CHAPTER

14 Clustering Algorithms III: Schemes Based on Function Optimization

14.1 INTRODUCTION

One of the most commonly used families of clustering schemes relies on the optimization of a cost function J using differential calculus techniques (e.g., see [Duda 01, Bezd 80, Bobr 91, Kris 95a, Kris 95b]). The cost J is a function of the vectors of the data set X and it is parameterized in terms of an unknown parameter vector, $\boldsymbol{\theta}$. For most of the schemes of the family, the number of clusters, m, is assumed to be known.

Our goal is the estimation of $\boldsymbol{\theta}$ that characterizes best the clusters underlying X. The parameter vector $\boldsymbol{\theta}$ is strongly dependent on the shape of the clusters. For example, for compact clusters (see Figure 14.1a), it is reasonable to adopt as parameters a set of m points, $\boldsymbol{m}_i$, in the l-dimensional space, each corresponding to a cluster—thus, $\boldsymbol{\theta} = [\boldsymbol{m}_1{}^T, \boldsymbol{m}_2{}^T, \ldots, \boldsymbol{m}_m{}^T]^T$. On the other hand, if ring-shaped clusters are expected (see Figure 14.1b), it is reasonable to use m hyperspheres $C(\boldsymbol{c}_i, r_i)$, $i = 1, \ldots, m$, as representatives, where $\boldsymbol{c}_i$ and r_i are the center and the radius of the ith hypersphere, respectively. In this case, $\boldsymbol{\theta} = [\boldsymbol{c}_1{}^T, r_1, \boldsymbol{c}_2{}^T, r_2, \ldots, \boldsymbol{c}_m{}^T, r_m]^T$.

Spherical or, in general, shell-shaped clusters[1] are encountered in many robot vision applications. The basic problem here is the identification of objects (patterns) lying in a *scene* (which is a region in the three-dimensional space), and the estimation of their relative positions, using a single or several *images* (two-dimensional projections of the scene). An important task of this problem is the identification of the boundaries of the objects in the image. Given an image (see, e.g., Figure 14.2a), we may identify the pixels that constitute the boundary of the objects using appropriate operators (see, e.g., [Horn 86, Kare 94]) (see Figure 14.2b). Then, the boundaries of the objects may be considered as shell-shaped or linear-shaped clusters and clustering algorithms may be mobilized in order to recover their exact shape and location in the image. In fact, clustering techniques have exhibited

[1] These may be hyperellipsoids, hyperparabolas, etc.

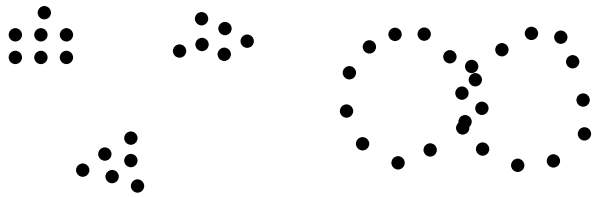

FIGURE 14.1

(a) Compact clusters. (b) Spherical clusters.

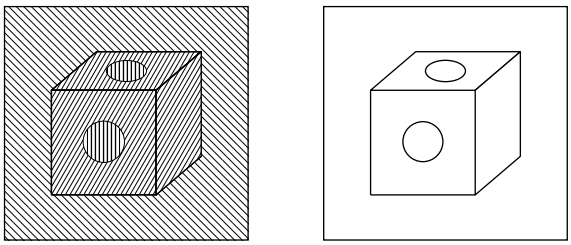

FIGURE 14.2

(a) The original image of a scene. (b) The image after the application of appropriate operators.

satisfactory results at least when the boundaries are known to be shell shaped (e.g., [Kris 95a]).

A distinct characteristic of most of the algorithms of this chapter, compared with the algorithms of the previous chapter, is that the cluster representatives are computed using *all* the available vectors of X, and not only the vectors that have been assigned to the respective cluster. We will focus on four major categories of algorithms: the mixture decomposition, the fuzzy, the possibilistic and the hard clustering algorithms. In the first, the cost function is constructed on the basis of random vectors, and assignment to clusters follows probabilistic arguments, in the spirit of the Bayesian classification. The conditional probabilities here result from the optimization process. In the fuzzy approach a proximity function between a vector and a cluster is defined, and the "grade of membership" of a vector in a cluster is provided by the set of membership functions. As is always the case with fuzzy approaches, the values of the membership functions of a vector in the various clusters are interrelated. This constraint is removed in the case of the possibilistic approach. Finally, hard clustering may be viewed as a special case of the fuzzy clustering approach, where each vector belongs exclusively to a cluster. This category includes the celebrated *k-means* clustering algorithm.

14.2 MIXTURE DECOMPOSITION SCHEMES

The basic reasoning behind this algorithmic family springs from our familiar Bayesian philosophy. We assume that there are m clusters, $C_j, j = 1, \ldots, m$, underlying the data set.[2] Each vector $\boldsymbol{x}_i$, $i = 1, \ldots, N$, belongs to a cluster C_j with probability $P(C_j|\boldsymbol{x}_i)$. A vector $\boldsymbol{x}_i$ is appointed to the cluster C_j if

$$P(C_j|\boldsymbol{x}_i) > P(C_k|\boldsymbol{x}_i), \quad k = 1, \ldots, m, k \neq j$$

The differences from the classification task of Chapter 2 are that (a) no training data with known cluster labeling are available and (b) the *a priori* probabilities $P(C_j) \equiv P_j$ are not known either. Thus, although the goal is the same, the tools have to be different. Basically, this is a typical task with an incomplete training data set. We are missing the corresponding cluster labeling information for each data point $\boldsymbol{x}_i$. Thus, the task fits nicely in the framework introduced in Section 2.5.5.

