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

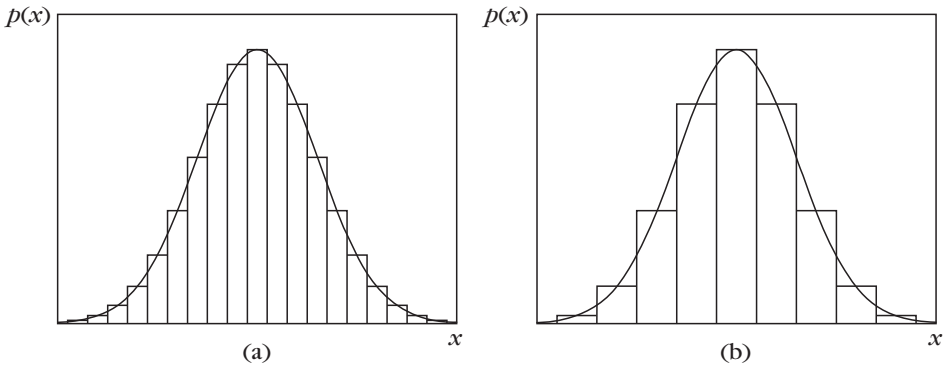

FIGURE 2.18

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 $\mathcal{N}(\boldsymbol{0}, I)$, kernel. For such a choice, the approximate expansion of the unknown

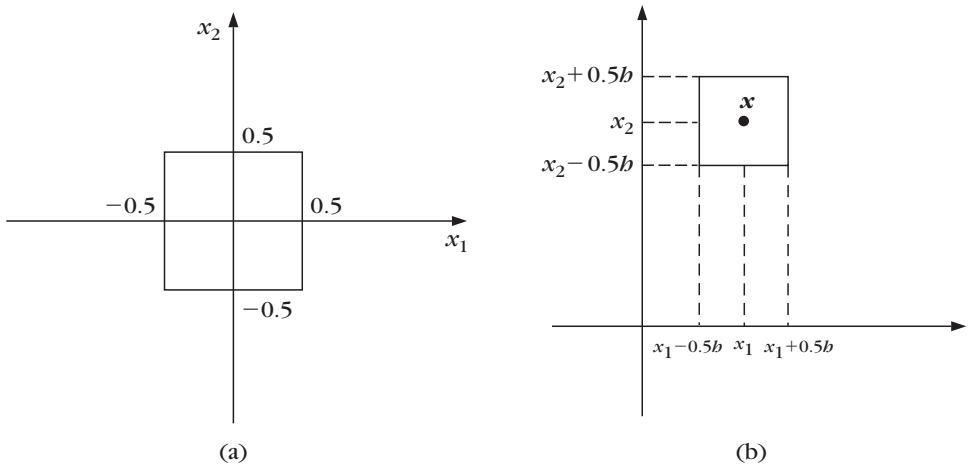

FIGURE 2.19

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.

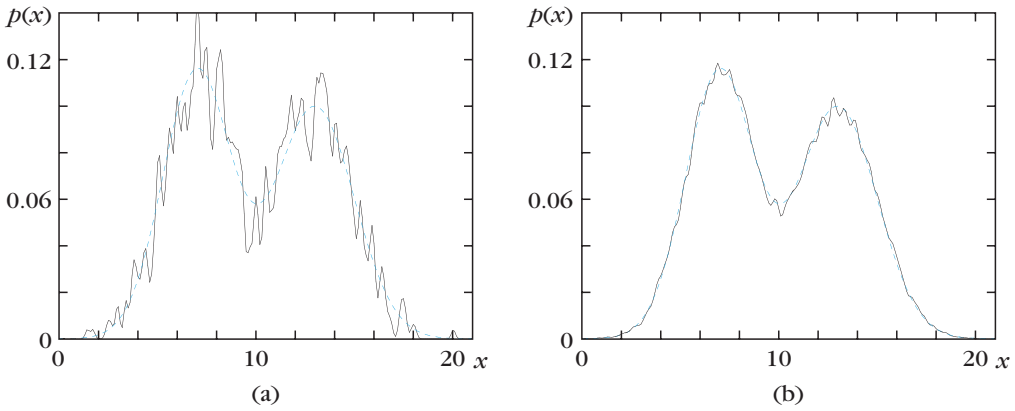

FIGURE 2.20

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.

