compute posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$ through Bayes' theorem.

First, consider problems having two classes. The posterior probability for class

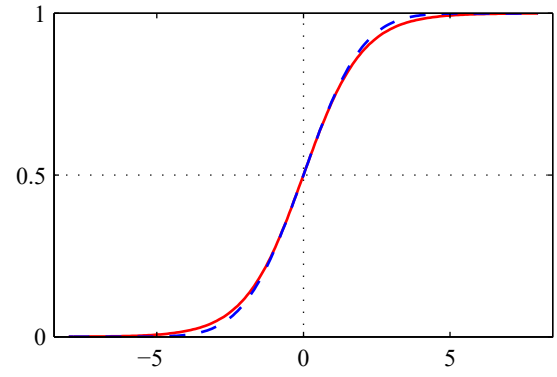

Figure 5.12 Plot of the logistic sigmoid function $\sigma(a)$ defined by (5.42), shown in red, together with the scaled probit function $\Phi(\lambda a)$, for $\lambda^2 = \pi/8$, shown in dashed blue, where $\Phi(a)$ is defined by (5.86). The scaling factor $\pi/8$ is chosen so that the derivatives of the two curves are equal for $a = 0$.

$\mathcal{C}_1$ can be written as

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a)} = \sigma(a) \end{aligned} \tag{5.40}$$

where we have defined

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{5.41}$$

and $\sigma(a)$ is the *logistic sigmoid* function defined by

$$\sigma(a) = \frac{1}{1+\exp(-a)}, \tag{5.42}$$

which is plotted in Figure 5.12. The term 'sigmoid' means S-shaped. This type of function is sometimes also called a 'squashing function' because it maps the whole real axis into a finite interval. The logistic sigmoid has been encountered already in earlier chapters and plays an important role in many classification algorithms. It satisfies the following symmetry property:

$$\sigma(-a) = 1 - \sigma(a) \tag{5.43}$$

as is easily verified. The inverse of the logistic sigmoid is given by

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right) \tag{5.44}$$

and is known as the *logit* function. It represents the log of the ratio of probabilities $\ln\left[p(\mathcal{C}_1|\mathbf{x})/p(\mathcal{C}_2|\mathbf{x})\right]$ for the two classes, also known as the *log odds*.

Note that in (5.40), we have simply rewritten the posterior probabilities in an equivalent form, and so the appearance of the logistic sigmoid may seem artificial.

However, it will have significance provided $a(\mathbf{x})$ has a constrained functional form. We will shortly consider situations in which $a(\mathbf{x})$ is a linear function of $\mathbf{x}$, in which case the posterior probability is governed by a generalized linear model.

If there are $K > 2$ classes, we have

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)}, \end{aligned} \tag{5.45}$$

which is known as the *normalized exponential* and can be regarded as a multi-class generalization of the logistic sigmoid. Here the quantities a_k are defined by

$$a_k = \ln\left(p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)\right). \tag{5.46}$$

The normalized exponential is also known as the *softmax function*, as it represents a smoothed version of the 'max' function because, if $a_k \gg a_j$ for all $j \neq k$, then $p(\mathcal{C}_k|\mathbf{x}) \simeq 1$, and $p(\mathcal{C}_j|\mathbf{x}) \simeq 0$.

We now investigate the consequences of choosing specific forms for the class-conditional densities, looking first at continuous input variables $\mathbf{x}$ and then discussing briefly discrete inputs.

5.3.1 Continuous inputs

Let us assume that the class-conditional densities are Gaussian. We will then explore the resulting form for the posterior probabilities. To start with, we will assume that all classes share the same covariance matrix $\mathbf{\Sigma}$. Thus, the density for class $\mathcal{C}_k$ is given by

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}. \tag{5.47}$$

First, suppose that we have two classes. From (5.40) and (5.41), we have

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0) \tag{5.48}$$

where we have defined

$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{5.49}$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}. \tag{5.50}$$

We see that the quadratic terms in $\mathbf{x}$ from the exponents of the Gaussian densities have cancelled (due to the assumption of common covariance matrices), leading to a linear function of $\mathbf{x}$ in the argument of the logistic sigmoid. This result is illustrated for a two-dimensional input space $\mathbf{x}$ in Figure 5.13. The resulting decision boundaries correspond to surfaces along which the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$

