CHAPTER

Classifiers Based on Bayes Decision Theory

1

1.1 INTRODUCTION

In this chapter, we discuss techniques inspired by Bayes decision theory. The theoretical developments of the associated algorithms were given in [Theo 09, Chapter 2]. To the newcomer in the field of pattern recognition the chapter's algorithms and exercises are very important for developing a basic understanding and familiarity with some fundamental notions associated with classification. Most of the algorithms are simple in both structure and physical reasoning.

In a classification task, we are given a *pattern* and the task is to classify it into one out of c classes. The number of classes, c, is assumed to be known a priori. Each pattern is represented by a set of feature values, $x(i),\ i = 1, 2, \ldots, l$, which make up the l-dimensional *feature vector*[1] $x = [x(1), x(2), \ldots, x(l)]^T \in \mathcal{R}^l$. We assume that each pattern is represented *uniquely* by a single feature vector and that it can belong to only one class.

Given $x \in \mathcal{R}^l$ and a set of c classes, $\omega_i,\ i = 1, 2, \ldots, c$, the Bayes theory states that

$$P(\omega_i|x)p(x) = p(x|\omega_i)P(\omega_i) \tag{1.1}$$

where

$$p(x) = \sum_{i=1}^{c} p(x|\omega_i)P(\omega_i)$$

where $P(\omega_i)$ is the a priori probability of class ω_i; $i = 1, 2, \ldots, c$, $P(\omega_i|x)$ is the a posteriori probability of class ω_i given the value of x; $p(x)$ is the probability density function (pdf) of x; and $p(x|\omega_i)$, $i = 1 = 2, \ldots, c$, is the class conditional pdf of x given ω_i (sometimes called the likelihood of ω_i with respect to x).

1.2 BAYES DECISION THEORY

We are given a pattern whose class label is unknown and we let $x \equiv [x(1), x(2), \ldots, x(l)]^T \in \mathcal{R}^l$ be its corresponding feature vector, which results from some measurements. Also, we let the number of possible classes be equal to c, that is, $\omega_1, \ldots, \omega_c$.

[1] In contrast to [Theo 09], vector quantities are not boldfaced here in compliance with MATLAB notation.

DOI: 10.1016/B978-0-12-374486-9.00001-4

According to the Bayes decision theory, x is assigned to the class ω_i if

$$P(\omega_i|x) > P(\omega_j|x), \quad \forall j \neq i \tag{1.2}$$

or, taking into account Eq. (1.1) and given that $p(x)$ is positive and the same for all classes, if

$$p(x|\omega_i)P(\omega_i) > p(x|\omega_j)P(\omega_j), \quad \forall j \neq i \tag{1.3}$$

Remark

- The Bayesian classifier is optimal in the sense that it minimizes the probability of error [Theo 09, Chapter 2].

1.3 THE GAUSSIAN PROBABILITY DENSITY FUNCTION

The *Gaussian* pdf [Theo 09, Section 2.4.1] is extensively used in pattern recognition because of its mathematical tractability as well as because of the central limit theorem. The latter states that the pdf of the sum of a number of statistically independent random variables tends to the Gaussian one as the number of summands tends to infinity. In practice, this is approximately true for a large enough number of summands.

The multidimensional Gaussian pdf has the form

$$p(x) = \frac{1}{(2\pi)^{l/2}|S|^{1/2}} \exp\left(-\frac{1}{2}(x-m)^T S^{-1}(x-m)\right) \tag{1.4}$$

where $m = E[x]$ is the mean vector, S is the covariance matrix defined as $S = E[(x-m)(x-m)^T]$, $|S|$ is the determinant of S.

Often we refer to the Gaussian pdf as the *normal* pdf and we use the notation $\mathcal{N}(m,S)$. For the 1-dimensional case, $x \in \mathcal{R}$, the above becomes

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right) \tag{1.5}$$

where σ^2 is the variance of the random variable x.

Example 1.3.1. Compute the value of a Gaussian pdf, $\mathcal{N}(m,S)$, at $x_1 = [0.2, 1.3]^T$ and $x_2 = [2.2, -1.3]^T$, where

$$m = [0, 1]^T, \quad S = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Solution. Use the function *comp_gauss_dens_val* to compute the value of the Gaussian pdf. Specifically, type

```
m=[0 1]'; S=eye(2);
x1=[0.2 1.3]'; x2=[2.2 -1.3]';
pg1=comp_gauss_dens_val(m,S,x1);
pg2=comp_gauss_dens_val(m,S,x2);
```

The resulting values for *pg*1 and *pg*2 are 0.1491 and 0.001, respectively. ■

Example 1.3.2. Consider a 2-class classification task in the 2-dimensional space, where the data in both classes, ω_1, ω_2, are distributed according to the Gaussian distributions $\mathcal{N}(m_1, S_1)$ and $\mathcal{N}(m_2, S_2)$, respectively. Let

$$m_1 = [1,\ 1]^T, \quad m_2 = [3,\ 3]^T, \quad S_1 = S_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Assuming that $P(\omega_1) = P(\omega_2) = 1/2$, classify $x = [1.8,\ 1.8]^T$ into ω_1 or ω_2.

Solution. Utilize the function *comp_ gauss_dens_val* by typing

```
P1=0.5;
P2=0.5;
m1=[1 1]'; m2=[3 3]'; S=eye(2); x=[1.8 1.8]';
p1=P1*comp_gauss_dens_val(m1,S,x);
p2=P2*comp_gauss_dens_val(m2,S,x);
```

The resulting values for $p1$ and $p2$ are 0.042 and 0.0189, respectively, and x is classified to ω_1 according to the Bayesian classifier. ■

Exercise 1.3.1

Repeat Example 1.3.2 for $P(\omega_1) = 1/6$ and $P(\omega_2) = 5/6$, and for $P(\omega_1) = 5/6$ and $P(\omega_2) = 1/6$. Observe the dependance of the classification result on the a priori probabilities [Theo 09, Section 2.4.2].

Example 1.3.3. Generate $N = 500$ 2-dimensional data points that are distributed according to the Gaussian distribution $\mathcal{N}(m, S)$, with mean $m = [0,\ 0]^T$ and covariance matrix $S = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$, for the following cases:

$\sigma_1^2 = \sigma_2^2 = 1, \sigma_{12} = 0$

$\sigma_1^2 = \sigma_2^2 = 0.2, \sigma_{12} = 0$

$\sigma_1^2 = \sigma_2^2 = 2, \sigma_{12} = 0$

$\sigma_1^2 = 0.2, \sigma_2^2 = 2, \sigma_{12} = 0$

$\sigma_1^2 = 2, \sigma_2^2 = 0.2, \sigma_{12} = 0$

$\sigma_1^2 = \sigma_2^2 = 1, \sigma_{12} = 0.5$

$\sigma_1^2 = 0.3, \sigma_2^2 = 2, \sigma_{12} = 0.5$

$\sigma_1^2 = 0.3, \sigma_2^2 = 2, \sigma_{12} = -0.5$

Plot each data set and comment on the shape of the clusters formed by the data points.

Solution. To generate the first data set, use the built-in MATLAB function *mvnrnd* by typing

```
randn('seed',0) %Initialization of the randn function
m=[0 0]';
S=[1 0;0 1];
N=500;
X = mvnrnd(m,S,N)';
```

where X is the matrix that contains the data vectors in its columns.

To ensure reproducibility of the results, the *randn* MATLAB function, which generates random numbers following the Gaussian distribution, with zero mean and unit variance, is initialized to a specific number via the first command (in the previous code *randn* is called by the *mvnrnd* MATLAB function).

To plot the data set, type

```
figure(1), plot(X(1,:),X(2,:),'.');
figure(1), axis equal
figure(1), axis([-7 7 -7 7])
```

Working similarly for the second data set, type

```
m=[0 0]';
S=[0.2 0;0 0.2];
N=500;
X = mvnrnd(m,S,N)';
figure(2), plot(X(1,:),X(2,:),'.');
figure(2), axis equal
figure(2), axis([-7 7 -7 7])
```

The rest of the data sets are obtained similarly. All of them are depicted in Figure 1.1, from which one can observe the following:

- When the two coordinates of x are uncorrelated ($\sigma_{12} = 0$) and their variances are equal, the data vectors form "spherically shaped" clusters (Figure 1.1(a–c)).
- When the two coordinates of x are uncorrelated ($\sigma_{12} = 0$) and their variances are unequal, the data vectors form "ellipsoidally shaped" clusters. The coordinate with the highest variance corresponds to the "major axis" of the ellipsoidally shaped cluster, while the coordinate with the lowest variance corresponds to its "minor axis." In addition, the major and minor axes of the cluster are parallel to the axes (Figure 1.1(d, e)).