From Eq. (2.81) and adopting the notation for the needs of the current chapter we have

$$Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}(t)) = \sum_{i=1}^{N} \sum_{j=1}^{m} P(C_j|\boldsymbol{x}_i; \boldsymbol{\Theta}(t)) \ln \left(p(\boldsymbol{x}_i|C_j; \boldsymbol{\theta}) P_j \right), \tag{14.1}$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_m^T]^T$, with $\boldsymbol{\theta}_j$ being the parameter vector corresponding to the j-th cluster, $\boldsymbol{P} = [P_1, \ldots, P_m]^T$, with P_j being the *a priori* probability for the jth cluster and $\boldsymbol{\Theta} = [\boldsymbol{\theta}^T, \boldsymbol{P}^T]^T$. The above equation results from application of the E-step of the EM algorithm. The M-step of the algorithm is

$$\boldsymbol{\Theta}(t+1) = arg \max_{\boldsymbol{\Theta}} Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}(t)). \tag{14.2}$$

Assuming that all pairs of $\boldsymbol{\theta}_k, \boldsymbol{\theta}_j$'s are functionally independent, that is, no $\boldsymbol{\theta}_k$ gives any information about $\boldsymbol{\theta}_j (j \neq i)$, we estimate $\boldsymbol{\theta}_j$ from Eq. (14.2) as follows:

$$\sum_{i=1}^{N} \sum_{j=1}^{m} P(C_j|\boldsymbol{x}_i; \boldsymbol{\Theta}(t)) \frac{\partial}{\partial \boldsymbol{\theta}_j} \ln p(\boldsymbol{x}_i|C_j; \boldsymbol{\theta}_j) = \boldsymbol{0} \tag{14.3}$$

Maximization with respect to $\boldsymbol{P}$ is a constraint optimization problem since

$$P_k \geq 0, \quad k = 1, \ldots, m, \quad \text{and} \quad \sum_{k=1}^{m} P_k = 1 \tag{14.4}$$

The corresponding Lagrangian function is

$$\mathcal{Q}(\boldsymbol{P}, \lambda) = Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}(t)) - \lambda \left(\sum_{k=1}^{m} P_k - 1 \right) \tag{14.5}$$

[2] Recall that the number m is assumed to be known.

Taking the partial derivative of $\mathcal{Q}(\boldsymbol{P}, \lambda)$ with respect to P_j and setting it equal to 0, and after some algebra we obtain

$$P_j = \frac{1}{\lambda} \sum_{i=1}^{N} P(C_j|\boldsymbol{x}_i; \boldsymbol{\Theta}(t)), \quad j = 1, \ldots, m \tag{14.6}$$

Substituting the above equations into Eq. (14.4), we obtain

$$\lambda = \sum_{i=1}^{N} \sum_{j=1}^{m} P(C_j|\boldsymbol{x}_i; \boldsymbol{\Theta}(t)) = N \tag{14.7}$$

Thus, Eq. (14.6) gives

$$P_j = \frac{1}{N} \sum_{i=1}^{N} P(C_j|\boldsymbol{x}_i; \boldsymbol{\Theta}(t)) \quad j = 1, \ldots, m \tag{14.8}$$

Taking into account Eqs. (14.3), (14.8), and (2.87), the EM algorithm for this case may be stated as

Generalized Mixture Decomposition Algorithmic Scheme (GMDAS)

- Choose initial estimates, $\boldsymbol{\theta} = \boldsymbol{\theta}(0)$ and $\boldsymbol{P} = \boldsymbol{P}(0)$.[3]
- $t = 0$
- Repeat
  - Compute

$$P(C_j|\boldsymbol{x}_i; \boldsymbol{\Theta}(t)) = \frac{p(\boldsymbol{x}_i|C_j; \boldsymbol{\theta}_j(t))P_j(t)}{\sum_{k=1}^{m} p(\boldsymbol{x}_i|C_k, \boldsymbol{\theta}_k(t))P_k(t)} \tag{14.9}$$

    $i = 1, \ldots, N, \quad j = 1, \ldots, m.$

  - Set $\boldsymbol{\theta}_j(t+1)$ equal to the solution of the equation

$$\sum_{i=1}^{N} \sum_{j=1}^{m} P(C_j|\boldsymbol{x}_i; \boldsymbol{\Theta}(t)) \frac{\partial}{\partial \boldsymbol{\theta}_j} \ln p(\boldsymbol{x}_i|C_j; \boldsymbol{\theta}_j) = \mathbf{0} \tag{14.10}$$

    with respect to $\boldsymbol{\theta}_j$, for $j = 1, \ldots, m$.

  - Set

$$P_j(t+1) = \frac{1}{N} \sum_{i=1}^{N} P(C_j|\boldsymbol{x}_i; \boldsymbol{\Theta}(t)), \quad j = 1, \ldots, m \tag{14.11}$$

  - $t = t + 1$
- Until convergence, with respect to $\boldsymbol{\Theta}$, is achieved.

[3] Initial conditions must satisfy the constraints.

A suitable termination criterion for the algorithm is the following:

$$\|\boldsymbol{\Theta}(t+1) - \boldsymbol{\Theta}(t)\| < \varepsilon$$

where $\|\cdot\|$ is an appropriate vector norm and ε is a "small" user-defined constant. This scheme is guaranteed to converge to a global or a local maximum of the log-likelihood function. However, even if a local maximum solution is reached, it may still capture satisfactorily the underlying clustering structure of X.

Once the algorithm has converged, vectors are assigned to clusters according to the final estimates $P(C_j|\boldsymbol{x}_i)$ of Eq. (14.9). Hence, the task now becomes a typical Bayesian classification problem, if we treat each cluster as a separate class.

