CHAPTER

Classifiers Based on Bayes Decision Theory

2

2.1 INTRODUCTION

This is the first chapter, out of three, dealing with the design of the classifier in a pattern recognition system. The approach to be followed builds upon probabilistic arguments stemming from the statistical nature of the generated features. As has already been pointed out in the introductory chapter, this is due to the statistical variation of the patterns as well as to the noise in the measuring sensors. Adopting this reasoning as our kickoff point, we will design classifiers that classify an unknown pattern in the most probable of the classes. Thus, our task now becomes that of defining what "most probable" means.

Given a classification task of M classes, $\omega_1, \omega_2, \ldots, \omega_M$, and an unknown pattern, which is represented by a feature vector $\boldsymbol{x}$, we form the M conditional probabilities $P(\omega_i|\boldsymbol{x}), i = 1, 2, \ldots, M$. Sometimes, these are also referred to as *a posteriori probabilities*. In words, each of them represents the probability that the unknown pattern belongs to the respective class ω_i, given that the corresponding feature vector takes the value $\boldsymbol{x}$. Who could then argue that these conditional probabilities are not sensible choices to quantify the term *most probable*? Indeed, the classifiers to be considered in this chapter compute either the maximum of these M values or, equivalently, the maximum of an appropriately defined function of them. The unknown pattern is then assigned to the class corresponding to this maximum.

The first task we are faced with is the computation of the conditional probabilities. The Bayes rule will once more prove its usefulness! A major effort in this chapter will be devoted to techniques for estimating probability density functions (pdf), based on the available experimental evidence, that is, the feature vectors corresponding to the patterns of the training set.

2.2 BAYES DECISION THEORY

We will initially focus on the two-class case. Let ω_1, ω_2 be the two classes in which our patterns belong. In the sequel, we assume that the *a priori probabilities*

$P(\omega_1), P(\omega_2)$ are known. This is a very reasonable assumption, because even if they are not known, they can easily be estimated from the available training feature vectors. Indeed, if N is the total number of available training patterns, and N_1, N_2 of them belong to ω_1 and ω_2, respectively, then $P(\omega_1) \approx N_1/N$ and $P(\omega_2) \approx N_2/N$.

The other statistical quantities assumed to be known are the class-conditional probability density functions $p(\boldsymbol{x}|\omega_i), i = 1, 2$, describing the distribution of the feature vectors in each of the classes. If these are not known, they can also be estimated from the available training data, as we will discuss later on in this chapter. The pdf $p(\boldsymbol{x}|\omega_i)$ is sometimes referred to as the *likelihood function of ω_i with respect to $\boldsymbol{x}$*. Here we should stress the fact that an implicit assumption has been made. That is, the feature vectors can take any value in the l-dimensional feature space. In the case that feature vectors can take only discrete values, density functions $p(\boldsymbol{x}|\omega_i)$ become probabilities and will be denoted by $P(\boldsymbol{x}|\omega_i)$.

We now have all the ingredients to compute our conditional probabilities, as stated in the introduction. To this end, let us recall from our probability course basics the *Bayes rule* (Appendix A)

$$P(\omega_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_i)P(\omega_i)}{p(\boldsymbol{x})} \tag{2.1}$$

where $p(\boldsymbol{x})$ is the pdf of $\boldsymbol{x}$ and for which we have (Appendix A)

$$p(\boldsymbol{x}) = \sum_{i=1}^{2} p(\boldsymbol{x}|\omega_i)P(\omega_i) \tag{2.2}$$

The *Bayes classification rule* can now be stated as

$$\begin{aligned} &\text{If} \quad P(\omega_1|\boldsymbol{x}) > P(\omega_2|\boldsymbol{x}), \quad \boldsymbol{x} \text{ is classified to } \omega_1 \\ &\text{If} \quad P(\omega_1|\boldsymbol{x}) < P(\omega_2|\boldsymbol{x}), \quad \boldsymbol{x} \text{ is classified to } \omega_2 \end{aligned} \tag{2.3}$$

The case of equality is detrimental and the pattern can be assigned to either of the two classes. Using (2.1), the decision can equivalently be based on the inequalities

$$p(\boldsymbol{x}|\omega_1)P(\omega_1) \gtrless p(\boldsymbol{x}|\omega_2)P(\omega_2) \tag{2.4}$$

$p(\boldsymbol{x})$ is not taken into account, because it is the same for all classes and it does not affect the decision. Furthermore, if the *a priori* probabilities are equal, that is, $P(\omega_1) = P(\omega_2) = 1/2$, Eq. (2.4) becomes

$$p(\boldsymbol{x}|\omega_1) \gtrless p(\boldsymbol{x}|\omega_2) \tag{2.5}$$

Thus, the search for the maximum now rests on the values of the conditional pdfs evaluated at $\boldsymbol{x}$. Figure 2.1 presents an example of two equiprobable classes and shows the variations of $p(x|\omega_i), i = 1, 2$, as functions of x for the simple case of a single feature ($l = 1$). The dotted line at x_0 is a threshold partitioning the feature space into two regions, R_1 and R_2. According to the Bayes decision rule, for all values of x in R_1 the classifier decides ω_1 and for all values in R_2 it decides ω_2. However, it is obvious from the figure that decision errors are unavoidable. Indeed, there is

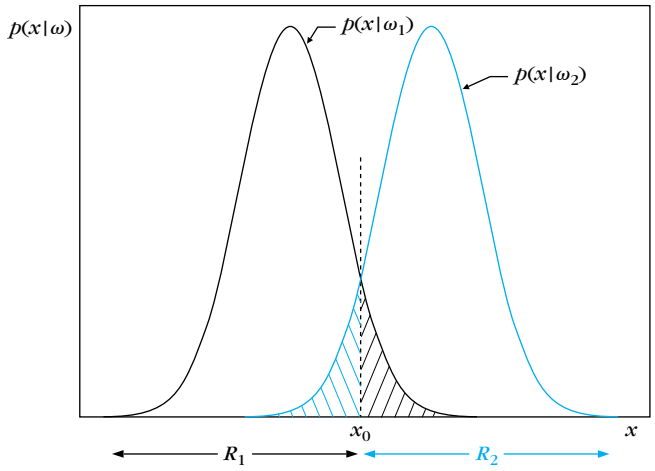

FIGURE 2.1

Example of the two regions R_1 and R_2 formed by the Bayesian classifier for the case of two equiprobable classes.

a finite probability for an x to lie in the R_2 region and at the same time to belong in class ω_1. Then our decision is in error. The same is true for points originating from class ω_2. It does not take much thought to see that the total probability, P_e, of committing a decision error for the case of two equiprobable classes, is given by

$$P_e = \frac{1}{2}\int_{-\infty}^{x_0} p(x|\omega_2)\,dx + \frac{1}{2}\int_{x_0}^{+\infty} p(x|\omega_1)\,dx \tag{2.6}$$

which is equal to the total shaded area under the curves in Figure 2.1. We have now touched on a very important issue. Our starting point to arrive at the Bayes classification rule was rather empirical, via our interpretation of the term *most probable*. We will now see that this classification test, though simple in its formulation, has a sounder mathematical interpretation.

Minimizing the Classification Error Probability

We will show that *the Bayesian classifier is optimal with respect to minimizing the classification error probability*. Indeed, the reader can easily verify, as an exercise, that moving the threshold away from x_0, in Figure 2.1, always increases the corresponding shaded area under the curves. Let us now proceed with a more formal proof.

Proof. Let R_1 be the region of the feature space in which we decide in favor of ω_1 and R_2 be the corresponding region for ω_2. Then an error is made if $\boldsymbol{x} \in R_1$, although it belongs to ω_2 or if $\boldsymbol{x} \in R_2$, although it belongs to ω_1. That is,

$$P_e = P(\boldsymbol{x} \in R_2, \omega_1) + P(\boldsymbol{x} \in R_1, \omega_2) \tag{2.7}$$

where $P(\cdot, \cdot)$ is the joint probability of two events. Recalling, once more, our probability basics (Appendix A), this becomes

$$\begin{aligned} P_e &= P(\boldsymbol{x} \in R_2|\omega_1)P(\omega_1) + P(\boldsymbol{x} \in R_1|\omega_2)P(\omega_2) \\ &= P(\omega_1) \int_{R_2} p(\boldsymbol{x}|\omega_1)\, d\boldsymbol{x} + P(\omega_2) \int_{R_1} p(\boldsymbol{x}|\omega_2)\, d\boldsymbol{x} \end{aligned} \tag{2.8}$$

or using the Bayes rule

$$P_e = \int_{R_2} P(\omega_1|\boldsymbol{x})p(\boldsymbol{x})\, d\boldsymbol{x} + \int_{R_1} P(\omega_2|\boldsymbol{x})p(\boldsymbol{x})\, d\boldsymbol{x} \tag{2.9}$$

It is now easy to see that the error is minimized if *the partitioning regions* R_1 and R_2 *of the feature space are chosen so that*

$$\begin{aligned} R_1&: P(\omega_1|\boldsymbol{x}) > P(\omega_2|\boldsymbol{x}) \\ R_2&: P(\omega_2|\boldsymbol{x}) > P(\omega_1|\boldsymbol{x}) \end{aligned} \tag{2.10}$$

Indeed, since the union of the regions R_1, R_2 covers all the space, from the definition of a probability density function we have that

$$\int_{R_1} P(\omega_1|\boldsymbol{x})p(\boldsymbol{x})\, d\boldsymbol{x} + \int_{R_2} P(\omega_1|\boldsymbol{x})p(\boldsymbol{x})\, d\boldsymbol{x} = P(\omega_1) \tag{2.11}$$

Combining Eqs. (2.9) and (2.11), we get

$$P_e = P(\omega_1) - \int_{R_1} (P(\omega_1|\boldsymbol{x}) - P(\omega_2|\boldsymbol{x}))\, p(\boldsymbol{x})\, d\boldsymbol{x} \tag{2.12}$$

This suggests that the probability of error is minimized if R_1 is the region of space in which $P(\omega_1|\boldsymbol{x}) > P(\omega_2|\boldsymbol{x})$. Then, R_2 becomes the region where the reverse is true. □

So far, we have dealt with the simple case of two classes. Generalizations to the multiclass case are straightforward. In a classification task with M classes, $\omega_1, \omega_2, \ldots, \omega_M$, an unknown pattern, represented by the feature vector $\boldsymbol{x}$, is assigned to class ω_i if

$$P(\omega_i|\boldsymbol{x}) > P(\omega_j|\boldsymbol{x}) \quad \forall j \neq i \tag{2.13}$$

It turns out that such a choice also minimizes the classification error probability (Problem 2.1).

Minimizing the Average Risk

The classification error probability is not always the best criterion to be adopted for minimization. This is because it assigns the same importance to all errors. However, there are cases in which some wrong decisions may have more serious implications than others. For example, it is much more serious for a doctor to make a wrong decision and a malignant tumor to be diagnosed as a benign one, than the other way round. If a benign tumor is diagnosed as a malignant one, the wrong decision will be cleared out during subsequent clinical examinations. However, the results

from the wrong decision concerning a malignant tumor may be fatal. Thus, in such cases it is more appropriate to assign a penalty term to weigh each error. For our example, let us denote by ω_1 the class of malignant tumors and as ω_2 the class of the benign ones. Let, also, R_1, R_2 be the regions in the feature space where we decide in favor of ω_1 and ω_2, respectively. The error probability P_e is given by Eq. (2.8). Instead of selecting R_1 and R_2 so that P_e is minimized, we will now try to minimize a modified version of it, that is,

$$r = \lambda_{12} P(\omega_1) \int_{R_2} p(\boldsymbol{x}|\omega_1) d\boldsymbol{x} + \lambda_{21} P(\omega_2) \int_{R_1} p(\boldsymbol{x}|\omega_2) d\boldsymbol{x} \tag{2.14}$$

where each of the two terms that contributes to the overall error probability is weighted according to its significance. For our case, the reasonable choice would be to have $\lambda_{12} > \lambda_{21}$. Thus errors due to the assignment of patterns originating from class ω_1 to class ω_2 will have a larger effect on the cost function than the errors associated with the second term in the summation.

Let us now consider an M-class problem and let $R_j, j = 1, 2, \ldots, M$, be the regions of the feature space assigned to classes ω_j, respectively. Assume now that a feature vector $\boldsymbol{x}$ that belongs to class ω_k lies in R_i, $i \neq k$. Then this vector is misclassified in ω_i and an error is committed. A penalty term λ_{ki}, known as *loss*, is associated with this wrong decision. The matrix L, which has at its (k, i) location the corresponding penalty term, is known as the *loss matrix*.[1] Observe that in contrast to the philosophy behind Eq. (2.14), we have now allowed weights across the diagonal of the loss matrix (λ_{kk}), which correspond to correct decisions. In practice, these are usually set equal to zero, although we have considered them here for the sake of generality. The *risk or loss* associated with ω_k is defined as

$$r_k = \sum_{i=1}^{M} \lambda_{ki} \int_{R_i} p(\boldsymbol{x}|\omega_k)\, d\boldsymbol{x} \tag{2.15}$$

Observe that the integral is the overall probability of a feature vector from class ω_k being classified in ω_i. This probability is weighted by λ_{ki}. Our goal now is to choose the partitioning regions R_j so that the *average risk*

$$\begin{aligned} r &= \sum_{k=1}^{M} r_k P(\omega_k) \\ &= \sum_{i=1}^{M} \int_{R_i} \left(\sum_{k=1}^{M} \lambda_{ki}\, p(\boldsymbol{x}|\omega_k) P(\omega_k) \right) d\boldsymbol{x} \end{aligned} \tag{2.16}$$

is minimized. This is achieved if each of the integrals is minimized, which is equivalent to selecting partitioning regions so that

$$\boldsymbol{x} \in R_i \quad \text{if} \quad l_i \equiv \sum_{k=1}^{M} \lambda_{ki}\, p(\boldsymbol{x}|\omega_k) P(\omega_k) < l_j \equiv \sum_{k=1}^{M} \lambda_{kj}\, p(\boldsymbol{x}|\omega_k) P(\omega_k) \quad \forall j \neq i \tag{2.17}$$

[1] The terminology comes from the general decision theory.

It is obvious that if $\lambda_{ki} = 1 - \delta_{ki}$, where δ_{ki} is *Kronecker's delta* (0 if $k \neq i$ and 1 if $k = i$), then minimizing the average risk becomes equivalent to minimizing the classification error probability.

The two-class case. For this specific case we obtain

$$\begin{aligned} l_1 &= \lambda_{11}\, p(\boldsymbol{x}|\omega_1)P(\omega_1) + \lambda_{21}\, p(\boldsymbol{x}|\omega_2)P(\omega_2) \\ l_2 &= \lambda_{12}\, p(\boldsymbol{x}|\omega_1)P(\omega_1) + \lambda_{22}\, p(\boldsymbol{x}|\omega_2)P(\omega_2) \end{aligned} \tag{2.18}$$

We assign $\boldsymbol{x}$ to ω_1 if $l_1 < l_2$, that is,

$$(\lambda_{21} - \lambda_{22})p(\boldsymbol{x}|\omega_2)P(\omega_2) < (\lambda_{12} - \lambda_{11})p(\boldsymbol{x}|\omega_1)P(\omega_1) \tag{2.19}$$

It is natural to assume that $\lambda_{ij} > \lambda_{ii}$ (correct decisions are penalized much less than wrong ones). Adopting this assumption, the decision rule (2.17) for the two-class case now becomes

$$\boldsymbol{x} \in \omega_1 (\omega_2) \quad \text{if} \quad l_{12} \equiv \frac{p(\boldsymbol{x}|\omega_1)}{p(\boldsymbol{x}|\omega_2)} > (<) \frac{P(\omega_2)}{P(\omega_1)} \frac{\lambda_{21} - \lambda_{22}}{\lambda_{12} - \lambda_{11}} \tag{2.20}$$

The ratio l_{12} is known as the *likelihood ratio* and the preceding test as the *likelihood ratio test*. Let us now investigate Eq. (2.20) a little further and consider the case of Figure 2.1. Assume that the loss matrix is of the form

$$L = \begin{bmatrix} 0 & \lambda_{12} \\ \lambda_{21} & 0 \end{bmatrix}$$

If misclassification of patterns that come from ω_2 is considered to have serious consequences, then we must choose $\lambda_{21} > \lambda_{12}$. Thus, patterns are assigned to class ω_2 if

$$p(\boldsymbol{x}|\omega_2) > p(\boldsymbol{x}|\omega_1)\frac{\lambda_{12}}{\lambda_{21}}$$

where $P(\omega_1) = P(\omega_2) = 1/2$ has been assumed. That is, $p(\boldsymbol{x}|\omega_1)$ is multiplied by a factor less than 1 and the effect of this is to move the threshold in Figure 2.1 to the left of x_0. In other words, region R_2 is increased while R_1 is decreased. The opposite would be true if $\lambda_{21} < \lambda_{12}$.

An alternative cost that sometimes is used for two class problems is the Neyman-Pearson criterion. The error for one of the classes is now constrained to be fixed and equal to a chosen value (Problem 2.6). Such a decision rule has been used, for example, in radar detection problems. The task there is to detect a target in the presence of noise. One type of error is the so-called *false alarm*—that is, to mistake the noise for a signal (target) present. Of course, the other type of error is to miss the signal and to decide in favor of the noise (*missed detection*). In many cases the error probability of false alarm is set equal to a predetermined threshold.

Example 2.1

In a two-class problem with a single feature x the pdfs are Gaussians with variance $\sigma^2 = 1/2$ for both classes and mean values 0 and 1, respectively, that is,

$$p(x|\omega_1) = \frac{1}{\sqrt{\pi}} \exp(-x^2)$$

$$p(x|\omega_2) = \frac{1}{\sqrt{\pi}} \exp(-(x-1)^2)$$

If $P(\omega_1) = P(\omega_2) = 1/2$, compute the threshold value x_0 (a) for minimum error probability and (b) for minimum risk if the loss matrix is

$$L = \begin{bmatrix} 0 & 0.5 \\ 1.0 & 0 \end{bmatrix}$$

Taking into account the shape of the Gaussian function graph (Appendix A), the threshold for the minimum probability case will be

$$x_0 : \exp(-x^2) = \exp(-(x-1)^2)$$

Taking the logarithm of both sides, we end up with $x_0 = 1/2$. In the minimum risk case we get

$$x_0 : \exp(-x^2) = 2\exp(-(x-1)^2)$$

or $x_0 = (1 - \ln 2)/2 < 1/2$; that is, the threshold moves to the left of 1/2. If the two classes are not equiprobable, then it is easily verified that if $P(\omega_1) > (<) P(\omega_2)$ the threshold moves to the right (left). That is, we expand the region in which we decide in favor of the most probable class, since it is better to make fewer errors for the most probable class.

2.3 DISCRIMINANT FUNCTIONS AND DECISION SURFACES

It is by now clear that minimizing either the risk or the error probability or the Neyman-Pearson criterion is equivalent to partitioning the feature space into M regions, for a task with M classes. If regions R_i, R_j happen to be contiguous, then they are separated by a *decision surface* in the multidimensional feature space. For the minimum error probability case, this is described by the equation

$$P(\omega_i|\boldsymbol{x}) - P(\omega_j|\boldsymbol{x}) = 0 \tag{2.21}$$

From the one side of the surface this difference is positive, and from the other it is negative. Sometimes, instead of working directly with probabilities (or risk functions), it may be more convenient, from a mathematical point of view, to work with an equivalent function of them, for example, $g_i(\boldsymbol{x}) \equiv f(P(\omega_i|\boldsymbol{x}))$, where $f(\cdot)$ is a monotonically increasing function. $g_i(\boldsymbol{x})$ is known as a *discriminant function*. The decision test (2.13) is now stated as

$$\text{classify } \boldsymbol{x} \text{ in } \omega_i \quad \text{if} \quad g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \forall j \neq i \tag{2.22}$$

The decision surfaces, separating contiguous regions, are described by

$$g_{ij}(\boldsymbol{x}) \equiv g_i(\boldsymbol{x}) - g_j(\boldsymbol{x}) = 0, \quad i,j = 1,2,\ldots,M, \quad i \neq j \tag{2.23}$$

So far, we have approached the classification problem via Bayesian probabilistic arguments and the goal was to minimize the classification error probability or the risk. However, as we will soon see, not all problems are well suited to such approaches. For example, in many cases the involved pdfs are complicated and their estimation is not an easy task. In such cases, it may be preferable to compute decision surfaces *directly by means of alternative costs*, and this will be our focus in Chapters 3 and 4. Such approaches give rise to discriminant functions and decision surfaces, which are entities with no (necessary) relation to Bayesian classification, and they are, in general, suboptimal with respect to Bayesian classifiers.

In the following we will focus on a particular family of decision surfaces associated with the Bayesian classification for the specific case of Gaussian density functions.

2.4 BAYESIAN CLASSIFICATION FOR NORMAL DISTRIBUTIONS

2.4.1 The Gaussian Probability Density Function

One of the most commonly encountered probability density functions in practice is the Gaussian or normal probability density function. The major reasons for its popularity are its computational tractability and the fact that it models adequately a large number of cases. One of the most celebrated theorems in statistics is the *central limit theorem*. The theorem states that if a random variable is the outcome of a summation of a number of *independent* random variables, its pdf approaches the Gaussian function as the number of summands tends to infinity (see Appendix A). In practice, it is most common to assume that the sum of random variables is distributed according to a Gaussian pdf, for a sufficiently large number of summing terms.

The one-dimensional or the univariate Gaussian, as it is sometimes called, is defined by

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.24}$$

The parameters μ and σ^2 turn out to have a specific meaning. The mean value of the random variable x is equal to μ, that is,

$$\mu = E[x] \equiv \int_{-\infty}^{+\infty} xp(x)dx \tag{2.25}$$

where $E[\cdot]$ denotes the mean (or expected) value of a random variable. The parameter σ^2 is equal to the variance of x, that is,

$$\sigma^2 = E[(x-\mu)^2] \equiv \int_{-\infty}^{+\infty} (x-\mu)^2 p(x)dx \tag{2.26}$$

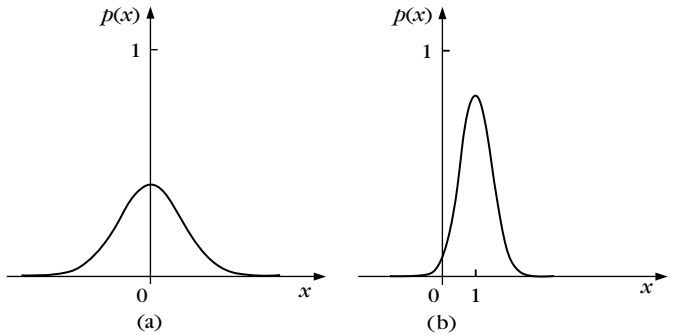

FIGURE 2.2

Graphs for the one-dimensional Gaussian pdf. (a) Mean value $\mu = 0$, $\sigma^2 = 1$, (b) $\mu = 1$ and $\sigma^2 = 0.2$. The larger the variance the broader the graph is. The graphs are symmetric, and they are centered at the respective mean value.

Figure 2.2a shows the graph of the Gaussian function for $\mu = 0$ and $\sigma^2 = 1$, and Figure 2.2b the case for $\mu = 1$ and $\sigma^2 = 0.2$. The larger the variance the broader the graph, which is symmetric, and it is always centered at μ (see Appendix A, for some more properties).

The multivariate generalization of a Gaussian pdf in the l-dimensional space is given by

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{l/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{2.27}$$

where $\boldsymbol{\mu} = E[\boldsymbol{x}]$ is the mean value and Σ is the $l \times l$ *covariance matrix* (Appendix A) defined as

$$\Sigma = E[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T] \tag{2.28}$$

where $|\Sigma|$ denotes the determinant of Σ. It is readily seen that for $l = 1$ the multivariate Gaussian coincides with the univariate one. Sometimes, the symbol $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is used to denote a Gaussian pdf with mean value $\boldsymbol{\mu}$ and covariance Σ.

To get a better feeling on what the multivariate Gaussian looks like, let us focus on some cases in the two-dimensional space, where nature allows us the luxury of visualization. For this case we have

$$\Sigma = E\left[\begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix}\begin{bmatrix} x_1-\mu_1, & x_2-\mu_2 \end{bmatrix}\right] \tag{2.29}$$

$$= \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix} \tag{2.30}$$

where $E[x_i] = \mu_i$, $i = 1, 2$, and by definition $\sigma_{12} = E[(x_1-\mu_1)(x_2-\mu_2)]$, which is known as the covariance between the random variables x_1 and x_2 and it is a measure

of their mutual statistical correlation. If the variables are statistically independent, their covariance is zero (Appendix A). Obviously, the diagonal elements of Σ are the variances of the respective elements of the random vector.

Figures 2.3–2.6 show the graphs for four instances of a two-dimensional Gaussian probability density function. Figure 2.3a corresponds to a Gaussian with a diagonal covariance matrix

$$\Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$$

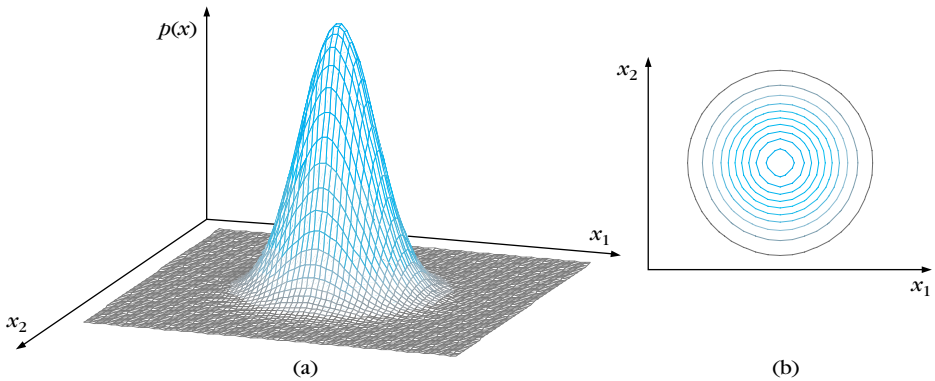

FIGURE 2.3

(a) The graph of a two-dimensional Gaussian pdf and (b) the corresponding isovalue curves for a diagonal Σ with $\sigma_1^2 = \sigma_2^2$. The graph has a spherical symmetry showing no preference in any direction.