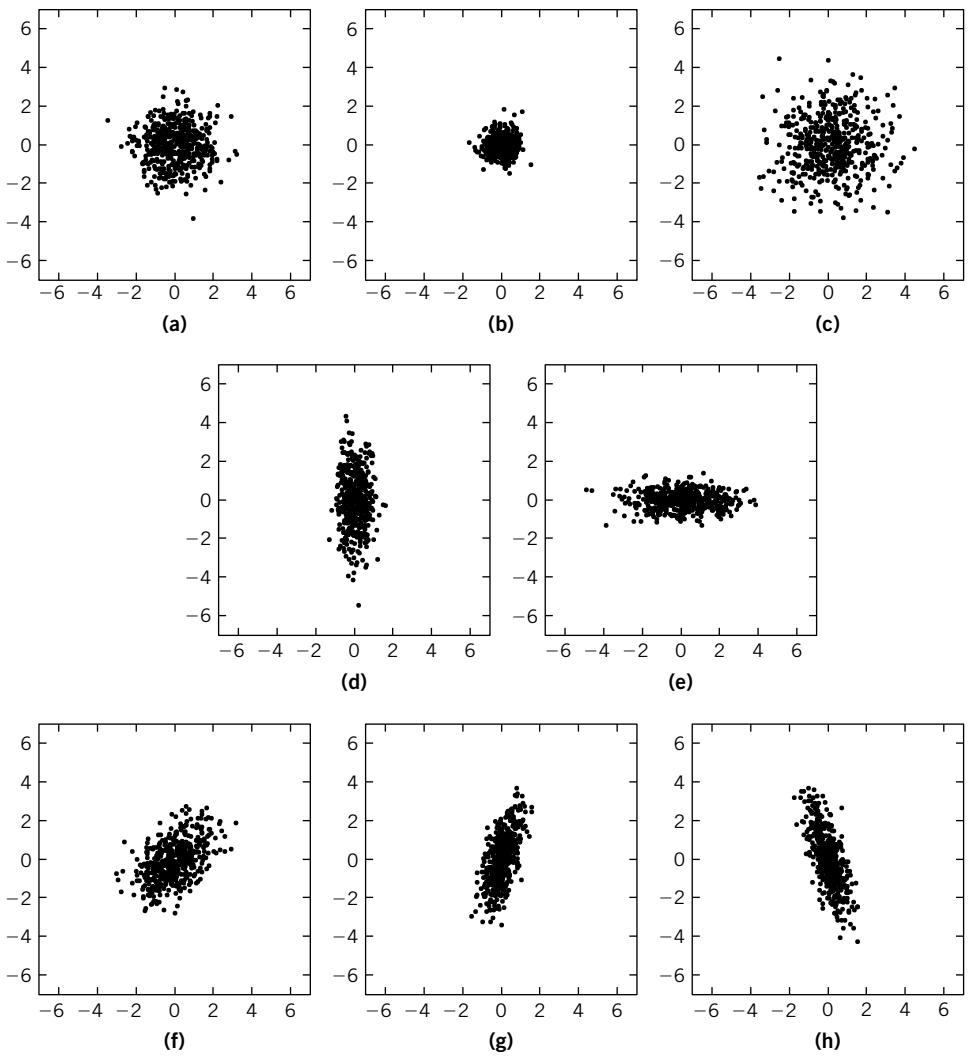

FIGURE 1.1

Eight data sets of Example 1.3.3.

- When the two coordinates of x are correlated ($\sigma_{12} \neq 0$), the major and minor axes of the ellipsoidally shaped cluster are no longer parallel to the axes. The degree of rotation with respect to the axes depends on the value of σ_{12} (Figure 1.1(f–h)). The effect of the value of σ_{12}, whether positive or negative, is demonstrated in Figure 1.1(g, h). Finally, as can be seen by comparing Figure 1.1(a, f), when $\sigma_{12} \neq 0$, the data form ellipsoidally shaped clusters despite the fact that the variances of each coordinate are the same.

1.4 MINIMUM DISTANCE CLASSIFIERS

1.4.1 The Euclidean Distance Classifier

The optimal Bayesian classifier is significantly simplified under the following assumptions:

- The classes are equiprobable.
- The data in *all* classes follow Gaussian distributions.
- The covariance matrix is the *same* for all classes.
- The covariance matrix is diagonal and *all* elements across the diagonal are *equal*. That is, $S = \sigma^2 I$, where I is the identity matrix.

Under these assumptions, it turns out that the optimal Bayesian classifier is equivalent to the minimum Euclidean distance classifier. That is, given an unknown x, assign it to class ω_i if

$$||x - m_i|| \equiv \sqrt{(x - m_i)^T (x - m_i)} < ||x - m_j||, \quad \forall i \neq j$$

It must be stated that the Euclidean classifier is often used, even if we know that the previously stated assumptions are not valid, because of its simplicity. It assigns a pattern to the class whose mean is closest to it with respect to the Euclidean norm.

1.4.2 The Mahalanobis Distance Classifier

If one relaxes the assumptions required by the Euclidean classifier and removes the last one, the one requiring the covariance matrix to be diagonal and with equal elements, the optimal Bayesian classifier becomes equivalent to the minimum Mahalanobis distance classifier. That is, given an unknown x, it is assigned to class ω_i if

$$\sqrt{(x - m_i)^T S^{-1} (x - m_i)} < \sqrt{(x - m_j)^T S^{-1} (x - m_j)}, \quad \forall j \neq i$$

where S is the common covariance matrix. The presence of the covariance matrix accounts for the shape of the Gaussians [Theo 09, Section 2.4.2].

Example 1.4.1. Consider a 2-class classification task in the 3-dimensional space, where the two classes, ω_1 and ω_2, are modeled by Gaussian distributions with means $m_1 = [0, 0, 0]^T$ and $m_2 = [0.5, 0.5, 0.5]^T$, respectively. Assume the two classes to be equiprobable. The covariance matrix for both distributions is

$$S = \begin{bmatrix} 0.8 & 0.01 & 0.01 \\ 0.01 & 0.2 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix}$$

Given the point $x = [0.1, 0.5, 0.1]^T$, classify x (1) according to the Euclidean distance classifier and (2) according to the Mahalanobis distance classifier. Comment on the results.

Solution. Take the following steps:

Step 1. Use the function *euclidean_classifier* by typing

```
x=[0.1 0.5 0.1]';
m1=[0 0 0]'; m2=[0.5 0.5 0.5]';
m=[m1 m2];
z=euclidean_classifier(m,x)
```

The answer is $z = 1$; that is, the point is classified to the ω_1 class.

Step 2. Use the function *mahalanobis_classifier* by typing

```
x=[0.1 0.5 0.1]';
m1=[0 0 0]'; m2=[0.5 0.5 0.5]';
m=[m1 m2];
S=[0.8 0.01 0.01;0.01 0.2 0.01; 0.01 0.01 0.2];
z=mahalanobis_classifier(m,S,x);
```

This time, the answer is $z = 2$, meaning the point is classified to the second class. For this case, the optimal Bayesian classifier is realized by the Mahalanobis distance classifier. The point is assigned to class ω_2 in spite of the fact that it lies closer to m_1 according to the Euclidean norm. ■

1.4.3 Maximum Likelihood Parameter Estimation of Gaussian pdfs

One problem often met in practice is that the pdfs describing the statistical distribution of the data in the classes are not known and must be estimated using the training data set. One approach to this function estimation task is to assume that a pdf has a specific functional form but we do not know the values of the parameters that define it. For example, we may know that the pdf is of Gaussian form but not the mean value and/or the elements of its covariance matrix.

The *maximum likelihood* (ML) technique [Theo 09, Section 2.5.1] is a popular method for such a *parametric estimation* of an unknown pdf. Focusing on Gaussian pdfs and assuming that we are given N points, $x_i \in \mathcal{R}^l$, $i = 1, 2, \ldots, N$, which are known to be normally distributed, the ML estimates of the unknown mean value and the associated covariance matrix are given by

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

and

$$S_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x_i - m_{ML})(x_i - m_{ML})^T$$

Often, instead of N, the summation associated with the covariance matrix is divided by $N - 1$ since this provides an unbiased estimate [Theo 09, Section 2.5.1]. The next example focuses on the estimation of the unknown parameters of the Gaussian pdf.

Example 1.4.2. Generate 50 2-dimensional feature vectors from a Gaussian distribution, $\mathcal{N}(m,S)$, where

$$m = [2,-2]^T,\ S = \begin{bmatrix} 0.9 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$$

Let X be the resulting matrix, having the feature vectors as columns. Compute the ML estimate of the mean value, m, and the covariance matrix, S, of $\mathcal{N}(m,S)$ and comment on the resulting estimates.