14.2.1 Compact and Hyperellipsoidal Clusters

In this section, we focus our attention on the case in which the vectors of X form compact clusters. A distribution that is suitable for clusters of this scheme is the normal distribution, that is,

$$p(\boldsymbol{x}|C_j;\boldsymbol{\theta}_j) = \frac{1}{(2\pi)^{l/2}|\Sigma_j|^{1/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T\Sigma_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\Big), \quad j=1,\ldots,m \tag{14.12}$$

or

$$\ln p(\boldsymbol{x}|C_j;\boldsymbol{\theta}_j) = \ln\frac{|\Sigma_j|^{-1/2}}{(2\pi)^{l/2}} - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T\Sigma_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j), \quad j=1,\ldots,m \tag{14.13}$$

In this case, each vector $\boldsymbol{\theta}_j$ consists of the l parameters of the mean $\boldsymbol{\mu}_j$ and the $l(l+1)/2$ independent parameters of Σ_j. A parameter reduction may be obtained by assuming that the covariance matrices are diagonal. If this assumption is too strict, another commonly used assumption is that all covariance matrices are equal. In the former case $\boldsymbol{\theta}$ consists of $2ml$ parameters, while in the latter it consists of $ml + l(l+1)/2$ parameters.

Combining Eq. (14.12) and Eq. (14.9) results in

$$P(C_j|\boldsymbol{x};\boldsymbol{\Theta}(t)) = \frac{|\Sigma_j(t)|^{-1/2}\exp\big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j(t))^T\Sigma_j^{-1}(t)(\boldsymbol{x}-\boldsymbol{\mu}_j(t))\big)P_j(t)}{\sum_{k=1}^{m}|\Sigma_k(t)|^{-1/2}\exp\big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k(t))^T\Sigma_k^{-1}(t)(\boldsymbol{x}-\boldsymbol{\mu}_k(t))\big)P_k(t)} \tag{14.14}$$

In the sequel, we consider the problem in its most general form; that is, we assume that all the means $\boldsymbol{\mu}_j$ and the covariance matrices Σ_j are unknown. We also assume that, in general, all Σ_j's are different from each other. Following an approach similar to the one described in Chapter 2, the updating equations for $\boldsymbol{\mu}_j$'s and Σ_j's from the M-step are

$$\boldsymbol{\mu}_j(t+1) = \frac{\sum_{k=1}^{N} P(C_j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))\boldsymbol{x}_k}{\sum_{k=1}^{N} P(C_j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))} \tag{14.15}$$

and

$$\Sigma_j(t+1) = \frac{\sum_{k=1}^{N} P(C_j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))(\boldsymbol{x}_k - \boldsymbol{\mu}_j(t))(\boldsymbol{x}_k - \boldsymbol{\mu}_j(t))^T}{\sum_{k=1}^{N} P(C_j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))} \tag{14.16}$$

$j = 1, \ldots, m$.

Thus, in the Gaussian case these two equations replace Eq. (14.10), and Eq. (14.14) replaces Eq. (14.9) in the corresponding steps of the GMDAS algorithm.

Remark

- Notice that this scheme is computationally very demanding, because at each iteration step the inverses of m covariance matrices are required for the computation of $P(C_j|\boldsymbol{x}_i; \boldsymbol{\Theta}(t))$. As stated earlier, one way to relax this demand is to assume that all covariance matrices are diagonal or that all are equal to each other. In the latter case, only one inversion is required at each iteration step.

Example 14.1

(a) Consider three 2-dimensional normal distributions with means $\boldsymbol{\mu}_1 = [1, 1]^T$, $\boldsymbol{\mu}_2 = [3.5, 3.5]^T$, $\boldsymbol{\mu}_3 = [6, 1]^T$ and covariance matrices

$$\Sigma_1 = \begin{bmatrix} 1 & -0.3 \\ -0.3 & 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix}, \quad \Sigma_3 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$

respectively.

A group of 100 vectors is generated from each distribution. These groups constitute the data set $\boldsymbol{X}$. Figure 14.3a is a plot of the generated data.

We initialize $P_j = 1/3$, $j = 1, 2, 3$. Also, we set $\boldsymbol{\mu}_i(0) = \boldsymbol{\mu}_i + \boldsymbol{y}_i$, where $\boldsymbol{y}_i$ is an 2×1 vector with random coordinates, uniformly distributed in the interval $[-1, 1]^T$. Similarly, we define $\Sigma_i(0)$, $i = 1, 2, 3$. We set $\varepsilon = 0.01$. Using these initial conditions, the GMDAS for Gaussian pdf's terminates after 38 iterations. The final parameter estimates obtained are $\boldsymbol{P}' = [0.35, 0.31, 0.34]^T$, $\boldsymbol{\mu}_1' = [1.28, 1.16]^T$, $\boldsymbol{\mu}_2' = [3.49, 3.68]^T$, $\boldsymbol{\mu}_3' = [5.96, 0.84]^T$, and

$$\Sigma_1' = \begin{bmatrix} 1.45 & 0.01 \\ 0.01 & 0.57 \end{bmatrix}, \quad \Sigma_2' = \begin{bmatrix} 0.62 & 0.09 \\ 0.09 & 0.74 \end{bmatrix}, \quad \Sigma_3' = \begin{bmatrix} 0.30 & 0.0024 \\ 0.0024 & 1.94 \end{bmatrix}$$

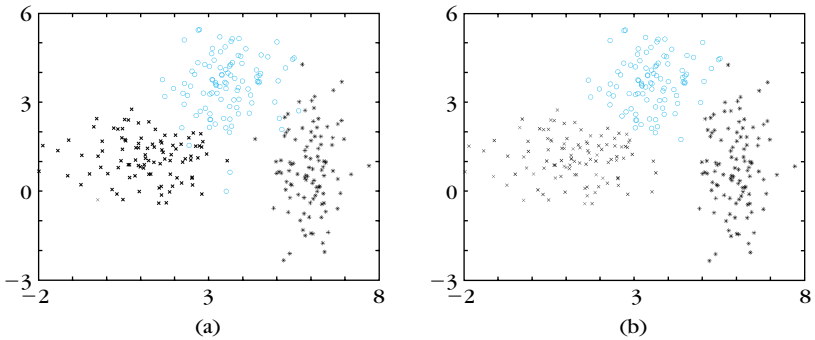

FIGURE 14.3

(a) A data set that consists of three groups of points. (b) The results from the application of GMDAS when normal mixtures are used.