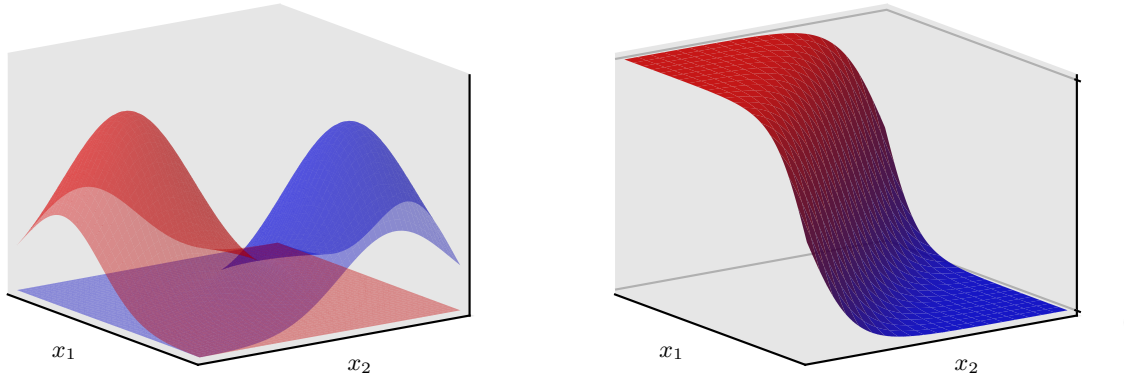

Figure 5.13 The left-hand plot shows the class-conditional densities for two classes, denoted red and blue. On the right is the corresponding posterior probability $p(\mathcal{C}_1|\mathbf{x})$, which is given by a logistic sigmoid of a linear function of $\mathbf{x}$. The surface in the right-hand plot is coloured using a proportion of red ink given by $p(\mathcal{C}_1|\mathbf{x})$ and a proportion of blue ink given by $p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$.

are constant and so will be given by linear functions of $\mathbf{x}$, and therefore the decision boundaries are linear in input space. The prior probabilities $p(\mathcal{C}_k)$ enter only through the bias parameter w_0, so that changes in the priors have the effect of making parallel shifts of the decision boundary and more generally of the parallel contours of constant posterior probability.

For the general case of K classes, the posterior probabilities are given by (5.45) where, from (5.46) and (5.47), we have

$$a_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0} \tag{5.51}$$

in which we have defined

$$\mathbf{w}_k = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \tag{5.52}$$

$$w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln p(\mathcal{C}_k). \tag{5.53}$$

We see that the $a_k(\mathbf{x})$ are again linear functions of $\mathbf{x}$ as a consequence of the cancellation of the quadratic terms due to the shared covariances. The resulting decision boundaries, corresponding to the minimum misclassification rate, will occur when two of the posterior probabilities (the two largest) are equal, and so will be defined by linear functions of $\mathbf{x}$. Thus, we again have a generalized linear model.

If we relax the assumption of a shared covariance matrix and allow each class-conditional density $p(\mathbf{x}|\mathcal{C}_k)$ to have its own covariance matrix $\boldsymbol{\Sigma}_k$, then the earlier cancellations will no longer occur, and we will obtain quadratic functions of $\mathbf{x}$, giving rise to a *quadratic discriminant*. The linear and quadratic decision boundaries are illustrated in Figure 5.14.

5.3.2 Maximum likelihood solution

Once we have specified a parametric functional form for the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$, we can then determine the values of the parameters, together with

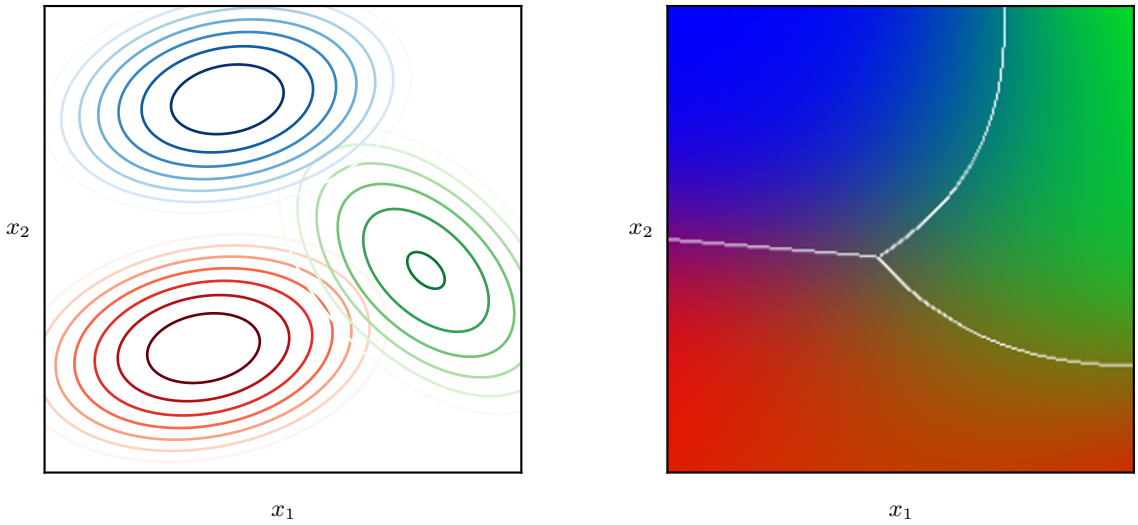

Figure 5.14 The left-hand plot shows the class-conditional densities for three classes each having a Gaussian distribution, coloured red, green, and blue, in which the red and blue classes have the same covariance matrix. The right-hand plot shows the corresponding posterior probabilities, in which each point on the image is coloured using proportions of red, blue, and green ink corresponding to the posterior probabilities for the respective three classes. The decision boundaries are also shown. Notice that the boundary between the red and blue classes, which have the same covariance matrix, is linear, whereas those between the other pairs of classes are quadratic.

the prior class probabilities $p(\mathcal{C}_k)$, using maximum likelihood. This requires a data set comprising observations of $\mathbf{x}$ along with their corresponding class labels.