Solution. To generate X, type

```
randn('seed',0)
m = [2 -2]; S = [0.9 0.2; 0.2 .3];
X = mvnrnd(m,S,50)';
```

To compute the ML estimates of m and S, type

```
[m_hat, S_hat]=Gaussian_ML_estimate(X);
```

The results are

$$m\_hat = [2.0495,\ -1.9418]^T,\ S\_hat = \begin{bmatrix} 0.8082 & 0.0885 \\ 0.0885 & 0.2298 \end{bmatrix}$$

It can be observed that the estimates that define the corresponding Gaussian pdf, although close to the true values of the parameters, cannot be trusted as good estimates. This is due to the fact that 50 points are not enough to result in reliable estimates. Note that the returned values depend on the initialization of the random generator (involved in function *mvnrnd*), so there is a slight deviation among experiments. ■

Exercise 1.4.1

Repeat Example 1.4.2 for $N = 500$ points and $N = 5000$ points. Comment on the results.

Example 1.4.3. Generate two data sets, X (training set) and X_1 (test set), each consisting of $N = 1000$ 3-dimensional vectors that stem from three *equiprobable* classes, ω_1, ω_2, and ω_3. The classes are modeled by Gaussian distributions with means $m_1 = [0,\ 0,\ 0]^T$, $m_2 = [1,\ 2,\ 2]^T$, and $m_3 = [3,\ 3,\ 4]^T$, respectively; their covariance matrices are

$$S_1 = S_2 = S_3 = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.8 \end{bmatrix} = \sigma^2 I$$

1. Using X, compute the maximum likelihood estimates of the mean values and the covariance matrices of the distributions of the three classes. Since the covariance matrices are known to be the same, estimate them for each class and compute their average. Use the latter as the estimate of the (common) covariance matrix.

2. Use the Euclidean distance classifier to classify the points of X_1 based on the ML estimates computed before.
3. Use the Mahalanobis distance classifier to classify the points of X_1 based on the ML estimates computed before.
4. Use the Bayesian classifier to classify the points of X_1 based on the ML estimates computed before.
5. For each case, compute the error probability and compare the results (all classifiers should result in almost the same performance. Why?).

Solution. To generate X, use the function *generate_gauss_classes* by typing

```
m=[0 0 0; 1 2 2; 3 3 4]';
S1=0.8*eye(3);
S(:,:,1)=S1;S(:,:,2)=S1;S(:,:,3)=S1;
P=[1/3 1/3 1/3]'; N=1000;
randn('seed',0)
[X,y]=generate_gauss_classes(m,S,P,N);
```

where

X is the $3 \times N$ matrix that contains the data vectors in its columns,

y is an N-dimensional vector that contains the class labels of the respective data vectors,

P is the vector of the respective class a priori probabilities.

The data set X_1 is generated similarly:

```
randn('seed',100);
[X1,y1]=generate_gauss_classes(m,S,P,N);
```

where *randn* is initialized using $seed = 100$.

Perform the following:

Step 1. To compute the ML estimates of the mean values and covariance matrix (common to all three classes), use *Gaussian_ML_estimate* by typing

```
class1_data=X(:,find(y==1));
[m1_hat, S1_hat]=Gaussian_ML_estimate(class1_data);
class2_data=X(:,find(y==2));
[m2_hat, S2_hat]=Gaussian_ML_estimate(class2_data);
class3_data=X(:,find(y==3));
[m3_hat, S3_hat]=Gaussian_ML_estimate(class3_data);
S_hat=(1/3)*(S1_hat+S2_hat+S3_hat);
m_hat=[m1_hat m2_hat m3_hat];
```

Step 2. For the Euclidean distance classifier, use the ML estimates of the means to classify the data vectors of X_1, typing

```
z_euclidean=euclidean_classifier(m_hat,X1);
```

where *z_euclidean* is an N-dimensional vector containing the labels of the classes where the respective data vectors are assigned by the Euclidean classifier.

Step 3. Similarly for the Mahalanobis distance classifier, type

```
z_mahalanobis=mahalanobis_classifier(m_hat,S_hat,X1);
```

Step 4. For the Bayesian classifier, use function *bayes_classifier* and provide as input the matrices m, S, P, which were used for the data set generation. In other words, use the true values of m, S, and P and not their estimated values. Type

```
z_bayesian=bayes_classifier(m,S,P,X1);
```

Step 5. To compute the error probability for each classifier, compare the vector y_1 of the true class labels of the vectors of X_1 with vectors *z_euclidean*, *z_mahalanobis*, and *z_bayesian*, respectively. For each comparison, examine the vector elements in pairs and count the number of matches (i.e., correct classifications); divide by the length of y_1. Type

```
err_euclidean = (1-length(find(y1==z_euclidean))/length(y1));
err_mahalanobis = (1-length(find(y1==z_mahalanobis))/length(y1));
err_bayesian = (1-length(find(y1==z_bayesian))/length(y1));
```

The error probabilities for the Euclidean, Mahalanobis, and Bayesian classifiers are 7.61%, 7.71%, and 7.61%, respectively. The results are almost equal since all of the four assumptions in Subsection 1.4.1 are valid, which implies that in the present case the three classifiers are equivalent. ■

Exercise 1.4.2

Repeat Example 1.4.3 using

$$S_1 = S_2 = S_3 = \begin{bmatrix} 0.8 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{bmatrix} \neq \sigma^2 I$$

Comment on the results.

Exercise 1.4.3

Repeat Example 1.4.3 using $P_1 = 1/2$, $P_2 = P_3 = 1/4$ to generate X and X_1. For this case, because the a priori probabilities are not equal, the Bayesian classifier should result in the best performance. Why?

Exercise 1.4.4

Repeat Example 1.4.3 using $P(\omega_1) = P(\omega_2) = P(\omega_3) = 1/3$ and

$$S_1 = \begin{bmatrix} 0.8 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{bmatrix}, \; S_2 = \begin{bmatrix} 0.6 & 0.01 & 0.01 \\ 0.01 & 0.8 & 0.01 \\ 0.01 & 0.01 & 0.6 \end{bmatrix}, \; S_3 = \begin{bmatrix} 0.6 & 0.1 & 0.1 \\ 0.1 & 0.6 & 0.1 \\ 0.1 & 0.1 & 0.6 \end{bmatrix}$$

Experiment with the mean values (bringing them closer or taking them farther away) and the a priori probabilities. Comment on the results.

1.5 MIXTURE MODELS

When the pdf that describes the data points in a class is not known, it has to be estimated prior to the application of the Bayesian classifier. In this section, we focus on a very popular method to model unknown probability density functions, known as *mixture modeling* [Theo 09, Section 2.5.5].

An arbitrary pdf can be modeled as a linear combination of J pdfs in the form

$$p(x) = \sum_{j=1}^{J} P_j p(x|j) \tag{1.6}$$

where

$$\sum_{j=1}^{J} P_j = 1, \int p(x|j)dx = 1$$

for sufficiently large J. In most cases, $p(x|j)$ are chosen to be Gaussians, $\mathcal{N}(m_j, S_j)$, $j = 1, 2, \ldots, J$.

The expansion in Eq. (1.6) points out a way to generate data from pdfs of a more complex functional form: *multimodal* (many-peaked) pdfs. The meaning of Eq. (1.6) is that the data are generated from each one of the (summand) pdfs, $p(x|j)$, with probability P_j.

Example 1.5.1. Consider the 2-dimensional pdf

$$p(x) = P_1 p(x|1) + P_2 p(x|2) \tag{1.7}$$

where $p(x|j), j = 1, 2$ are normal distributions with means $m_1 = [1, 1]^T$ and $m_2 = [3, 3]^T$ and covariance matrices

$$S_1 = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}, \; S_2 = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

with $\sigma_1^2 = 0.1$, $\sigma_2^2 = 0.2$, $\sigma_{12} = -0.08$, $\sigma^2 = 0.1$.

Generate and plot a set X consisting of $N = 500$ points that stem from $p(x)$ for (i) $P_1 = P_2 = 0.5$, and (ii) for $P_1 = 0.85$, $P_2 = 0.15$; and (iii) experiment by changing the parameters σ_1^2, σ_2^2, σ_{12}, σ^2 of the covariance matrices and the mixing probabilities P_1 and P_2.

Solution. To generate X, use the function *mixt_model* by typing

```
randn('seed',0); % used for the initialization of MATLAB's randn generator
m1=[1, 1]'; m2=[3, 3]';
m=[m1 m2];
S(:,:,1)=[0.1 -0.08; -0.08 0.2];
S(:,:,2)=[0.1 0; 0 0.1];
P=[1/2 1/2];
N=500;
```

```
sed=0; % used for the initialization of MATLAB's rand generator
[X,y]=mixt_model(m,S,P,N,sed);
plot(X(1,:),X(2,:),'.');
```

where

sed is the "seed" used for the initialization of the built-in MATLAB random generator function *rand*, which generates numbers from the uniform distribution in the interval [0, 1],

y is a vector whose *i*th element contains the label of the distribution that generated the *i*th data vector

The next steps are carried out in a similar manner. From Figure 1.2, one can verify the multimodal nature of the pdf of x. That is, x is spread over two well-separated regions in space. Comparing Figures 1.2(a, b), observe that in the latter case, since $P_1 \neq P_2$, one of the two high-density regions is sparser in data points.