For comparison, the sample mean values are $\hat{\boldsymbol{\mu}}_1 = [1.16, 1.13]^T$, $\hat{\boldsymbol{\mu}}_2 = [3.54, 3.56]^T$, $\hat{\boldsymbol{\mu}}_3 = [5.97, 0.76]^T$, respectively. Also, the sample covariance matrices are

$$\hat{\Sigma}_1 = \begin{bmatrix} 1.27 & -0.03 \\ -0.03 & 0.52 \end{bmatrix}, \quad \hat{\Sigma}_2 = \begin{bmatrix} 0.70 & 0.07 \\ 0.07 & 0.98 \end{bmatrix}, \quad \hat{\Sigma}_3 = \begin{bmatrix} 0.32 & 0.05 \\ 0.05 & 1.81 \end{bmatrix}$$

respectively.

As we can see, the final estimates of the algorithm are close enough to the means and the covariance matrices of the three groups of vectors.

Once the unknown parameters of the model have been estimated, the data vectors are assigned to clusters according to the estimated values of $P(C_j|\boldsymbol{x}_i)$. Figure 14.3b shows the assignment of points to the three clusters, which is in close agreement with the original structure. A way to assess the performance of the resulting model estimates is via the so-called *confusion matrix*. This is a matrix A whose (i, j) element is the number of vectors that originate from the ith distribution and are assigned to the jth cluster.[4] For our example this is

$$A_1 = \begin{bmatrix} 99 & 0 & 1 \\ 0 & 100 & 0 \\ 3 & 4 & 93 \end{bmatrix}$$

This example indicates that 99% of the data from the first distribution are assigned to the same cluster (the first one). Similarly, all the data from the second distribution are assigned to the same cluster (the second one) and, finally, 93% of the data from the third distribution are assigned to the same cluster (the third one).

[4] It should be noted here that in real clustering applications the confusion matrix cannot be defined, since we do not know *a priori* the cluster where each feature vector belongs. However, we may use it in artificial experiments such as this one, in order to evaluate the performance of the clustering algorithms.

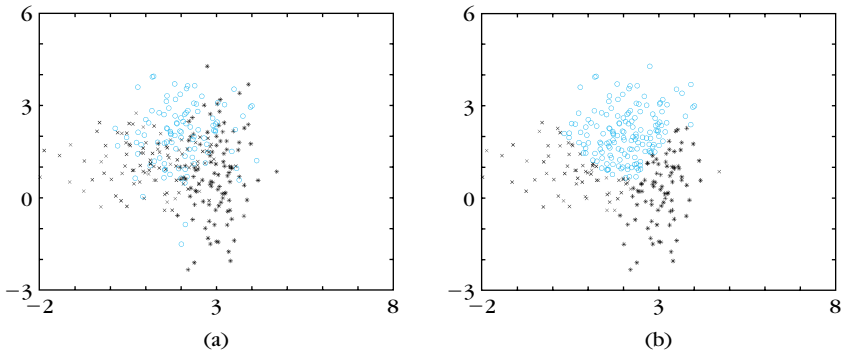

FIGURE 14.4

(a) The data set, which consists of three overlapping groups of points. (b) The results of the GMDAS when Gaussian mixtures are used.

(b) Let us now consider the case in which the three normal distributions are located closer, for example, $\boldsymbol{\mu}_1 = [1, 1]^T$, $\boldsymbol{\mu}_2 = [2, 2]^T$, $\boldsymbol{\mu}_3 = [3, 1]^T$, and with the same covariance matrices as before (Figure 14.4). We initialize $\boldsymbol{\mu}_i$ and Σ_i, $i = 1, 2, 3$, as in the previous case and run the GMDAS for Gaussian pdf's. The confusion matrix for this case is

$$A_2 = \begin{bmatrix} 85 & 4 & 11 \\ 35 & 56 & 9 \\ 26 & 0 & 74 \end{bmatrix}$$

As expected, each one of the obtained clusters contains a significant percentage of points from more than one distribution.

Example 14.2

The data set X, which is depicted in Figure 14.5a, consists of two intersecting ring-shaped clusters. Each cluster consists of 500 points. We run the GMDAS with Gaussians and $m = 2$ and $\varepsilon = 0.01$. The algorithm terminates after 72 iterations, and the results are shown in Figure 14.5b. *As expected, the algorithm fails to recover the underlying clustering structure of X, because it seeks compact clusters. Generally speaking, GMDAS using Gaussians reveals clusters that are as compact as possible, even though the clusters underlying X may have a different shape.* Even worse, it will identify clusters in X even though there is no clustering structure in it.[5]

[5] In general, before we apply any clustering algorithm to identify clusters contained in X, we should first check whether *there exists* any clustering structure in X. This procedure refers to *clustering tendency* and is considered in Chapter 16.

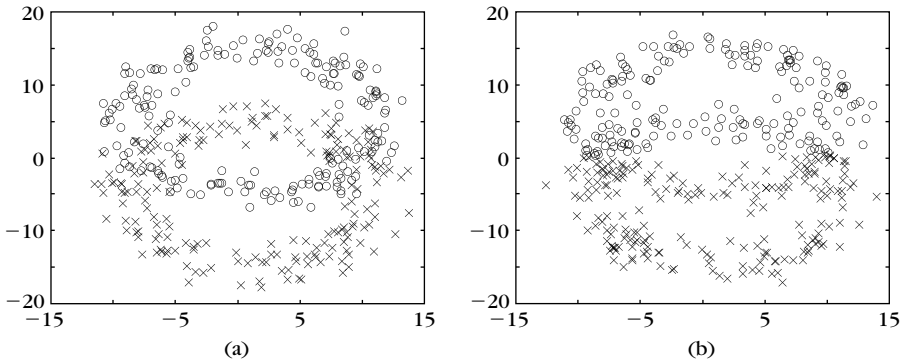

FIGURE 14.5

(a) A data set that consists of ring-shaped intersecting clusters. (b) The results from the application of GMDAS when Gaussian mixtures are used.