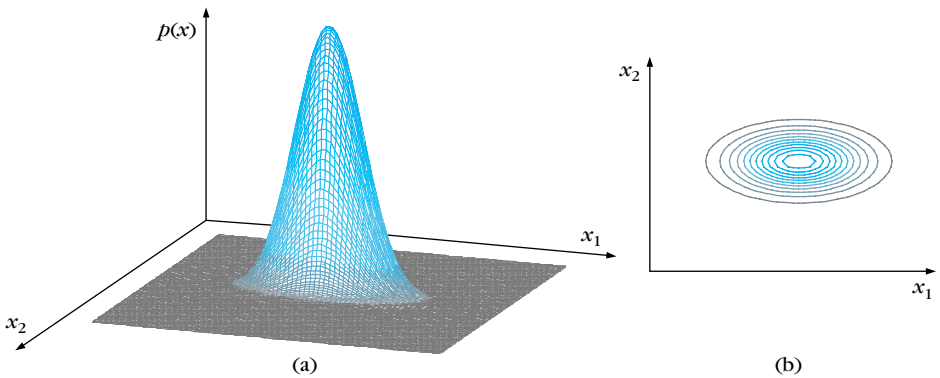

FIGURE 2.4

(a) The graph of a two-dimensional Gaussian pdf and (b) the corresponding isovalue curves for a diagonal Σ with $\sigma_1^2 >> \sigma_2^2$. The graph is elongated along the x_1 direction.

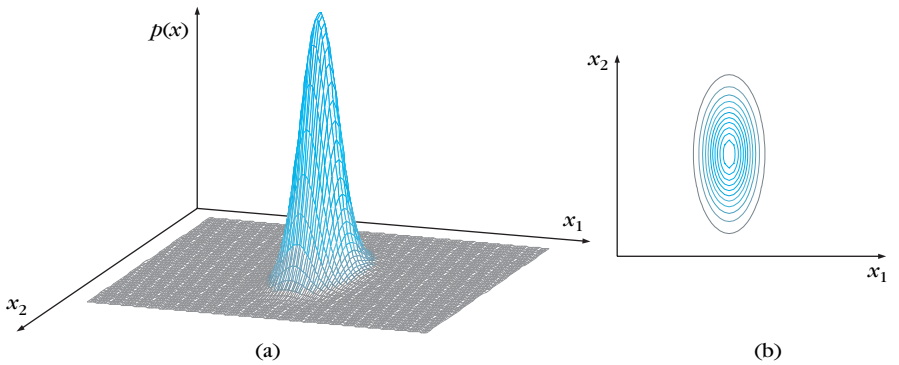

FIGURE 2.5

(a) The graph of a two-dimensional Gaussian pdf and (b) the corresponding isovalue curves for a diagonal Σ with $\sigma_1^2 << \sigma_2^2$. The graph is elongated along the x_2 direction.

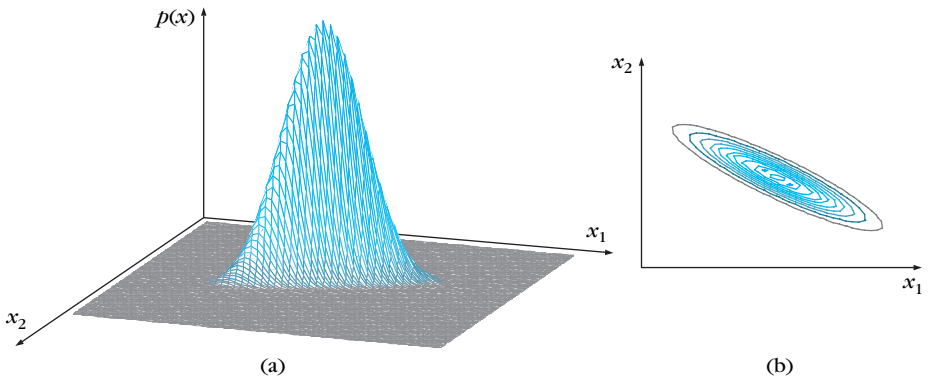

FIGURE 2.6

(a) The graph of a two-dimensional Gaussian pdf and (b) the corresponding isovalue curves for a case of a nondiagonal Σ. Playing with the values of the elements of Σ one can achieve different shapes and orientations.

that is, both features, x_1, x_2 have variance equal to 3 and their covariance is zero. The graph of the Gaussian is symmetric. For this case the isovalue curves (i.e., curves of equal probability density values) are circles (hyperspheres in the general l-dimensional space) and are shown in Figure 2.3b. The case shown in Figure 2.4a corresponds to the covariance matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

with $\sigma_1^2 = 15 >> \sigma_2^2 = 3$. The graph of the Gaussian is now elongated along the x_1-axis, which is the direction of the larger variance. The isovalue curves, shown

in Figure 2.4b, are ellipses. Figures 2.5a and 2.5b correspond to the case with $\sigma_1^2 = 3 << \sigma_2^2 = 15$. Figures 2.6a and 2.6b correspond to the more general case where

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$$

and $\sigma_1^2 = 15$, $\sigma_2^2 = 3$, $\sigma_{12} = 6$. Playing with σ_1^2, σ_2^2 and σ_{12} one can achieve different shapes and different orientations.

The isovalue curves are ellipses of different orientations and with different ratios of major to minor axis lengths. Let us consider, as an example, the case of a zero mean random vector with a diagonal covariance matrix. To compute the isovalue curves is equivalent to computing the curves of constant values for the exponent, that is,

$$\boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x} = [x_1, x_2] \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = C \tag{2.31}$$

or

$$\frac{x_1^2}{\sigma_1^2} + \frac{x_2^2}{\sigma_2^2} = C \tag{2.32}$$

for some constant C. This is the equation of an ellipse whose axes are determined by the the variances of the involved features. As we will soon see, the principal axes of the ellipses are controlled by the eigenvectors/eigenvalues of the covariance matrix. As we know from linear algebra (and it is easily checked), the eigenvalues of a diagonal matrix, which was the case for our example, are equal to the respective elements across its diagonal.

2.4.2 The Bayesian Classifier for Normally Distributed Classes

Our goal in this section is to study the optimal Bayesian classifier when the involved pdfs, $p(\boldsymbol{x}|\omega_i)$, $i = 1, 2, \ldots, M$ (likelihood functions of ω_i with respect to $\boldsymbol{x}$), describing the data distribution in each one of the classes, are multivariate normal distributions, that is, $\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$, $i = 1, 2, \ldots, M$. Because of the exponential form of the involved densities, it is preferable to work with the following discriminant functions, which involve the (monotonic) logarithmic function $\ln(\cdot)$:

$$g_i(\boldsymbol{x}) = \ln(p(\boldsymbol{x}|\omega_i)P(\omega_i)) = \ln p(\boldsymbol{x}|\omega_i) + \ln P(\omega_i) \tag{2.33}$$

or

$$g_i(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i) + \ln P(\omega_i) + c_i \tag{2.34}$$

where c_i is a constant equal to $-(l/2)\ln 2\pi - (1/2)\ln|\Sigma_i|$. Expanding, we obtain

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\boldsymbol{x}^T \Sigma_i^{-1} \boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T \Sigma_i^{-1} \boldsymbol{\mu}_i - \frac{1}{2}\boldsymbol{\mu}_i^T \Sigma_i^{-1} \boldsymbol{\mu}_i + \frac{1}{2}\boldsymbol{\mu}_i^T \Sigma_i^{-1} \boldsymbol{x} \quad + \ln P(\omega_i) + c_i \tag{2.35}$$

In general, this is a nonlinear quadratic form. Take, for example, the case of $l = 2$ and assume that

$$\Sigma_i = \begin{bmatrix} \sigma_i^2 & 0 \\ 0 & \sigma_i^2 \end{bmatrix}$$

Then (2.35) becomes

$$g_i(\boldsymbol{x}) = -\frac{1}{2\sigma_i^2}(x_1^2 + x_2^2) + \frac{1}{\sigma_i^2}(\mu_{i1}x_1 + \mu_{i2}x_2) - \frac{1}{2\sigma_i^2}(\mu_{i1}^2 + \mu_{i2}^2) + \ln P(\omega_i) + c_i \quad (2.36)$$

and obviously the associated decision curves $g_i(\boldsymbol{x}) - g_j(\boldsymbol{x}) = 0$ are *quadrics* (i.e., ellipsoids, parabolas, hyperbolas, pairs of lines). That is, in such cases, the Bayesian classifier is a *quadratic classifier*, in the sense that the partition of the feature space is performed via quadric decision surfaces. For $l > 2$ the decision surfaces are *hyperquadrics*. Figure 2.7a shows the decision curve corresponding to $P(\omega_1) = P(\omega_2), \boldsymbol{\mu}_1 = [0, 0]^T$ and $\boldsymbol{\mu}_2 = [4, 0]^T$. The covariance matrices for the two classes are

$$\Sigma_1 = \begin{bmatrix} 0.3 & 0.0 \\ 0.0 & 0.35 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 1.2 & 0.0 \\ 0.0 & 1.85 \end{bmatrix}$$

For the case of Figure 2.7b the classes are also equiprobable with $\boldsymbol{\mu}_1 = [0, 0]^T$, $\boldsymbol{\mu}_2 = [3.2, 0]^T$ and covariance matrices

$$\Sigma_1 = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.75 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 0.75 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}$$

Figure 2.8 shows the two pdfs for the case of Figure 2.7a. The red color is used for class ω_1 and indicates the points where $p(\boldsymbol{x}|\omega_1) > p(\boldsymbol{x}|\omega_2)$. The gray color is similarly used for class ω_2. It is readily observed that the decision curve is an ellipse, as shown in Figure 2.7a. The setup corresponding to Figure 2.7b is shown in Figure 2.9. In this case, the decision curve is a hyperbola.

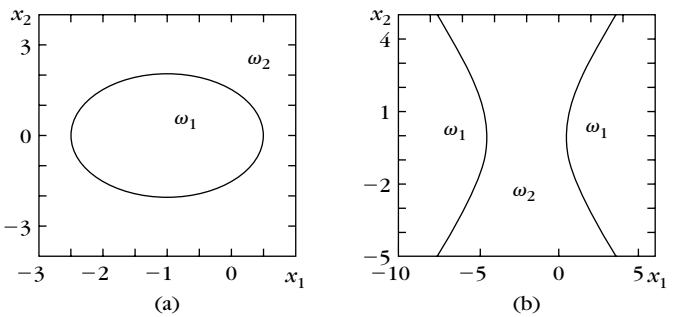

FIGURE 2.7

Examples of quadric decision curves. Playing with the covariance matrices of the Gaussian functions, different decision curves result, that is, ellipsoids, parabolas, hyperbolas, pairs of lines.

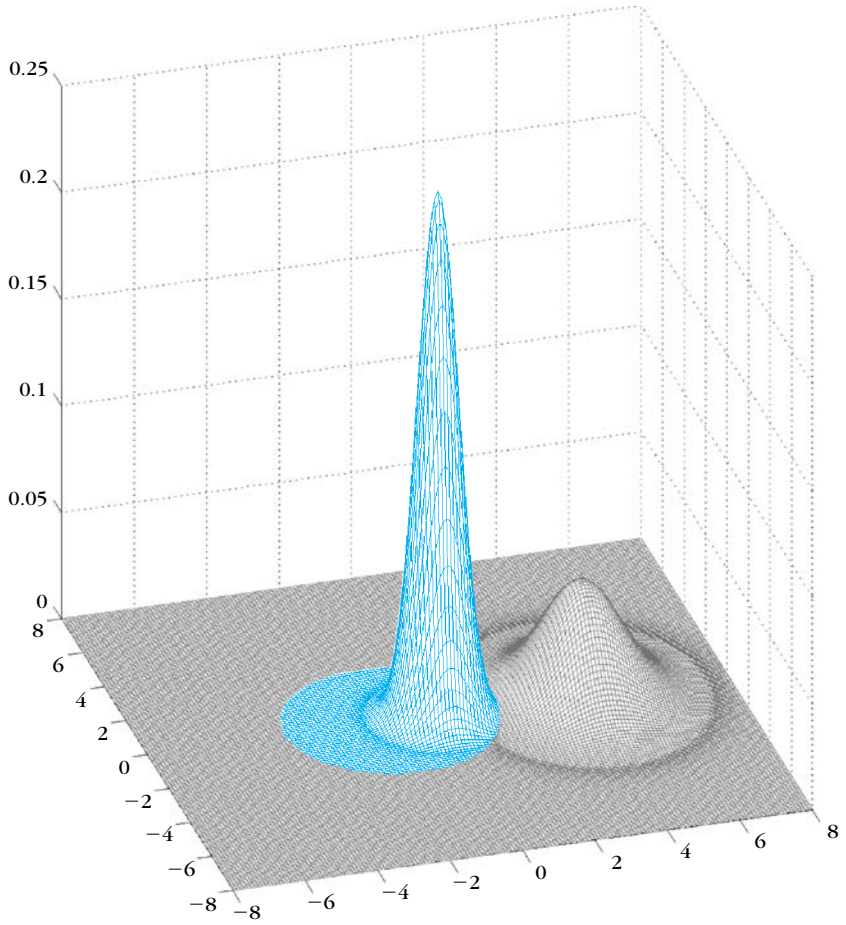

FIGURE 2.8

An example of the pdfs of two equiprobable classes in the two-dimensional space. The feature vectors in both classes are normally distributed with different covariance matrices. In this case, the decision curve is an ellipse and it is shown in Figure 2.7a. The coloring indicates the areas where the value of the respective pdf is larger.

Decision Hyperplanes

The only quadratic contribution in (2.35) comes from the term $\boldsymbol{x}^T\Sigma_i^{-1}\boldsymbol{x}$. If we now assume that the covariance matrix is the same in all classes, that is, $\Sigma_i = \Sigma$, the quadratic term will be the same in all discriminant functions. Hence, it does not enter into the comparisons for computing the maximum, and it cancels out in the decision surface equations. The same is true for the constants c_i. Thus, they can be omitted and we may redefine $g_i(\boldsymbol{x})$ as

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} \tag{2.37}$$

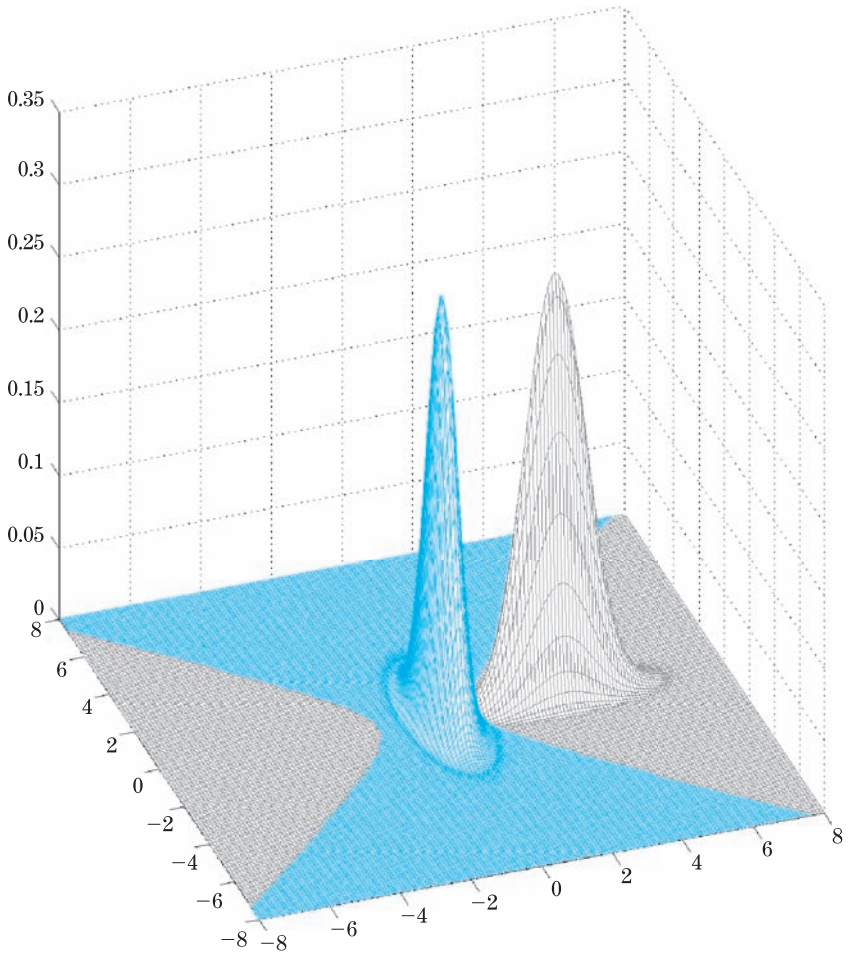

FIGURE 2.9

An example of the pdfs of two equiprobable classes in the two-dimensional space. The feature vectors in both classes are normally distributed with different covariance matrices. In this case, the decision curve is a hyperbola and it is shown in Figure 2.7b.

where

$$\boldsymbol{w}_i = \Sigma^{-1}\boldsymbol{\mu}_i \tag{2.38}$$

and

$$w_{i0} = \ln P(\omega_i) - \frac{1}{2}\boldsymbol{\mu}_i^T\Sigma^{-1}\boldsymbol{\mu}_i \tag{2.39}$$

Hence $g_i(\boldsymbol{x})$ is a *linear function* of $\boldsymbol{x}$ and the respective decision surfaces are *hyperplanes*. Let us investigate this a bit more.

- *Diagonal covariance matrix with equal elements*: Assume that the individual features, constituting the feature vector, are *mutually uncorrelated and of the same variance* ($E[(x_i - \mu_i)(x_j - \mu_j)] = \sigma^2\delta_{ij}$). Then, as discussed in Appendix A, $\Sigma = \sigma^2 I$, where I is the l-dimensional identity matrix, and (2.37) becomes

$$g_i(\boldsymbol{x}) = \frac{1}{\sigma^2}\boldsymbol{\mu}_i^T\boldsymbol{x} + w_{i0} \tag{2.40}$$

Thus, the corresponding decision hyperplanes can now be written as (verify it)

$$g_{ij}(\boldsymbol{x}) \equiv g_i(\boldsymbol{x}) - g_j(\boldsymbol{x}) = \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}_0) = 0 \tag{2.41}$$

where

$$\boldsymbol{w} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \tag{2.42}$$

and

$$\boldsymbol{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j) - \sigma^2 \ln\left(\frac{P(\omega_i)}{P(\omega_j)}\right)\frac{\boldsymbol{\mu}_i - \boldsymbol{\mu}_j}{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2} \tag{2.43}$$

where $\|\boldsymbol{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_l^2}$ denotes the Euclidean norm of $\boldsymbol{x}$. Thus, the decision surface is a *hyperplane* passing through the point $\boldsymbol{x}_0$. Obviously, if $P(\omega_i) = P(\omega_j)$, then $\boldsymbol{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j)$, and the hyperplane passes through the average of $\boldsymbol{\mu}_i, \boldsymbol{\mu}_j$, that is, the middle point of the segment joining the mean values. On the other hand, if $P(\omega_j) > P(\omega_i)$ ($P(\omega_i) > P(\omega_j)$) the hyperplane is located closer to $\boldsymbol{\mu}_i(\boldsymbol{\mu}_j)$. In other words, the area of the region where we decide in favor of the more probable of the two classes is increased.

The geometry is illustrated in Figure 2.10 for the two-dimensional case and for two cases, that is, $P(\omega_j) = P(\omega_i)$ (black line) and $P(\omega_j) > P(\omega_i)$ (red line). We observe that for both cases the decision hyperplane (straight line) is orthogonal to $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j$. Indeed, for any point $\boldsymbol{x}$ lying on the decision hyperplane, the vector $\boldsymbol{x} - \boldsymbol{x}_0$ also lies on the hyperplane and

$$g_{ij}(\boldsymbol{x}) = 0 \Rightarrow \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}_0) = (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T(\boldsymbol{x} - \boldsymbol{x}_0) = 0$$

That is, $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j$ is orthogonal to the decision hyperplane. Furthermore, if σ^2 is small with respect to $\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|$, the location of the hyperplane is rather insensitive to the values of $P(\omega_i), P(\omega_j)$. This is expected, because small variance indicates that the random vectors are clustered within a small radius around their mean values. Thus a small shift of the decision hyperplane has a small effect on the result.

Figure 2.11 illustrates this. For each class, the circles around the means indicate regions where samples have a high probability, say 98%,

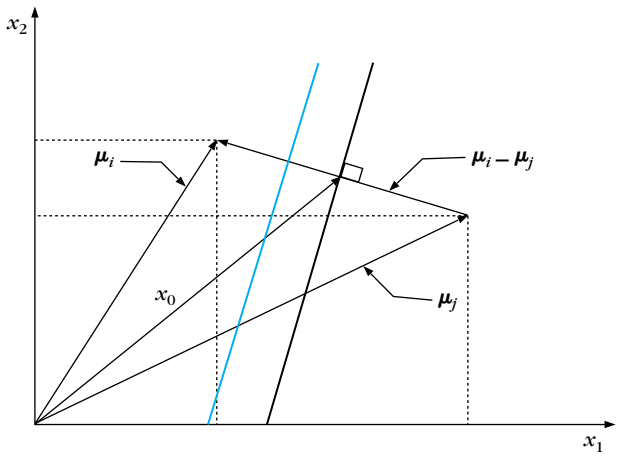

FIGURE 2.10

Decision lines for normally distributed vectors with $\Sigma = \sigma^2 I$. The black line corresponds to the case of $P(\omega_j) = P(\omega_i)$ and it passes through the middle point of the line segment joining the mean values of the two classes. The red line corresponds to the case of $P(\omega_j) > P(\omega_i)$ and it is closer to μ_i, leaving more "room" to the more probable of the two classes. If we had assumed $P(\omega_j) < P(\omega_i)$, the decision line would have moved closer to μ_j.

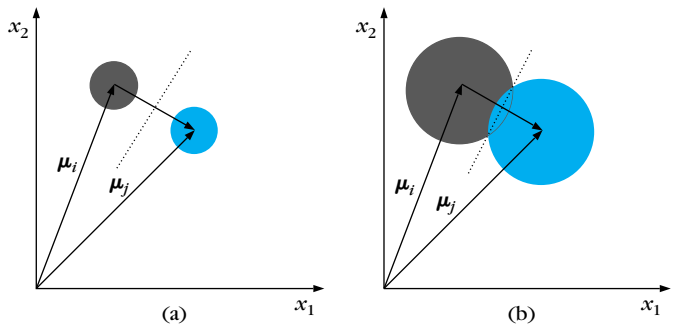

FIGURE 2.11

Decision line (a) for compact and (b) for noncompact classes. When classes are compact around their mean values, the location of the hyperplane is rather insensitive to the values of $P(\omega_1)$ and $P(\omega_2)$. This is not the case for noncompact classes, where a small movement of the hyperplane to the right or to the left may be more critical.

of being found. The case of Figure 2.11a corresponds to small variance, and that of Figure 2.11b to large variance. No doubt the location of the decision hyperplane in Figure 2.11b is much more critical than that in Figure 2.11a.

- *Nondiagonal covariance matrix*: Following algebraic arguments similar to those used before, we end up with hyperplanes described by

$$g_{ij}(\boldsymbol{x}) = \boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{x}_0) = 0 \tag{2.44}$$

where

$$\boldsymbol{w} = \Sigma^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) \tag{2.45}$$

and

$$\boldsymbol{x}_0 = \frac{1}{2}(\boldsymbol{\mu}_i + \boldsymbol{\mu}_j) - \ln\left(\frac{P(\omega_i)}{P(\omega_j)}\right)\frac{\boldsymbol{\mu}_i - \boldsymbol{\mu}_j}{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2_{\Sigma^{-1}}} \tag{2.46}$$

where $\|\boldsymbol{x}\|_{\Sigma^{-1}} \equiv (\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x})^{1/2}$ denotes the so-called Σ^{-1} norm of $\boldsymbol{x}$. The comments made before for the case of the diagonal covariance matrix are still valid, with one exception. *The decision hyperplane is no longer orthogonal to the vector $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j$ but to its linear transformation $\Sigma^{-1}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)$.*

Figure 2.12 shows two Gaussian pdfs with equal covariance matrices, describing the data distribution of two equiprobable classes. In both classes, the data are distributed around their mean values in *exactly* the same way and the optimal decision curve is a straight line.

Minimum Distance Classifiers

We will now view the task from a slightly different angle. Assuming equiprobable classes with the same covariance matrix, $g_i(\boldsymbol{x})$ in (2.34) is simplified to

$$g_i(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^T\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i) \tag{2.47}$$

where constants have been neglected.

- $\Sigma = \sigma^2 I$: In this case maximum $g_i(\boldsymbol{x})$ implies minimum

$$\textit{Euclidean distance:} \quad d_\epsilon = \|\boldsymbol{x} - \boldsymbol{\mu}_i\| \tag{2.48}$$

Thus, feature vectors are assigned to classes according to their Euclidean distance from the respective mean points. Can you verify that this result ties in with the geometry of the hyperplanes discussed before?

Figure 2.13a shows curves of equal distance $d_\epsilon = c$ from the mean points of each class. They are obviously circles of radius c (hyperspheres in the general case).