First, suppose we have two classes, each having a Gaussian class-conditional density with a shared covariance matrix, and suppose we have a data set $\{\mathbf{x}_n, t_n\}$ where $n = 1, \ldots, N$. Here $t_n = 1$ denotes class $\mathcal{C}_1$ and $t_n = 0$ denotes class $\mathcal{C}_2$. We denote the prior class probability $p(\mathcal{C}_1) = \pi$, so that $p(\mathcal{C}_2) = 1 - \pi$. For a data point $\mathbf{x}_n$ from class $\mathcal{C}_1$, we have $t_n = 1$ and hence

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}).$$

Similarly for class $\mathcal{C}_2$, we have $t_n = 0$ and hence

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}).$$

Thus, the likelihood function is given by

$$p(\mathbf{t}, \mathbf{X}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})\right]^{t_n} \left[(1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})\right]^{1-t_n} \quad (5.54)$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$. As usual, it is convenient to maximize the log of the likelihood function. Consider first the maximization with respect to π. The terms in

the log likelihood function that depend on π are

$$\sum_{n=1}^{N} \{t_n \ln \pi + (1 - t_n) \ln(1 - \pi)\} . \tag{5.55}$$

Setting the derivative with respect to π equal to zero and rearranging, we obtain

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2} \tag{5.56}$$

where N_1 denotes the total number of data points in class $\mathcal{C}_1$, and N_2 denotes the total number of data points in class $\mathcal{C}_2$. Thus, the maximum likelihood estimate for π is simply the fraction of points in class $\mathcal{C}_1$ as expected. This result is easily generalized to the multi-class case where again the maximum likelihood estimate of the prior probability associated with class $\mathcal{C}_k$ is given by the fraction of the training set points assigned to that class.

Exercise 5.13

Now consider the maximization with respect to $\boldsymbol{\mu}_1$. Again, we can pick out of the log likelihood function those terms that depend on $\boldsymbol{\mu}_1$:

$$\sum_{n=1}^{N} t_n \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2} \sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const.} \tag{5.57}$$

Setting the derivative with respect to $\boldsymbol{\mu}_1$ to zero and rearranging, we obtain

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n \mathbf{x}_n, \tag{5.58}$$

which is simply the mean of all the input vectors $\mathbf{x}_n$ assigned to class $\mathcal{C}_1$. By a similar argument, the corresponding result for $\boldsymbol{\mu}_2$ is given by

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^{N} (1 - t_n) \mathbf{x}_n, \tag{5.59}$$

which again is the mean of all the input vectors $\mathbf{x}_n$ assigned to class $\mathcal{C}_2$.

Finally, consider the maximum likelihood solution for the shared covariance matrix $\boldsymbol{\Sigma}$. Picking out the terms in the log likelihood function that depend on $\boldsymbol{\Sigma}$, we have

$$\begin{aligned} &-\frac{1}{2} \sum_{n=1}^{N} t_n \ln |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) \\ &-\frac{1}{2} \sum_{n=1}^{N} (1 - t_n) \ln |\boldsymbol{\Sigma}| - \frac{1}{2} \sum_{n=1}^{N} (1 - t_n) (\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2) \\ &= -\frac{N}{2} \ln |\boldsymbol{\Sigma}| - \frac{N}{2} \mathrm{Tr} \left\{ \boldsymbol{\Sigma}^{-1} \mathbf{S} \right\} \end{aligned} \tag{5.60}$$

where we have defined

$$\mathbf{S} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2 \tag{5.61}$$

$$\mathbf{S}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \tag{5.62}$$

$$\mathbf{S}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}}. \tag{5.63}$$

Using the standard result for the maximum likelihood solution for a Gaussian distribution, we see that $\boldsymbol{\Sigma} = \mathbf{S}$, which represents a weighted average of the covariance matrices associated with each of the two classes separately.

This result is easily extended to the K-class problem to obtain the corresponding maximum likelihood solutions for the parameters in which each class-conditional density is Gaussian with a shared covariance matrix. Note that the approach of fitting Gaussian distributions to the classes is not robust to outliers, because the maximum likelihood estimation of a Gaussian is not robust.

Exercise 5.14

Section 5.1.4

5.3.3 Discrete features

Let us now consider discrete feature values x_i. For simplicity, we begin by looking at binary feature values $x_i \in \{0, 1\}$ and discuss the extension to more general discrete features shortly. If there are D inputs, then a general distribution would correspond to a table of 2^D numbers for each class and have $2^D - 1$ independent variables (due to the summation constraint). Because this grows exponentially with the number of features, we can seek a more restricted representation. Here we will make the *naive Bayes* assumption in which the feature values are treated as independent and conditioned on the class $\mathcal{C}_k$. Thus, we have class-conditional distributions of the form

Section 11.2.3

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{i=1}^{D}\mu_{ki}^{x_i}(1 - \mu_{ki})^{1-x_i}, \tag{5.64}$$

which contain D independent parameters for each class. Substituting into (5.46) then gives

$$a_k(\mathbf{x}) = \sum_{i=1}^{D}\{x_i \ln \mu_{ki} + (1 - x_i)\ln(1 - \mu_{ki})\} + \ln p(\mathcal{C}_k), \tag{5.65}$$

which again are linear functions of the input values x_i. For $K = 2$ classes, we can alternatively consider the logistic sigmoid formulation given by (5.40). Analogous results are obtained for discrete variables that take $L > 2$ states.