In [Zhua 96], the case of Gaussian pdf's, contaminated by unknown outlier distributions, $h(\boldsymbol{x}_i|C_j)$, is considered. In this case, we can write $p(\boldsymbol{x}|C_j) = (1-\varepsilon_j)G(\boldsymbol{x}|C_j) + \varepsilon_j h(\boldsymbol{x}|C_j)$, where ε_j is the level of contamination and $G(\boldsymbol{x}|C_j)$ is the jth Gaussian distribution. Under the assumption that all $h(\boldsymbol{x}_i|C_j)$ are constant, that is, $h(\boldsymbol{x}_i|C_j) = c_j$, $i = 1, \ldots, N$, $p(\boldsymbol{x}|C_j)$ may be written as $p(\boldsymbol{x}|C_j) = (1-\varepsilon_j)G(\boldsymbol{x}|C_j) + \varepsilon_j c_j$. Then we may use the preceding methodology in order to identify the mean and the covariance matrices of the normal distributions $G(\boldsymbol{x}|C_j)$ as well as the values of ε_j and c_j.

In [Figu 02] an alternative mixture decomposition scheme is proposed, which does not demand *a priori* knowledge of m and, in addition, it does not require careful initialization.

14.2.2 A Geometrical Interpretation

As mentioned earlier, the conditional probability, $P(C_j|\boldsymbol{x}_i)$, indicates how likely it is that $\boldsymbol{x}_i \in X$ belongs to C_j, $i = 1, \ldots, N$, subject, of course, to the constraint

$$\sum_{j=1}^{m} P(C_j|\boldsymbol{x}_i) = 1 \tag{14.17}$$

This may be viewed as the equation of an $(m-1)$-dimensional hyperplane. For notational purposes, let $P(C_j|\boldsymbol{x}_i) \equiv y_j, j = 1, \ldots, m$. Then Eq. (14.17) may be written as

$$\boldsymbol{a}^T \boldsymbol{y} = 1 \tag{14.18}$$

where $\boldsymbol{y}^T = [y_1, \ldots, y_m]$ and $\boldsymbol{a}^T = [1, 1, \ldots, 1]$. That is, $\boldsymbol{y}$ is allowed to move on the hyperplane defined by the previous equation. In addition, since $0 \leq y_j \leq 1$, $j = 1, \ldots, m$, $\boldsymbol{y}$ lies also inside the unit hypercube (see Figure 14.6).

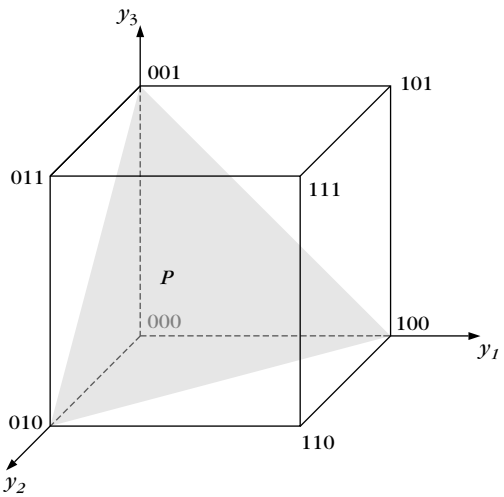

FIGURE 14.6

The hypercube for $m = 3$. The point $\boldsymbol{y}$ is allowed to move only on the shaded region of P.

This interpretation allows us to derive some useful conclusions for the so-called *noisy feature vectors* or *outliers*. Let $\boldsymbol{x}_i$ be such a vector. Since Eq. (14.17) holds for $\boldsymbol{x}_i$, at least one of the y_j's, $j = 1, \ldots, m$, is significant (it lies in the interval $[1/m, 1]$). Thus, $\boldsymbol{x}_i$ will affect, at least, the estimates for the corresponding cluster C_j, through Eqs. (14.9), (14.10), and (14.11), and this makes GMDAS sensitive to outliers. The following example clarifies this idea further.

Example 14.3

Consider the data set X shown in Figure 14.7. It consists of 22 vectors. The first (next) 10 vectors are drawn from the normal distribution with mean $\boldsymbol{\mu}_1 = [0, 0]^T$ ($\boldsymbol{\mu}_2 = [4.5, 4.5]^T$) and covariance matrix $\Sigma_1 = I$ ($\Sigma_2 = I$), where I is the 2×2 identity matrix. The last two points are $\boldsymbol{x}_{21} = [-6, 5]^T$ and $\boldsymbol{x}_{22} = [11, 0]^T$, respectively. We run the GMDAS for Gaussian pdf's on X. The estimates of $\boldsymbol{P}$, $\boldsymbol{\mu}_j$, and Σ_j, $j = 1, 2$, obtained after five iterations, are

$$\boldsymbol{P}' = [0.5, 0.5]^T, \quad \boldsymbol{\mu}_1' = [-0.58, 0.35]^T, \quad \boldsymbol{\mu}_2' = [4.98, 4.00]^T$$

$$\Sigma_1' = \begin{bmatrix} 4.96 & -2.01 \\ -2.01 & 2.63 \end{bmatrix}, \quad \Sigma_2' = \begin{bmatrix} 3.40 & -2.53 \\ -2.53 & 3.27 \end{bmatrix}$$

The resulting values of $P(C_j|\boldsymbol{x}_i)$, $j = 1, 2$, $i = 1, \ldots, 22$, are shown in Table 14.1. Although $\boldsymbol{x}_{21}$ and $\boldsymbol{x}_{22}$ may be considered as outliers, since they lie away from the two clusters, we get that $P(C_1|\boldsymbol{x}_{21}) = 1$ and $P(C_2|\boldsymbol{x}_{22}) = 1$, due to the constraint $\sum_{j=1}^{2} P(C_j|\boldsymbol{x}_i) = 1$. This