- Nondiagonal Σ: For this case maximizing $g_i(\boldsymbol{x})$ is equivalent to minimizing the Σ^{-1} norm, known as the

$$\textit{Mahalanobis distance:} \quad d_m = \left((\boldsymbol{x} - \boldsymbol{\mu}_i)^T\Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_i)\right)^{1/2} \tag{2.49}$$

In this case, the constant distance $d_m = c$ curves are ellipses (hyperellipses). Indeed, the covariance matrix is symmetric and, as discussed in Appendix B, it can always be diagonalized by a unitary transform

$$\Sigma = \Phi\Lambda\Phi^T \tag{2.50}$$

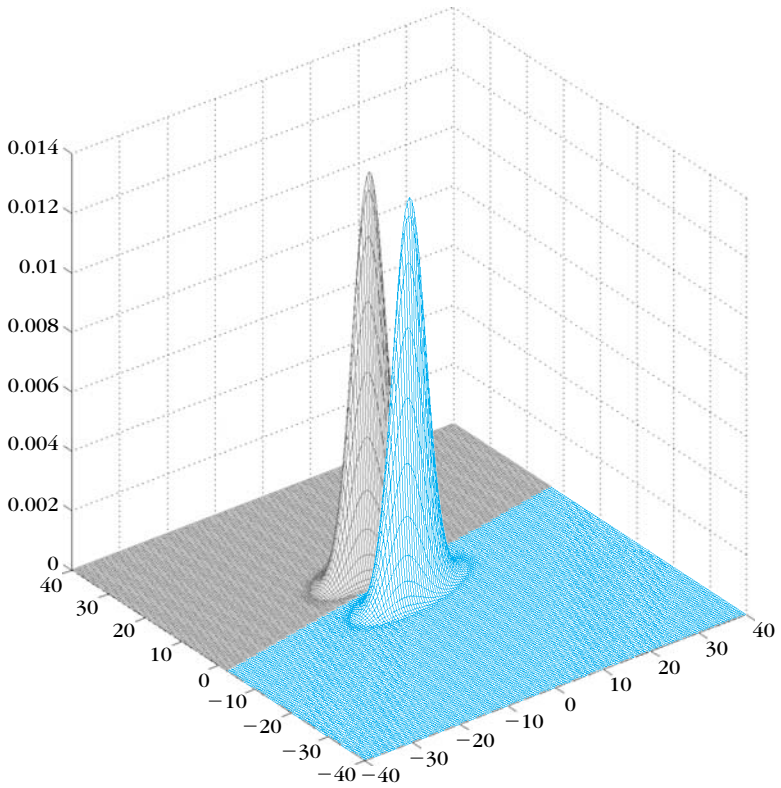

FIGURE 2.12

An example of two Gaussian pdfs with the same covariance matrix in the two-dimensional space. Each one of them is associated with one of two equiprobable classes. In this case, the decision curve is a straight line.

where $\Phi^T = \Phi^{-1}$ and Λ is the diagonal matrix whose elements are the eigenvalues of Σ. Φ has as its columns the corresponding (orthonormal) eigenvectors of Σ

$$\Phi = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_l] \tag{2.51}$$

Combining (2.49) and (2.50), we obtain

$$(\boldsymbol{x} - \boldsymbol{\mu}_i)^T \Phi \Lambda^{-1} \Phi^T (\boldsymbol{x} - \boldsymbol{\mu}_i) = c^2 \tag{2.52}$$

Define $\boldsymbol{x}' = \Phi^T \boldsymbol{x}$. The coordinates of $\boldsymbol{x}'$ are equal to $\boldsymbol{v}_k^T \boldsymbol{x}, k = 1, 2, \ldots, l$, that is, the projections of $\boldsymbol{x}$ onto the eigenvectors. In other words, they are the coordinates of $\boldsymbol{x}$ with respect to a new coordinate system whose axes are determined by $\boldsymbol{v}_k, k = 1, 2, \ldots, l$. Equation (2.52) can now be written as

$$\frac{(x_1' - \mu_{i1}')^2}{\lambda_1} + \ldots + \frac{(x_l' - \mu_{il}')^2}{\lambda_l} = c^2 \tag{2.53}$$

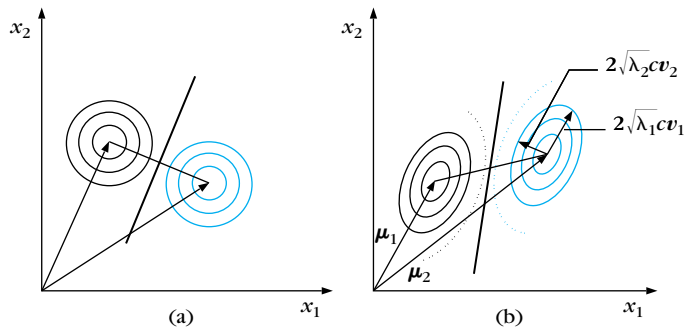

FIGURE 2.13

Curves of (a) equal Euclidean distance and (b) equal Mahalanobis distance from the mean points of each class. In the two-dimensional space, they are circles in the case of Euclidean distance and ellipses in the case of Mahalanobis distance. Observe that in the latter case the decision line is no longer orthogonal to the line segment joining the mean values. It turns according to the shape of the ellipses.

This is the equation of a hyperellipsoid in the new coordinate system. Figure 2.13b shows the $l = 2$ case. The center of mass of the ellipse is at $\boldsymbol{\mu}_i$, and the principal axes are aligned with the corresponding eigenvectors and have lengths $2\sqrt{\lambda_k}c$, respectively. Thus, *all points having the same Mahalanobis distance from a specific point are located on an ellipse*.

Example 2.2

In a two-class, two-dimensional classification task, the feature vectors are generated by two normal distributions sharing the same covariance matrix

$$\Sigma = \begin{bmatrix} 1.1 & 0.3 \\ 0.3 & 1.9 \end{bmatrix}$$

and the mean vectors are $\boldsymbol{\mu}_1 = [0, 0]^T$, $\boldsymbol{\mu}_2 = [3, 3]^T$, respectively.

(a) Classify the vector $[1.0, 2.2]^T$ according to the Bayesian classifier.

It suffices to compute the Mahalanobis distance of $[1.0, 2.2]^T$ from the two mean vectors. Thus,

$$d_m^2(\boldsymbol{\mu}_1, \boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_1)$$

$$= [1.0, 2.2] \begin{bmatrix} 0.95 & -0.15 \\ -0.15 & 0.55 \end{bmatrix} \begin{bmatrix} 1.0 \\ 2.2 \end{bmatrix} = 2.952$$

Similarly,

$$d_m^2(\boldsymbol{\mu}_2, \boldsymbol{x}) = [-2.0, -0.8] \begin{bmatrix} 0.95 & -0.15 \\ -0.15 & 0.55 \end{bmatrix} \begin{bmatrix} -2.0 \\ -0.8 \end{bmatrix} = 3.672 \tag{2.54}$$

Thus, the vector is assigned to the class with mean vector $[0, 0]^T$. *Notice that the given vector* $[1.0, 2.2]^T$ *is closer to* $[3, 3]^T$ *with respect to the Euclidean distance.*

(b) Compute the principal axes of the ellipse centered at $[0, 0]^T$ that corresponds to a constant Mahalanobis distance $d_m = \sqrt{2.952}$ from the center.
To this end, we first calculate the eigenvalues of Σ.

$$\det\left(\begin{bmatrix} 1.1-\lambda & 0.3 \\ 0.3 & 1.9-\lambda \end{bmatrix}\right) = \lambda^2 - 3\lambda + 2 = 0$$

or $\lambda_1 = 1$ and $\lambda_2 = 2$. To compute the eigenvectors we substitute these values into the equation

$$(\Sigma - \lambda I)\boldsymbol{v} = \mathbf{0}$$

and we obtain the unit norm eigenvectors

$$\boldsymbol{v}_1 = \begin{bmatrix} \frac{3}{\sqrt{10}} \\ -\frac{1}{\sqrt{10}} \end{bmatrix}, \quad \boldsymbol{v}_2 = \begin{bmatrix} \frac{1}{\sqrt{10}} \\ \frac{3}{\sqrt{10}} \end{bmatrix}$$

It can easily be seen that they are mutually orthogonal. The principal axes of the ellipse are parallel to $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ and have lengths 3.436 and 4.859, respectively.

Remarks

- In practice, it is quite common to assume that the data in each class are adequately described by a Gaussian distribution. As a consequence, the associated Bayesian classifier is either linear or quadratic in nature, depending on the adopted assumptions concerning the covariance matrices. That is, if they are all equal or different. In statistics, this approach to the classification task is known as *linear discriminant analysis* (LDA) or *quadratic discriminant analysis* (QDA), respectively. *Maximum likelihood* is usually the method mobilized for the estimation of the unknown parameters that define the mean values and the covariance matrices (see Section 2.5 and Problem 2.19).
- A major problem associated with LDA and even more with QDA is the large number of the unknown parameters that have to be estimated in the case of high-dimensional spaces. For example, there are l parameters in each of the mean vectors and approximately $l^2/2$ in each (symmetric) covariance matrix. Besides the high demand for computational resources, obtaining good estimates of a large number of parameters dictates a large number of training points, N. This is a major issue that also embraces the design of other types of classifiers, for most of the cases, and we will come to it in greater detail in Chapter 5. In an effort to reduce the number of parameters to be estimated, a number of approximate techniques have been suggested over the years, including [Kimu 87, Hoff 96, Frie 89, Liu 04]. Linear discrimination will be approached from a different perspective in Section 5.8.
- LDA and QDA exhibit good performance in a large set of diverse applications and are considered to be among the most popular classifiers. No doubt, it is hard to accept that in all these cases the Gaussian assumption provides a reasonable modeling for the data statistics. The secret of the success seems

to lie in the fact that linear or quadratic decision surfaces offer a reasonably good partition of the space, from the classification point of view. Moreover, as pointed out in [Hast 01], the estimates associated with Gaussian models have some good statistical properties (i.e., bias variance trade-off, Section 3.5.3) compared to other techniques.

2.5 ESTIMATION OF UNKNOWN PROBABILITY DENSITY FUNCTIONS

So far, we have assumed that the probability density functions are known. However, this is not the most common case. In many problems, the underlying pdf has to be estimated from the available data. There are various ways to approach the problem. Sometimes we may know the type of the pdf (e.g., Gaussian, Rayleigh), but we do not know certain parameters, such as the mean values or the variances. In contrast, in other cases we may not have information about the type of the pdf but we may know certain statistical parameters, such as the mean value and the variance. Depending on the available information, different approaches can be adopted. This will be our focus in the next subsections.

2.5.1 Maximum Likelihood Parameter Estimation

Let us consider an M-class problem with feature vectors distributed according to $p(\boldsymbol{x}|\omega_i), i = 1, 2, \ldots, M$. We assume that these likelihood functions are given in a *parametric* form and that the corresponding parameters form the vectors $\boldsymbol{\theta}_i$ which are unknown. To show the dependence on $\boldsymbol{\theta}_i$ we write $p(\boldsymbol{x}|\omega_i; \boldsymbol{\theta}_i)$. Our goal is to estimate the unknown parameters using a set of known feature vectors in each class. If we further assume that data from one class do not affect the parameter estimation of the others, we can formulate the problem independent of classes and simplify our notation. At the end, one has to solve one such problem for each class independently.

Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ be random samples drawn from pdf $p(\boldsymbol{x}; \boldsymbol{\theta})$. We form the joint pdf $p(X; \boldsymbol{\theta})$, where $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is the set of the samples. Assuming *statistical independence* between the different samples, we have

$$p(X; \boldsymbol{\theta}) \equiv p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N; \boldsymbol{\theta}) = \prod_{k=1}^{N} p(\boldsymbol{x}_k; \boldsymbol{\theta}) \tag{2.55}$$

This is a function of $\boldsymbol{\theta}$, and it is also known as the likelihood function of $\boldsymbol{\theta}$ with respect to X. *The maximum likelihood (ML) method estimates $\boldsymbol{\theta}$ so that the likelihood function takes its maximum value*, that is,

$$\hat{\boldsymbol{\theta}}_{ML} = arg \max_{\boldsymbol{\theta}} \prod_{k=1}^{N} p(\boldsymbol{x}_k; \boldsymbol{\theta}) \tag{2.56}$$

A necessary condition that $\hat{\boldsymbol{\theta}}_{ML}$ must satisfy in order to be a maximum is the gradient of the likelihood function with respect to $\boldsymbol{\theta}$ to be zero, that is

$$\frac{\partial \prod_{k=1}^{N} p(\boldsymbol{x}_k; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbf{0} \tag{2.57}$$

Because of the monotonicity of the logarithmic function, we define the *log-likelihood function* as

$$L(\boldsymbol{\theta}) \equiv \ln \prod_{k=1}^{N} p(\boldsymbol{x}_k; \boldsymbol{\theta}) \tag{2.58}$$

and (2.57) is equivalent to

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{N} \frac{\partial \ln p(\boldsymbol{x}_k; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{N} \frac{1}{p(\boldsymbol{x}_k; \boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_k; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbf{0} \tag{2.59}$$

Figure 2.14 illustrates the method for the single unknown parameter case. The ML estimate corresponds to the peak of the log-likelihood function.

Maximum likelihood estimation has some very desirable properties. If $\boldsymbol{\theta}_0$ is the true value of the unknown parameter in $p(\boldsymbol{x}; \boldsymbol{\theta})$, it can be shown that under generally valid conditions the following are true [Papo 91].

- The ML estimate is *asymptotically unbiased*, which by definition means that

$$\lim_{N \to \infty} E[\hat{\boldsymbol{\theta}}_{ML}] = \boldsymbol{\theta}_0 \tag{2.60}$$

  Alternatively, we say that the estimate *converges in the mean* to the true value. The meaning of this is as follows. The estimate $\hat{\boldsymbol{\theta}}_{ML}$ is itself a random vector, because for different sample sets X different estimates will result. An estimate is called *unbiased* if its mean is the true value of the unknown parameter. In the ML case this is true only asymptotically ($N \to \infty$).

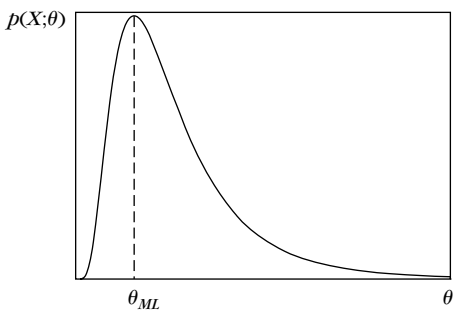

FIGURE 2.14

The maximum likelihood estimator θ_{ML} corresponds to the peak of $p(X; \theta)$.

- The ML estimate is *asymptotically consistent*, that is, it satisfies

$$\lim_{N\to\infty} \text{prob}\{\|\hat{\boldsymbol{\theta}}_{ML} - \boldsymbol{\theta}_0\| \leq \epsilon\} = 1 \tag{2.61}$$

  where ϵ is arbitrarily small. Alternatively, we say that the estimate converges *in probability*. In other words, for large N it is highly probable that the resulting estimate will be arbitrarily close to the true value. A stronger condition for consistency is also true:

$$\lim_{N\to\infty} E[\|\hat{\boldsymbol{\theta}}_{ML} - \boldsymbol{\theta}_0\|^2] = 0 \tag{2.62}$$

  In such cases we say that the estimate converges in the *mean square*. In words, for large N, the variance of the ML estimates tends to zero.

  Consistency is very important for an estimator, because it may be unbiased, but the resulting estimates exhibit large variations around the mean. In such cases we have little confidence in the result obtained from a single set X.

- The ML estimate is asymptotically *efficient*; that is, it achieves the Cramer–Rao lower bound (Appendix A). This is the lowest value of variance, which *any* estimate can achieve.

- The pdf of the ML estimate as $N \to \infty$ approaches the Gaussian distribution with mean $\boldsymbol{\theta}_0$ [Cram 46]. This property is an offspring of (a) the central limit theorem (Appendix A) and (b) the fact that the ML estimate is related to the *sum* of random variables, that is, $\partial \ln(p(\boldsymbol{x}_k; \boldsymbol{\theta}))/\partial\boldsymbol{\theta}$ (Problem 2.16).

In summary, the ML estimator is unbiased, is normally distributed, and has the minimum possible variance. However, all these nice properties are valid *only* for large values of N.

Example 2.3

Assume that N data points, $x_1, x_2, \ldots, x_N$, have been generated by a one-dimensional Gaussian pdf of known mean, μ, but of unknown variance. Derive the ML estimate of the variance.

The log-likelihood function for this case is given by

$$L(\sigma^2) = \ln \prod_{k=1}^{N} p(x_k; \sigma^2) = \ln \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi}\sqrt{\sigma^2}} \exp\left(-\frac{(x_k - \mu)^2}{2\sigma^2}\right)$$

or

$$L(\sigma^2) = -\frac{N}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{k=1}^{N}(x_k - \mu)^2$$

Taking the derivative of the above with respect to σ^2 and equating to zero, we obtain

$$-\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{k=1}^{N}(x_k - \mu)^2 = 0$$

and finally the ML estimate of σ^2 results as the solution of the above,

$$\hat{\sigma}^2_{ML} = \frac{1}{N}\sum_{k=1}^{N}(x_k - \mu)^2 \tag{2.63}$$

Observe that, for finite N, $\hat{\sigma}^2_{ML}$ in Eq. (2.63) is a biased estimate of the variance. Indeed,

$$E[\hat{\sigma}^2_{ML}] = \frac{1}{N}\sum_{k=1}^{N}E[(x_k - \mu)^2] = \frac{N-1}{N}\sigma^2$$

where σ^2 is the true variance of the Gaussian pdf. However, for large values of N, we have

$$E[\hat{\sigma}^2_{ML}] = (1 - \frac{1}{N})\sigma^2 \approx \sigma^2$$

which is in line with the theoretical result of asymptotic consistency of the ML estimator.

Example 2.4

Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ be vectors stemmed from a normal distribution with known covariance matrix and unknown mean, that is,

$$p(\boldsymbol{x}_k; \boldsymbol{\mu}) = \frac{1}{(2\pi)^{l/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{x}_k - \boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}_k - \boldsymbol{\mu})\right)$$

Obtain the ML estimate of the unknown mean vector.

For N available samples we have

$$L(\boldsymbol{\mu}) \equiv \ln\prod_{k=1}^{N}p(\boldsymbol{x}_k; \boldsymbol{\mu}) = -\frac{N}{2}\ln((2\pi)^l|\Sigma|) - \frac{1}{2}\sum_{k=1}^{N}(\boldsymbol{x}_k - \boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}_k - \boldsymbol{\mu}) \tag{2.64}$$

Taking the gradient with respect to $\boldsymbol{\mu}$, we obtain

$$\frac{\partial L(\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} \equiv \begin{bmatrix} \frac{\partial L}{\partial \mu_1} \\ \frac{\partial L}{\partial \mu_2} \\ \vdots \\ \frac{\partial L}{\partial \mu_l} \end{bmatrix} = \sum_{k=1}^{N}\Sigma^{-1}(\boldsymbol{x}_k - \boldsymbol{\mu}) = 0 \tag{2.65}$$

or

$$\hat{\boldsymbol{\mu}}_{ML} = \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{x}_k \tag{2.66}$$

That is, the ML estimate of the mean, for Gaussian densities, is the sample mean. However, this very "natural approximation" is not necessarily ML optimal for non-Gaussian density functions.

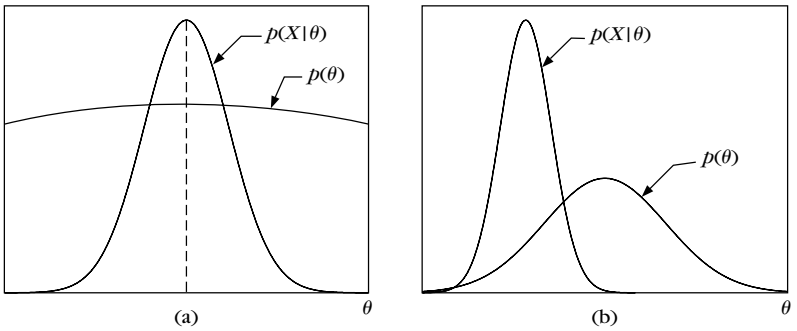

FIGURE 2.15

ML and MAP estimates of θ will be approximately the same in (a) and different in (b).

2.5.2 Maximum *a Posteriori* Probability Estimation

For the derivation of the maximum likelihood estimate, we considered $\boldsymbol{\theta}$ as an unknown parameter. In this subsection we will consider it as a random vector, and we will estimate its value on the condition that samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ have occurred. Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. Our starting point is $p(\boldsymbol{\theta}|X)$. From our familiar Bayes theorem we have

$$p(\boldsymbol{\theta})p(X|\boldsymbol{\theta}) = p(X)p(\boldsymbol{\theta}|X) \tag{2.67}$$

or

$$p(\boldsymbol{\theta}|X) = \frac{p(\boldsymbol{\theta})p(X|\boldsymbol{\theta})}{p(X)} \tag{2.68}$$

The *maximum a posteriori probability* (MAP) estimate $\hat{\boldsymbol{\theta}}_{MAP}$ is defined at the point where $p(\boldsymbol{\theta}|X)$ becomes maximum,

$$\hat{\boldsymbol{\theta}}_{MAP} : \frac{\partial}{\partial \boldsymbol{\theta}} p(\boldsymbol{\theta}|X) = 0 \quad \text{or} \quad \frac{\partial}{\partial \boldsymbol{\theta}} (p(\boldsymbol{\theta})p(X|\boldsymbol{\theta})) = 0 \tag{2.69}$$

Note that $p(X)$ is not involved since it is independent of $\boldsymbol{\theta}$. The difference between the ML and the MAP estimates lies in the involvement of $p(\boldsymbol{\theta})$ in the latter case. If we assume that this obeys the uniform distribution, that is, is constant for all $\boldsymbol{\theta}$, both estimates yield identical results. This is also approximately true if $p(\boldsymbol{\theta})$ exhibits small variation. However, in the general case, the two methods yield different results. Figures 2.15a and 2.15b illustrate the two cases.

Example 2.5

Let us assume that in Example 2.4 the unknown mean vector $\boldsymbol{\mu}$ is known to be normally distributed as

$$p(\boldsymbol{\mu}) = \frac{1}{(2\pi)^{l/2}\sigma_\mu^l} \exp\left(-\frac{1}{2}\frac{\|\boldsymbol{\mu} - \boldsymbol{\mu}_0\|^2}{\sigma_\mu^2}\right)$$

The MAP estimate is given by the solution of

$$\frac{\partial}{\partial \boldsymbol{\mu}} \ln\left(\prod_{k=1}^{N} p(\boldsymbol{x}_k|\boldsymbol{\mu})p(\boldsymbol{\mu})\right) = \mathbf{0}$$

or, for $\Sigma = \sigma^2 I$,

$$\sum_{k=1}^{N} \frac{1}{\sigma^2}(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}}) - \frac{1}{\sigma_\mu^2}(\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}_0) = \mathbf{0} \Rightarrow$$

$$\hat{\boldsymbol{\mu}}_{MAP} = \frac{\boldsymbol{\mu}_0 + \frac{\sigma_\mu^2}{\sigma^2}\sum_{k=1}^{N} \boldsymbol{x}_k}{1 + \frac{\sigma_\mu^2}{\sigma^2}N}$$

We observe that if $\frac{\sigma_\mu^2}{\sigma^2} >> 1$, that is, the variance σ_μ^2 is very large and the corresponding Gaussian is very wide with little variation over the range of interest, then

$$\hat{\boldsymbol{\mu}}_{MAP} \approx \hat{\boldsymbol{\mu}}_{ML} = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_k$$

Furthermore, observe that this is also the case for $N \to \infty$, regardless of the values of the variances. Thus, the MAP estimate tends asymptotically to the ML one. This is a more general result. For large values of N, the likelihood term $\prod_{k=1}^{N} p(\boldsymbol{x}_k|\boldsymbol{\mu})$ becomes sharply peaked around the true value (of the unknown parameter) and is the term that basically determines where the maximum occurs. This can be better understood by mobilizing the properties of the ML estimate given before.

2.5.3 Bayesian Inference

Both methods considered in the preceding subsections compute a specific estimate of the unknown parameter vector $\boldsymbol{\theta}$. In the current method, a different path is adopted. Given the set X of the N training vectors and the *a priori* information about the pdf $p(\boldsymbol{\theta})$, the goal is to compute the conditional pdf $p(\boldsymbol{x}|X)$. After all, this is what we actually need to know. To this end, and making use of known identities from our statistics basics, we have the following set of relations at our disposal:

$$p(\boldsymbol{x}|X) = \int p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|X)\, d\boldsymbol{\theta} \tag{2.70}$$

with

$$p(\boldsymbol{\theta}|X) = \frac{p(X|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(X)} = \frac{p(X|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(X|\boldsymbol{\theta})p(\boldsymbol{\theta})\, d\boldsymbol{\theta}} \tag{2.71}$$

$$p(X|\boldsymbol{\theta}) = \prod_{k=1}^{N} p(\boldsymbol{x}_k|\boldsymbol{\theta}) \tag{2.72}$$

The conditional density $p(\boldsymbol{\theta}|X)$ is also known as the a posteriori pdf estimate, since it is updated "knowledge" about the statistical properties of $\boldsymbol{\theta}$, after having observed the data set X. Once more, Eq. (2.72) presupposes statistical independence among the training samples.

In general, the computation of $p(\boldsymbol{x}|X)$ requires the integration in the right-hand side of (2.70). However, analytical solutions are feasible only for very special forms of the involved functions. For most of the cases, analytical solutions for (2.70), as well as for the denominator in (2.71), are not possible, and one has to resort to numerical approximations. To this end, a large research effort has been invested in developing efficient techniques for the numerical computation of such statistical quantities. Although a detailed presentation of such approximation schemes is beyond the scope of this book, we will attempt to highlight the main philosophy behind these techniques in relation to our own problem.