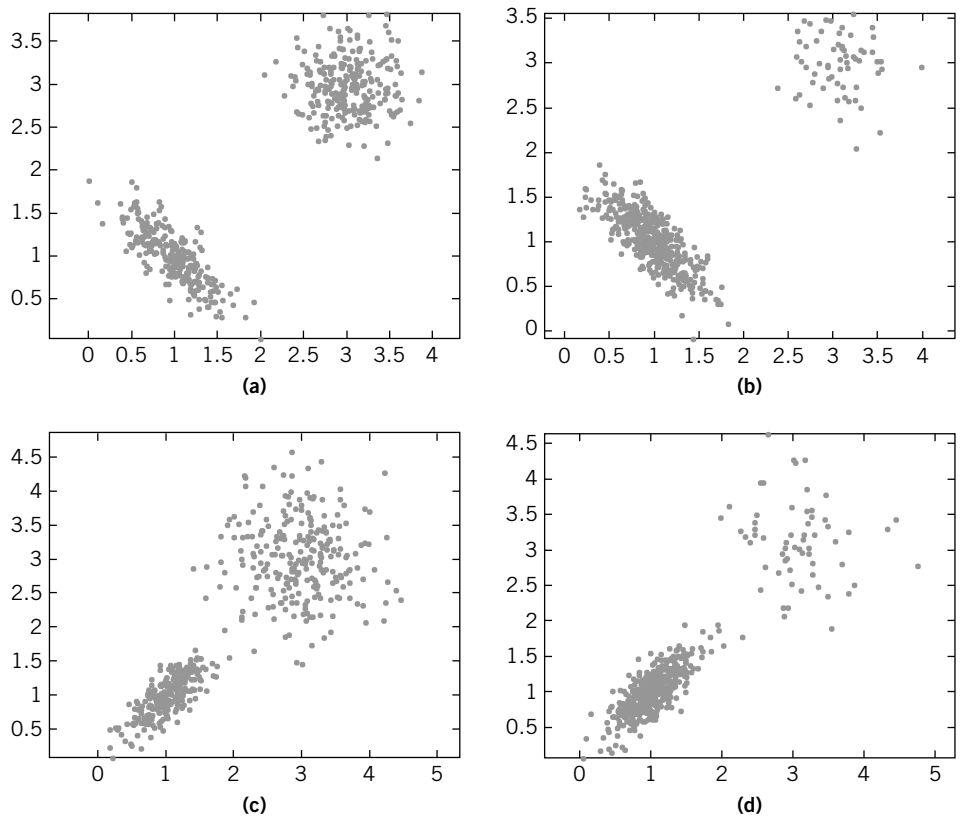

FIGURE 1.2

Example 1.5.1 (a) results obtained for the setup of case (i), (b) setup of case (ii), and (c)–(d) for some values of case (iii).

1.6 THE EXPECTATION-MAXIMIZATION ALGORITHM

Let us assume that we are given a set of N points, $x_i \in \mathcal{R}^l,\ i = 1, 2, \ldots, N$, whose statistical properties are described by a pdf that is expanded as in Eq. (1.6). Adopting a value for J, the task is to use these data points to estimate the parameters that enter in the expansion—that is, the probability parameters $P_j,\ j = 1, 2, \ldots, J$ and the parameters associated with each one of the terms $p(x|j),\ j = 1, 2, \ldots, J$. For example, if we assume each one of the summand pdfs to be a Gaussian distribution with $\sigma_j^2 I$ covariance matrix:

$$p(x|j) = \frac{1}{(2\pi)^{l/2}\sigma_j^l} \exp\left(-\frac{(x - m_j)^T (x - m_j)}{2\sigma_j^2}\right), \quad j = 1, 2, \ldots, J$$

then the associated unknown parameters are the mean values $m_j,\ j = 1, 2, \ldots, J$ (lJ parameters in total) and the J covariances $\sigma_j^2,\ j = 1, 2, \ldots, J$ (J parameters in total).

The expectation-maximization (EM) algorithm iteratively computes the corresponding estimates, starting from some user-defined initial values [Theo 09, Section 2.5.5].

Example 1.6.1. Generate a set X of $N = 500$ 2-dimensional points that stem from the following pdf:

$$p(x) = \sum_{j=1}^{3} P_j p(x|j)$$

where the $p(x|j)$'s, $j = 1, 2, 3$ are (2-dimensional) normal distributions with mean values $m_1 = [1,\ 1]^T$, $m_2 = [3,\ 3]^T$, $m_3 = [2,\ 6]^T$ and covariance matrices $S_1 = 0.1I$, $S_2 = 0.2I$, $S_3 = 0.3I$, respectively (I is the 2×2 identity matrix). In addition, $P_1 = 0.4$, $P_2 = 0.4$, and $P_3 = 0.2$.

The idea is to use the previously generated data and pretend that we do not know how they were generated. We assume that the pdf $p(x)$ underlying X is a weighted sum of J (Eq. (1.6)) normal distributions with covariance matrices of the form $S_i = \sigma_i^2 I$, and we employ the EM algorithm to estimate the unknown parameters in the adopted model of $p(x)$. The goal is to demonstrate the dependence of the EM algorithm on the initial conditions and the parameter J. To this end, we use the following sets of initial parameter estimates:

- $J = 3$, $m_{1,ini} = [0,\ 2]^T$, $m_{2,ini} = [5,\ 2]^T$, $m_{3,ini} = [5,\ 5]^T$, $S_{1,ini} = 0.15I$, $S_{2,ini} = 0.27I$, $S_{3,ini} = 0.4I$ and $P_{1,ini} = P_{2,ini} = P_{3,ini} = 1/3$
- $J = 3$, $m_{1,ini} = [1.6,\ 1.4]^T$, $m_{2,ini} = [1.4,\ 1.6]^T$, $m_{3,ini} = [1.3,\ 1.5]^T$, $S_{1,ini} = 0.2I$, $S_{2,ini} = 0.4I$, $S_{3,ini} = 0.3I$ and $P_{1,ini} = 0.2$, $P_{2,ini} = 0.4$, $P_{3,ini} = 0.4$
- $J = 2$, $m_{1,ini} = [1.6,\ 1.4]^T$, $m_{2,ini} = [1.4,\ 1.6]^T$, $S_{1,ini} = 0.2I$, $S_{2,ini} = 0.4I$ and $P_{1,ini} = P_{2,ini} = 1/2$

Comment on the results.

Solution. To generate and plot the data set X, type

```
randn('seed',0);
m1=[1, 1]'; m2=[3, 3]';m3=[2, 6]';
m=[m1 m2 m3];
S(:,:,1)=0.1*eye(2);
S(:,:,2)=0.2*eye(2);
S(:,:,3)=0.3*eye(2);
P=[0.4 0.4 0.2];
N=500;
sed=0;
[X,y]=mixt_model(m,S,P,N,sed);
plot_data(X,y,m,1)
```

Then do the following:

Step 1. Use the function *em_alg_ function* to estimate the mixture model parameters by typing

```
m1_ini=[0; 2];m2_ini=[5; 2];m3_ini=[5; 5];
m_ini=[m1_ini m2_ini m3_ini];
s_ini=[.15 .27 .4];
Pa_ini=[1/3 1/3 1/3];
e_min=10^(-5);
[m_hat,s_hat,Pa,iter,Q_tot,e_tot]=...
em_alg_function(X,m_ini,s_ini,Pa_ini,e_min);
```

where

m_hat is an $l \times J$ matrix whose jth column is the estimate for the mean of the jth distribution,

s is a J-dimensional vector whose jth element is the variance for the jth distribution (it is assumed that the covariance matrices of the distributions are of the form $s(j) * I$, where I is the identity matrix),

Pa is a J-dimensional vector with a jth element that is the estimate of the a priori probability of the jth distribution.

The final estimates obtained by the EM algorithm are (rounded to the second decimal):

- $\hat{m}_1 = [1.02,\ 0.98]^T$, $\hat{m}_2 = [2.94,\ 3.02]^T$, $\hat{m}_3 = [2.03,\ 6.00]^T$
- $\hat{S}_1 = 0.10I$, $\hat{S}_2 = 0.22I$, $\hat{S}_3 = 0.30I$
- $\hat{P}_1 = 0.39$, $\hat{P}_2 = 0.43$, $\hat{P}_3 = 0.18$

The algorithm converged after 12 iterations (see Figure 1.3(a)).