Exercise 5.16

5.3.4 Exponential family

As we have seen, for both Gaussian distributed and discrete inputs, the posterior class probabilities are given by generalized linear models with logistic sigmoid ($K =$

2 classes) or softmax ($K \geqslant 2$ classes) activation functions. These are particular cases of a more general result obtained by assuming that the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ are members of the subset of the exponential family of distributions given by

Section 3.4

$$p(\mathbf{x}|\boldsymbol{\lambda}_k, s) = \frac{1}{s} h\left(\frac{1}{s}\mathbf{x}\right) g(\boldsymbol{\lambda}_k) \exp\left\{\frac{1}{s}\boldsymbol{\lambda}_k^{\mathrm{T}}\mathbf{x}\right\}. \tag{5.66}$$

Here the scaling parameter s is shared across all the classes.

For the two-class problem, we substitute this expression for the class-conditional densities into (5.41) and we see that the posterior class probability is again given by a logistic sigmoid acting on a linear function $a(\mathbf{x})$, which is given by

$$a(\mathbf{x}) = (\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)^{\mathrm{T}}\mathbf{x} + \ln g(\boldsymbol{\lambda}_1) - \ln g(\boldsymbol{\lambda}_2) + \ln p(\mathcal{C}_1) - \ln p(\mathcal{C}_2). \tag{5.67}$$

Similarly, for the K-class problem, we substitute the class-conditional density expression into (5.46) to give

$$a_k(\mathbf{x}) = \boldsymbol{\lambda}_k^{\mathrm{T}}\mathbf{x} + \ln g(\boldsymbol{\lambda}_k) + \ln p(\mathcal{C}_k) \tag{5.68}$$

and so again is a linear function of $\mathbf{x}$.

5.4. Discriminative Classifiers

For the two-class classification problem, we have seen that the posterior probability of class $\mathcal{C}_1$ can be written as a logistic sigmoid acting on a linear function of $\mathbf{x}$, for a wide choice of class-conditional distributions $p(\mathbf{x}|\mathcal{C}_k)$ from the exponential family. Similarly, for the multi-class case, the posterior probability of class $\mathcal{C}_k$ is given by a softmax transformation of linear functions of $\mathbf{x}$. For specific choices of the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$, we have used maximum likelihood to determine the parameters of the densities as well as the class priors $p(\mathcal{C}_k)$ and then used Bayes' theorem to find the posterior class probabilities. This represents an example of *generative* modelling, because we could take such a model and generate synthetic data by drawing values of $\mathbf{x}$ from the marginal distribution $p(\mathbf{x})$ or from any of the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$.

However, an alternative approach is to use the functional form of the generalized linear model explicitly and to determine its parameters directly by using maximum likelihood. In this direct approach, we maximize a likelihood function defined through the conditional distribution $p(\mathcal{C}_k|\mathbf{x})$, which represents a form of *discriminative* probabilistic modelling. One advantage of the discriminative approach is that there will typically be fewer learnable parameters to be determined, as we will see shortly. It may also lead to improved predictive performance, particularly when the assumed forms for the class-conditional densities represent a poor approximation to the true distributions.

5.4.1 Activation functions

Chapter 4

In linear regression, the model prediction $y(\mathbf{x}, \mathbf{w})$ is given by a linear function of the parameters

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0, \tag{5.69}$$

which gives a continuous-valued output in the range $(-\infty, \infty)$. 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 $\mathbf{w}$ and w_0 using a nonlinear function $f(\,\cdot\,)$ so that

$$y(\mathbf{x}, \mathbf{w}) = f\left(\mathbf{w}^{\mathrm{T}}\mathbf{w} + w + 0\right). \tag{5.70}$$

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 $\mathbf{w}^{\mathrm{T}}\mathbf{x} = \text{constant}$, 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 (5.70) are called *generalized linear models* (McCullagh and Nelder, 1989). However, in contrast to the models used for regression, they are no longer linear in the parameters due to 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 much more flexible nonlinear models that will be studied in subsequent chapters.

5.4.2 Fixed basis functions

So far in this chapter, we have considered classification models that work directly with the original input vector $\mathbf{x}$. However, all the algorithms are equally applicable if we first make a fixed nonlinear transformation of the inputs using a vector of basis functions $\boldsymbol{\phi}(\mathbf{x})$. The resulting decision boundaries will be linear in the feature space $\boldsymbol{\phi}$, and these correspond to nonlinear decision boundaries in the original $\mathbf{x}$ space, as illustrated in Figure 5.15. Classes that are linearly separable in the feature space $\boldsymbol{\phi}(\mathbf{x})$ need not be linearly separable in the original observation space $\mathbf{x}$.