Looking more carefully at (2.70) and assuming that $p(\boldsymbol{\theta}|X)$ is known, then $p(\boldsymbol{x}|X)$ is nothing but the average of $p(\boldsymbol{x}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, that is,

$$p(\boldsymbol{x}|X) = E_{\boldsymbol{\theta}}\left[p(\boldsymbol{x}|\boldsymbol{\theta})\right]$$

If we assume that a large enough number of samples $\boldsymbol{\theta}_i,\ i = 1, 2 \ldots, L$, of the random vector $\boldsymbol{\theta}$ are available, one can compute the corresponding values $p(\boldsymbol{x}|\boldsymbol{\theta}_i)$ and then approximate the expectation as the mean value

$$p(\boldsymbol{x}|X) \approx \frac{1}{L}\sum_{i=1}^{L} p(\boldsymbol{x}|\boldsymbol{\theta}_i)$$

The problem now becomes that of generating a set of samples, $\boldsymbol{\theta}_i,\ i = 1, 2 \ldots, L$. For example, if $p(\boldsymbol{\theta}|X)$ were a Gaussian pdf, one could use a Gaussian pseudorandom generator to generate the L samples. The difficulty in our case is that, in general, the exact form of $p(\boldsymbol{\theta}|X)$ is not known, and its computation presupposes the numerical integration of the normalizing constant in the denominator of (2.71). This difficulty is bypassed by a set of methods known as *Markov chain Monte Carlo* (MCMC) techniques. The main rationale behind these techniques is that one can generate samples from (2.71) in a sequential manner that *asymptotically* follow the distribution $p(\boldsymbol{\theta}|X)$, even without knowing the normalizing factor. The Gibbs sampler and the Metropolis-Hastings algorithms are two of the most popular schemes of this type. For more details on such techniques, the interested reader may consult, for example, [Bish 06].

Further insight into the Bayesian methods can be gained by focusing on the Gaussian one-dimensional case.

Example 2.6

Let $p(x|\mu)$ be a univariate Gaussian $\mathcal{N}(\mu, \sigma^2)$ with unknown parameter the mean, which is also assumed to follow a Gaussian $\mathcal{N}(\mu_0, \sigma_0^2)$. From the theory exposed before we have

$$p(\mu|X) = \frac{p(X|\mu)p(\mu)}{p(X)} = \frac{1}{\alpha}\prod_{k=1}^{N} p(x_k|\mu)p(\mu)$$

where for a given training data set, X, $p(X)$ is a constant denoted as α, or

$$p(\mu|X) = \frac{1}{\alpha}\prod_{k=1}^{N}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x_k-\mu)^2}{\sigma^2}\right)\frac{1}{\sqrt{2\pi}\sigma_0}\exp\left(-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right)$$

It is a matter of some algebra (Problem 2.25) to show that, given a number of samples, N, $p(\mu|X)$ turns out to be also Gaussian, that is,

$$p(\mu|X) = \frac{1}{\sqrt{2\pi}\sigma_N}\exp\left(-\frac{(\mu-\mu_N)^2}{2\sigma_N^2}\right) \tag{2.73}$$

with mean value

$$\mu_N = \frac{N\sigma_0^2\bar{x}_N + \sigma^2\mu_0}{N\sigma_0^2+\sigma^2} \tag{2.74}$$

and variance

$$\sigma_N^2 = \frac{\sigma^2\sigma_0^2}{N\sigma_0^2+\sigma^2} \tag{2.75}$$

where $\bar{x}_N = \frac{1}{N}\sum_{k=1}^{N} x_k$. Letting N vary from 1 to ∞, we generate a sequence of Gaussians $\mathcal{N}(\mu_N, \sigma_N^2)$, whose mean values move away from μ_0 and tend, in the limit, to the sample mean, which, asymptotically, becomes equal to the true mean value. Furthermore, their variance keeps decreasing at the rate σ^2/N for large N. Hence, for large values of N, $p(\mu|X)$ becomes sharply peaked around the sample mean. Recall that the latter is the ML estimate of the mean value.

Once $p(\mu|X)$ has been computed, it can be shown, by substituting (2.73) into (2.70) (problem 2.25), that

$$p(x|X) = \frac{1}{\sqrt{2\pi(\sigma^2+\sigma_N^2)}}\exp\left(-\frac{1}{2}\frac{(x-\mu_N)^2}{\sigma^2+\sigma_N^2}\right)$$

which is a Gaussian pdf with mean value μ_N and variance $\sigma^2+\sigma_N^2$.

Observe that as N tends to infinity, the unknown mean value of the Gaussian tends to the ML estimate $\bar{x}_N$ (and asymptotically to the true mean) and the variance to the true value σ^2. For finite values of N, the variance is larger than σ^2 to account for our extra uncertainty about x due to the unknown value of the mean μ. Figure 2.16 shows the posterior pdf estimate $p(\mu|X)$ obtained for different sizes of the training data set. Data were generated using a pseudorandom number generator following a Gaussian pdf with mean value equal to $\mu = 2$ and variance $\sigma^2 = 4$. The mean value was assumed to be unknown, and the prior pdf was adopted to be Gaussian with $\mu_0 = 0$ and $\sigma_0^2 = 8$. We observe that as N increases $p(\mu|X)$ gets narrower (in accordance to (2.75)). The respective mean value estimate (Eq. (2.74)) depends on N and $\bar{x}_N$. For small values of N, the ML estimate of the mean, $\bar{x}_N$, can vary a lot, which has a direct effect in moving around the centers of the Gaussians. However, as N increases, $\bar{x}_N$ tends to the true value of the mean $(\mu = 2)$ with a decreasing variance.

It can be shown (Problem 2.27) that the results of this example can be generalized for the case of multivariate Gaussians. More specifically, one can show that Eqs. (2.74) and (2.75) are generalized to the following

$$p(\boldsymbol{\mu}|X) \sim \mathcal{N}(\boldsymbol{\mu}_N, \Sigma_N) \tag{2.76}$$

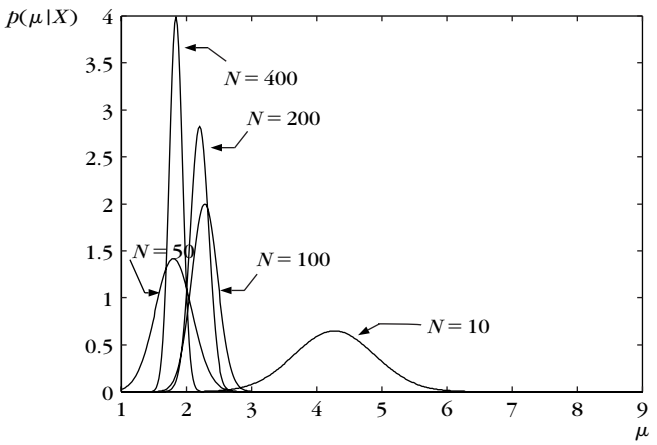

FIGURE 2.16

A sequence of the posterior pdf estimates (Eq. (2.73)), for the case of Example 2.6. As the number of training points increases, the posterior pdf becomes more spiky (the ambiguity decreases) and its center moves toward the true mean value of the data.

where

$$\boldsymbol{\mu}_N = N\Sigma_0[N\Sigma_0 + \Sigma]^{-1}\bar{\boldsymbol{x}}_N + \Sigma[N\Sigma_0 + \Sigma]^{-1}\boldsymbol{\mu}_0 \tag{2.77}$$

and

$$\Sigma_N = \Sigma_0[N\Sigma_0 + \Sigma]^{-1}\Sigma \tag{2.78}$$

and also

$$p(\boldsymbol{x}|X) \sim \mathcal{N}(\boldsymbol{\mu}_N, \Sigma + \Sigma_N) \tag{2.79}$$

Remarks

- If $p(\boldsymbol{\theta}|X)$ in Eq. (2.71) is sharply peaked at a $\hat{\boldsymbol{\theta}}$ and we treat it as a delta function, Eq. (2.70) becomes $p(\boldsymbol{x}|X) \approx p(\boldsymbol{x}|\hat{\boldsymbol{\theta}})$; that is, the parameter estimate is approximately equal to the MAP estimate. This happens, for example, if $p(X|\boldsymbol{\theta})$ is concentrated around a sharp peak and $p(\boldsymbol{\theta})$ is broad enough around this peak. Then the resulting estimate approximates the ML one. The latter was also verified by our previous example. This is a more general property valid for most of the pdfs used in practice, for which the posterior probability of the unknown parameter vector $p(\boldsymbol{\theta}|X)$ tends to a delta function as N tends to $+\infty$. *Thus, all three methods considered so far result, asymptotically, in the same estimate. However, the results are different for small numbers N of training samples.*
- An obvious question concerns the choice of the prior $p(\boldsymbol{\theta})$. In practice, the choice depends on the form of the likelihood function $p(\boldsymbol{x}|\boldsymbol{\theta})$, so that the posterior pdf $p(\boldsymbol{\theta}|X)$ can be of a tractable form. The set of prior distributions

for which the adopted model $p(\boldsymbol{x}|\boldsymbol{\theta})$ is of the same functional form as the posterior distribution $p(\boldsymbol{\theta}|X)$ is known as *conjugate* with respect to the model. Some commonly used forms of the conjugate priors are discussed, for example, in [Bern 94].

- For data sets of limited length, ML and MAP estimators are simpler to use, and they result in a single estimate of the unknown parameters vector, which is the outcome of a maximization procedure. On the other hand, Bayesian methods make use of more information and, provided that this information is reliable, these techniques are expected to give better results, albeit at the expense of higher complexity. Due to the advances in computer technology, Bayesian methods have gained a lot of popularity over the recent years.

2.5.4 Maximum Entropy Estimation

The concept of *entropy* is known from Shannon's information theory. It is a measure of the uncertainty concerning an event and, from another viewpoint, a measure of randomness of the messages (feature vectors in our case) occurring at the output of a system. If $p(\boldsymbol{x})$ is the density function, the associated entropy H is given by

$$H = -\int_{\boldsymbol{x}} p(\boldsymbol{x}) \ln p(\boldsymbol{x})\, d\boldsymbol{x} \tag{2.80}$$

Assume now that $p(\boldsymbol{x})$ is unknown but we know a number of related constraints (mean value, variance, etc.). The *maximum entropy* estimate of the unknown pdf is the one that maximizes the entropy, subject to the given constraints. According to the principle of maximum entropy, stated by Jaynes [Jayn 82], such an estimate corresponds to the distribution that exhibits the highest possible randomness, subject to the available constraints.

Example 2.7

The random variable x is nonzero for $x_1 \leq x \leq x_2$ and zero otherwise. Compute the maximum entropy estimate of its pdf.

We have to maximize (2.80) subject to the constraint

$$\int_{x_1}^{x_2} p(x)\, dx = 1 \tag{2.81}$$

Using Lagrange multipliers (Appendix C), this is equivalent to maximizing

$$H_L = -\int_{x_1}^{x_2} p(x)(\ln p(x) - \lambda)\, dx \tag{2.82}$$

Taking the derivative with respect to $p(x)$, we obtain

$$\frac{\partial H_L}{\partial p(x)} = -\int_{x_1}^{x_2} \{(\ln p(x) - \lambda) + 1\}\, dx \tag{2.83}$$

Equating to zero, we obtain

$$\hat{p}(x) = \exp(\lambda - 1) \tag{2.84}$$

To compute λ, we substitute this into the constraint equation (2.81), and we get $\exp(\lambda - 1) = \frac{1}{x_2 - x_1}$. Thus

$$\hat{p}(x) = \begin{cases} \frac{1}{x_2 - x_1} & \text{if } x_1 \leq x \leq x_2 \\ 0 & \text{otherwise} \end{cases} \tag{2.85}$$

That is, the maximum entropy estimate of the unknown pdf is the uniform distribution. This is within the maximum entropy spirit. Since we have imposed no other constraint but the obvious one, the resulting estimate is the one that maximizes randomness and all points are equally probable. It turns out that if the mean value and the variance are given as the second and third constraints, the resulting maximum entropy estimate of the pdf, for $-\infty < x < +\infty$, is the Gaussian (Problem 2.30).

2.5.5 Mixture Models

An alternative way to model an unknown $p(\boldsymbol{x})$ is via a linear combination of density functions in the form of

$$p(\boldsymbol{x}) = \sum_{j=1}^{J} p(\boldsymbol{x}|j)P_j \tag{2.86}$$

where

$$\sum_{j=1}^{J} P_j = 1, \quad \int_{\boldsymbol{x}} p(\boldsymbol{x}|j)\, d\boldsymbol{x} = 1 \tag{2.87}$$

In other words, it is assumed that J distributions contribute to the formation of $p(\boldsymbol{x})$. Thus, this modeling implicitly assumes that each point $\boldsymbol{x}$ may be "drawn" from any of the J model distributions with probability $P_j, j = 1, 2, \ldots, J$. It can be shown that this modeling can approximate arbitrarily closely any continuous density function for a sufficient number of *mixtures* J and appropriate model parameters. The first step of the procedure involves the choice of the set of density components $p(\boldsymbol{x}|j)$ in parametric form, that is, $p(\boldsymbol{x}|j; \boldsymbol{\theta})$, and then the computation of the unknown parameters, $\boldsymbol{\theta}$ and $P_j, j = 1, 2, \ldots, J$, based on the set of the available training samples $\boldsymbol{x}_k$. There are various ways to achieve this. A typical maximum likelihood formulation, maximizing the likelihood function $\prod_k p(\boldsymbol{x}_k; \boldsymbol{\theta}, P_1, P_2, \ldots, P_J)$ with respect to $\boldsymbol{\theta}$ and the P_j's, is a first thought. The difficulty here arises from the fact that the unknown parameters enter the maximization task in a *nonlinear fashion*; thus, nonlinear optimization iterative techniques have to be adopted (Appendix C). A review of related techniques is given in [Redn 84]. The source of this complication is the lack of information concerning the labels of the available training samples, that is, the specific mixture from which each sample is contributed. This is the issue that makes the current problem different from the ML case treated in Section 2.5.1. There, the class labels were known, and this led to a *separate* ML problem for each

of the classes. In the same way, if the mixture labels were known, we could collect all data from the same mixture and carry out J separate ML tasks. The missing label information makes our current problem a typical task with an *incomplete data set*.

In the sequel, we will focus on the so-called EM algorithm, which has attracted a great deal of interest over the past few years in a wide range of applications involving tasks with incomplete data sets.

The Expectation Maximization (EM) Algorithm

This algorithm is ideally suited for cases in which the available data set is incomplete. Let us first state the problem in more general terms and then apply it to our specific task. Let us denote by $\boldsymbol{y}$ the *complete data* samples, with $\boldsymbol{y} \in Y \subseteq \mathcal{R}^m$, and let the corresponding pdf be $p_{\boldsymbol{y}}(\boldsymbol{y}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is an unknown parameter vector. The samples $\boldsymbol{y}$, however, *cannot be directly observed*. What we observe instead are samples $\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{y}) \in X_{ob} \subseteq \mathcal{R}^l, l < m$. We denote the corresponding pdf $p_{\boldsymbol{x}}(\boldsymbol{x}; \boldsymbol{\theta})$. This is a *many-to-one mapping*. Let $Y(\boldsymbol{x}) \subseteq Y$ be the subset of all the $\boldsymbol{y}$'s corresponding to a specific $\boldsymbol{x}$. Then the pdf of the incomplete data is given by

$$p_{\boldsymbol{x}}(\boldsymbol{x}; \boldsymbol{\theta}) = \int_{Y(\boldsymbol{x})} p_{\boldsymbol{y}}(\boldsymbol{y}; \boldsymbol{\theta})\, d\boldsymbol{y} \tag{2.88}$$

As we already know, the maximum likelihood estimate of $\boldsymbol{\theta}$ is given by

$$\hat{\boldsymbol{\theta}}_{ML}: \sum_k \frac{\partial \ln(p_{\boldsymbol{y}}(\boldsymbol{y}_k; \boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} = \mathbf{0} \tag{2.89}$$

However, the $\boldsymbol{y}$'s are not available. So, the EM algorithm maximizes the *expectation* of the log-likelihood function, *conditioned on the observed samples and the current iteration estimate of* $\boldsymbol{\theta}$. The two steps of the algorithm are:

- *E-step*: At the $(t+1)$th step of the iteration, where $\boldsymbol{\theta}(t)$ is available, compute the expected value of

$$Q(\boldsymbol{\theta}; \boldsymbol{\theta}(t)) \equiv E\left[\sum_k \ln(p_{\boldsymbol{y}}(\boldsymbol{y}_k; \boldsymbol{\theta} | X; \boldsymbol{\theta}(t))\right] \tag{2.90}$$

  This is the so-called *expectation step* of the algorithm.

- *M-step*: Compute the next $(t+1)$th estimate of $\boldsymbol{\theta}$ by maximizing $Q(\boldsymbol{\theta}; \boldsymbol{\theta}(t))$, that is,

$$\boldsymbol{\theta}(t+1): \frac{\partial Q(\boldsymbol{\theta}; \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}} = \mathbf{0} \tag{2.91}$$

  This is the *maximization step*, where, obviously, differentiability has been assumed.

To apply the EM algorithm, we start from an initial estimate $\boldsymbol{\theta}(0)$, and iterations are terminated if $\|\boldsymbol{\theta}(t+1) - \boldsymbol{\theta}(t)\| \leq \epsilon$ for an appropriately chosen vector norm and ϵ.

Remark

- It can be shown that the successive estimates $\boldsymbol{\theta}(t)$ never decrease the likelihood function. The likelihood function keeps increasing until a maximum (local or global) is reached and the EM algorithm converges. The convergence proof can be found in the seminal paper [Demp 77] and further discussions in [Wu 83, Boyl 83]. Theoretical results as well as practical experimentation confirm that the convergence is slower than the quadratic convergence of Newton-type searching algorithms (Appendix C), although near the optimum a speedup may be possible. However, the great advantage of the algorithm is that its convergence is smooth and is not vulnerable to instabilities. Furthermore, it is computationally more attractive than Newton-like methods, which require the computation of the Hessian matrix. The keen reader may obtain more information on the EM algorithm and some of its applications from [McLa 88, Titt 85, Moon 96].

Application to the Mixture Modeling Problem

In this case, the complete data set consists of the joint events $(\boldsymbol{x}_k, j_k), k = 1, 2, \ldots, N$, and j_k takes integer values in the interval $[1, J]$, and it denotes the mixture from which $\boldsymbol{x}_k$ is generated. Employing our familiar rule, we obtain

$$p(\boldsymbol{x}_k, j_k; \boldsymbol{\theta}) = p(\boldsymbol{x}_k | j_k; \boldsymbol{\theta}) P_{j_k} \tag{2.92}$$

Assuming mutual independence among samples of the data set, the log-likelihood function becomes

$$L(\boldsymbol{\theta}) = \sum_{k=1}^{N} \ln\left(p(\boldsymbol{x}_k | j_k; \boldsymbol{\theta}) P_{j_k}\right) \tag{2.93}$$

Let $\boldsymbol{P} = [P_1, P_2, \ldots, P_J]^T$. In the current framework, the unknown parameter vector is $\boldsymbol{\Theta}^T = [\boldsymbol{\theta}^T, \boldsymbol{P}^T]^T$. Taking the expectation over the *unobserved data*, conditioned on the training samples and the current estimates, $\boldsymbol{\Theta}(t)$, of the unknown parameters, we have

$$\text{E-step: } Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}(t)) = E\left[\sum_{k=1}^{N} \ln(p(\boldsymbol{x}_k | j_k; \boldsymbol{\theta}) P_{j_k})\right]$$

$$= \sum_{k=1}^{N} E[\ln(p(\boldsymbol{x}_k | j_k; \boldsymbol{\theta}) P_{j_k})] \tag{2.94}$$

$$= \sum_{k=1}^{N} \sum_{j_k=1}^{J} P(j_k | \boldsymbol{x}_k; \boldsymbol{\Theta}(t)) \ln(p(\boldsymbol{x}_k | j_k; \boldsymbol{\theta}) P_{j_k}) \tag{2.95}$$

The notation can now be simplified by dropping the index k from j_k. This is because, for each k, we sum up over all possible J values of j_k and these are the same for all k. We will demonstrate the algorithm for the case of Gaussian mixtures with diagonal

covariance matrices of the form $\Sigma_j = \sigma_j^2 I$, that is,

$$p(\boldsymbol{x}_k|j;\boldsymbol{\theta}) = \frac{1}{\left(2\pi\sigma_j^2\right)^{l/2}} \exp\left(-\frac{\|\boldsymbol{x}_k - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \tag{2.96}$$

Assume that besides the prior probabilities, P_j, the respective mean values $\boldsymbol{\mu}_j$ as well as the variances $\sigma_j^2, j = 1, 2, \ldots, J$, of the Gaussians are also unknown. Thus, $\boldsymbol{\theta}$ is a $J(l+1)$-dimensional vector. Combining Eqs. (2.95) and (2.96) and omitting constants, we get

E-step:

$$Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}(t)) = \sum_{k=1}^{N}\sum_{j=1}^{J} P(j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))\left(-\frac{l}{2}\ln\sigma_j^2 - \frac{1}{2\sigma_j^2}\|\boldsymbol{x}_k - \boldsymbol{\mu}_j\|^2 + \ln P_j\right) \tag{2.97}$$

M-step: Maximizing the above with respect to $\boldsymbol{\mu}_j$, σ_j^2, and P_j results in (Problem 2.31)

$$\boldsymbol{\mu}_j(t+1) = \frac{\sum_{k=1}^{N} P(j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))\boldsymbol{x}_k}{\sum_{k=1}^{N} P(j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))} \tag{2.98}$$

$$\sigma_j^2(t+1) = \frac{\sum_{k=1}^{N} P(j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))\|\boldsymbol{x}_k - \boldsymbol{\mu}_j(t+1)\|^2}{l\sum_{k=1}^{N} P(j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))} \tag{2.99}$$

$$P_j(t+1) = \frac{1}{N}\sum_{k=1}^{N} P(j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t)) \tag{2.100}$$

For the iterations to be complete we need only to compute $P(j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))$. This is easily obtained from

$$P(j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t)) = \frac{p(\boldsymbol{x}_k|j; \boldsymbol{\theta}(t))P_j(t)}{p(\boldsymbol{x}_k; \boldsymbol{\Theta}(t))} \tag{2.101}$$

$$p(\boldsymbol{x}_k; \boldsymbol{\Theta}(t)) = \sum_{j=1}^{J} p(\boldsymbol{x}_k|j; \boldsymbol{\theta}(t))P_j(t) \tag{2.102}$$

Equations (2.98)–(2.102) constitute the EM algorithm for the estimation of the unknown parameters of the Gaussian mixtures in (2.86). The algorithm starts with valid initial guesses for the unknown parameters. Valid means that probabilities must add to one.

Remark

- Modeling unknown probability density functions via a mixture of Gaussian components and the EM algorithm has been very popular in a number of applications. Besides some convergence issues associated with the EM algorithm,

as previously discussed, another difficulty may arise in deciding about the exact number of components, J. In the context of supervised learning, one may use different values and choose the model that results in the best error probability. The latter can be computed by employing an error estimation technique (Chapter 10).