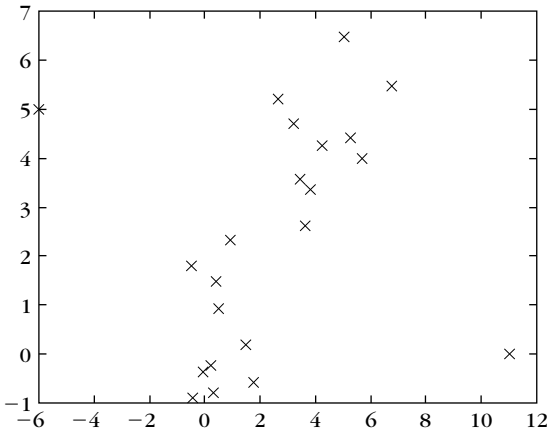

FIGURE 14.7
The data set for Example 14.3.

Table 14.1 The Resulting *a Posteriori* Probabilities for the Data Set of Example 14.3

feat. vec.	$P(C_1\|x)$	$P(C_2\|x)$	feat. vec.	$P(C_1\|x)$	$P(C_2\|x)$
x_1	0	1	x_{12}	1	0
x_2	0	1	x_{13}	1	0
x_3	0	1	x_{14}	1	0
x_4	0	1	x_{15}	1	0
x_5	0	1	x_{16}	1	0
x_6	0	1	x_{17}	1	0
x_7	0	1	x_{18}	1	0
x_8	0	1	x_{19}	0.99	0.01
x_9	0	1	x_{20}	1	0
x_{10}	0	1	x_{21}	0	1
x_{11}	1	0	x_{22}	1	0

implies that these points have a nonnegligible impact on $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, Σ_1, and Σ_2. Indeed, if we run GMDAS for Gaussian pdf's on $X_1 = \{\boldsymbol{x}_i : i = 1, \ldots, 20\}$ (i.e., we exclude the last two points), using the same initial conditions, we obtain after five iterations:

$$\boldsymbol{P}'' = [0.5, 0.5]^T, \quad \boldsymbol{\mu}_1'' = [-0.03, -0.12]^T, \quad \boldsymbol{\mu}_2'' = [4.37, 4.40]^T$$

$$\Sigma_1'' = \begin{bmatrix} 0.50 & -0.01 \\ -0.01 & 1.22 \end{bmatrix}, \qquad \Sigma_2'' = \begin{bmatrix} 1.47 & 0.44 \\ 0.44 & 1.13 \end{bmatrix}$$

Comparing the results of the two experiments, it is easily observed that the last setup gives more accurate estimates of the unknown parameters.

Another interesting observation can be derived by examining the following situation. Let $l = 1$. Consider two clusters described by normal distributions $p(x|C_j), j = 1, 2$, with the same variance and means μ_1 and μ_2, respectively ($\mu_1 < \mu_2$). Also let $P_1 = P_2$. It is not difficult to prove that for $x < (>) \frac{\mu_1+\mu_2}{2}$, $P(C_1|x) > (<) P(C_2|x)$. Now consider the points $x_1 = \frac{3\mu_1+\mu_2}{4}$ and $x_2 = \frac{5\mu_1-\mu_2}{4}$. Although these points have the same distance from μ_1 (i.e., they are symmetric with respect to μ_1), it is not hard to show that $P(C_1|x_1) > P(C_1|x_2)$. This happens because $P(C_1|x)$ and $P(C_2|x)$ are *related* through Eq. (14.17). Thus, the probability of having x in one cluster is affected by the probability of belonging to the other. We will soon see that we can free ourselves from such an interrelation.

14.3 FUZZY CLUSTERING ALGORITHMS

One of the difficulties associated with the previously discussed probabilistic algorithms is the involvement of the pdf's, for which a suitable model has to be assumed. In addition, it is not easy to handle cases where the clusters are not compact but are shell shaped. A family of clustering algorithms that emancipates itself from such constraints is that of fuzzy clustering algorithms. These schemes have been the subject of intensive research during the past three decades. The major point that differentiates the two approaches is that in the fuzzy schemes a vector *belongs simultaneously* to more than one cluster, whereas in the probabilistic schemes, each vector belongs *exclusively* to a single cluster.

As already discussed in Chapter 11, a *fuzzy m-clustering of X* is defined by a set of functions $u_j : X \to A, j = 1, \ldots, m$, where $A = [0, 1]$.

In the case where $A = \{0, 1\}$, a *hard m-clustering of X* is defined. In this case, each vector belongs exclusively to a single cluster.

As in the previous section, it is assumed that the number of clusters as well as their shape is known *a priori*. The shape of the clusters is characterized by the adopted set of parameters. For example, if we deal with compact clusters, a point representative is used to represent each cluster; that is, each cluster is represented by l parameters. On the other hand, if we deal with noncompact but, say, hyperspherical clusters, a hypersphere is used as a representative of each cluster. In this case, each cluster is represented by $l + 1$ parameters (l for the center of the hypersphere and 1 for its radius).