Note that as in our discussion of linear models for regression, one of the basis functions is typically set to a constant, say $\phi_0(\mathbf{x}) = 1$, so that the corresponding parameter w_0 plays the role of a bias.

For many problems of practical interest, there is significant overlap in $\mathbf{x}$-space between the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$. This corresponds to posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$, which, for at least some values of $\mathbf{x}$, are not 0 or 1. In such cases, the optimal solution is obtained by modelling the posterior probabilities accurately and then applying standard decision theory. Note that nonlinear transformations $\boldsymbol{\phi}(\mathbf{x})$ cannot remove such a class overlap, although they can increase the level of overlap or create an overlap where none existed in the original observation space. However, suitable choices of nonlinearity can make the process of modelling the posterior probabilities easier. However, such fixed basis function models have important limitations, and these will be resolved in later chapters by allowing the basis functions themselves to adapt to the data.

Section 5.2

Section 6.1

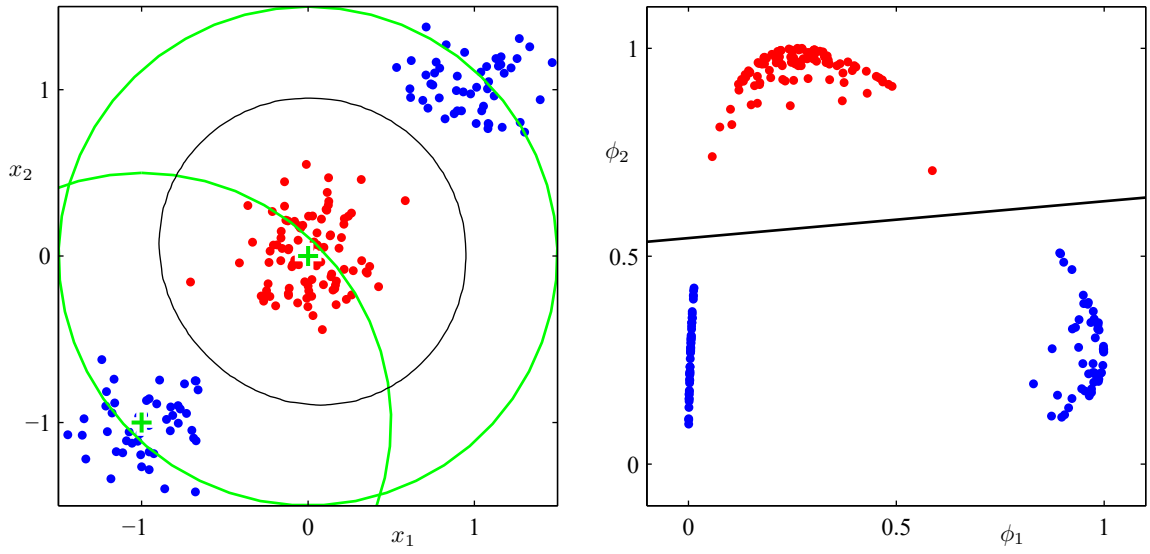

Figure 5.15 Illustration of the role of nonlinear basis functions in linear classification models. The left-hand plot shows the original input space (x_1, x_2) together with data points from two classes labelled red and blue. Two 'Gaussian' basis functions $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ are defined in this space with centres shown by the green crosses and with contours shown by the green circles. The right-hand plot shows the corresponding feature space (ϕ_1, ϕ_2) together with the linear decision boundary obtained given by a logistic regression model of the form discussed in Section 5.4.3. This corresponds to a nonlinear decision boundary in the original input space, shown by the black curve in the left-hand plot.

5.4.3 Logistic regression

We first consider the problem of two-class classification. In our discussion of generative approaches in Section 5.3, we saw that under rather general assumptions, the posterior probability of class $\mathcal{C}_1$ can be written as a logistic sigmoid acting on a linear function of the feature vector $\boldsymbol{\phi}$ so that

$$p(\mathcal{C}_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma\left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}\right) \tag{5.71}$$

with $p(\mathcal{C}_2|\boldsymbol{\phi}) = 1 - p(\mathcal{C}_1|\boldsymbol{\phi})$. Here $\sigma(\cdot)$ is the *logistic sigmoid* function defined by (5.42). In the terminology of statistics, this model is known as *logistic regression*, although it should be emphasized that this is a model for classification rather than for continuous variable.

For an M-dimensional feature space $\boldsymbol{\phi}$, this model has M adjustable parameters. By contrast, if we had fitted Gaussian class-conditional densities using maximum likelihood, we would have used $2M$ parameters for the means and $M(M+1)/2$ parameters for the (shared) covariance matrix. Together with the class prior $p(\mathcal{C}_1)$, this gives a total of $M(M+5)/2+1$ parameters, which grows quadratically with M, in contrast to the linear dependence on M of the number of parameters in logistic regression. For large values of M, there is a clear advantage in working with the logistic regression model directly.