Example 2.8

Figure 2.17a shows $N = 100$ points in the two-dimensional space, which have been drawn from a multimodal distribution. The samples were generated using two Gaussian random generators $\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$, $\mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$, with

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 2.0 \\ 2.0 \end{bmatrix}$$

and covariance matrices

$$\Sigma_1 = \Sigma_2 = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}$$

respectively. Each time a sample $\boldsymbol{x}_k$, $k = 1, 2, \ldots, N$, is to be generated a coin is tossed. The corresponding probabilities for heads or tails are $P(H) \equiv P = 0.8, P(T) = 1 - P = 0.2$, respectively. If the outcome of the coin flip is heads, the sample $\boldsymbol{x}_k$ is generated from $\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$. Otherwise, it is drawn from $\mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$. This is the reason that in Figure 2.17a the space around the point $[1.0, 1.0]^T$ is more densely populated. The pdf of the data set can obviously be written as

$$p(\boldsymbol{x}) = g(\boldsymbol{x}; \boldsymbol{\mu}_1, \sigma_1^2)P + g(\boldsymbol{x}; \boldsymbol{\mu}_2, \sigma_2^2)(1 - P) \tag{2.103}$$

where $g(\cdot; \boldsymbol{\mu}, \sigma^2)$ denotes the Gaussian pdf with parameters the mean value $\boldsymbol{\mu}$ and a diagonal covariance matrix, $\Sigma = \text{diag}\{\sigma^2\}$, having σ^2 across the diagonal and zeros

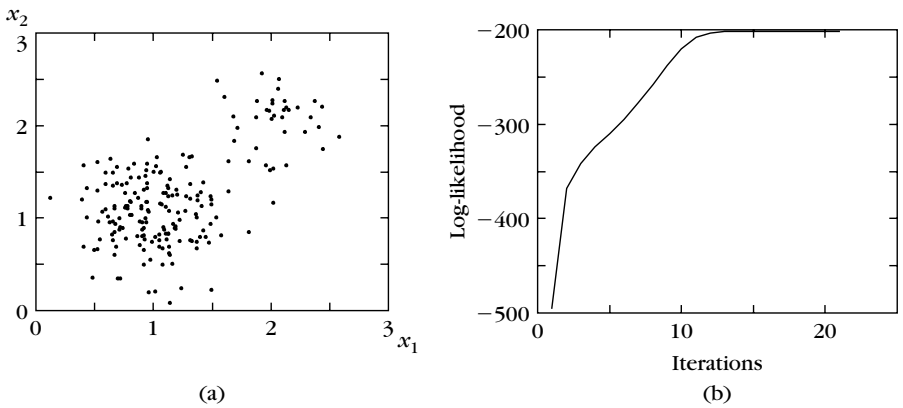

FIGURE 2.17

(a) The data set of Example 2.8 and (b) the log-likelihood as a function of the number of iteration steps.

elsewhere. Equation (2.103) is a special case of the more general formulation given in (2.86). The goal is to compute the maximum likelihood estimate of the unknown parameters vector

$$\boldsymbol{\Theta}^T = [P, \boldsymbol{\mu}_1^T, \sigma_1^2, \boldsymbol{\mu}_2^T, \sigma_2^2]$$

based on the available $N = 100$ points. The full training data set consists of the sample pairs $(\boldsymbol{x}_k, j_k)$, $k = 1, 2, \ldots, N$, where $j_k \in \{1, 2\}$, and it indicates the origin of each observed sample. However, only the points $\boldsymbol{x}_k$ are at our disposal, with the "label" information being hidden from us. To understand this issue better and gain more insight into the rationale behind the EM methodology, it may be useful to arrive at Eq. (2.95) from a slightly different route. Each of the random vectors, $\boldsymbol{x}_k$, can be thought of as the result of a linear combination of two other random vectors; namely,

$$\boldsymbol{x}_k = \alpha_k \boldsymbol{x}_k^1 + (1 - \alpha_k)\boldsymbol{x}_k^2$$

where $\boldsymbol{x}_k^1$ is drawn from $\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and $\boldsymbol{x}_k^2$ from $\mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$. The binary coefficients $\alpha_k \in \{0, 1\}$ are randomly chosen with probabilities $P(1) = P = 0.8$, $P(0) = 0.2$. If the values of the α_ks, $k = 1, 2, \ldots, N$, were known to us, the log-likelihood function in (2.93) would be written as

$$L(\boldsymbol{\Theta}; \alpha) = \sum_{k=1}^{N} \alpha_k \ln \left\{ g(\boldsymbol{x}_k; \boldsymbol{\mu}_1, \sigma_1^2) P \right\} + \sum_{k=1}^{N} (1 - \alpha_k) \ln \left\{ g(\boldsymbol{x}_k; \boldsymbol{\mu}_2, \sigma_2^2)(1 - P) \right\} \quad (2.104)$$

since we can split the summation in two parts, depending on the origin of each sample $\boldsymbol{x}_k$. However, this is just an "illusion" since the α_ks are unknown to us. Motivated by the spirit behind the EM algorithm, we substitute in (2.104) the respective mean values $E[\alpha_k | \boldsymbol{x}_k; \hat{\boldsymbol{\Theta}}]$, given an estimate, $\hat{\boldsymbol{\Theta}}$, of the unknown parameter vector. For the needs of our example we have

$$E[\alpha_k | \boldsymbol{x}_k; \hat{\boldsymbol{\Theta}}] = 1 \times P(1 | \boldsymbol{x}_k; \hat{\boldsymbol{\Theta}}) + 0 \times (1 - P(1 | \boldsymbol{x}_k; \hat{\boldsymbol{\Theta}})) = P(1 | \boldsymbol{x}_k; \hat{\boldsymbol{\Theta}}) \quad (2.105)$$

Substitution of (2.105) into (2.104) results in (2.95) for the case of $J = 2$.

We are now ready to apply the EM algorithm [Eqs. (2.98)–(2.102)] to the needs of our example. The initial values were chosen to be

$$\boldsymbol{\mu}_1(0) = [1.37, 1.20]^T, \quad \boldsymbol{\mu}_2(0) = [1.81, 1.62]^T, \quad \sigma_1^2 = \sigma_2^2 = 0.44, \quad P = 0.5$$

Figure 2.17b shows the log-likelihood as a function of the number of iterations. After convergence, the obtained estimates for the unknown parameters are

$$\boldsymbol{\mu}_1 = [1.05, 1.03]^T, \ \boldsymbol{\mu}_2 = [1.90, 2.08]^T, \ \sigma_1^2 = 0.10, \ \sigma_2^2 = 0.06, \ P = 0.844 \quad (2.106)$$

2.5.6 Nonparametric Estimation

So far in our discussion a pdf parametric modeling has been incorporated, in one way or another, and the associated unknown parameters have been estimated. In

the current subsection we will deal with nonparametric techniques. These are basically variations of the *histogram* approximation of an unknown pdf, which is familiar to us from our statistics basics. Let us take, for example, the simple one-dimensional case. Figure 2.18 shows two examples of a pdf and its approximation by the histogram method. That is, the x-axis (one-dimensional space) is first divided into successive bins of length h. Then the probability of a sample x being located in a bin is estimated for each of the bins. If N is the total number of samples and k_N of these are located inside a bin, the corresponding probability is approximated by the *frequency ratio*

$$P \approx k_N/N \tag{2.107}$$

This approximation converges to the true P as $N \to \infty$ (Problem 2.32). The corresponding pdf value is assumed constant throughout the bin and is approximated by

$$\hat{p}(x) \equiv \hat{p}(\hat{x}) \approx \frac{1}{h}\frac{k_N}{N}, \quad |x - \hat{x}| \leq \frac{h}{2} \tag{2.108}$$

where $\hat{x}$ is the midpoint of the bin. This determines the amplitude of the histogram curve over the bin. This is a reasonable approximation for continuous $p(x)$ and small enough h so that the assumption of constant $p(x)$ in the bin is sensible. It can be shown that $\hat{p}(x)$ converges to the true value $p(x)$ as $N \to \infty$ provided:

- $h_N \to 0$
- $k_N \to \infty$
- $\frac{k_N}{N} \to 0$

where h_N is used to show the dependence on N. These conditions can be understood from simple reasoning, without having to resort to mathematical details. The first has already been discussed. The other two show the way that k_N must grow

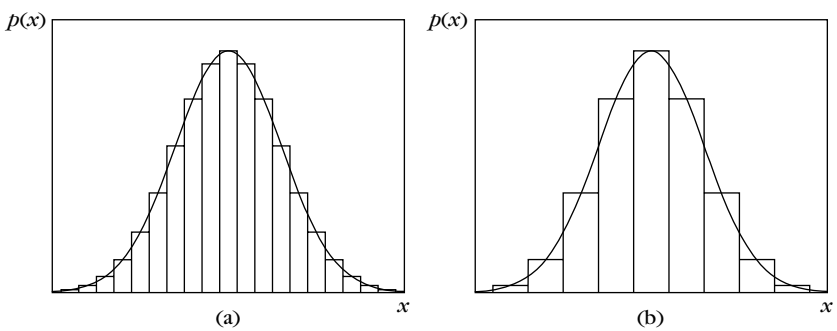

FIGURE 2.18

Probability density function approximation by the histogram method with (a) small and (b) large-size intervals (bins).

to guarantee convergence. Indeed, at all points where $p(x) \neq 0$ fixing the size h_N, however small, the probability P of points occurring in this bin is finite. Hence, $k_N \approx PN$ and k_N tends to infinity as N grows to infinity. On the other hand, as the size h_N of the bin tends to zero, the corresponding probability also goes to zero, justifying the last condition. In practice, the number N of data points is finite. The preceding conditions indicate the way that the various parameters must be chosen. N must be "large enough," h_N "small enough," and the number of points falling in each bin "large enough" too. How small and how large depend on the type of the pdf function and the degree of approximation one is satisfied with. Two popular approaches used in practice are described next.

Parzen Windows

In the multidimensional case, instead of bins of size h, the l-dimensional space is divided into hypercubes with length of side h and volume h^l. Let $\boldsymbol{x}_i, i = 1, 2, \ldots, N$, be the available feature vectors. Define the function $\phi(\boldsymbol{x})$ so that

$$\phi(\boldsymbol{x}_i) = \begin{cases} 1 & \text{for } |x_{ij}| \leq 1/2 \\ 0 & \text{otherwise} \end{cases} \tag{2.109}$$

where $x_{ij}, j = 1, \ldots, l$, are the components of $\boldsymbol{x}_i$. In words, the function is equal to 1 for all points inside the unit side hypercube centered at the origin and 0 outside it. This is shown in Figure 2.19(a). Then (2.108) can be "rephrased" as

$$\hat{p}(\boldsymbol{x}) = \frac{1}{h^l}\left(\frac{1}{N}\sum_{i=1}^{N}\phi\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}}{h}\right)\right) \tag{2.110}$$

The interpretation of this is straightforward. We consider a hypercube with length of side h centered at $\boldsymbol{x}$, the point where the pdf is to be estimated. This is illustrated in Figure 2.19(b) for the two-dimensional space. The summation equals k_N, that is, the number of points falling inside this hypercube. Then the pdf estimate results from dividing k_N by N and the respective hypercube volume h^l. However, viewing Eq. (2.110) from a slightly different perspective, we see that we try to approximate a continuous function $p(\boldsymbol{x})$ via an expansion in terms of discontinuous step functions $\phi(\cdot)$. Thus, the resulting estimate will suffer from this "ancestor's sin." This led Parzen [Parz 62] to generalize (2.110) by using smooth functions in the place of $\phi(\cdot)$. It can be shown that, provided

$$\phi(\boldsymbol{x}) \geq 0 \quad \text{and} \tag{2.111}$$

$$\int_{\boldsymbol{x}} \phi(\boldsymbol{x})\, d\boldsymbol{x} = 1 \tag{2.112}$$

the resulting estimate is a legitimate pdf. Such smooth functions are known as *kernels* or *potential functions* or *Parzen windows*. A typical example is the Gaussian $\mathcal{N}(\mathbf{0}, I)$, kernel. For such a choice, the approximate expansion of the unknown

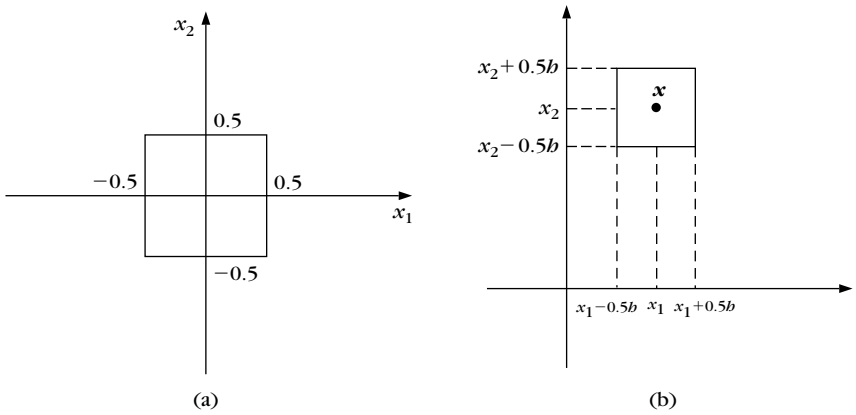

FIGURE 2.19

In the two-dimensional space (a) the function $\phi(\boldsymbol{x}_i)$ is equal to one for every point, $\boldsymbol{x}_i$, inside the square of unit side length, centered at the origin and equal to zero for every point outside it. (b) The function $\phi\left(\frac{\boldsymbol{x}_i-\boldsymbol{x}}{h}\right)$ is equal to unity for every point $\boldsymbol{x}_i$ inside the square with side length equal to h, centered at $\boldsymbol{x}$ and zero for all the other points.

$p(\boldsymbol{x})$ will be

$$\hat{p}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{\frac{l}{2}}h^l}\exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^T(\boldsymbol{x}-\boldsymbol{x}_i)}{2h^2}\right)$$

In other words, the unknown pdf is approximated as an average of N Gaussians, each one centered at a different point of the training set. Recall that as the parameter h becomes smaller, the shape of the Gaussians becomes narrower and more "spiky" (Appendix A) and the influence of each individual Gaussian is more localized in the feature space around the area of its mean value. On the other hand, the larger the value of h, the broader their shape becomes and more global in space their influence is. The expansion of a pdf in a sum of Gaussians was also used in 2.5.5. However, here, the number of Gaussians coincides with the number of points, and the unknown parameter, h, is chosen by the user. In the EM algorithm concept, the number of Gaussians is chosen independently of the number of training points, and the involved parameters are computed via an optimization procedure.

In the sequel, we will examine the limiting behavior of the approximation. To this end, let us take the mean value of (2.110)

$$E[\hat{p}(\boldsymbol{x})] = \frac{1}{h^l}\left(\frac{1}{N}\sum_{i=1}^{N}E\left[\phi\left(\frac{\boldsymbol{x}_i-\boldsymbol{x}}{h}\right)\right]\right)$$

$$\equiv \int_{\boldsymbol{x}'}\frac{1}{h^l}\phi\left(\frac{\boldsymbol{x}'-\boldsymbol{x}}{h}\right)p(\boldsymbol{x}')\,d\boldsymbol{x}' \tag{2.113}$$

Thus, the mean value is a *smoothed version* of the true pdf $p(\boldsymbol{x})$. However as $h \to 0$ the function $\frac{1}{h^l}\phi\left(\frac{\boldsymbol{x}'-\boldsymbol{x}}{h}\right)$ tends to the delta function $\delta(\boldsymbol{x}' - \boldsymbol{x})$. Indeed, its amplitude goes to infinity, its width tends to zero, and its integral from (2.112) remains equal to one. Thus, in this limiting case and for well-behaved continuous pdfs, $\hat{p}(\boldsymbol{x})$ is an unbiased estimate of $p(\boldsymbol{x})$. *Note that this is independent of the size N of the data set*. Concerning the variance of the estimate (Problem 2.38), the following remarks are valid:

- For fixed N, the smaller the h the higher the variance, and this is indicated by the noisy appearance of the resulting pdf estimate, for example, Figures 2.20a and 2.21a as well as Figures 2.22c and 2.22d. This is because $p(\boldsymbol{x})$ is approximated by a finite sum of δ-like spiky functions, centered at the training sample points. Thus, as one moves $\boldsymbol{x}$ in space the response of $\hat{p}(\boldsymbol{x})$ will be very high near the training points, and it will decrease very rapidly as one moves away, leading to this noiselike appearance. Large values of h smooth out local variations in density.
- For a fixed h, the variance decreases as the number of sample points N increases. This is illustrated in Figures 2.20a and 2.20b as well as in Figures 2.22b and 2.22c. This is because the space becomes dense in points, and the spiky functions are closely located. Furthermore, for a large enough number of samples, the smaller the h the better the accuracy of the resulting estimate, for example, Figures 2.20b and 2.21b.
- It can be shown, for example, [Parz 62, Fuku 90] that, under some mild conditions imposed on $\phi(\cdot)$, which are valid for most density functions, if h tends to zero but in such a way that $hN \to \infty$, the resulting estimate is both unbiased and asymptotically consistent.

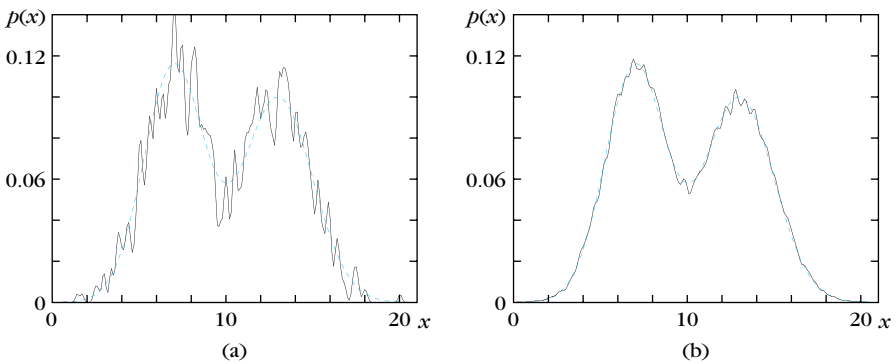

FIGURE 2.20

Approximation (full-black line) of a pdf (dotted-red line) via Parzen windows, using Gaussian kernels with (a) $h = 0.1$ and 1,000 training samples and (b) $h = 0.1$ and 20,000 samples. Observe the influence of the number of samples on the smoothness of the resulting estimate.

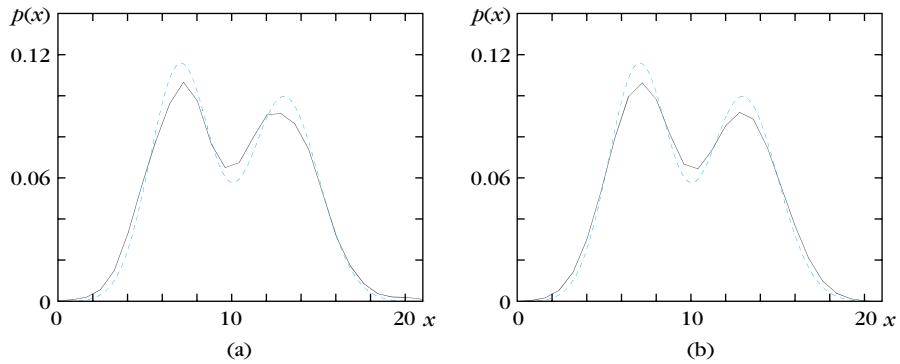

FIGURE 2.21

Approximation (full-black line) of a pdf (dotted-red line) via Parzen windows, using Gaussian kernels with (a) $h = 0.8$ and $1{,}000$ training samples and (b) $h = 0.8$ and $20{,}000$ samples. Observe that, in this case, increasing the number of samples has little influence on the smoothness as well as the approximation accuracy of the resulting estimate.

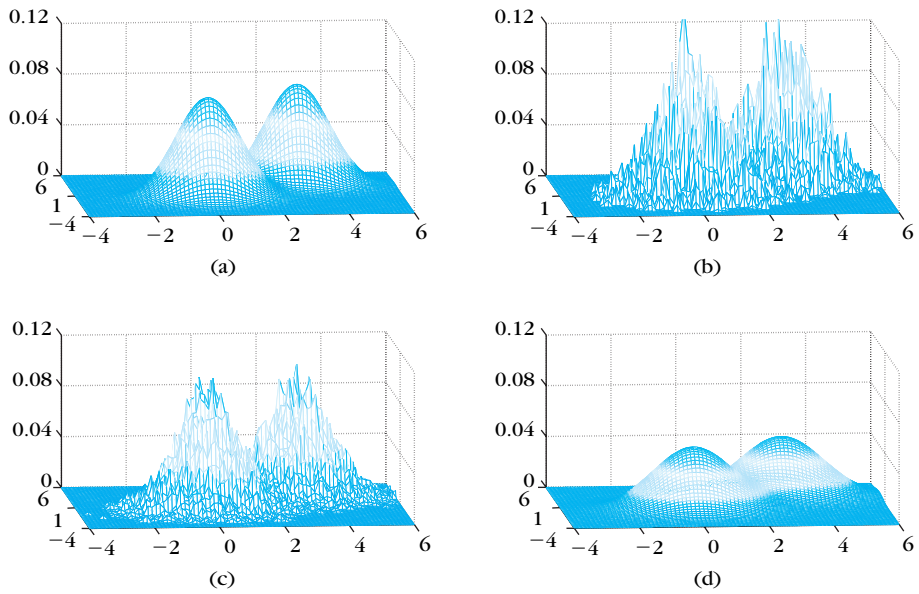

FIGURE 2.22

Approximation of a two-dimensional pdf, shown in (a), via Parzen windows, using two-dimensional Gaussian kernels with (b) $h = 0.05$ and $N = 1000$ samples, (c) $h = 0.05$ and $N = 20000$ samples and (d) $h = 0.8$ and $N = 20000$ samples. Large values of h lead to smooth estimates, but the approximation accuracy is low (the estimate is highly biased), as one can observe by comparing (a) with (d). For small values of h, the estimate is more noisy in appearance, but it becomes smoother as the number of samples increases, (b) and (c). The smaller the h and the larger the N, the better the approximation accuracy.

Remarks

- In practice, where only a finite number of samples is possible, a compromise between h and N must be made. The choice of suitable values for h is crucial, and several approaches have been proposed in the literature, for example, [Wand 95]. A straightforward way is to start with an initial estimate of h and then modify it iteratively to minimize the resulting misclassification error. The latter can be estimated by appropriate manipulation of the training set. For example, the set can be split into two subsets, one for training and one for testing. We will say more on this in Chapter 10.

- Usually, a large N is necessary for acceptable performance. This number grows exponentially with the dimensionality l. If a one-dimensional interval needs, say, N equidistant points to be considered as a densely populated one, the corresponding two-dimensional square will need N^2, the three-dimensional cube N^3, and so on. We usually refer to this as the *curse of dimensionality*. To our knowledge, this term was first used by Bellman in the context of Control theory [Bell 61]. To get a better feeling about the curse of dimensionality problem, let us consider the l-dimensional unit hypercube and let us fill it randomly with N points drawn from a uniform distribution. It can be shown ([Frie 89]) that the average Euclidean distance between a point and its nearest neighbor is given by

$$d(l, N) = 2\left(\frac{l\Gamma(l/2)}{2\pi^{\frac{l}{2}}N}\right)^{\frac{1}{l}}$$

where $\Gamma(\cdot)$ is the gamma function (Appendix A). In words, the average distance to locate the nearest neighbor to a point, for fixed l, shrinks as $N^{-\frac{1}{l}}$. To get a more quantitative feeling, let us fix N to the value $N = 10^{10}$. Then for $l = 2,\ 10,\ 20$ and 40, $d(l, N)$ becomes 10^{-5}, 0.18, 0.76, and 1.83, respectively. Figure 2.23a shows 50 points lying within the unit-length segment in the one-dimensional space. The points were randomly generated by the uniform distribution. Figure 2.23b shows the same number of points lying in the unit-length square. These points were also generated by a uniform distribution in the two-dimensional space. It is readily seen that the points in the one-dimensional segment are, on average, more closely located compared to the same number of points in the two-dimensional square.

The large number of data points required for a relatively high-dimensional feature space to be sufficiently covered puts a significant burden on complexity requirements, since one has to consider one Gaussian centered at each point. To this end, some techniques have been suggested that attempt to approximate the unknown pdf by using a reduced number of kernels, see, for example, [Babi 96].

Another difficulty associated with high-dimensional spaces is that, in practice, due to the lack of enough training data points, some regions in the feature space may be sparsely represented in the data set. To cope with

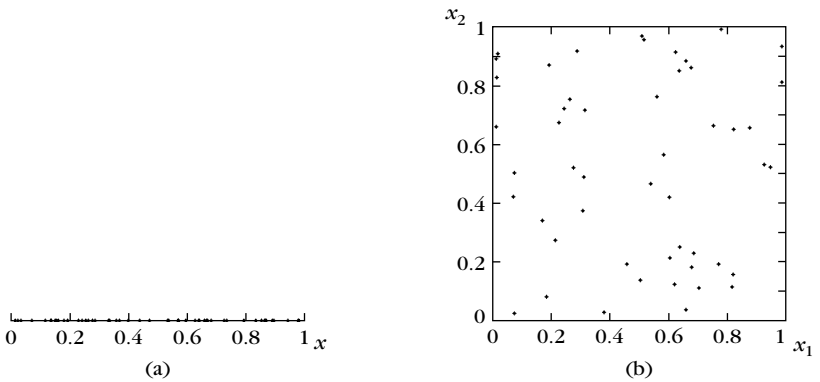

FIGURE 2.23

Fifty points generated by a uniform distribution lying in the (a) one-dimensional unit-length segment and (b) the unit-length square. In the two-dimensional space the points are more spread compared to the same number of points in the one-dimensional space.

such scenarios, some authors have adopted a variable value for h. In regions where data are sparse, a large value of h is used, while in more densely populated areas a smaller value is employed. To this end, a number of mechanisms for adjusting the value of h have been adopted, see, for example, [Brei 77, Krzy 83, Terr 92, Jone 96].

Application to classification: On the reception of a feature vector $\boldsymbol{x}$ the likelihood test in (2.20) becomes

$$\text{assign } \boldsymbol{x} \text{ to } \omega_1(\omega_2) \quad \text{if} \quad l_{12} \approx \left(\frac{\frac{1}{N_1 h^l} \sum_{i=1}^{N_1} \phi\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}}{h}\right)}{\frac{1}{N_2 h^l} \sum_{i=1}^{N_2} \phi\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}}{h}\right)} \right) > (<) \frac{P(\omega_2)}{P(\omega_1)} \frac{\lambda_{21} - \lambda_{22}}{\lambda_{12} - \lambda_{11}} \tag{2.114}$$

where N_1, N_2 are the training vectors in class ω_1, ω_2, respectively. The risk-related terms are ignored when the Bayesian minimum error probability classifier is used. For large N_1, N_2 this computation is a very demanding job, in both processing time and memory requirements.

k Nearest Neighbor Density Estimation

In the Parzen estimation of the pdf in (2.110), the volume around the points $\boldsymbol{x}$ was considered fixed (h^l) and the number of points k_N, falling inside the volume, was left to vary randomly from point to point. Here we will reverse the roles. The number of points $k_N = k$ will be fixed, and the size of the volume around $\boldsymbol{x}$ will be adjusted each time, to include k points. Thus, *in low-density areas the volume will be large and in high-density areas it will be small.* We can also consider more general types of regions, besides the hypercube. The estimator can now be

written as

$$\hat{p}(\boldsymbol{x}) = \frac{k}{NV(\boldsymbol{x})} \tag{2.115}$$

where the dependence of the volume $V(\boldsymbol{x})$ on $\boldsymbol{x}$ is explicitly shown. Again it can be shown [Fuku 90] that asymptotically ($\lim k = +\infty, \lim N = +\infty, \lim(k/N) = 0$) this is an unbiased and consistent estimate of the true pdf, and it is known as the *k Nearest Neighbor (kNN) density estimate*. Results concerning the finite k, N case have also been derived; see [Fuku 90, Butu 93]. A selection of seminal papers concerning NN classification techniques can be found in [Dasa 91].