In the sequel we use the following notation: $\boldsymbol{\theta}_j$ is the parameterized representative of the jth cluster, $\boldsymbol{\theta} \equiv [\boldsymbol{\theta}_1{}^T, \ldots, \boldsymbol{\theta}_m{}^T]^T$, U is an $N \times m$ matrix whose (i, j)

element equals $u_j(\boldsymbol{x}_i)$, $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ is the dissimilarity between $\boldsymbol{x}_i$ and $\boldsymbol{\theta}_j$, and $q(>1)$ is a parameter called a *fuzzifier*. The role of the latter will be clarified shortly. Most of the well-known fuzzy clustering algorithms are those derived by minimizing a cost function of the form

$$J_q(\boldsymbol{\theta}, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^q d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) \tag{14.19}$$

with respect to $\boldsymbol{\theta}$ and U, subject to the constraints

$$\sum_{j=1}^{m} u_{ij} = 1, \quad i = 1, \ldots, N \tag{14.20}$$

where

$$\begin{aligned} u_{ij} \in [0, 1], \quad i = 1, \ldots, N, \quad j = 1, \ldots, m, \\ 0 < \sum_{i=1}^{N} u_{ij} < N, \quad j = 1, 2, \ldots, m \end{aligned} \tag{14.21}$$

In other words, the grade of membership of $\boldsymbol{x}_i$ in the jth cluster is related to the grade of membership of $\boldsymbol{x}_i$ to the rest $m - 1$ clusters through Eq. (14.20). Different values of q in Eq. (14.19) bias $J_q(\boldsymbol{\theta}, U)$ toward either the fuzzy or the hard clusterings. More specifically, for fixed $\boldsymbol{\theta}$, *if* $q = 1$, *no fuzzy clustering is better than the best hard clustering in terms of* $J_q(\boldsymbol{\theta}, U)$. *However, if* $q > 1$, *there are cases in which fuzzy clusterings lead to lower values of* $J_q(\boldsymbol{\theta}, U)$ *than the best hard clustering.* Let us clarify these ideas further using the following example.

Example 14.4

Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}$, where $\boldsymbol{x}_1 = [0, 0]^T$, $\boldsymbol{x}_2 = [2, 0]^T$, $\boldsymbol{x}_3 = [0, 3]^T$, $\boldsymbol{x}_4 = [2, 3]^T$. Let $\boldsymbol{\theta}_1 = [1, 0]^T$, $\boldsymbol{\theta}_2 = [1, 3]^T$ be the cluster representatives. Suppose also that the Euclidean distance between a vector and a representative is in use. The hard two-cluster clustering that minimizes $J_q(\boldsymbol{\theta}, U)$, for the above choice of $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$, can be represented by

$$U_{hard} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

The value of $J_q(\boldsymbol{\theta}, U)$ in this case Eq. (14.19) is $J_q^{hard}(\boldsymbol{\theta}, U) = 4$. Obviously, hard clusterings do not depend on q.

Assume now that $q = 1$ and u_{ij}'s are between 0 and 1. Then the value of the cost function becomes

$$J_1^{fuzzy}(\boldsymbol{\theta}, U) = \sum_{i=1}^{2} (u_{i1} + u_{i2}\sqrt{10}) + \sum_{i=3}^{4} (u_{i1}\sqrt{10} + u_{i2})$$

Since for each $\boldsymbol{x}_i$ both u_{i1} and u_{i2} are positive and $u_{i1} + u_{i2} = 1$, it easily follows that $J_1^{fuzzy}(\boldsymbol{\theta}, U) > 4$. Thus, the hard clustering always results in better values of $J_q^{fuzzy}(\boldsymbol{\theta}, U)$, compared with their fuzzy counterparts, when $q = 1$.

Assume now that $q = 2$. The reader should easily verify that when $u_{i2} \in [0, 0.48]$ for $i = 1, 2$ and $u_{i1} \in [0, 0.48]$ for $i = 3, 4$, and, of course, $u_{i1} = 1 - u_{i2}$, for each $\boldsymbol{x}_i$, then the value of $J_2^{fuzzy}(\boldsymbol{\theta}, U)$ is less than 4 (see Problem 14.7). Thus, in this case fuzzy clusterings are favored over hard ones.

Finally, let $q = 3$. In this case, it is easily verified that when $u_{i2} \in [0, 0.67]$ for $i = 1, 2$ and $u_{i1} \in [0, 0.67]$ for $i = 3, 4$ and $u_{i1} = 1 - u_{i2}$, for each $\boldsymbol{x}_i$, then the value of $J_3^{fuzzy}(\boldsymbol{\theta}, U)$ is also less than 4.

Minimization of $J_q(\boldsymbol{\theta}, U)$

We first assume that no $\boldsymbol{x}_i$ coincides with any of the representatives. More formally, for an $\boldsymbol{x}_i$ let Z_i be the set that contains the indices of the representatives $\boldsymbol{\theta}_j$ for which $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = 0$. According to our assumption, $Z_i = \emptyset$, for all i. In the sequel, for ease of notation, we drop the index q from $J_q(\boldsymbol{\theta}, U)$. Let us consider first U. Minimization of $J_q(\boldsymbol{\theta}, U)$ with respect to U, subject to the constraint (14.20), leads to the following Lagrangian function:

$$\mathcal{J}(\boldsymbol{\theta}, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) - \sum_{i=1}^{N} \lambda_i \left(\sum_{j=1}^{m} u_{ij} - 1 \right) \tag{14.22}$$

The partial derivative of $\mathcal{J}(\boldsymbol{\theta}, U)$ with respect to u_{rs} is

$$\frac{\partial \mathcal{J}(\boldsymbol{\theta}, U)}{\partial u_{rs}} = q u_{rs}^{q-1} d(\boldsymbol{x}_r, \boldsymbol{\theta}_s) - \lambda_r \tag{14.23}$$

Setting $\partial \mathcal{J}(\boldsymbol{\theta}, U)/\partial u_{rs}$ equal to 0 and solving with respect to u_{rs}, we obtain

$$u_{rs} = \left(\frac{\lambda_r}{q d(\boldsymbol{x}_r, \boldsymbol{\theta}_s)} \right)^{\frac{1}{q-1}}, \quad s = 1, \ldots, m \tag{14.24}$$

Substituting u_{rs} from the previous equation in the constraint equation $\sum_{j=1}^{m} u_{rj} = 1$, we obtain

$$\sum_{j=1}^{m} \left(\frac{\lambda_r}{q d(\boldsymbol{x}_r, \boldsymbol{\theta}_j)} \right)^{\frac{1}{q-1}} = 1$$

or

$$\lambda_r = \frac{q}{\left(\sum_{j=1}^{m} \left(\frac{1}{d(\boldsymbol{x}_r, \boldsymbol{\theta}_j)} \right)^{\frac{1}{q-1}} \right)^{q-1}} \tag{14.25}$$

Combining Eq. (14.25) with (14.24) and using a bit of algebra, we obtain

$$u_{rs} = \frac{1}{\sum_{j=1}^{m} \left(\frac{d(\boldsymbol{x}_r, \boldsymbol{\theta}_s)}{d(\boldsymbol{x}_r, \boldsymbol{\theta}_j)} \right)^{\frac{1}{q-1}}} \tag{14.26}$$

$r = 1, \ldots, N, s = 1, \ldots, m$.