We now use maximum likelihood to determine the parameters of the logistic regression model. To do this, we will make use of the derivative of the logistic sigmoid function, which can conveniently be expressed in terms of the sigmoid function itself:

Exercise 5.18

$$\frac{\mathrm{d}\sigma}{\mathrm{d}a} = \sigma(1-\sigma). \tag{5.72}$$

For a data set $\{\boldsymbol{\phi}_n, t_n\}$, where $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$ and $t_n \in \{0, 1\}$, with $n = 1, \ldots, N$, the likelihood function can be written

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} \{1 - y_n\}^{1-t_n} \tag{5.73}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$ and $y_n = p(\mathcal{C}_1|\boldsymbol{\phi}_n)$. As usual, we can define an error function by taking the negative logarithm of the likelihood, which gives the *cross-entropy* error function:

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} \tag{5.74}$$

where $y_n = \sigma(a_n)$ and $a_n = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n$. Taking the gradient of the error function with respect to $\mathbf{w}$, we obtain

Exercise 5.19

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\boldsymbol{\phi}_n \tag{5.75}$$

where we have made use of (5.72). We see that the factor involving the derivative of the logistic sigmoid has cancelled, leading to a simplified form for the gradient of the log likelihood. In particular, the contribution to the gradient from data point n is given by the 'error' $y_n - t_n$ between the target value and the prediction of the model times the basis function vector $\boldsymbol{\phi}_n$. Furthermore, comparison with (4.12) shows that this takes precisely the same form as the gradient of the sum-of-squares error function for the linear regression model.

Section 4.1.3

The maximum likelihood solution corresponds to $\nabla E(\mathbf{w}) = 0$. However, from (5.75) we see that this no longer corresponds to a set of linear equations, due to the nonlinearity in $y(\cdot)$, and so this equation does not have a closed-form solution. One approach to finding a maximum likelihood solution would be to use stochastic gradient descent, in which ∇E_n is the nth term on the right-hand side of (5.75).

Chapter 7

Stochastic gradient descent will be the principal approach to training the highly non-linear neural networks discussed in later chapters. However, the maximum likelihood equation is only 'slightly' nonlinear, and in fact the error function (5.74), in which the model is defined by (5.71), is a convex function of the parameters, which allows the error function to be minimized using a simple algorithm called *iterative reweighted least squares* or IRLS (Bishop, 2006). However, this does not easily generalize to more complex models such as deep neural networks.

Note that maximum likelihood can exhibit severe over-fitting for data sets that are linearly separable. This arises because the maximum likelihood solution occurs when the hyperplane corresponding to $\sigma = 0.5$, equivalent to $\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi} = 0$, separates the two classes and the magnitude of $\mathbf{w}$ goes to infinity. In this case, the logistic sigmoid function becomes infinitely steep in feature space, corresponding to a Heaviside step function, so that every training point from each class k is assigned a posterior probability $p(\mathcal{C}_k|\mathbf{x}) = 1$. Furthermore, there is typically a continuum of such solutions because any separating hyperplane will give rise to the same posterior probabilities at the training data points. Maximum likelihood provides no way to favour one such solution over another, and which solution is found in practice will depend on the choice of optimization algorithm and on the parameter initialization. Note that the problem will arise even if the number of data points is large compared with the number of parameters in the model, so long as the training data set is linearly separable. The singularity can be avoided by adding a regularization term to the error function.

Exercise 5.20

Chapter 9

5.4.4 Multi-class logistic regression

Section 5.3

In our discussion of generative models for multi-class classification, we have seen that, for a large class of distributions from the exponential family, the posterior probabilities are given by a softmax transformation of linear functions of the feature variables, so that

$$p(\mathcal{C}_k|\boldsymbol{\phi}) = y_k(\boldsymbol{\phi}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \tag{5.76}$$

where the pre-activations a_k are given by

$$a_k = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}. \tag{5.77}$$

There we used maximum likelihood to determine separately the class-conditional densities and the class priors and then found the corresponding posterior probabilities using Bayes' theorem, thereby implicitly determining the parameters $\{\mathbf{w}_k\}$. Here we consider the use of maximum likelihood to determine the parameters $\{\mathbf{w}_k\}$ of this model directly. To do this, we will require the derivatives of y_k with respect to all the pre-activations a_j. These are given by

Exercise 5.21

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j) \tag{5.78}$$

where I_{kj} are the elements of the identity matrix.

Next we write down the likelihood function. This is most easily done using the 1-of-K coding scheme in which the target vector $\mathbf{t}_n$ for a feature vector $\boldsymbol{\phi}_n$ belonging to class $\mathcal{C}_k$ is a binary vector with all elements zero except for element k, which equals one. The likelihood function is then given by

$$p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(\mathcal{C}_k|\boldsymbol{\phi}_n)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}} \tag{5.79}$$

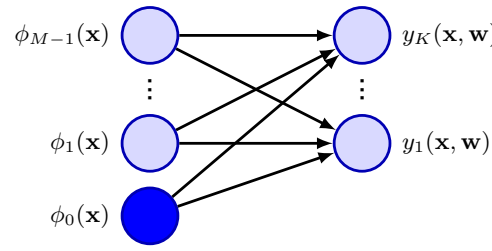

Figure 5.16 Representation of a multi-class linear classification model as a neural network having a single layer of connections. Each basis function is represented by a node, with the solid node representing the 'bias' basis function ϕ_0, whereas each output $y_1, \ldots, y_N$ is also represented by a node. The links between the nodes represent the corresponding weight and bias parameters.

where $y_{nk} = y_k(\boldsymbol{\phi}_n)$, and $\mathbf{T}$ is an $N \times K$ matrix of target variables with elements t_{nk}. Taking the negative logarithm then gives

$$E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}, \tag{5.80}$$

which is known as the *cross-entropy* error function for the multi-class classification problem.