From a practical point of view, at the reception of an unknown feature vector $\boldsymbol{x}$, we compute its distance d, for example, Euclidean, from *all* the training vectors of the various classes, for example, ω_1, ω_2. Let r_1 be the radius of the hypersphere, centered at $\boldsymbol{x}$, that contains k points from ω_1 and r_2 the corresponding radius of the hypersphere containing k points from class ω_2 (k may not necessarily be the same for all classes). If we denote by V_1, V_2 the respective hypersphere volumes, the likelihood ratio test becomes

$$\begin{aligned} \text{assign } \boldsymbol{x} \text{ to } \omega_1(\omega_2) \quad \text{if } l_{12} \approx \frac{kN_2V_2}{kN_1V_1} > (<) \frac{P(\omega_2)}{P(\omega_1)} \frac{\lambda_{21} - \lambda_{22}}{\lambda_{12} - \lambda_{11}} \\ \frac{V_2}{V_1} > (<) \frac{N_1}{N_2} \frac{P(\omega_2)}{P(\omega_1)} \frac{\lambda_{21} - \lambda_{22}}{\lambda_{12} - \lambda_{11}} \end{aligned} \tag{2.116}$$

If the Mahalanobis distance is alternatively adopted, we will have hyperellipsoids in the place of the hyperspheres.

The volume of a hyperellipsoid, corresponding to Mahalanobis distance equal to r, is given by ([Fuku 90])

$$V = V_0 |\Sigma|^{\frac{1}{2}} r^l \tag{2.117}$$

where V_0 is the volume of the hypersphere of unit radius given by

$$V_0 = \begin{cases} \pi^{\frac{l}{2}}/(l/2)!, & l \text{ even} \\ 2^l \pi^{\frac{l-1}{2}} (\frac{l-1}{2})!/l!, & l \text{ odd} \end{cases} \tag{2.118}$$

Verify that Eq. (2.117) results to $4\pi r^3/3$ for the volume of a sphere of radius r in the three-dimensional space.

Remark

- The nonparametric probability density function estimation techniques, discussed in this section, are among the techniques that are still in use in practical applications. It is interesting to note that, although the performance of the methods, as density estimators, degrades in high-dimensional spaces due to the lack of sufficient data, their performance as classifiers may be sufficiently good. After all, lack of enough training data points affects, in one way or another, all the methods.

More recently, the so-called *probabilistic neural networks* have been suggested as efficient implementations for the computation of the classifier given in (2.114), by exploiting the intrinsic parallelism of the neural network architectures and will be discussed in Chapter 4.

Example 2.9

The points shown in Figure 2.24 belong to either of two equiprobable classes. Black points belong to class ω_1 and red points belong to class ω_2. For the needs of the example we assume that all points are located at the nodes of a grid. We are given the point denoted by a "star", with coordinates $(0.7, 0.6)$, which is to be classified in one of the two classes. The Bayesian (minimum error probability) classifier and the k-nearest neighbor density estimation technique, for $k = 5$, will be employed.

Adopting the Euclidean distance, we find the five nearest neighbors to the unknown point $(0.7, 0.6)$ from all the points in class ω_2. These are the points $(0.8, 0.6)$, $(0.7, 0.7)$, $(0.6, 0.5)$, $(0.6, 0.6)$, $(0.6, 0.7)$. The full line circle encircles the five nearest neighbors, and its radius is equal to the distance of the point that is furthest from $(0.7, 0.6)$, that is, $\rho = \sqrt{0.1^2 + 0.1^2} = 0.1\sqrt{2}$. In the sequel, we repeat the procedure for the points in class ω_1. The nearest points are the ones with coordinates $(0.7, 0.5)$, $(0.8, 0.4)$, $(0.8, 0.7)$,

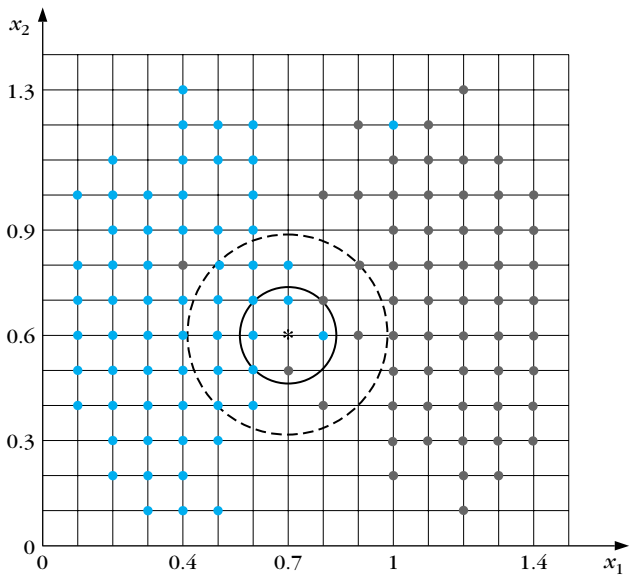

FIGURE 2.24

The setup for the example 2.9. The point denoted by a "star" is classified to the class ω_2 of the red points. The $k = 5$ nearest neighbors from this class lie within a smaller area compared to the five nearest neighbors coming from the other class.

(0.9, 0.6), (0.9, 0.8). The dotted circle is the one that encircles all five points, and its radius is equal to $\sqrt{0.2^2 + 0.2^2} = 0.2\sqrt{2} = 2\rho$.

There are $N_1 = 59$ points in class ω_1 and $N_2 = 61$ in class ω_2. The areas (volumes) of the two circles are $V_1 = 4\pi\rho^2$ and $V_2 = \pi\rho^2$, respectively, for the two classes. Hence, according to Eq. (2.116) and ignoring the risk related terms, we have

$$\frac{V_2}{V_1} = \frac{\pi\rho^2}{4\pi\rho^2} = 0.25$$

and since 0.25 is less than 59/61 and the classes are equiprobable, the point (0.7, 0.6) is classified to class ω_2.

2.5.7 The Naive-Bayes Classifier

The goal in this section, so far, was to present various techniques for the estimation of the probability density functions $p(\boldsymbol{x}|\omega_i)$, $i = 1, 2, \ldots, M$, required by the Bayes classification rule, based on the available training set, X. As we have already stated, in order to safeguard good estimates of the pdfs the number of training samples, N, must be large enough. To this end, the demand for data increases exponentially fast with the dimension, l, of the feature space. Crudely speaking, if N could be regarded as a good number of training data points for obtaining sufficiently accurate estimates of a pdf in an one-dimensional space, then N^l points would be required for an l-dimensional space. Thus, large values of l make the accurate estimation of a multidimensional pdf a bit of an "illusion" since in practice data is hard to obtain. Loosely speaking, data can be considered to be something like money. It is never enough! Accepting this reality, one has to make concessions about the degree of accuracy that is expected from the pdf estimates. One widely used approach is to assume that individual features x_j, $j = 1, 2, \ldots, l$, are statistically independent. Under this assumption, we can write

$$p(\boldsymbol{x}|\omega_i) = \prod_{j=1}^{l} p(x_j|\omega_i), \quad i = 1, 2, \ldots, M$$

The scenario is now different. To estimate l one-dimensional pdfs, for each of the classes, lN data points would be enough in order to obtain good estimates, instead of N^l. This leads to the so-called *naive-Bayes* classifier, which assigns an unknown sample $\boldsymbol{x} = [x_1, x_2, \ldots, x_l]^T$ to the class

$$\omega_m = \arg \max_{\omega_i} \prod_{j=1}^{l} p(x_j|\omega_i), \quad i = 1, 2, \ldots, M$$

It turns out that the naive-Bayes classifier can be very robust to violations of its independence assumption, and it has been reported to perform well for many real-world data sets. See, for example, [Domi 97].

Example 2.10

The discrete features case: In Section 2.2, it was stated that in the case of discrete-valued features the only required change in the Bayesian classification rule is to replace probability density functions with probabilities. In this example, we will see how the associated with the naive Bayes classifier assumption of statistical independence among the features simplifies the Bayesian classification rule.

Consider the feature vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_l]^T$ with binary features, that is, $x_i \in \{0, 1\}$, $i = 1, 2, \ldots, l$. Also let the respective class-conditional probabilities be $P(x_i = 1|\omega_1) = p_i$ and $P(x_i = 1|\omega_2) = q_i$. According to the Bayesian rule, given the value of $\boldsymbol{x}$, its class is decided according to the value of the likelihood ratio

$$\frac{P(\omega_1)P(\boldsymbol{x}|\omega_1)}{P(\omega_2)P(\boldsymbol{x}|\omega_2)} > (<) 1 \tag{2.119}$$

for the minimum probability error rule (the minimum risk rule could also be used).

The number of values that $\boldsymbol{x}$ can take, for all possible combinations of x_i, amounts to 2^l. If we do not adopt the independence assumption, then one must have enough training data in order to obtain probability estimates for each one of these values (probabilities add to one, thus $2^l - 1$ estimates are required). However, adopting statistical independence among the features, we can write

$$P(\boldsymbol{x}|\omega_1) = \prod_{i=1}^{l} p_i^{x_i}(1 - p_i)^{1-x_i}$$

and

$$P(\boldsymbol{x}|\omega_2) = \prod_{i=1}^{l} q_i^{x_i}(1 - q_i)^{1-x_i}$$

Hence, the number of required probability estimates is now $2l$, that is, the p_i's and q_i's. It is interesting to note that, taking the logarithm of both sides in (2.119), one ends up with a *linear* discriminant function similar to the hyperplane classifier of Section 2.4, that is,

$$g(\boldsymbol{x}) = \sum_{i=1}^{l} \left(x_i \ln \frac{p_i}{q_i} + (1 - x_i) \ln \frac{1 - p_i}{1 - q_i} \right) + \ln \frac{P(\omega_1)}{P(\omega_2)} \tag{2.120}$$

which can easily be brought into the form of

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 \tag{2.121}$$

where

$$\boldsymbol{w} = \left[\ln \frac{p_1(1 - q_1)}{q_1(1 - p_1)}, \ldots, \ln \frac{p_l(1 - q_l)}{q_l(1 - p_l)} \right]^T$$

and

$$w_0 = \sum_{i=1}^{l} \ln \frac{1 - p_i}{1 - q_i} + \ln \frac{P(\omega_1)}{P(\omega_2)}$$

Binary features are used in a number of applications where one has to decide based on the presence or not of certain attributes. For example, in medical diagnosis, **1** can represent a normal value in a medical test and a **0** an abnormal one.

2.6 THE NEAREST NEIGHBOR RULE

A variation of the kNN density estimation technique results in a *suboptimal*, yet popular in practice, nonlinear classifier. Although this does not fall in the Bayesian framework, it fits nicely at this point. In a way, this section could be considered as a bridge with Chapter 4. The algorithm for the so-called *nearest neighbor rule* is summarized as follows. Given an unknown feature vector $\boldsymbol{x}$ and a distance measure, then:

- Out of the N training vectors, identify the k nearest neighbors, *regardless* of class label. k is chosen to be odd for a two class problem, and in general not to be a multiple of the number of classes M.
- Out of these k samples, identify the number of vectors, k_i, that belong to class $\omega_i, i = 1, 2, \ldots, M$. Obviously, $\sum_i k_i = k$.
- Assign $\boldsymbol{x}$ to the class ω_i with the maximum number k_i of samples.

Figure 2.25 illustrates the k-NN rule for the case of $k = 11$. Various distance measures can be used, including the Euclidean and Mahalanobis distance.

The simplest version of the algorithm is for $k = 1$, known as the *nearest neighbor (NN)* rule. In other words, a feature vector $\boldsymbol{x}$ is assigned to the class of its nearest neighbor! Provided that the number of training samples is large enough, this simple

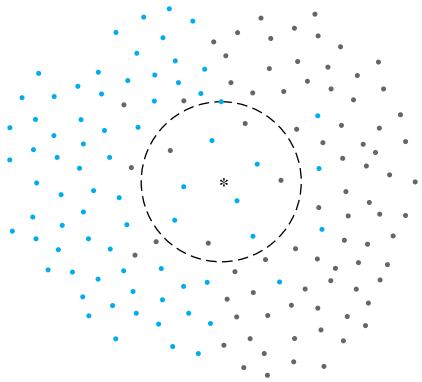

FIGURE 2.25

Using the 11-NN rule, the point denoted by a "star" is classified to the class of the red points. Out of the eleven nearest neighbors seven are red and four are black. The circle indicates the area within which the eleven nearest neighbors lie.

rule exhibits good performance. This is also substantiated by theoretical findings. It can be shown [Duda 73, Devr 96] that, as $N \to \infty$, the classification error probability, for the NN rule, P_{NN}, is bounded by

$$P_B \leq P_{NN} \leq P_B\left(2 - \frac{M}{M-1}P_B\right) \leq 2P_B \tag{2.122}$$

where P_B is the optimal Bayesian error. Thus, the error committed by the NN classifier is (*asymptotically*) at most twice that of the optimal classifier. The asymptotic performance of the kNN is better than that of the NN, and a number of interesting bounds have been derived. For example, for the two-class case it can be shown, for example, [Devr 96] that

$$P_B \leq P_{kNN} \leq P_B + \frac{1}{\sqrt{ke}} \quad \text{or} \quad P_B \leq P_{kNN} \leq P_B + \sqrt{\frac{2P_{NN}}{k}} \tag{2.123}$$

Both of these suggest that as $k \to \infty$ the performance of the kNN tends to the optimal one. Furthermore, for small values of Bayesian errors, the following approximations are valid [Devr 96]:

$$P_{NN} \approx 2P_B \tag{2.124}$$

$$P_{3NN} \approx P_B + 3(P_B)^2 \tag{2.125}$$

Thus, for large N and small Bayesian errors, we expect the 3NN classifier to give performance almost identical to that of the Bayesian classifier. As an example, let us say that the error probability of the Bayesian classifier is of the order of 1%; then the error resulting from a 3NN classifier will be of the order of 1.03%! The approximation improves for higher values of k. A little thought can provide justification for this without too much mathematics. Under the assumption of large N, the radius of the hypersphere (Euclidean distance) centered at $\boldsymbol{x}$ and containing its k nearest neighbors tends to zero [Devr 96]. This is natural, because for very large N we expect the space to be densely filled with samples. Thus, the k (a very small portion of N) neighbors of $\boldsymbol{x}$ will be located very close to it, and the conditional class probabilities, at all points inside the hypersphere around $\boldsymbol{x}$, will be approximately equal to $P(\omega_i|\boldsymbol{x})$ (assuming continuity). Furthermore, for large k (yet an infinitesimally small fraction of N), the majority of the points in the region will belong to the class corresponding to the maximum conditional probability. Thus, the kNN rule tends to the Bayesian classifier. Of course, all these are true asymptotically. In the finite sample case there are even counterexamples (Problem 2.34) where the kNN results in higher error probabilities than the NN. However, in conclusion, it can be stated that the nearest neighbor techniques are among the serious candidates to be adopted as classifiers in a number of applications. A comparative study of the various statistical classifiers, considered in this chapter as well as others, can be found in [Aebe 94].

Remarks

- A serious drawback associated with (k)NN techniques is the complexity in search of the nearest neighbor(s) among the N available training samples.

Brute-force searching amounts to operations proportional to kN ($O(kN)$).[2] The problem becomes particularly severe in high-dimensional feature spaces. To reduce the computational burden, a number of efficient searching schemes have been suggested; see, for example, [Fuku 75, Dasa 91, Brod 90, Djou 97, Nene 97, Hatt 00, Kris 00, Same 08]. In [Vida 94, Mico 94] a preprocessing stage is suggested that computes a number of *base prototypes* that are in some sense maximally separated from among the set of training feature vectors. A summary of efficient searching techniques and a comparative study is given in [McNa 01].

- Although, due to its asymptotic error performance, the kNN rule achieves good results when the data set is large (compared to the dimension of the feature space), the performance of the classifier may degrade dramatically when the value of N is relatively small [Devr 96]. Also, in practice, one may have to reduce the number of training patterns due to the constraints imposed by limited computer resources. To this end, a number of techniques, also known as *prototype editing* or *condensing*, have been proposed. The idea is to reduce the number of training points in a way that a cost related to the error performance is optimized; see, for example, [Yan 93, Huan 02, Pare 06a] and the references therein. Besides computational savings, reducing the size of a finite set appropriately may offer performance improvement advantages, by making the classifier less sensitive to outliers. A simple method, which also makes transparent the reason for such a potential improvement, has been suggested in [Wils 72]. This editing procedure tests a sample using a kNN rule against the rest of the data. The sample is discarded if it is misclassified. The edited data set is then used for a NN classification of unknown samples.

  A direction to cope with the performance degradation associated with small values of N is to employ distance measures that are optimized on the available training set. The goal is to find a data-adaptive distance metric that leads to an optimal performance, according to an adopted cost. Such *trained* metrics can be *global* ones (i.e., the same at every point), *class-dependent* (i.e., shared by all points of the same class), and/or *locally dependent* (i.e., the metric varies according to the position in the feature space); see, for example, [Hast 96, Dome 05, Pare 06] and the references therein. An in depth treatment of the topic is given in [Frie 94].

- When the $k = 1$ nearest neighbor rule is used, the training feature vectors $\boldsymbol{x}_i, i = 1, 2, \ldots, N$, define a partition of the l-dimensional space into N regions, R_i. Each of these regions is defined by

$$R_i = \{\boldsymbol{x}: d(\boldsymbol{x}, \boldsymbol{x}_i) < d(\boldsymbol{x}, \boldsymbol{x}_j), i \neq j\} \tag{2.126}$$

  that is, R_i contains all points in space that are closer to $\boldsymbol{x}_i$ than any other point of the training set, with respect to the distance d. This partition of the

[2] $O(n)$ denotes order of n calculations.

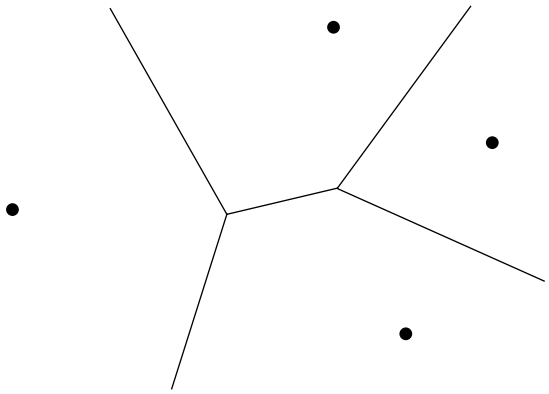

FIGURE 2.26

An example of Voronoi tessellation in the two-dimensional space and for Euclidean distance.

feature space is known as *Voronoi tessellation*. Figure 2.26 is an example of the resulting *Voronoi tessellation* for the case of $l = 2$ and the Euclidean distance.

2.7 BAYESIAN NETWORKS

In Section 2.5.7 the naive-Bayes classifier was introduced as a means of coping with the curse of dimensionality and to exploiting more efficiently the available training data set. However, by adopting the naive-Bayes classifier, one goes from one extreme (fully dependent features) to another (features mutually independent). Common sense drives us to search for approximations that lie between these two extremes.

The essence of the current section is to introduce a methodology that allows one to develop models that can accommodate built-in independence assumptions with respect to the features x_i, $i = 1, 2, \ldots, l$. Recall the well-known probability chain rule [Papo 91, p. 192]

$$p(x_1, x_2, \ldots, x_l) = p(x_l|x_{l-1}, \ldots, x_1)p(x_{l-1}|x_{l-2} \ldots, x_1) \ldots, p(x_2|x_1)p(x_1) \qquad (2.127)$$

This rule applies always and does not depend on the order in which features are presented. The rule states that the joint probability density function can be expressed in terms of the product of conditional pdfs and a marginal one ($p(x_1)$).[3] This important and elegant rule opens the gate through which assumptions will infiltrate the problem. The conditional dependence for each feature, x_i, will be limited into a subset of the features appearing in each term in the product. Under this assumption,

[3] In the study of several random variables, the statistics of each are called marginal.

Eq. (2.127) can now be written as

$$p(\boldsymbol{x}) = p(x_1)\prod_{i=2}^{l} p(x_i|A_i) \tag{2.128}$$

where

$$A_i \subseteq \{x_{i-1}, x_{i-2}, \ldots, x_1\} \tag{2.129}$$

For example, let $l = 6$ and

$$p(x_6|x_5, \ldots, x_1) = p(x_6|x_5, x_4) \tag{2.130}$$

$$p(x_5|x_4, \ldots, x_1) = p(x_5|x_4) \tag{2.131}$$

$$p(x_4|x_3, x_2, x_1) = p(x_4|x_2, x_1) \tag{2.132}$$

$$p(x_3|x_2, x_1) = p(x_3|x_2) \tag{2.133}$$

$$p(x_2|x_1) = p(x_2) \tag{2.134}$$

Then,

$$A_6 = \{x_5, x_4\},\ A_5 = \{x_4\},\ A_4 = \{x_2, x_1\},\ A_3 = \{x_2\}, A_2 = \emptyset$$

where $\emptyset$ denotes the empty set. These assumptions are represented graphically in Figure 2.27. Nodes correspond to features. The *parents* of a feature, x_i, are those features with directed links toward x_i and are the members of the set A_i. In other words, x_i *is conditionally independent* of any combination of its *nondescendants*, *given its parents*. There is a subtle point concerning conditional independence. Take, for example, that $p(x_3|x_2, x_1) = p(x_3|x_2)$. This does not necessarily mean that x_3 and x_1 are independent. They may be dependent while x_2 is unknown, but they become independent once the value of x_2 is disclosed to us. This is not surprising since by measuring the value of a random variable part of the randomness is removed.

Under the previous assumptions, the problem of estimating the joint pdf has broken into the product of simpler terms. Each of them involves, in general, a much smaller number of features compared to the original number. For example,

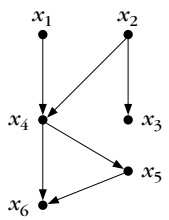

FIGURE 2.27

Graphical model illustrating conditional dependencies.

for the case of Eqs. (2.130)–(2.134) none of the products involves more than three features. Hence, the estimation of each pdf term in the product takes place in a low-dimensional space and the problems arising from the curse of dimensionality can be handled easier. To get a feeling for the computational size reduction implied by the independence assumptions, encoded in the graphical model of Figure 2.27, let us assume that variables $x_i,\ i = 1, 2, \ldots, 6$, are binary. Then the pdfs in (2.127)–(2.134) become probabilities. Complete knowledge of $P(x_1, \ldots, x_6)$ requires the estimation of 63 $(2^l - 1)$ probability values. It is 63 and not 64 due to the constraint that probabilities must add to one. This is also suggested by the right-hand side of Eq. (2.127). The number of the required probability values is $2^{l-1} + 2^{l-2} + \cdots + 1 = 2^l - 1$. In contrast to that, the assumptions in (2.130)–(2.134) reduce the number of the required probability values to be estimated to 13 (Why?). For large values of l, such a saving can be very significant.

The naive-Bayes classifier is a special case for which $A_i = \emptyset,\ i = 2, \ldots, l$, and the product in (2.128) becomes a product of marginal pdfs. Examples of classifiers that exploit the idea of conditional independence with respect to a subset of features are given in, for example, [Frie 97, Webb 05, Roos 05].

Although our original goal was to seek for ways for the approximate estimation of a joint pdf, it turns out that the adopted assumptions (nicely condensed in a graphical representation such as in Figure 2.27), have much more interesting consequences. For the rest of the section and for the sake of simplicity, we will assume that the features can only take values from a discrete set. Thus, pdfs give their place to probabilities.

Definition: A *Bayesian network* is a directed acyclic graph (DAG) where the nodes correspond to random variables (features). Each node is associated with a set of conditional probabilities, $P(x_i|A_i)$, where x_i is the variable associated with the specific node and A_i is the set of its parents in the graph.

Acyclic means that there are no cycles in the graph. For example, the graph in Figure 2.27 is an acyclic one, and it will cease to be so if one draws an arc directed from x_6 to, say, x_1. The complete specification of a Bayesian network requires knowledge of (a) the marginal probabilities of the root nodes (those without a parent) and (b) the conditional probabilities of the nonroot nodes, given their parents for *all* possible combinations of their values. The joint probability of the variables can now be obtained by multiplying all conditional probabilities with the prior probabilities of the root nodes. All that is needed is to perform a *topological sorting* of the random variables; that is, to order the variables such that every variable comes before its descendants in the related graph.

Bayesian networks have been used in a variety of applications. The network in Figure 2.28 corresponds to an example inspired by the discipline of medical diagnosis, a scientific area where Bayesian networks have been very popular. S stands for smokers, C for lung cancer, and H for heart disease. H1 and H2 are heart disease medical tests, and C1 and C2 are cancer medical tests. The table of the root node shows the population percentage (probability) of smokers (True) and

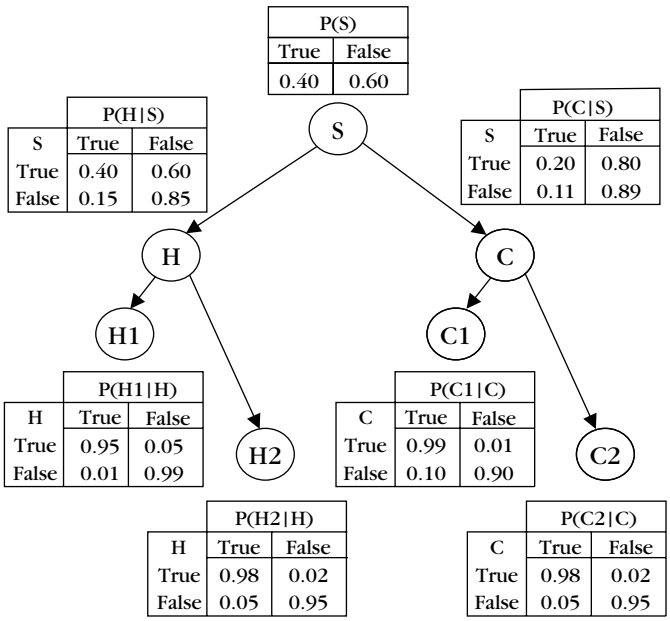

FIGURE 2.28

Bayesian network modeling conditional dependencies for an example concerning smokers (S), tendencies to develop cancer (C), and heart disease (H), together with variables corresponding to heart (H1, H2) and cancer (C1, C2) medical tests.

nonsmokers (False). The tables along the nodes of the tree are the respective conditional probabilities. For example, $P(C : \text{True}|S : \text{True}) = 0.20$ is the probability of a smoker (True) to develop cancer (True). (The probabilities used in Figure 2.28 may not correspond to true values having resulted from statistical studies.)