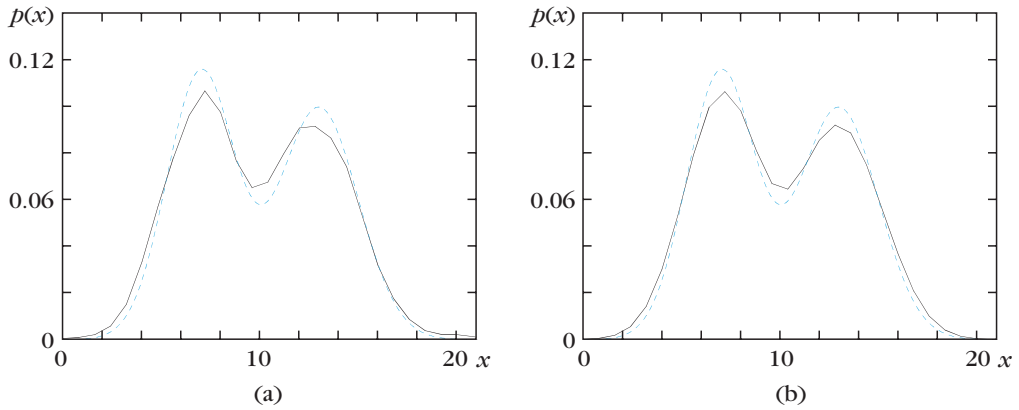

FIGURE 2.21

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.

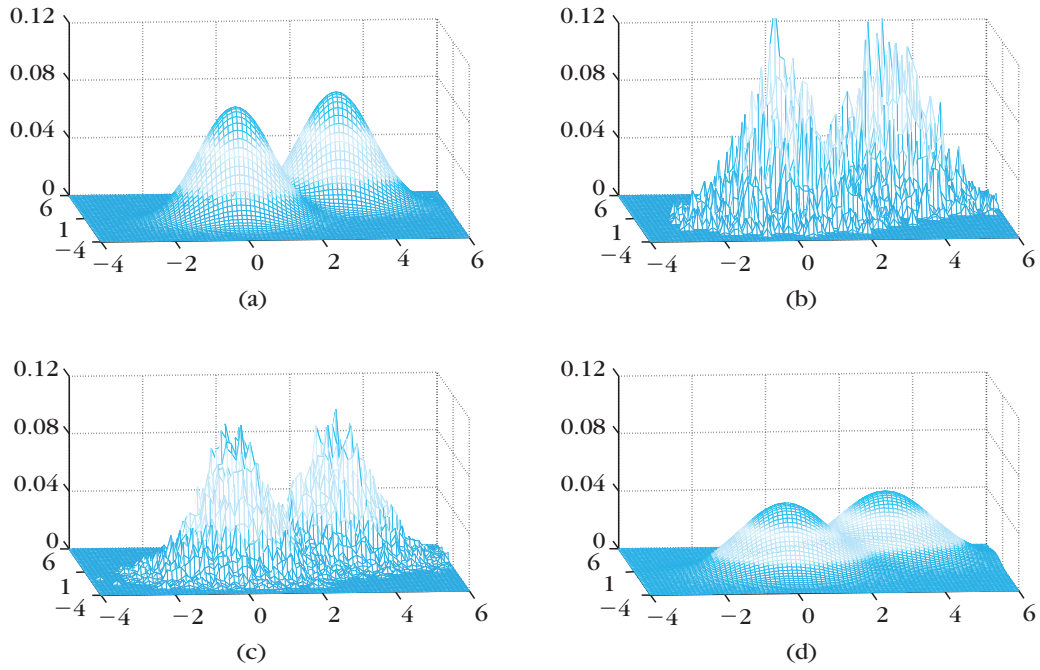

FIGURE 2.22

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

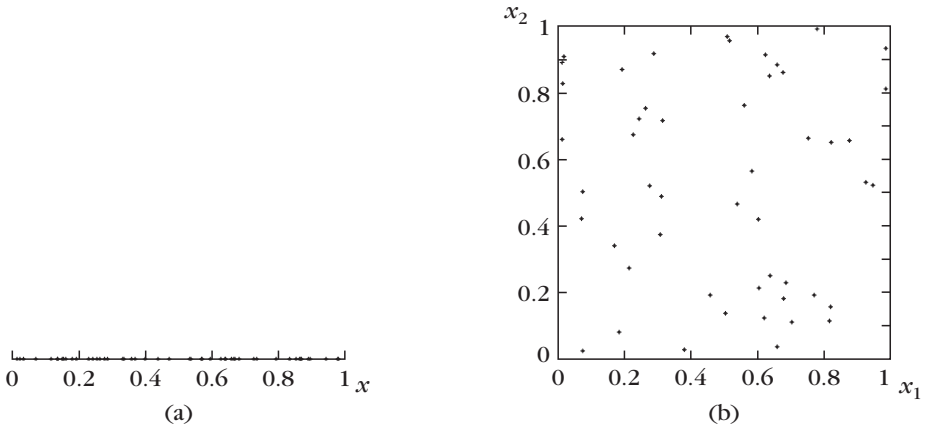

FIGURE 2.23

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