Step 2. Working as in step 1, obtain the results

- $\hat{m}_1 = [1.01,\ 0.84]^T$, $\hat{m}_2 = [2.66,\ 3.86]^T$, $\hat{m}_3 = [1.02,\ 1.26]^T$
- $\hat{S}_1 = 0.09I$, $\hat{S}_2 = 1.28I$, $\hat{S}_3 = 0.07I$
- $\hat{P}_1 = 0.26$, $\hat{P}_2 = 0.62$, $\hat{P}_3 = 0.12$

The algorithm converged after 533 iterations (see Figure 1.3(b)).

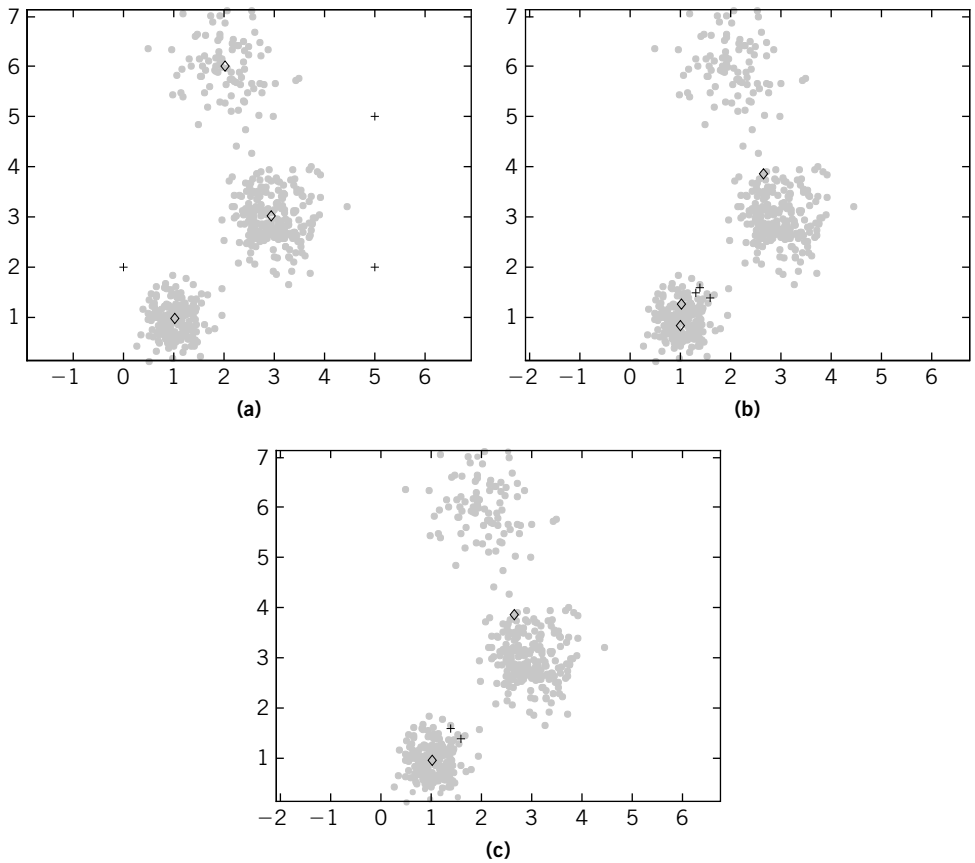

FIGURE 1.3

Example 1.6.1 initial (+) and final (◇) estimates of the mean values of the normal distributions for all three cases.

Step 3. Working as in step 1, obtain the results

- $\hat{m}_1 = [1.01,\ 0.97]^T$, $\hat{m}_2 = [2.66,\ 3.86]^T$
- $\hat{S}_1 = 0.10I$, $\hat{S}_2 = 1.27I$
- $\hat{P}_1 = 0.38$, $\hat{P}_2 = 0.62$

The algorithm converged after 10 iterations (see Figure 1.3(c)). ■

In the first case, the good initialization of the algorithm led to parameter estimates that are very close to the true parameters, which were used for the generation of the data set. In the second case, the bad initialization led to poor parameter estimates. In the third case, the wrong choice of the order of the model (number of involved normal distributions) led to bad estimates. Using the EM algorithm, one has to be cautious about parameter initialization as well as the choice of the value of J.

Some popular methods for estimating the correct order of the problem (in our case J) are based on so-called information-based criteria. For an application of these criteria in the current framework, see [Theo 09, Chapter 16], where such methods are used to identify the number of dense regions (clusters) formed by a set of data vectors.

Example 1.6.2. In this example, the EM algorithm is used in a classification application. A 2-class problem is considered. The data set X consists of $N = 1000$ 2-dimensional vectors. Of these, 500 stem from class ω_1, which is modeled as $p_1(x) = \sum_{j=1}^{3} P_{1j} p_1(x|j)$, where $p_1(x|j)$, $j = 1,2,3$ are normal distributions with mean values $m_{11} = [1.25,\ 1.25]^T$, $m_{12} = [2.75,\ 2.75]^T$, $m_{13} = [2,\ 6]^T$, and covariance matrices $S_{1j} = \sigma_{1j}^2 I$, $j = 1,2,3$, where $\sigma_{11}^2 = 0.1$, $\sigma_{12}^2 = 0.2$, $\sigma_{13}^2 = 0.3$, respectively. The mixing probabilities are $P_{11} = 0.4$, $P_{12} = 0.4$, and $P_{13} = 0.2$.

The other 500 data vectors stem from class ω_2, which is modeled as $p_2(x) = \sum_{j=1}^{3} P_{2j} p_2(x|j)$, where $p_2(x|j)$, $j = 1,2,3$ are also normal distributions with means $m_{21} = [1.25,\ 2.75]^T$, $m_{22} = [2.75,\ 1.25]^T$, $m_{23} = [4,\ 6]^T$, and covariance matrices $S_{2j} = \sigma_{2j}^2 I$, $j = 1,2,3$, where $\sigma_{21}^2 = 0.1$, $\sigma_{22}^2 = 0.2$, $\sigma_{23}^2 = 0.3$, respectively. The mixing probabilities are $P_{21} = 0.2$, $P_{22} = 0.3$, and $P_{23} = 0.5$.

The setup of the problem is shown in Figure 1.4. Each class consists of points that are spread to more than one dense region. Such a setup is a typical scenario where mixture modeling and the EM algorithm are used to estimate the corresponding pdfs for each class.

The data set X is used as the training set, and we pretend that we do not know how it was generated. We assume that, somehow, we have a priori information about the number of dense regions in each class, so we adopt a mixture model with three Gaussian components to model the pdf in each class. The data set X is used by the EM algorithm for estimating the "unknown" parameters involved in the respective model pdf expansions.

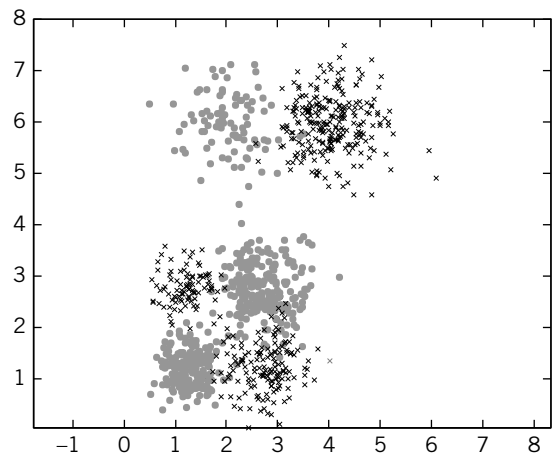

FIGURE 1.4

Data set X of Example 1.6.2.

With the pdf estimates for each class obtained, the Bayesian classifier is mobilized. An additional data set Z of 1000 data vectors is also generated such that the first half stem from $p_1(x)$ and the rest stem from $p_2(x)$. The set Z is used for testing the performance of the resulting classifier.

Use the EM algorithm to estimate $p_1(x)$ and $p_2(x)$ based on the data set X using the following initial parameter estimates:

- For $p_1(x)$: Three normal distributions with initial mean estimates $m_{11,ini} = [0, 2]^T$, $m_{12,ini} = [5, 2]^T$, $m_{13,ini} = [5, 5]^T$; initial variance estimates $\sigma^2_{11,ini} = 0.15$, $\sigma^2_{12,ini} = 0.27$, $\sigma^2_{13,ini} = 0.4$; and mixing probabilities $P_{11,ini} = P_{12,ini} = P_{13,ini} = 1/3$.
- For $p_2(x)$: Three normal distributions with initial mean estimates $m_{21,ini} = [5, 2]^T$, $m_{22,ini} = [3, 4]^T$, $m_{23,ini} = [2, 5]^T$; initial variance estimates $\sigma^2_{21,ini} = 0.15$, $\sigma^2_{22,ini} = 0.27$, $\sigma^2_{23,ini} = 0.35$; and mixing probabilities $P_{21,ini} = P_{22,ini} = P_{23,ini} = 1/3$.