Once a DAG has been constructed, the Bayesian network allows one to calculate *efficiently* the conditional probability of *any* node in the graph, given that the values of some other nodes have been *observed*. Such questions arise in the field of artificial intelligence closely related to pattern recognition. The computational efficiency stems from the existing probability relations encoded in the graph. A detailed treatment of the topic is beyond the scope of this book; the interested reader may consult more specialized texts, such as [Neap 04]. The remainder of this section aims at providing the reader with a flavor of the related theory.

Probability Inference: This is the most common task that Bayesian networks help us to solve efficiently. Given the values of some of the variables, known as *evidence*, the goal is to compute the conditional probabilities for some (or all) of the other variables in the graph, *given* the evidence.

Example 2.11

Let us take the simple Bayesian network of Figure 2.29. For notational simplicity we avoid subscripts, and the involved variables are denoted by x, y, z, w. Each variable is assumed to be binary. We also use the symbol $x1$ instead of $x = 1$ and $x0$ instead of $x = 0$, and similarly for the rest of the variables. The Bayesian network is fully specified by the marginal probabilities of the root node (x) and the conditional probabilities shown in Figure 2.29. Note that only the values above the graph need to be specified. Those below the graph can be derived. Take, for example, the y node.

$$P(y1) = P(y1|x1)P(x1) + P(y1|x0)P(x0) = (0.4)(0.6) + (0.3)(0.4) = 0.36$$

$$P(y0) = 1 - P(y1) = 0.64$$

Also,

$$P(y0|x1) = 1 - P(y1|x1)$$

The rest are similarly derived. Note that all of these parameters should be available prior to performing probability inference. Suppose now that:

(a) x is measured and let its value be $x1$ (the evidence). We seek to compute $P(z1|x1)$ and $P(w0|x1)$.

(b) w is measured and let its value be $w1$. We seek to compute $P(x0|w1)$ and $P(z1|w1)$.

To answer (a), the following calculations are in order.

$$\begin{aligned} P(z1|x1) &= P(z1|y1, x1)P(y1|x1) + P(z1|y0, x1)P(y0|x1) \\ &= P(z1|y1)P(y1|x1) + P(z1|y0)P(y0|x1) \\ &= (0.25)(0.4) + (0.6)(0.6) = 0.46 \end{aligned} \tag{2.135}$$

Though not explicitly required, $P(z0|x1)$ must also be evaluated, as we will soon realize.

$$P(z0|x1) = 1 - P(z1|x1) = 0.54 \tag{2.136}$$

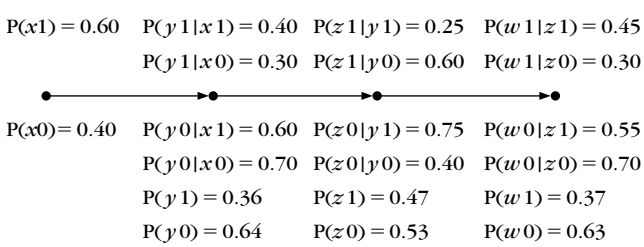

FIGURE 2.29

A simple Bayesian network where conditional dependencies are restricted to a single variable.

In a similar way, we obtain

$$\begin{aligned} P(w0|x1) &= P(w0|z1, x1)P(z1|x1) + P(w0|z0, x1)P(z0|x1) \\ &= P(w0|z1)P(z1|x1) + P(w0|z0)P(z0|x1) \\ &= (0.55)(0.46) + (0.7)(0.54) = 0.63 \end{aligned} \tag{2.137}$$

We can think of the algorithm as a process that *passes messages* (i.e., probabilities) downward from one node to the next. The first two computations, (2.135) and (2.136), "are performed in node z" and then "passed" to the last node, where (2.137) is performed.

To answer (b), the direction of "message propagation" is reversed since, now, the evidence is provided from node w and the required information, $P(x0|w1)$, $P(z1|w1)$ concerns nodes x and z, respectively.

$$P(z1|w1) = \frac{P(w1|z1)P(z1)}{P(w1)} = \frac{(0.45)(0.47)}{0.37} = 0.57$$

The activity is then passed to node y, where the following needs to be performed.

$$P(y1|w1) = \frac{P(w1|y1)P(y1)}{P(w1)}$$

$P(w1|y1)$ is unknown and can be computed as discussed in the "downward" message propagation. That is,

$$\begin{aligned} P(w1|y1) &= P(w1|z1, y1)P(z1|y1) + P(w1|z0, y1)P(z0|y1) \\ &= P(w1|z1)P(z1|y1) + P(w1|z0)P(z0|y1) \\ &= (0.45)(0.25) + (0.3)(0.75) = 0.34 \end{aligned}$$

In a similar way, $P(w1|y0) = 0.39$ is obtained. These values are then "passed" over to node x, and it is left as an exercise to show that $P(x0|w1) = 0.4$.

This idea can be carried out to any net of any size of the form given in Figure 2.29.

For Bayesian networks that have a tree structure, probability inference is achieved via a combination of downward and upward computations propagated through the tree. A number of algorithms have been proposed for the general case of Bayesian networks based on this "message-passing" philosophy. See, for example, [Pear 88, Laur 96]. For the case of singly connected graphs, these algorithms have complexity that is linear in the number of nodes. A singly connected graph is one that has no more than one path between any two nodes. For example, the graph in Figure 2.27 is not singly connected since there are two paths connecting x_1 and x_6. An alternative approach to derive efficient algorithms for probability inference, which exploits the structure of the DAG, has been taken in [Li 94]. Although it is beyond our scope to focus on algorithmic details, it is quite instructive to highlight the basic idea around which this type of algorithm evolves.

Let us take as an example the DAG shown in Figure 2.30, with nodes corresponding to the variables s, u, v, x, y, w, z with the joint probability $P(s, u, v, x, y, w, z)$. This can be obtained, as we have already stated, as the product of all conditional

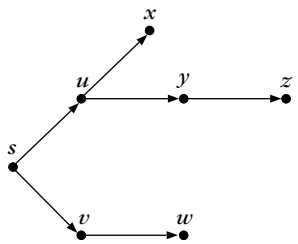

FIGURE 2.30

A Bayesian network with a tree structure.

probabilities defining the network. Suppose one wishes to compute the conditional probability $P(s|z = z_0)$, where $z = z_0$ is the evidence. From the Bayes rule we have

$$P(s|z = z_0) = \frac{P(s, z = z_0)}{P(z = z_0)} = \frac{P(s, z = z_0)}{\sum_s P(s, z = z_0)} \tag{2.138}$$

To obtain $P(s, z = z_0)$, one has to marginalize (Appendix A) the joint probability over all possible values of u, v, x, y, w; that is,

$$P(s, z = z_0) = \sum_{u,v,x,y,w} P(s, u, v, x, y, w, z = z_0) \tag{2.139}$$

Assuming, for simplicity, that each of the discrete variables can take, say, L values, the complexity of the previous computations amounts to L^5 operations. For more variables and a large number of values, L, this can be a prohibitively large number. Let us now exploit the structure of the Bayesian network in order to reduce this computational burden. Taking into account the relations implied by the topology of the graph shown in Figure 2.30 and the Bayes chain rule in (2.128) (for probabilities), we obtain

$$\sum_{u,v,x,y,w} P(s, u, v, x, y, w, z = z_0) =$$

$$\sum_{u,v,x,y,w} P(s)P(u|s)P(v|s)P(w|v)P(x|u)P(y|u)P(z = z_0|y) =$$

$$P(s) \underbrace{\sum_{u,v} P(u|s)P(v|s) \underbrace{\sum_{w} P(w|v)}_{v} \underbrace{\sum_{x} P(x|u)}_{u} \underbrace{\sum_{y} P(y|u)P(z = z_0|y)}_{u}}_{s} \tag{2.140}$$

or

$$\sum_{u,v,x,y,w} P(s, u, v, x, y, w, z = z_0) = P(s) \sum_{u,v} P(u|s)P(v|s)\phi_1(v)\phi_2(u)\phi_3(u) \tag{2.141}$$

where the definitions of $\phi_i(\cdot)$, $i = 1, 2, 3$, are readily understood by inspection. Underbraces indicate what variable the result of each summation depends on. To obtain $\phi_3(u)$ for each value of u, one needs to perform L operations (products and summations). Hence, a total number of L^2 operations is needed to compute $\phi_3(u)$ for all possible values of u. This is also true for the $\phi_2(u)$, $\phi_1(v)$. Thus, the total number of operations required to compute (2.141) is, after the factorization, of the order of L^2, instead of the order of L^5 demanded for the brute-force computation in (2.139). This procedure could be viewed as an effort to decompose a "global" sum into products of "local" sums to make computations tractable. Each summation can be viewed as a processing stage that removes a variable and provides as output a function. The essence of the algorithm given in [Li 94] is to search for the factorization that requires the minimal number of operations. This algorithm also has linear complexity in the number of nodes for singly connected networks. In general, for multiply connected networks the probability inference problem is NP-hard [Coop 90]. In light of this result, one tries to seek approximate solutions, as in [Dagu 93].

Training: Training of a Bayesian network consists of two parts. The first is to learn the network topology. The topology can either be fixed by an expert who can provide knowledge about dependencies or by use of optimization techniques based on the training set. Once the topology has been fixed, the unknown parameters (i.e., conditional probabilities and marginal probabilities) are estimated from the available training data points. For example, the fraction (frequency) of the number of instances that an event occurs over the total number of trials performed is a way to approximate probabilities. In Bayesian networks, other refined techniques are usually encountered. A review of learning procedures can be found in [Heck 95]. For the reader who wishes to delve further into the exciting world of Bayesian networks, the books of [Pear 88, Neap 04, Jens 01] will prove indispensable tools.

2.8 PROBLEMS

2.1 Show that in a multiclass classification task, the Bayes decision rule minimizes the error probability.

Hint: It is easier to work with the probability of correct decision.

2.2 In a two-class one-dimensional problem, the pdfs are the Gaussians $\mathcal{N}(0, \sigma^2)$ and $\mathcal{N}(1, \sigma^2)$ for the two classes, respectively. Show that the threshold x_0 minimizing the average risk is equal to

$$x_0 = 1/2 - \sigma^2 \ln \frac{\lambda_{21} P(\omega_2)}{\lambda_{12} P(\omega_1)}$$

where $\lambda_{11} = \lambda_{22} = 0$ has been assumed.

2.3 Consider a two equiprobable class problem with a loss matrix L. Show that if ϵ_1 is the probability of error corresponding to feature vectors from class ω_1

and ϵ_2 for those from class ω_2, then the average risk r is given by

$$r = P(\omega_1)\lambda_{11} + P(\omega_2)\lambda_{22} + P(\omega_1)(\lambda_{12} - \lambda_{11})\epsilon_1 + P(\omega_2)(\lambda_{21} - \lambda_{22})\epsilon_2$$

2.4 Show that in a multiclass problem with M classes the probability of classification error for the optimum classifier is bounded by

$$P_e \leq \frac{M-1}{M}$$

Hint: Show first that for each $\boldsymbol{x}$ the maximum of $P(\omega_i|\boldsymbol{x}), i = 1, 2, \ldots, M$, is greater than or equal to $1/M$. Equality holds if all $P(\omega_i|\boldsymbol{x})$ are equal.

2.5 Consider a two (equiprobable) class, one-dimensional problem with samples distributed according to the Rayleigh pdf in each class, that is,

$$p(x|\omega_i) = \begin{cases} \frac{x}{\sigma_i^2} \exp\left(\frac{-x^2}{2\sigma_i^2}\right) & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Compute the decision boundary point $g(x) = 0$.

2.6 In a two-class classification task, we constrain the error probability for one of the classes to be fixed, that is, $\epsilon_1 = \epsilon$. Then show that minimizing the error probability of the other class results in the likelihood test

$$\text{decide } \boldsymbol{x} \text{ in } \omega_1 \text{ if } \frac{P(\omega_1|\boldsymbol{x})}{P(\omega_2|\boldsymbol{x})} > \theta$$

where θ is chosen so that the constraint is fulfilled. This is known as the *Neyman–Pearson test*, and it is similar to the Bayesian minimum risk rule.

Hint: Use a Lagrange multiplier to show that this problem is equivalent to minimizing the quantity

$$q = \theta(\epsilon_1 - \epsilon) + \epsilon_2$$

2.7 In a three-class, two-dimensional problem the feature vectors in each class are normally distributed with covariance matrix

$$\Sigma = \begin{bmatrix} 1.2 & 0.4 \\ 0.4 & 1.8 \end{bmatrix}$$

The mean vectors for each class are $[0.1, 0.1]^T, [2.1, 1.9]^T, [-1.5, 2.0]^T$. Assuming that the classes are equiprobable, (a) classify the feature vector $[1.6, 1.5]^T$ according to the Bayes minimum error probability classifier; (b) draw the curves of equal Mahalanobis distance from $[2.1, 1.9]^T$.

2.8 In a two-class, three-dimensional classification problem, the feature vectors in each class are normally distributed with covariance matrix

$$\Sigma = \begin{bmatrix} 0.3 & 0.1 & 0.1 \\ 0.1 & 0.3 & -0.1 \\ 0.1 & -0.1 & 0.3 \end{bmatrix}$$

The respective mean vectors are $[0,0,0]^T$ and $[0.5,0.5,0.5]^T$. Derive the corresponding linear discriminant functions and the equation describing the decision surface.

2.9 In a two equiprobable class classification problem, the feature vectors in each class are normally distributed with covariance matrix Σ, and the corresponding mean vectors are $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$. Show that for the Bayesian minimum error classifier, the error probability is given by

$$P_B = \int_{(1/2)d_m}^{+\infty} \frac{1}{\sqrt{2\pi}} \exp(-z^2/2)\, dz$$

where d_m is the Mahalanobis distance between the mean vectors. Observe that this is a decreasing function of d_m.
Hint: Compute the log-likelihood ratio $u = \ln p(\boldsymbol{x}|\omega_1) - \ln p(\boldsymbol{x}|\omega_2)$. Observe that u is also a random variable normally distributed as $\mathcal{N}((1/2)d_m^2, d_m^2)$ if $\boldsymbol{x} \in \omega_1$ and as $\mathcal{N}(-(1/2)d_m^2, d_m^2)$ if $\boldsymbol{x} \in \omega_2$. Use this information to compute the error probability.

2.10 Show that in the case in which the feature vectors follow Gaussian pdfs, the likelihood ratio test in (2.20)

$$\boldsymbol{x} \in \omega_1(\omega_2) \quad \text{if} \quad l_{12} \equiv \frac{p(\boldsymbol{x}|\omega_1)}{p(\boldsymbol{x}|\omega_2)} > (<)\theta$$

is equivalent to

$$d_m^2(\boldsymbol{\mu}_1, \boldsymbol{x}|\Sigma_1) - d_m^2(\boldsymbol{\mu}_2, \boldsymbol{x}|\Sigma_2) + \ln \frac{|\Sigma_1|}{|\Sigma_2|} < (>) - 2\ln\theta$$

where $d_m(\boldsymbol{\mu}_i, \boldsymbol{x}|\Sigma_i)$ is the Mahalanobis distance between $\boldsymbol{\mu}_i$ and $\boldsymbol{x}$ with respect to the Σ_i^{-1} norm.

2.11 If $\Sigma_1 = \Sigma_2 = \Sigma$, show that the criterion of the previous problem becomes

$$(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \Sigma^{-1} \boldsymbol{x} > (<) \Theta$$

where

$$\Theta = \ln\theta + 1/2(\|\boldsymbol{\mu}_1\|_{\Sigma^{-1}} - \|\boldsymbol{\mu}_2\|_{\Sigma^{-1}})$$

2.12 Consider a two-class, two-dimensional classification task, where the feature vectors in each of the classes ω_1, ω_2 are distributed according to

$$p(\boldsymbol{x}|\omega_1) = \frac{1}{\left(\sqrt{2\pi\sigma_1^2}\right)^2} \exp\left(-\frac{1}{2\sigma_1^2}(\boldsymbol{x} - \boldsymbol{\mu}_1)^T(\boldsymbol{x} - \boldsymbol{\mu}_1)\right)$$

$$p(\boldsymbol{x}|\omega_2) = \frac{1}{\left(\sqrt{2\pi\sigma_2^2}\right)^2} \exp\left(-\frac{1}{2\sigma_2^2}(\boldsymbol{x} - \boldsymbol{\mu}_2)^T(\boldsymbol{x} - \boldsymbol{\mu}_2)\right)$$

with

$$\boldsymbol{\mu}_1 = [1, 1]^T, \quad \boldsymbol{\mu}_2 = [1.5, 1.5]^T, \quad \sigma_1^2 = \sigma_2^2 = 0.2$$

Assume that $P(\omega_1) = P(\omega_2)$ and design a Bayesian classifier

(a) that minimizes the error probability

(b) that minimizes the average risk with loss matrix

$$\Lambda = \begin{bmatrix} 0 & 1 \\ 0.5 & 0 \end{bmatrix}$$

Using a pseudorandom number generator, produce 100 feature vectors from each class, according to the preceding pdfs. Use the classifiers designed to classify the generated vectors. What is the percentage error for each case? Repeat the experiments for $\boldsymbol{\mu}_2 = [3.0, 3.0]^T$.

2.13 Repeat the preceding experiment if the feature vectors are distributed according to

$$p(\boldsymbol{x}|\omega_i) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i)\Big)$$

with

$$\Sigma = \begin{bmatrix} 1.01 & 0.2 \\ 0.2 & 1.01 \end{bmatrix}$$

and $\boldsymbol{\mu}_1 = [1, 1]^T, \boldsymbol{\mu}_2 = [1.5, 1.5]^T$.

Hint: To generate the vectors, recall from [Papo 91, p. 144] that a linear transformation of Gaussian random vectors also results in Gaussian vectors. Note also that

$$\begin{bmatrix} 1.01 & 0.2 \\ 0.2 & 1.01 \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0.1 \\ 0.1 & 1 \end{bmatrix}$$

2.14 Consider a two-class problem with normally distributed vectors with the same Σ in both classes. Show that the decision hyperplane at the point $\boldsymbol{x}_0$, Eq. (2.46), is tangent to the constant Mahalanobis distance hyperellipsoids.

Hint: (a) Compute the gradient of Mahalanobis distance with respect to $\boldsymbol{x}$. (b) Recall from vector analysis that $\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$ is normal to the tangent of the surface $f(\boldsymbol{x}) =$ constant.

2.15 Consider a two-class, one-dimensional problem with $p(x|\omega_1)$ being $\mathcal{N}(\mu, \sigma^2)$ and $p(x|\omega_2)$ a uniform distribution between a and b. Show that the Bayesian error probability is bounded by $G\left(\frac{b-\mu}{\sigma}\right) - G\left(\frac{a-\mu}{\sigma}\right)$, where $G(x) \equiv P(y \leq x)$ and y is $\mathcal{N}(0, 1)$.

2.16 Show that the mean value of the random vector $\frac{\partial \ln(p(\boldsymbol{x};\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}$ is zero.

2.17 In a heads or tails coin-tossing experiment the probability of occurrence of a head (1) is q and that of a tail (0) is $1 - q$. Let $x_i, i = 1, 2, \ldots, N$, be the resulting experimental outcomes, $x_i \in \{0, 1\}$. Show that the ML estimate of q is

$$q_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Hint: The likelihood function is

$$P(X : q) = \prod_{i=1}^{N} q^{x_i}(1 - q)^{(1-x_i)}$$

Then show that the ML results from the solution of the equation

$$q^{\sum_i x_i}(1 - q)^{(N-\sum_i x_i)}\left(\frac{\sum_i x_i}{q} - \frac{N - \sum_i x_i}{1 - q}\right) = 0$$

2.18 The random variable x is normally distributed $\mathcal{N}(\mu, \sigma^2)$, where μ is considered unknown. Given N measurements of the variable, compute the Cramer–Rao bound $-E\left[\frac{\partial^2 L(\mu)}{\partial^2 \mu}\right]$ (Appendix A). Compare the bound with the variance of the resulting ML estimate of μ. Repeat this if the unknown parameter is the variance σ^2. Comment on the results.

2.19 Show that if the likelihood function is Gaussian with unknowns the mean $\boldsymbol{\mu}$ as well as the covariance matrix Σ, then the ML estimates are given by

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_k$$

$$\hat{\Sigma} = \frac{1}{N}\sum_{k=1}^{N}(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})^T$$

2.20 Prove that the covariance estimate

$$\hat{\Sigma} = \frac{1}{N-1}\sum_{k=1}^{N}(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})^T$$

is an unbiased one, where

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_k$$

2.21 Prove that the ML estimates of the mean value and the covariance matrix (Problem 2.19) can be computed recursively, that is,

$$\hat{\boldsymbol{\mu}}_{N+1} = \hat{\boldsymbol{\mu}}_N + \frac{1}{N+1}(\boldsymbol{x}_{N+1} - \hat{\boldsymbol{\mu}}_N)$$

and

$$\hat{\Sigma}_{N+1} = \frac{N}{N+1}\hat{\Sigma}_N + \frac{N}{(N+1)^2}(\boldsymbol{x}_{N+1} - \hat{\boldsymbol{\mu}}_N)(\boldsymbol{x}_{N+1} - \hat{\boldsymbol{\mu}}_N)^T$$

where the subscript in the notation of the estimates, $\hat{\boldsymbol{\mu}}_N$, $\hat{\Sigma}_N$ indicates the number of samples used for their computation.

2.22 The random variable x follows the Erlang pdf

$$p(x;\theta) = \theta^2 x \exp(-\theta x)u(x)$$

where $u(x)$ is the unit-step function,

$$u(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Show that the maximum likelihood estimate of θ, given N measurements, $x_1, \ldots, x_N$, of x, is

$$\hat{\theta}_{ML} = \frac{2N}{\sum_{k=1}^{N} x_k}$$

2.23 In the ML estimation, the zero of the derivative of the log pdf derivative was computed. Using a multivariate Gaussian pdf, show that this corresponds to a maximum and not to a minimum.

2.24 Prove that the sum $z = x + y$ of two independent random variables, x and y, where $x \sim \mathcal{N}(\mu_x, \sigma_x^2)$ and $y \sim \mathcal{N}(\mu_y, \sigma_y^2)$, is also a Gaussian one with mean value and variance equal to $\mu_x + \mu_y$ and $\sigma_x^2 + \sigma_y^2$, respectively.

2.25 Show relations (2.74) and (2.75). Then show that $p(x|X)$ is also normal with mean μ_N and variance $\sigma^2 + \sigma_N^2$. Comment on the result.

2.26 Show that the posterior pdf estimate in the Bayesian inference task, for independent variables, can be computed recursively, that is,

$$p(\boldsymbol{\theta}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \frac{p(\boldsymbol{x}_N|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-1})}{p(\boldsymbol{x}_N|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N-1})}$$

2.27 Show Eqs. (2.76)–(2.79).

2.28 The random variable x is normally distributed as $\mathcal{N}(\mu, \sigma^2)$, with μ being the unknown parameter described by the Rayleigh pdf

$$p(\mu) = \frac{\mu \exp(-\mu^2/2\sigma_\mu^2)}{\sigma_\mu^2}$$

Show that the maximum *a posteriori* probability estimate of μ is given by

$$\hat{\mu}_{MAP} = \frac{Z}{2R}\left(1 + \sqrt{1 + \frac{4R}{Z^2}}\right)$$

where

$$Z = \frac{1}{\sigma^2}\sum_{k=1}^{N} x_k, \quad R = \frac{N}{\sigma^2} + \frac{1}{\sigma_\mu^2}$$

2.29 Show that for the lognormal distribution

$$p(x) = \frac{1}{\sigma x\sqrt{2\pi}} \exp\left(-\frac{(\ln x - \theta)^2}{2\sigma^2}\right), \quad x > 0$$

the ML estimate is given by

$$\hat{\theta}_{ML} = \frac{1}{N}\sum_{k=1}^{N} \ln x_k$$

2.30 Show that if the mean value and the variance of a random variable are known, that is,

$$\mu = \int_{-\infty}^{+\infty} xp(x)\, dx, \quad \sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 p(x)\, dx$$

the maximum entropy estimate of the pdf is the Gaussian $\mathcal{N}(\mu, \sigma^2)$.

2.31 Show Eqs. (2.98), (2.99), and (2.100).
Hint: For the latter, note that the probabilities add to one; thus a Lagrangian multiplier must be used.

2.32 Let P be the probability of a random point x being located in a certain interval h. Given N of these points, the probability of having k of them inside h is given by the binomial distribution

$$\text{prob}\{k\} = \frac{N!}{k!(N-k)!} P^k (1-P)^{N-k}$$

Show that $E[k/N] = P$ and that the variance around the mean is $\sigma^2 = E[(k/N - P)^2] = P(1-P)/N$. That is, the probability estimator $P = k/N$ is unbiased and asymptotically consistent.

2.33 Consider three Gaussian pdfs: $\mathcal{N}(1.0, 0.1), \mathcal{N}(3.0, 0.1)$, and $\mathcal{N}(2.0, 0.2)$. Generate 500 samples according to the following rule. The first two samples are generated from the second Gaussian, the third sample from the first one, and the fourth sample from the last Gaussian. This rule repeats until all 500 samples have been generated. The pdf underlying the random samples is modeled as a mixture

$$\sum_{i=1}^{3} \mathcal{N}(\mu_i, \sigma_i^2) P_i$$

Use the EM algorithm and the generated samples to estimate the unknown parameters μ_i, σ_i^2, P_i.