We now take the gradient of the error function with respect to one of the parameter vectors $\mathbf{w}_j$. Making use of the result (5.78) for the derivatives of the softmax function, we obtain

Exercise 5.22

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\, \boldsymbol{\phi}_n \tag{5.81}$$

where we have made use of $\sum_k t_{nk} = 1$. Again, we could optimize the parameters through stochastic gradient descent.

Chapter 7

Once again, we see the same form arising for the gradient as was found for the sum-of-squares error function with the linear model and for the cross-entropy error with the logistic regression model, namely the product of the error $(y_{nj} - t_{nj})$ times the basis function activation $\boldsymbol{\phi}_n$. These are examples of a more general result that we will explore later.

Section 5.4.6

Linear classification models can be represented as single-layer neural networks as shown in Figure 5.16. If we consider the derivative of the error function with respect to a weight w_{ik}, which links basis function $\phi_i(\mathbf{x})$ to output unit t_k, we have from (5.81)

$$\frac{\partial E(\mathbf{w}_1, \ldots, \mathbf{w}_K)}{\partial w_{ij}} = \sum_{n=1}^{N} (y_{nk} - t_{nk})\, \phi_i(\mathbf{x}_n). \tag{5.82}$$

Comparing this with Figure 5.16, we see that, for each data point n this gradient takes the form of the output of the basis function at the input end of the weight link with the 'error' $(y_{nk} - t_{nk})$ at the output end.

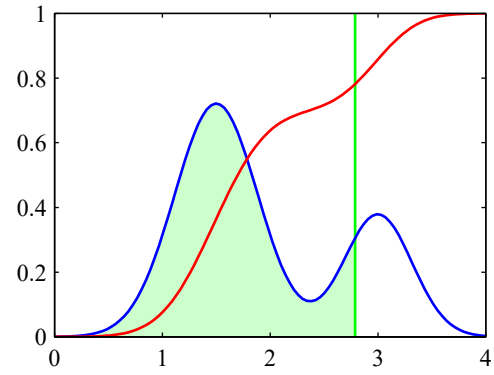

Figure 5.17 Schematic example of a probability density $p(\theta)$ shown by the blue curve, given in this example by a mixture of two Gaussians, along with its cumulative distribution function $f(a)$, shown by the red curve. Note that the value of the blue curve at any point, such as that indicated by the vertical green line, corresponds to the slope of the red curve at the same point. Conversely, the value of the red curve at this point corresponds to the area under the blue curve indicated by the shaded green region. In the stochastic threshold model, the class label takes the value $t = 1$ if the value of $a = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}$ exceeds a threshold, otherwise it takes the value $t = 0$. This is equivalent to an activation function given by the cumulative distribution function $f(a)$.

5.4.5 Probit regression

We have seen that, for a broad range of class-conditional distributions described by the exponential family, the resulting posterior class probabilities are given by a logistic (or softmax) transformation acting on a linear function of the feature variables. However, not all choices of class-conditional density give rise to such a simple form for the posterior probabilities, which suggests that it might be worth exploring other types of discriminative probabilistic model. Consider the two-class case, again remaining within the framework of generalized linear models, so that

$$p(t = 1|a) = f(a) \tag{5.83}$$

where $a = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}$, and $f(\cdot)$ is the activation function.

One way to motivate an alternative choice for the link function is to consider a noisy threshold model, as follows. For each input $\boldsymbol{\phi}_n$, we evaluate $a_n = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n$ and then we set the target value according to

$$\begin{cases} t_n = 1, & \text{if } a_n \geqslant \theta, \\ t_n = 0, & \text{otherwise.} \end{cases} \tag{5.84}$$

If the value of θ is drawn from a probability density $p(\theta)$, then the corresponding activation function will be given by the cumulative distribution function

$$f(a) = \int_{-\infty}^{a} p(\theta)\,\mathrm{d}\theta \tag{5.85}$$

as illustrated in Figure 5.17.

As a specific example, suppose that the density $p(\theta)$ is given by a zero-mean, unit-variance Gaussian. The corresponding cumulative distribution function is given by

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta|0, 1)\,\mathrm{d}\theta, \tag{5.86}$$

which is known as the *probit* function. It has a sigmoidal shape and is compared with the logistic sigmoid function in Figure 5.12. Note that the use of a Gaussian distribution with general mean and variances does not change the model because this is equivalent to a re-scaling of the linear coefficients $\mathbf{w}$. Many numerical packages can evaluate a closely related function defined by

$$\text{erf}(a) = \frac{2}{\sqrt{\pi}} \int_0^a \exp(-\theta^2/2)\,\mathrm{d}\theta \tag{5.87}$$

and known as the *erf function* or *error function* (not to be confused with the error function of a machine learning model). It is related to the probit function by

Exercise 5.23

$$\Phi(a) = \frac{1}{2}\left\{1 + \frac{1}{\sqrt{2}}\text{erf}(a)\right\}. \tag{5.88}$$

The generalized linear model based on a probit activation function is known as *probit regression*. We can determine the parameters of this model using maximum likelihood by a straightforward extension of the ideas discussed earlier. In practice, the results found using probit regression tend to be like those of logistic regression.