Having obtained the estimates of $p_1(x)$ and $p_2(x)$, employ the Bayes classification rule to classify the vectors in Z and compute the classification error.

Solution. To generate the subset X_1 of X, which contains the data points from class ω_1, type

```
m11=[1.25 1.25]'; m12=[2.75 2.75]';m13=[2 6]';
m1=[m11 m12 m13];
S1(:,:,1)=0.1*eye(2);
S1(:,:,2)=0.2*eye(2);
S1(:,:,3)=0.3*eye(2);
P1=[0.4 0.4 0.2];
N1=500;
sed=0;
[X1,y1]=mixt_model(m1,S1,P1,N1,sed);
```

The subset X_2 of X with the points from class ω_2 is generated similarly (again use $sed = 0$). Let $X2$ and $y2$ be the two MATLAB variables resulting from the *mixt_model* function in this case.

To generate the set Z, we work in a similar manner. Specifically, Z is generated in two steps: First, 500 points are generated from class ω_1 via the following code:

```
mZ11=[1.25 1.25]'; mZ12=[2.75 2.75]';mZ13=[2 6]';
mZ1=[mZ11 mZ12 mZ13];
SZ1(:,:,1)=0.1*eye(2);
SZ1(:,:,2)=0.2*eye(2);
SZ1(:,:,3)=0.3*eye(2);
wZ1=[0.4 0.4 0.2];
NZ1=500;
sed=100;
[Z1,yz1]=mixt_model(mZ1,SZ1,wZ1,NZ1,sed);
```

The remaining 500 points from the second class are generated similarly (with $sed = 100$). In this case, let $Z2$, $yz2$ be the two corresponding MATLAB variables resulting from the *mixt_model* function.

Finally, type Z = [Z1 Z2]; and do the following:

Step 1. To estimate the Gaussian mixture model of each class, type

```
m11_ini=[0; 2]; m12_ini=[5; 2]; m13_ini=[5; 5];
m1_ini=[m11_ini m12_ini m13_ini];
S1_ini=[0.15 0.27 0.4];
w1_ini=[1/3 1/3 1/3];

m21_ini=[5; 2]; m22_ini=[3; 4]; m23_ini=[2; 5];
m2_ini=[m21_ini m22_ini m23_ini];
S2_ini=[0.15 0.27 0.35];
w2_ini=[1/3 1/3 1/3];

m_ini{1}=m1_ini;
m_ini{2}=m2_ini;
S_ini{1}=S1_ini;
S_ini{2}=S2_ini;
w_ini{1}=w1_ini;
w_ini{2}=w2_ini;
[m_hat,S_hat,w_hat,P_hat]=...
EM_pdf_est([X1 X2],[ones(1,500) 2*ones(1,500)],m_ini,S_ini,w_ini);
```

The estimated values of the remaining parameters involved in $p_1(x)$ and $p_2(x)$ are

- For $p_1(x)$: The mean values are $\hat{m}_{11} = [1.27,\ 1.22]^T$, $\hat{m}_{12} = [2.69,\ 2.76]^T$, $\hat{m}_{13} = [2.03,\ 6.00]^T$; the variances are $\hat{\sigma}_{11} = 0.10$, $\hat{\sigma}_{12} = 0.22$, $\hat{\sigma}_{13} = 0.31$; the mixing probabilities are $\hat{P}_{11} = 0.38$, $\hat{P}_{12} = 0.44$, $\hat{P}_{13} = 0.18$.
- For $p_2(x)$: The mean values are $\hat{m}_{21} = [1.22,\ 2.77]^T$, $\hat{m}_{22} = [2.75,\ 1.22]^T$, $\hat{m}_{23} = [4.03,\ 5.97]^T$; the variances are $\hat{\sigma}_{21} = 0.11$, $\hat{\sigma}_{22} = 0.20$, $\hat{\sigma}_{23} = 0.30$; the mixing probabilities are $\hat{P}_{21} = 0.19$, $\hat{P}_{22} = 0.31$, $\hat{P}_{23} = 0.50$.

The a priori class probability $P(\omega_i)$, $i = 1,2$ for each class is estimated as the number of vectors in the respective class divided by the total number of vectors. In our case, $P(\omega_1) = P(\omega_2) = 0.5$.

Step 2. Use function *mixture_Bayes* to classify the data vectors of Z and function *compute_error* to obtain the classification error. Type

```
for j=1:2
  le=length(S_hat{j});
  te=[];
  for i=1:le
    te(:,:,i)=S_hat{j}(i)*eye(2);
  end
  S{j}=te;
end
```

```
[y_est]=mixture_Bayes(m_hat,S,w_hat,P_hat,Z);
[classification_error]=compute_error([ones(1,500) 2*ones(1,500)],y_est);
```

The computed classification error is equal to 4.20%. ■

Remark

- In a classification task, when the number of summands in the mixture model is not known, the task is run a number of times with different values of J; the value that results in the lowest classification error over the test set is adopted.

Exercise 1.6.1

Repeat Example 1.6.2 using $X1$, $X2$,$Z1$, $Z2$ with the following initial parameter values:

- For $p_1(x)$: Three normal distributions with initial mean estimates $m_{11,ini} = [5, 5]^T$, $m_{12,ini} = [5.5, 5.5]^T$, $m_{13,ini} = [5, 5]^T$; initial variance estimates $\sigma_{11,ini} = 0.2$, $\sigma_{12,ini} = 0.4$, $\sigma_{13,ini} = 0.3$; and mixing probabilities $P_{11,ini} = 0.2$, $P_{12,ini} = 0.4$, $P_{13,ini} = 0.4$.
  For $p_2(x)$: Three normal distributions with initial mean estimates $m_{21,ini} = [2, 2]^T$, $m_{22,ini} = [1.98, 1.98]^T$, $m_{23,ini} = [2.4, 2.4]^T$; initial variance estimates $\sigma_{21,ini} = 0.06$, $\sigma_{22,ini} = 0.05$, $\sigma_{23,ini} = 0.4$; and mixing probabilities $P_{21,ini} = 0.8$, $P_{22,ini} = 0.1$, $P_{23,ini} = 0.1$.
- For $p_1(x)$: Three normal distributions with initial mean estimates $m_{11,ini} = [1.6, 1.4]^T$, $m_{12,ini} = [1.4, 1.6]^T$, $m_{13,ini} = [1.3, 1.5]^T$; initial variance estimates $\sigma_{11,ini} = 0.2$, $\sigma_{12,ini} = 0.4$, $\sigma_{13,ini} = 0.3$; and mixing probabilities $P_{11,ini} = 0.2$, $P_{12,ini} = 0.4$, $P_{13,ini} = 0.4$.
  For $p_2(x)$: Three normal distributions with initial mean estimates $m_{21,ini} = [1.5, 1.7]^T$, $m_{22,ini} = [1.7, 1.5]^T$, $m_{23,ini} = [1.6, 1.6]^T$; initial variance estimates $\sigma_{21,ini} = 0.6$, $\sigma_{22,ini} = 0.05$, $\sigma_{23,ini} = 0.02$; and mixing probabilities $P_{21,ini} = 0.1$, $P_{22,ini} = 0.8$, $P_{23,ini} = 0.1$.
- For $p_1(x)$: Four normal distributions with initial mean estimates $m_{11,ini} = [0, 2]^T$, $m_{12,ini} = [5, 2]^T$, $m_{13,ini} = [5, 5]^T$, $m_{14,ini} = [3, 4]^T$; initial variance estimates $\sigma_{11,ini} = 0.15$, $\sigma_{12,ini} = 0.27$, $\sigma_{13,ini} = 0.4$, $\sigma_{14,ini} = 0.2$; and mixing probabilities $P_{11,ini} = P_{12,ini} = P_{13,ini} = P_{14,ini} = 1/4$.
  For $p_2(x)$: Four normal distributions with initial mean estimates $m_{21,ini} = [1, 2]^T$, $m_{22,ini} = [3.2, 1.5]^T$, $m_{23,ini} = [1, 4]^T$, $m_{24,ini} = [4, 2]^T$; initial variance estimates $\sigma_{21,ini} = 0.15$, $\sigma_{22,ini} = 0.08$, $\sigma_{23,ini} = 0.27$, $\sigma_{24,ini} = 0.05$; and mixing probabilities $P_{21,ini} = P_{22,ini} = P_{23,ini} = P_{24,ini} = 1/4$.
- For $p_1(x)$: Two normal distributions with initial mean estimates $m_{11,ini} = [0, 2]^T$, $m_{12,ini} = [5, 2]^T$; initial variance estimates $\sigma_{11,ini} = 0.15$, $\sigma_{12,ini} = 0.27$; and mixing probabilities $P_{11,ini} = P_{12,ini} = 1/2$.
  For $p_2(x)$: One normal distribution with initial mean estimate $m_{21,ini} = [1, 2]^T$; initial variance estimate $\sigma_{21,ini} = 0.15$; and mixing probability $P_{21,ini} = 1$.
- For $p_1(x)$: One normal distribution with initial mean estimate $m_{11,ini} = [2, 2]^T$; initial variance estimates $\sigma_{11,ini} = 0.4$; and mixing probability $P_{11,ini} = 1$.
  For $p_2(x)$: One normal distribution with initial mean estimate $m_{21,ini} = [1, 2]^T$; initial variance estimate $\sigma_{21,ini} = 0.15$; and mixing probability $P_{21,ini} = 1$.