2.34 Consider two classes ω_1, ω_2 in the two-dimensional space. The data from class ω_1 are uniformly distributed inside a circle of radius r. The data of class ω_2 are also uniformly distributed inside another circle of radius r. The distance between the centers of the circles is greater than $4r$. Let N be the number of the available training samples. Show that the probability of error of the NN classifier is always smaller than that of the kNN, for any $k \geq 3$.

2.35 Generate 50 feature vectors for each of the two classes of Problem 2.12, and use them as training points. In the sequel, generate 100 vectors from each class and classify them according to the NN and 3NN rules. Compute the classification error percentages.

2.36 The pdf of a random variable is given by

$$p(x) = \begin{cases} \frac{1}{2} & \text{for } 0 < x < 2 \\ 0 & \text{otherwise} \end{cases}$$

Use the Parzen window method to approximate it using as the kernel function the Gaussian $\mathcal{N}(0, 1)$. Choose the smoothing parameter to be (a) $h = 0.05$ and (b) $h = 0.2$. For each case, plot the approximation based on $N = 32, N = 256$, and $N = 5000$ points, which are generated from a pseudorandom generator according to $p(x)$.

2.37 Repeat the preceding problem by generating $N = 5000$ points and using k nearest neighbor estimation with $k = 32, 64$, and 256, respectively.

2.38 Show that the variance $\sigma_N^2(\boldsymbol{x})$ of the pdf estimate, given by Eq. (2.110), is upper bounded by:

$$\sigma_N^2(\boldsymbol{x}) \leq \frac{\sup(\phi) E[\hat{p}(\boldsymbol{x})]}{N h^l}$$

where $\sup(\cdot)$ is the supremum of the associated function. Observe that for large values of h the variance is small. On the other hand, we can make the variance small for small values of h, provided N tends to infinity and if, also, the product Nh^l tends to infinity.

2.39 Recall Equation (2.128)

$$p(\boldsymbol{x}) = p(x_1) \prod_{i=2}^{l} p(x_i | A_i)$$

Assume $l = 6$ and

$$p(x_6 | x_5, \ldots, x_1) = p(x_6 | x_5, x_1) \tag{2.142}$$

$$p(x_5 | x_4, \ldots, x_1) = p(x_5 | x_4, x_3) \tag{2.143}$$

$$p(x_4 | x_3, x_2, x_1) = p(x_4 | x_3, x_2, x_1) \tag{2.144}$$

$$p(x_3 | x_2, x_1) = p(x_3) \tag{2.145}$$

$$p(x_2|x_1) = p(x_2) \tag{2.146}$$

Write the respective sets A_i, $i = 1, 2, \ldots, 6$, and construct the corresponding DAG.

2.40 In the DAG defined in Figure 2.29, assume that the variable z is measured to be $z0$. Compute $P(x1|z0)$ and $P(w0|z0)$.

2.41 In the example associated with the tree-structured DAG of Figure 2.28, assume that the patient undergoes the medical test H_1 and that this turns out to be positive (True). Based on this test, compute the probability that the patient has developed cancer. In other words, compute the conditional probability $P(C = \text{True}|H_1 = \text{True})$.

MATLAB PROGRAMS AND EXERCISES

Computer Exercises

A number of MATLAB functions are provided, which will help the interested reader to experiment on some of the most important issues discussed in the present chapter. Needless to say that there may be other implementations of these functions. Short comments are also given along with the code. In addition, we have used the symbols m and S to denote the mean vector (given as a column vector) and the covariance matrix, respectively, instead of the symbols $\boldsymbol{\mu}$ and Σ, which are used in the text. In the following, unless otherwise stated, each class is represented by an integer in $\{1, \ldots, c\}$ where c is the number of classes.

2.1 *Gaussian generator.* Generate N l-dimensional vectors from a Gaussian distribution with mean m and covariance matrix S, using the *mvnrnd* MATLAB function.

Solution

Just type

```
mvnrnd(m,S,N)
```

2.2 *Gaussian function evaluation.* Write a MATLAB function that computes the value of the Gaussian distribution $\mathcal{N}(m, S)$, at a given vector x.

Solution

```
function z=comp_gauss_dens_val(m,S,x)
  [l,q]=size(m);  % l=dimensionality
  z=(1/((2*pi)^(l/2)*det(S)^0.5))...
      *exp(-0.5*(x-m)'*inv(S)*(x-m));
```

2.3 *Data set generation from Gaussian classes.* Write a MATLAB function that generates a data set of Nl-dimensional vectors that stem from c different Gaussian distributions $\mathcal{N}(m_i, S_i)$, with corresponding *a priori* probabilities $P_i, i = 1, \ldots, c$.

Solution

In the sequel:

- m is an $l \times c$ matrix, the i-th column of which is the mean vector of the i-th class distribution.
- S is an $l \times l \times c$ (three-dimensional) matrix, whose ith two-dimensional $l \times l$ component is the covariance of the distribution of the ith class. In MATLAB $S(:, :, i)$ denotes the i-th two-dimensional $l \times l$ matrix of S.
- P is the c dimensional vector that contains the *a priori* probabilities of the classes. m_i, S_i, P_i, and c are provided as inputs.

The following function returns:

- A matrix X with (approximately) N columns, each column of which is an l-dimensional data vector.
- A row vector y whose ith entry denotes the class from which the ith data vector stems.

```
function [X,y]=generate_gauss_classes(m,S,P,N)
  [l,c]=size(m);
  X=[];
  y=[];
  for j=1:c
  % Generating the [p(j)*N)] vectors from each distribution
    t=mvnrnd(m(:,j),S(:,:,j),fix(P(j)*N));
    % The total number of points may be slightly less than N
    % due to the fix operator
    X=[X t];
    y=[y ones(1,fix(P(j)*N))*j];
  end
```

2.4 *Plot of data.* Write a MATLAB function that takes as inputs: (a) a matrix X and a vector y defined as in the previous function, (b) the mean vectors of c class distributions. It plots: (a) the data vectors of X using a different color for each class, (b) the mean vectors of the class distributions. It is assumed that the data live in the two-dimensional space.

Solution

```
% CAUTION: This function can handle up to
% six different classes
```

```
function plot_data(X,y,m)
  [l,N]=size(X); % N=no. of data vectors, l=dimensionality
  [l,c]=size(m); % c=no. of classes
  if(l ~ =2)
    fprintf('NO PLOT CAN BE GENERATED\n')
    return
  else
    pale=['r.'; 'g.'; 'b.'; 'y.'; 'm.'; 'c.'];
    figure(1)
    % Plot of the data vectors
    hold on
    for i=1:N
      plot(X(1,i),X(2,i),pale(y(i),:))
    end
    % Plot of the class means
    for j=1:c
    plot(m(1,j),m(2,j),'k+')
    end
  end
```

2.5 *Bayesian classifier (for Gaussian Processes).* Write a MATLAB function that will take as inputs: (a) the mean vectors, (b) the covariance matrices of the class distributions of a c-class problem, (c) the *a priori* probabilities of the c classes, and (d) a matrix X containing column vectors that stem from the above classes. It will give as output an N-dimensional vector whose ith component contains the class where the corresponding vector is assigned, according to the Bayesian classification rule.

Solution

Caution: While inserting the following function, **do not** type the labels (A), (B) and (C). They are used to serve as references, as we will see later on.

```
(A) function z=bayes_classifier(m,S,P,X)
  [l,c]=size(m); % l=dimensionality, c=no. of classes
  [l,N]=size(X); % N=no. of vectors
  for i=1:N
    for j=1:c
      (B) t(j)=P(j)*comp_gauss_dens_val(m(:,j),...
          S(:,:,j),X(:,i));
    end
    % Determining the maximum quantity Pi*p(x|wi)
    (C) [num,z(i)]=max(t);
end
```

2.6 *Euclidean distance classifier.* Write a MATLAB function that will take as inputs: (a) the mean vectors, and (b) a matrix X containing column vectors that stem from the above classes. It will give as output an N-dimensional vector whose ith component contains the class where the corresponding vector is assigned, according to the minimum Euclidean distance classifier.

Solution

The requested function may be obtained by the *bayes_classifier* function by replacing (A), (B), and (C) with

- `function z=euclidean_classifier(m,X)`
- `t(j)=sqrt((X(:,i)-m(:,j))'*(X(:,i)-m(:,j)));`

(computation of the Euclidean distances from all class representatives)

- `[num,z(i)]=min(t);`

(determination of the closest class mean),
respectively.

2.7 *Mahalanobis distance classifier.* Write a MATLAB function that will take as inputs: (a) the mean vectors, (b) the covariance matrix of the class distributions of a c-class problem, and (c) a matrix X containing column vectors that stem from the above classes. It will give as output an N-dimensional vector whose ith component contains the class where the corresponding vector is assigned according to the minimum Mahalanobis distance classifier.

Solution

The requested function may be obtained by the *bayes_classifier* function by replacing (A), (B) and (C) with

- `function z=mahalanobis_classifier(m,S,X)`
- `t(j)=sqrt((X(:,i)-m(:,j))'*inv(S(:,:,j))*...`
  `(X(:,i)-m(:,j)));`

(computation of the Mahalanobis distances from all class representatives)

- `[num,z(i)]=min(t);`

(determination of the closest class mean), respectively.

2.8 *k-nearest neighbor classifier.* Write a MATLAB function that takes as inputs: (a) a set of N_1 vectors packed as columns of a matrix Z, (b) an N_1-dimensional vector containing the classes where each vector in Z belongs, (c) the value for the parameter k of the classifier, (d) a set of N vectors packed as columns in the

matrix X. It returns an N-dimensional vector whose ith component contains the class where the corresponding vector of X is assigned, according to the k-nearest neighbor classifier.

Solution

```
function z=k_nn_classifier(Z,v,k,X)
  [l,N1]=size(Z);
  [l,N]=size(X);
  c=max(v); % The number of classes
  % Computation of the (squared) Euclidean distance
  % of a point from each reference vector
  for i=1:N
    dist=sum((X(:,i)*ones(1,N1)-Z).^2);
    %Sorting the above distances in ascending order
    [sorted,nearest]=sort(dist);
    % Counting the class occurrences among the k-closest
    % reference vectors Z(:,i)
    refe=zeros(1,c); %Counting the reference vectors per class
    for q=1:k
      class=v(nearest(q));
      refe(class)=refe(class)+1;
    end
    [val,z(i)]=max(refe);
  end
```

2.9 *Classification error evaluation.* Write a MATLAB function that will take as inputs: (a) an N-dimensional vector, each component of which contains the class where the corresponding data vector belongs and (b) a similar N-dimensional vector each component of which contains the class where the corresponding data vector is assigned from a certain classifier. Its output will be the percentage of the places where the two vectors differ (i.e., the classification error of the classifier).

Solution

```
function clas_error=compute_error(y,y_est)
  [q,N]=size(y);   % N= no. of vectors
  c=max(y);   % Determining the number of classes
  clas_error=0; % Counting the misclassified vectors
  for i=1:N
    if(y(i)~=y_est(i))
      clas_error=clas_error+1;
    end
  end
```

```
% Computing the classification error
clas_error=clas_error/N;
```

Computer Experiments

Notes: In the sequel, it is advisable to use the command

```
randn('seed',0)
```

before generating the data sets, in order to initialize the Gaussian random number generator to 0 (or any other fixed number). This is important for the reproducibility of the results.

2.1 a. Generate and plot a data set of $N = 1{,}000$ two-dimensional vectors that stem from three equiprobable classes modeled by normal distributions with mean vectors $m_1 = [1,\ 1]^T$, $m_2 = [7,\ 7]^T$, $m_3 = [15,\ 1]^T$ and covariance matrices $S_1 = \begin{bmatrix} 12 & 0 \\ 0 & 1 \end{bmatrix}, S_2 = \begin{bmatrix} 8 & 3 \\ 3 & 2 \end{bmatrix}, S_3 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$.

b. Repeat (a) when the *a priori* probabilities of the classes are given by the vector $P = [0.6,\ 0.3,\ 0.1]^T$.

Solution

Figure (2.31)a–b display the vectors from each class. Note the "shape" of the clusters formed by the vectors of each class. This is directly affected by the corresponding covariance matrix. Also note that, in the first case, each class has roughly the same number of the vectors, while in the latter case,

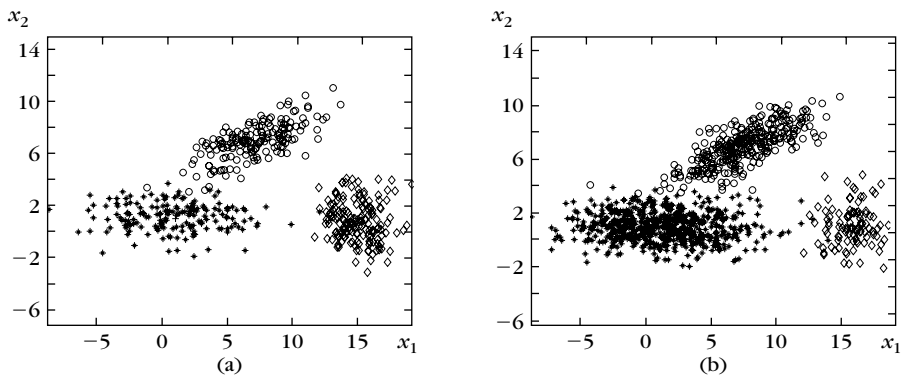

FIGURE 2.31

(a) The equiprobable classes case. (b) The case where the a-priori probabilities differ.

the leftmost and the rightmost classes are more "dense" and more "sparse" compared to the previous case, respectively.

2.2 a. Generate a data set X_1 of $N = 1{,}000$ two-dimensional vectors that stem from three equiprobable classes modeled by normal distributions with mean vectors $m_1 = [1, 1]^T$, $m_2 = [12, 8]^T$, $m_3 = [16,1]^T$ and covariance matrices $S_1 = S_2 = S_3 = 4I$, where I is the 2×2 identity matrix.

b. Apply the Bayesian, the Euclidean, and the Mahalanobis classifiers on X_1.

c. Compute the classification error for each classifier.

2.3 a. Generate a data set X_2 of $N = 1{,}000$ two-dimensional vectors that stem from three equiprobable classes modeled by normal distributions with mean vectors $m_1 = [1, 1]^T$, $m_2 = [14, 7]^T$, $m_3 = [16, 1]^T$ and covariance matrices $S_1 = S_2 = S_3 = \begin{bmatrix} 5 & 3 \\ 3 & 4 \end{bmatrix}$.

(b)–(c) Repeat steps b) and (c) of experiment 2.2, for X_2.

2.4 a. Generate a data set X_3 of $N = 1{,}000$ two-dimensional vectors that stem from three equiprobable classes modeled by normal distributions with mean vectors $m_1 = [1, 1]^T, m_2 = [8, 6]^T, m_3 = [13, 1]^T$ and covariance matrices $S_1 = S_2 = S_3 = 6I$, where I is the 2×2 identity matrix.

(b)–(c) Repeat (b) and (c) from experiment 2.2, for X_3.

2.5 a. Generate a data set X_4 of $N = 1{,}000$ two-dimensional vectors that stem from three equiprobable classes modeled by normal distributions with mean vectors $m_1 = [1, 1]^T, m_2 = [10, 5]^T, m_3 = [11, 1]^T$ and covariance matrices $S_1 = S_2 = S_3 = \begin{bmatrix} 7 & 4 \\ 4 & 5 \end{bmatrix}$.

(b)–(c) Repeat steps (b) and (c) of experiment 2.2, for X_4.

2.6 Study carefully the results obtained by experiments (2.2)–(2.5) and draw your conclusions.

2.7 a. Generate two data sets X_5 and X_5' of $N = 1{,}000$ two-dimensional vectors each that stem from three classes modeled by normal distributions with mean vectors $m_1 = [1,\ 1]^T, m_2 = [4,\ 4]^T, m_3 = [8,\ 1]^T$ and covariance matrices $S_1 = S_2 = S_3 = 2I$. In the generation of X_5, the classes are assumed to be equiprobable, while in the generation of X_5', the *a priori* probabilities of the classes are given by the vector $P = [0.8,\ 0.1,\ 0.1]^T$.

b. Apply the Bayesian and the Euclidean classifiers on both X_5 and X_5'.

c. Compute the classification error for each classifier for both data sets and draw your conclusions.

2.8 Consider the data set X_3 (from experiment (2.4)). Using the same settings, generate a data set Z, where the class from which a data vector stems is known. Apply the k nearest neighbor classifier on X_3 for $k = 1$ and $k = 11$ using Z as the training set and draw your conclusions.

REFERENCES

[Aebe 94] Aeberhard S., Coomans D., Devel O. "Comparative analysis of statistical pattern recognition methods in high dimensional setting," *Pattern Recognition*, Vol. 27(8), pp. 1065–1077, 1994.

[Babi 96] Babich G.A., Camps O.I. "Weighted Parzen windows for pattern classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18(5), pp. 567–570, 1996.

[Bell 61] Bellman R. *Adaptive Control Processes: A Guided Tour*, Princeton University Press, 1961.

[Bern 94] Bernardo J.M., Smith A.F.M *Bayesian Theory*, John Wiley, 1994.

[Bish 06] Bishop C.M. *Pattern Recognition and Machine Learning*, Springer, 2006.

[Boyl 83] Boyles R.A. "On the convergence of the EM algorithm," *J. Royal Statistical Society B*, Vol. 45(1), pp. 47–55, 1983.

[Brei 77] Breiman L., Meisel W., Purcell E. "Variable kernel estimates of multivariate densities," *Technometrics*, Vol. 19(2), pp. 135–144, 1977.

[Brod 90] Broder A. "Strategies for efficient incremental nearest neighbor search," *Pattern Recognition*, Vol. 23, pp. 171–178, 1990.

[Butu 93] Buturovic L.J. "Improving k-nearest neighbor density and error estimates," *Pattern Recognition*, Vol. 26(4), pp. 611–616, 1993.

[Coop 90] Cooper G.F. "The computational complexity of probabilistic inference using Bayesian belief networks," *Artifical Intelligence*, Vol. 42, pp. 393–405, 1990.

[Cram 46] Cramer H. *Mathematical Methods of Statistics*, Princeton University Press, 1941.

[Dagu 93] Dagum P., Chavez R.M. "Approximating probabilistic inference in Bayesian belief networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15(3), pp. 246–255, 1993.

[Dasa 91] Dasarasthy B. *Nearest Neighbor Pattern Classification Techniques*, IEEE Computer Society Press, 1991.

[Demp 77] Dempster A.P., Laird N.M., Rubin D.B. "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Statistical Society*, Vol. 39(1), pp. 1–38, 1977.

[Devr 96] Devroye L., Gyorfi L., Lugosi G. *A Probabilistic Theory of Pattern Recognition*, Springer-Verlag, 1996.

[Djou 97] Djouadi A., Bouktache E. "A fast algorithm for the nearest neighbor classifier," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19(3), pp. 277–282, 1997.

[Dome 05] Domeniconi C., Gunopoulos D., Peng J. "Large margin nearest neighbor classifiers," *IEEE Transactions on Neural Networks*, Vol. 16(4), pp. 899–909, 2005.

[Domi 97] Domingos P., Pazzani M. "Beyond independence: Conditions for the optimality of the simple Bayesian classifier," *Machine Learning*, Vol. 29, pp. 103–130, 1997.

[Duda 73] Duda R., Hart P.E. *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.

[Frie 94] Friedman J.H. "Flexible metric nearest neighbor classification," *Technical Report, Department of Statistics, Stanford University*, 1994.

[Frie 89] Friedman J.H. "Regularized discriminant analysis," *Journal of American Statistical Association*, Vol. 84(405), pp. 165–175, 1989.

[Frie 97] Friedman N., Geiger D., Goldszmidt M. "Bayesian network classifiers," *Machine Learning*, Vol. 29, pp. 131–163, 1997.

[Fuku 75] Fukunaga F., Narendra P.M. "A branch and bound algorithm for computing *k*-nearest neighbors," *IEEE Transactions on Computers*, Vol. 24, pp. 750–753, 1975.

[Fuku 90] Fukunaga F. *Introduction to Statistical Pattern Recognition*, 2nd ed., Academic Press, 1990.

[Hast 96] Hastie T., Tibshirani R. "Discriminant adaptive nearest neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18(6), pp. 607–616, 1996.

[Hast 01] Hastie T., Tibshirani R., Friedman J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer, 2001.

[Hatt 00] Hattori K., Takahashi M. "A new edited *k*-nearest neighbor rule in the pattern classification problem," *Pattern Recognition*, Vol. 33, pp. 521–528, 2000.

[Heck 95] Heckerman D. "A tutorial on learning Bayesian networks," *Technical Report #MSR-TR-95-06*, Microsoft Research, Redmond, Washington, 1995.

[Hoff 96] Hoffbeck J.P., Landgrebe D.A. "Covariance matrix estimation and classification with limited training data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18(7), pp. 763–767, 1996.

[Huan 02] Huang Y.S., Chiang C.C., Shieh J.W., Grimson E. "Prototype optimization for nearest-neighbor classification," *Pattern Recognition*, Vol. 35, pp. 1237–1245, 2002.

[Jayn 82] Jaynes E.T. "On the rationale of the maximum entropy methods," *Proceedings of the IEEE*, Vol. 70(9), pp. 939–952, 1982.

[Jens 01] Jensen F.V. *Bayesian Networks and Decision Graphs*, Springer, 2001.

[Jone 96] Jones M.C., Marron J.S., Seather S.J. "A brief survey of bandwidth selection for density estimation," *Journal of the American Statistical Association*, Vol. 91, pp. 401–407, 1996.

[Kimu 87] Kimura F., Takashina K., Tsuruoka S., Miyake Y. "Modified quadratic discriminant functions and the application to Chinese character recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 9(1), pp. 149–153, 1987.

[Kris 00] Krishna K., Thathachar M.A.L., Ramakrishnan K.R. "Voronoi networks and their probability of misclassification," *IEEE Transactions on Neural Networks*, Vol. 11(6), pp. 1361–1372, 2000.

[Krzy 83] Krzyzak A. "Classification procedures using multivariate variable kernel density estimate," *Pattern Recognition Letters*, Vol. 1, pp. 293–298, 1983.

[Laur 96] Lauritzen S.L. *Graphical Models*, Oxford University Press, 1996.

[Li 94] Li Z., D'Abrosio B. "Efficient inference in Bayes' networks as a combinatorial optimization problem," *International Journal of Approximate Inference*, Vol. 11, 1994.

[Liu 04] Liu C.-L., Sako H., Fusisawa H. "Discriminative learning quadratic discriminant function for handwriting recognition," *IEEE Transactions on Neural Networks*, Vol. 15(2), pp. 430–444, 2004.

[McLa 88] McLachlan G.J., Basford K.A. *Mixture Models: Inference and Applications to Clustering*, Marcel Dekker, 1988.

[McNa 01] McNames J. "A Fast nearest neighbor algorithm based on principal axis search tree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23(9), pp. 964–976, 2001.

[Mico 94] Mico M.L., Oncina J., Vidal E. "A new version of the nearest neighbor approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements," *Pattern Recognition Letters*, Vol. 15, pp. 9–17, 1994.

[Moon 96] Moon T. "The expectation maximization algorithm," *Signal Processing Magazine*, Vol. 13(6), pp. 47–60, 1996.

[Neap 04] Neapolitan R.D. *Learning Bayesian Networks*, Prentice Hall, 2004.

[Nene 97] Nene S.A., Nayar S.K. "A simple algorithm for nearest neighbor search in high dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19(9), pp. 989–1003, 1997.

[Papo 91] Papoulis A. *Probability Random Variables and Stochastic Processes*, 3rd ed., McGraw-Hill 1991.

[Pare 06] Paredes R., Vidal E. "Learning weighted metrics to minimize nearest neighbor classification error," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28(7), pp. 1100–1111, 2006.

[Pare 06a] Paredes R., Vidal E. "Learning prototypes and distances: A prototype reduction technique based on nearest neighbor error minimization," *Pattern Recognition*, Vol. 39, pp. 180–188, 2006.

[Parz 62] Parzen E. "On the estimation of a probability density function and mode," *Ann. Math. Stat.* Vol. 33, pp. 1065–1076, 1962.

[Pear 88] Pearl J. *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, 1988.

[Redn 84] Redner R.A., Walker H.F. "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Review*, Vol. 26(2), pp. 195–239, 1984.

[Roos 05] Roos T., Wettig H., Grunwald P., Myllymaki P., Tirri H. "On discriminative Bayesian network classifiers and logistic regression," *Machine Learning*, Vol. 59, pp. 267–296, 2005.

[Same 08] Samet H. "*k*-Nearest neighbor finding using MaxNearestDist," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30(2), pp. 243–252, 2008.

[Terr 92] Terrell G.R., Scott D.W. "Variable kernel density estimation," *Annals of Statistics*, Vol. 20(3), pp. 1236–1265, 1992.

[Titt 85] Titterington D.M., Smith A.F.M., Makov U.A. *Statistical Analysis of Finite Mixture Distributions*, John Wiley & Sons, 1985.

[Vida 94] Vidal E. "New formulation and improvements of the nearest neighbor approximating and eliminating search algorithm (AESA)," *Pattern Recognition Letters*, Vol. 15, pp. 1–7, 1994.

[Wand 95] Wand M., Jones M. *Kernel Smoothing*, Chapman & Hall, London, 1995.

[Webb 05] Webb G.I., Boughton J.R., Wang Z. "Not so naive Bayes: Aggregating one dependence estimators," *Machine Learning*, Vol. 58, pp. 5–24, 2005.

[Wils 72] Wilson D.L. "Asymptotic properties of NN rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 2, pp. 408–421, 1972.

[Wu 83] Wu C. "On the convergence properties of the EM algorithm," *Annals of Statistics*, Vol. 11(1), pp. 95–103, 1983.

[Yan 93] Yan H. "Prototype optimization for nearest neighbor classifiers using a two layer perceptron," *Pattern Recognition*, Vol. 26(2), pp. 317–324, 1993.

This page intentionally left blank