One issue that can occur in practical applications is that of *outliers*, which can arise for instance through errors in measuring the input vector $\mathbf{x}$ or through mislabelling of the target value t. Because such points can lie a long way to the wrong side of the ideal decision boundary, they can seriously distort the classifier. The logistic and probit regression models behave differently in this respect because the tails of the logistic sigmoid decay asymptotically like $\exp(-x)$ for $|x| \to \infty$, whereas for the probit activation function, they decay like $\exp(-x^2)$, and so the probit model can be significantly more sensitive to outliers.

5.4.6 Canonical link functions

For the linear regression model with a Gaussian noise distribution, the error function, corresponding to the negative log likelihood, is given by (4.11). If we take the derivative with respect to the parameter vector $\mathbf{w}$ of the contribution to the error function from a data point n, this takes the form of the 'error' $y_n - t_n$ times the feature vector $\boldsymbol{\phi}_n$, where $y_n = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n$. Similarly, for the combination of the logistic-sigmoid activation function and the cross-entropy error function (5.74) and for the softmax activation function with the multi-class cross-entropy error function (5.80), we again obtain this same simple form. We now show that this is a general result of assuming a conditional distribution for the target variable from the exponential family along with a corresponding choice for the activation function known as the *canonical link function*.

We again make use of the restricted form (3.169) of exponential family distributions. Note that here we are applying the assumption of exponential family distribution to the target variable t, in contrast to Section 5.3.4 where we applied it to the input vector $\mathbf{x}$. We therefore consider conditional distributions of the target variable of the form

$$p(t|\eta, s) = \frac{1}{s}h\left(\frac{t}{s}\right) g(\eta)\exp\left\{\frac{\eta t}{s}\right\}. \tag{5.89}$$

Using the same line of argument as led to the derivation of the result (3.172), we see that the conditional mean of t, which we denote by y, is given by

$$y \equiv \mathbb{E}[t|\eta] = -s\frac{d}{d\eta}\ln g(\eta). \tag{5.90}$$

Thus, y and η must related, and we denote this relation through $\eta = \psi(y)$.

Following Nelder and Wedderburn (1972), we define a *generalized linear model* to be one for which y is a nonlinear function of a linear combination of the input (or feature) variables so that

$$y = f(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) \tag{5.91}$$

where $f(\cdot)$ is known as the activation function in the machine learning literature, and $f^{-1}(\cdot)$ is known as the link function in statistics.

Now consider the log likelihood function for this model, which, as a function of η, is given by

$$\ln p(\mathbf{t}|\eta, s) = \sum_{n=1}^{N} \ln p(t_n|\eta, s) = \sum_{n=1}^{N} \left\{ \ln g(\eta_n) + \frac{\eta_n t_n}{s} \right\} + \text{const} \tag{5.92}$$

where we are assuming that all observations share a common scale parameter (which corresponds to the noise variance for a Gaussian distribution, for instance) and so s is independent of n. The derivative of the log likelihood with respect to the model parameters $\mathbf{w}$ is then given by

$$\begin{aligned} \nabla_{\mathbf{w}} \ln p(\mathbf{t}|\eta, s) &= \sum_{n=1}^{N} \left\{ \frac{\mathrm{d}}{\mathrm{d}\eta_n} \ln g(\eta_n) + \frac{t_n}{s} \right\} \frac{\mathrm{d}\eta_n}{\mathrm{d}y_n} \frac{\mathrm{d}y_n}{\mathrm{d}a_n} \nabla_{\mathbf{w}} a_n \\ &= \sum_{n=1}^{N} \frac{1}{s} \{t_n - y_n\} \psi'(y_n) f'(a_n) \boldsymbol{\phi}_n \end{aligned} \tag{5.93}$$

where $a_n = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n$, and we have used $y_n = f(a_n)$ together with the result (5.90) for $\mathbb{E}[t|\eta]$. We now see that there is a considerable simplification if we choose a particular form for the link function $f^{-1}(y)$ given by

$$f^{-1}(y) = \psi(y), \tag{5.94}$$

which gives $f(\psi(y)) = y$ and hence $f'(\psi)\psi'(y) = 1$. Also, because $a = f^{-1}(y)$, we have $a = \psi$ and hence $f'(a)\psi'(y) = 1$. In this case, the gradient of the error function reduces to

$$\nabla \ln E(\mathbf{w}) = \frac{1}{s} \sum_{n=1}^{N} \{y_n - t_n\} \boldsymbol{\phi}_n. \tag{5.95}$$

We have seen that there is a natural pairing between the choice of error function and the choice of output-unit activation function. Although we have derived this result in the context of single-layer network models, the same considerations apply to deep neural networks discussed in later chapters.