For each scenario comment on the EM estimates and find the classification error of the Bayesian classifier.

1.7 PARZEN WINDOWS

This section and the following section deal with *nonparametric* estimation of an unknown pdf associated with a given set of data points. According to the Parzen windows pdf estimation method, if we are given N data points, $x_i \in \mathcal{R}^l$, $i = 1, 2, \ldots, N$, that follow an unknown distribution, their pdf can be estimated

using the expansion

$$p(x) \approx \frac{1}{Nh^l} \sum_{i=1}^{N} \phi\left(\frac{x - x_i}{h}\right) \tag{1.8}$$

for sufficiently *large* N and sufficiently *small* values of h, which is a user-defined parameter [Theo 09, Section 2.5.6], $\phi(\cdot)$ is an appropriately defined *kernel* function. A commonly used kernel function is the Gaussian, and in this case the expansion becomes

$$p(x) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{1}{(2\pi)^{l/2} h^l} \exp\left(-\frac{(x - x_i)^T (x - x_i)}{2h^2}\right) \tag{1.9}$$

Example 1.7.1. Generate $N = 1000$ data points lying in the real axis, $x_i \in \mathcal{R}$, $i = 1, 2, \ldots, N$, from the following pdf, and plot $p(x)$:

$$p(x) = \frac{1}{3} \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{x^2}{2\sigma_1^2}\right) + \frac{2}{3} \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{(x-2)^2}{2\sigma_2^2}\right)$$

where $\sigma_1^2 = \sigma_2^2 = 0.2$.

Use the Parzen windows approximation of Eq. (1.9), with $h = 0.1$, and plot the obtained estimate.

Solution. The pdf is actually a Gaussian mixture model. Use the function *generate_gauss_classes* to generate the required data set, typing

```
m=[0; 2]';
S(:,:,1)=[0.2];

S(:,:,2)=[0.2];
P=[1/3 2/3];
N=1000;
randn('seed',0);
[X]=generate_gauss_classes(m,S,P,N);
```

Step 1. To plot the pdf, assume $x \in [-5, 5]$ and type

```
x=-5:0.1:5;
pdfx=(1/3)*(1/sqrt(2*pi*0.2))*exp(-(x.^2)/0.4)
     +(2/3)*(1/sqrt(2*pi*0.2))*exp(-((x-2).^2)/0.4);
plot(x,pdfx); hold;
```

Step 2. To compute and plot the approximation of the pdf for $h = 0.1$ and $x \in [-5, 5]$, use function *Parzen_gauss_kernel* as follows:

```
h=0.1;
pdfx_approx=Parzen_gauss_kernel(X,h,-5,5);
plot(-5:h:5,pdfx_approx,'r');
```

Exercise 1.7.1
Repeat the experiment in Example 1.7.1 with $h = 0.01$, $N = 1000$ and $h = 0.1$, $N = 10{,}000$. Comment on the results. The choice of h for a given N needs careful consideration. Tips related to this choice are provided in [Theo 09, Section 2.5.6] and the references therein.

Exercise 1.7.2
Generate $N = 1000$ data points from the following 2-dimensional pdf:

$$p(x) \equiv p(x(1),x(2)) = \frac{1}{3}\frac{1}{2\pi\sigma^2}\exp\left\{-\frac{x^2(1)+x^2(2)}{2\sigma^2}\right\} + \frac{2}{3}\frac{1}{2\pi\sigma^2}\exp\left\{-\frac{x^2(1)+(x(2)-2)^2}{2\sigma^2}\right\}$$

Repeat the experiment in Example 1.7.1.

Exercise 1.7.3
Use the setup for the classification task in Example 1.4.3. Classify the data points of the set X_1 using the Bayesian classifier, where the estimate of the required values $p(x|\omega_1)$, $p(x|\omega_2)$ for each point in X_1 is obtained via the Parzen window estimation method. Use different values of h and choose the one that results in the best error performance of the classifier.

1.8 k-NEAREST NEIGHBOR DENSITY ESTIMATION

Let us consider a set of N points, $x_1, x_2, \ldots, x_N \in \mathcal{R}^l$, that stem from a statistical distribution unknown to us. The goal is to *estimate* the value of the unknown pdf at a given point x. According to the k-nearest neighbor estimation technique, the following steps are performed:

1. Choose a value for k.
2. Find the distance between x and all training points x_i, $i = 1, 2, \ldots, N$. Any distance measure can be used (e.g., Euclidean, Mahalanobis).
3. Find the k-nearest points to x.
4. Compute the volume $V(x)$ in which the k-nearest neighbors lie.
5. Compute the estimate by

$$p(x) \approx \frac{k}{NV(x)}$$

If the Euclidean distance is employed and the distance between the k-furthest neighbor and x is ρ, the volume $V(x)$ is equal to

$$V(x) = 2\rho \quad \text{in the 1-dimensional space}$$

$$V(x) = \pi\rho^2 \quad \text{in the 2-dimensional space}$$

or

$$V(x) = \frac{4}{3}\pi\rho^3 \quad \text{in the 3-dimensional space}$$

For the more general case of l dimensions and/or Mahalanobis distance, see [Theo 09, Section 2.5.6].

Example 1.8.1. Consider the data set generated in Example 1.7.1 and use the k-nearest neighbor density estimator to estimate the required pdf with $k = 21$.

Solution. To generate the set X of the data vectors, work as in Example 1.7.1. Assuming that we are interested in approximating the pdf for $x \in [-5, 5]$ (as in Example 1.7.1), we use the function *knn_density_estimate*, typing

```
pdfx_approx=knn_density_estimate(X,21,-5,5,0.1);
plot(-5:0.1:5,pdfx_approx,'r');
```

Exercise 1.8.1
Repeat Example 1.8.1 for $k = 5, 100$. Repeat with $N = 5000$.

Exercise 1.8.2
Use the setup for the classification task in Example 1.4.3. Classify the data points of set X_1 using the Bayesian classifier. Estimate the required values $p(x|\omega_1)$, $p(x|\omega_2)$ for each point in X_1 via the k-nearest neighbor density estimation method. Use different values of k and choose the one that results in the best error performance of the classifier.

1.9 THE NAIVE BAYES CLASSIFIER

In the naive Bayes classification scheme, the required estimate of the pdf at a point $x = [x(1), \ldots, x(l)]^T \in \mathcal{R}^l$ is given as

$$p(x) = \prod_{j=1}^{l} p(x(j))$$

That is, the components (features) of the feature vector x are assumed to be *statistically independent*. This assumption is convenient in high-dimensional spaces, where, because of the curse of dimensionality [Theo 09, Section 2.5.6], a large number of training points should be available to obtain a reliable estimate of the corresponding multidimensional pdf. Instead, with the naive Bayes classifier, although the independence assumption may not be valid, the final performance may still be good since reliable estimates of the 1-dimensional pdfs can be obtained with relatively few data points.

Example 1.9.1. Generate a set X_1 that consists of $N_1 = 50$ 5-dimensional data vectors that stem from two equiprobable classes, ω_1 and ω_2. The classes are modeled by Gaussian distributions with means

$m_1 = [0,0,0,0,0]^T$ and $m_2 = [1,1,1,1,1]^T$ and respective covariance matrices

$$S_1 = \begin{bmatrix} 0.8 & 0.2 & 0.1 & 0.05 & 0.01 \\ 0.2 & 0.7 & 0.1 & 0.03 & 0.02 \\ 0.1 & 0.1 & 0.8 & 0.02 & 0.01 \\ 0.05 & 0.03 & 0.02 & 0.9 & 0.01 \\ 0.01 & 0.02 & 0.01 & 0.01 & 0.8 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 0.9 & 0.1 & 0.05 & 0.02 & 0.01 \\ 0.1 & 0.8 & 0.1 & 0.02 & 0.02 \\ 0.05 & 0.1 & 0.7 & 0.02 & 0.01 \\ 0.02 & 0.02 & 0.02 & 0.6 & 0.02 \\ 0.01 & 0.02 & 0.01 & 0.02 & 0.7 \end{bmatrix}$$

In a similar manner, generate a data set X_2 consisting of $N_2 = 10{,}000$ data points. X_1 is used for training; X_2, for testing.

In the spirit of the naive Bayes classifier, we assume that for each class the features of the feature vectors are statistically independent (although we know this is not true), and that each follows a 1-dimensional Gaussian distribution. For each of the five dimensions and for each of the two classes, use the training set X_1 to compute the maximum likelihood estimates of the mean values m_{1j}, m_{2j}, $j = 1,2,\ldots,5$ and the variances σ_{1j}^2, σ_{2j}^2, $j = 1,2,\ldots,5$.

Perform the following steps:

Step 1. Classify the points of the test set X_2 using the naive Bayes classifier, where for a given x, $p(x|\omega_i)$ is estimated as

$$p(x|\omega_i) = \prod_{j=1}^{5} \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp\left(-\frac{(x(j) - m_{ij})^2}{2\sigma_{ij}^2}\right), \; i = 1,2$$

where $x(j)$ is the jth component of x. Compute the error probability.

Step 2. Compute the ML estimates of m_1, m_2, S_1, and S_2 using X_1. Employ the ML estimates in the Bayesian classifier in the 5-dimensional space. Compute the error probability.

Step 3. Compare the results obtained in steps 1 and 2.

Solution. To generate sets X_1 and X_2, type

```
m=[zeros(5,1) ones(5,1)];
S(:,:,1)=[0.8 0.2 0.1 0.05 0.01;
          0.2 0.7 0.1 0.03 0.02;
          0.1 0.1 0.8 0.02 0.01;
          0.05 0.03 0.02 0.9 0.01;
          0.01 0.02 0.01 0.01 0.8];
S(:,:,2)=[0.9 0.1 0.05 0.02 0.01;
          0.1 0.8 0.1  0.02 0.02;
          0.05 0.1 0.7 0.02 0.01;
          0.02 0.02 0.02 0.6 0.02;
          0.01 0.02 0.01 0.02 0.7];
P=[1/2 1/2]'; N_1=100;
randn('state',0);
[X1,y1]=generate_gauss_classes(m,S,P,N_1);
```

```
N_2=10000;
randn('state',100);
[X2,y2]=generate_gauss_classes(m,S,P,N_2);
```

Assuming that the features are independent, use function *Gaussian_ML_estimate* to compute the ML estimate of the mean and the variance per feature for each class (using set X_1). Type

```
for i=1:5
  [m1_hat(i), S1_hat(i)]=Gaussian_ML_estimate(X1(i,find(y1==1)));
end
m1_hat=m1_hat'; S1_hat=S1_hat';

for i=1:5
  [m2_hat(i), S2_hat(i)]=Gaussian_ML_estimate(X1(i,find(y1==2)));
end
m2_hat=m2_hat'; S2_hat=S2_hat';
```

Then, do the following:

Step 1. To classify each point in X_2 according to the naive Bayes classification scheme, type

```
for i=1:5
  perFeature1(i,:)=normpdf(X2(i,:),m1_hat(i),sqrt(S1_hat(i)));
  perFeature2(i,:)=normpdf(X2(i,:),m2_hat(i),sqrt(S2_hat(i)));
end
naive_probs1=prod(perFeature1);
naive_probs2=prod(perFeature2);
classified=ones(1,length(X2));
classified(find(naive_probs1<naive_probs2))=2;
```

To compute the classification error, type

```
true_labels=y2;
naive_error=sum(true_labels~=classified)/length(classified)
```

Step 2. To compute the maximum likelihood estimates of the "unknown" mean values and covariance matrices m_1, m_2, S_1, and S_2, based on X_1, type

```
[m1_ML, S1_ML]=Gaussian_ML_estimate(X1(:,find(y1==1)));
[m2_ML, S2_ML]=Gaussian_ML_estimate(X1(:,find(y1==2)));
```

To classify the data vectors of X_2 using the Bayesian classifier, which is based on the ML estimates of the respective parameters, type

```
m_ML(:,1)=m1_ML;
m_ML(:,2)=m2_ML;
```

```
S_ML(:,:,1)=S1_ML;
S_ML(:,:,2)=S2_ML;
P=[1/2 1/2];
z=bayes_classifier(m_ML,S_ML,P,X2);
```

To compute the classification error, type

```
true_labels=y2;
Bayes_ML_error=sum(true_labels~=z)/length(z)
```

Step 3. The resulting classification errors—*naive_error* and *Bayes_ML_error*—are 0.1320 and 0.1426, respectively. In other words, the naive classification scheme outperforms the standard ML-based scheme. If the experiment is repeated for the case where X_1 consists of 20 instead of 50 points, the difference between the performance of the two classifiers is even more noticeable in favor of the naive Bayes classifier. ■

Exercise 1.9.1

1. Classify the points of the set X_2 in Example 1.9.1, adopting the optimal Bayesian classifier. That is, use the true values of the means and covariance matrices associated with the 5-dimensional Gaussian pdfs. Compare the results with those obtained in Example 1.9.1.
2. Repeat Example 1.9.1 with X_1 consisting of $N_1 = 1000$ data vectors.

Remark

- The previous example is very important in the sense that it demonstrates that it is often preferable to use *suboptimal* searching techniques if the use of the optimal method results in excessive computations and/or poor estimates due to a limited amount of data. This is often the case in high-dimensional spaces because of the *curse of dimensionality* [Theo 09, Section 2.5.6].

1.10 THE NEAREST NEIGHBOR RULE

Nearest neighbor (NN) is one of the most popular classification rules, although it is an old technique. We are given c classes, ω_i, $i = 1, 2, \ldots, c$, and a point $x \in \mathcal{R}^l$, and N *training* points, x_i, $i = 1, 2, \ldots, N$, in the l-dimensional space, with the corresponding class labels. Given a point, x, whose class label is unknown, the task is to classify x in one of the c classes. The rule consists of the following steps:

1. Among the N training points, search for the k neighbors closest to x using a distance measure (e.g., Euclidean, Mahalanobis). The parameter k is user-defined. Note that it should not be a multiple of c. That is, for two classes k should be an odd number.
2. Out of the k-closest neighbors, identify the number k_i of the points that belong to class ω_i. Obviously, $\sum_{i=1}^{c} k_i = k$.
3. Assign x to class ω_i, for which $k_i > k_j$, $j \neq i$. In other words, x is assigned to the class in which the majority of the k-closest neighbors belong.

For large N (in theory $N \to \infty$), the larger k is the closer the performance of the k-NN classifier to the optimal Bayesian classifier is expected to be [Theo 09, Section 2.6]. However, for small values of N (in theory, for its finite values), a larger k may not result in better performance [Theo 09, Problem 2.34].

A major problem with the k-NN classifier, as well as with its close relative the k-NN density estimator, is the computational complexity associated with searching for the k-nearest neighbors, especially in high-dimensional spaces. This search is repeated every time a new point x is classified, for which a number of suboptimal techniques have been suggested [Theo 09, Section 2.6].

Example 1.10.1

1. Consider a 2-dimensional classification problem where the data vectors stem from two equiprobable classes, ω_1 and ω_2. The classes are modeled by Gaussian distributions with means $m_1 = [0,0]^T$, $m_2 = [1,2]^T$, and respective covariance matrices

$$S_1 = S_2 = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

   Generate two data sets X_1 and X_2 consisting of 1000 and 5000 points, respectively.
2. Taking X_1 as the training set, classify the points in X_2 using the k-NN classifier, with $k = 3$ and adopting the squared Euclidean distance. Compute the classification error.

Solution

Step 1. To generate sets X_1 and X_2, type

```
m=[0 0; 1 2]';
S=[0.8 0.2;0.2 0.8];
S(:,:,1)=S;S(:,:,2)=S;
P=[1/2 1/2]'; N_1=1000;
randn('seed',0)
[X1,y1]=generate_gauss_classes(m,S,P,N_1);
N_2=5000;
randn('seed',100)
[X2,y2]=generate_gauss_classes(m,S,P,N_2);
```

Step 2. For the classification task, use function *k_nn_classifier* and type

```
k=3;
z=k_nn_classifier(X1,y1,k,X2);
```

To compute the classification error, type

```
pr_err=sum(z~=y2)/length(y2)
```

The classification error is 15.12%. Note that different seeds for the *randn* function are likely to lead to slightly different results. ■

Exercise 1.10.1

Repeat Example 1.10.1 for $k = 1, 7, 15$. For each case compute the classification error rate. Compare the results with the error rate obtained by the optimal Bayesian classifier, using the true values of the mean and the covariance matrix.

Exercise 1.10.2

Compose your own example of a 2-class classification task in the 5-dimensional space. Assume the data to follow the Gaussian pdf in both classes. Choose the mean values and covariance matrices. Produce two data sets, one for training and one for testing. Use the nearest neighbor classifier. Experiment with different values of the mean values, the covariance matrix, the parameter k, and the length of the training data set. Comment on the obtained results as well as the computational time involved.