Now consider the parameter vector $\boldsymbol{\theta}_j$. Taking the gradient of $J(\boldsymbol{\theta}, U)$ with respect to $\boldsymbol{\theta}_j$ and setting it equal to zero, we obtain

$$\frac{\partial J(\boldsymbol{\theta}, U)}{\partial \boldsymbol{\theta}_j} = \sum_{i=1}^{N} u_{ij}^q \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} = \mathbf{0}, \quad j = 1, \ldots, m \tag{14.27}$$

Equations (14.26) and (14.27) are coupled and, in general, cannot give closed-form solutions. One way to proceed is to employ the following iterative algorithmic scheme, in order to obtain estimates for U and $\boldsymbol{\theta}$.

Generalized Fuzzy Algorithmic Scheme (GFAS)

- Choose $\boldsymbol{\theta}_j(0)$ as initial estimates for $\boldsymbol{\theta}_j, j = 1, \ldots, m$.
- $t = 0$
- Repeat
  - For $i = 1$ to N
    - For $j = 1$ to m
      - $u_{ij}(t) = \dfrac{1}{\sum_{k=1}^{m} \left(\frac{d(\boldsymbol{x}_i, \boldsymbol{\theta}_j(t))}{d(\boldsymbol{x}_i, \boldsymbol{\theta}_k(t))} \right)^{\frac{1}{q-1}}}$
    - End {For-j}
  - End {For-i}
  - $t = t + 1$
  - For $j = 1$ to m
    - *Parameter updating:* Solve

$$\sum_{i=1}^{N} u_{ij}^q (t-1) \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} = \mathbf{0} \tag{14.28}$$

      with respect to $\boldsymbol{\theta}_j$ and set $\boldsymbol{\theta}_j(t)$ equal to this solution.
  - End {For-j}
- Until a termination criterion is met.

As the termination criterion we may employ $\|\boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1)\| < \varepsilon$, where $\|\cdot\|$ is any vector norm and ε is a "small" user-defined constant.

Remarks

- If, for a given $\boldsymbol{x}_i$, $Z_i \neq \emptyset$, we arbitrarily choose u_{ij}'s, with $j \in Z_i$, such that $\sum_{j \in Z_i} u_{ij} = 1$ and $u_{ij} = 0$, for $j \notin Z_i$. That is, $\boldsymbol{x}_i$ is shared arbitrarily among the clusters whose representatives coincide with $\boldsymbol{x}_i$, subject to the constraint (14.20). In the case in which $\boldsymbol{x}_i$ coincides with a single representative, say $\boldsymbol{\theta}_j$, the condition becomes $u_{ij} = 1$, and $u_{ik} = 0, k \neq j$.
- The algorithmic scheme may also be initialized from *U(0)* instead of $\boldsymbol{\theta}_j(0)$, $j = 1, \ldots, m$, and start iterations with computing $\boldsymbol{\theta}_j$ first.
- The above iterative algorithmic scheme is also known as the *alternating optimization (AO) scheme*, since at each iteration step U is updated for fixed $\boldsymbol{\theta}$, and then $\boldsymbol{\theta}$ is updated for fixed U ([Bezd 95, Hopp 99]).

In the sequel the algorithm is specialized to three commonly encountered cases.

14.3.1 Point Representatives

In the case of compact clusters, a point representative is used for each cluster; that is, $\boldsymbol{\theta}_j$ consists of l parameters. In this case, the dissimilarity $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ may be any distance measure between two points. Two common choices for $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ are (see also Chapter 11)

$$d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = (\boldsymbol{x}_i - \boldsymbol{\theta}_j)^T A (\boldsymbol{x}_i - \boldsymbol{\theta}_j) \tag{14.29}$$

where A is a symmetric, positive definite matrix, and the *Minkowski distance*,

$$d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = \left(\sum_{k=1}^{l} |x_{ik} - \theta_{jk}|^p \right)^{\frac{1}{p}} \tag{14.30}$$

where p is a positive integer and x_{ik}, θ_{jk} are the kth coordinates of $\boldsymbol{x}_i$ and $\boldsymbol{\theta}_j$, respectively. Let us now see the specific form of GFAS under these choices.

- When the distance given in Eq. (14.29) is in use, we have

$$\frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} = 2A(\boldsymbol{\theta}_j - \boldsymbol{x}_i) \tag{14.31}$$

Substituting Eq. (14.31) into Eq. (14.28), we obtain

$$\sum_{i=1}^{N} u_{ij}^q (t-1) 2A(\boldsymbol{\theta}_j - \boldsymbol{x}_i) = \mathbf{0}$$

Since A is positive definite, it is invertible. Premultiplying both sides of this equation with A^{-1} and after some simple algebra, we obtain

$$\boldsymbol{\theta}_j(t) = \frac{\sum_{i=1}^{N} u_{ij}^q(t-1)\boldsymbol{x}_i}{\sum_{i=1}^{N} u_{ij}^q(t-1)} \tag{14.32}$$

The resulting algorithm is also known as *Fuzzy c-Means (FCM)*[6] or *Fuzzy k-means algorithm* and has been discussed extensively in the literature (e.g. [Bezd 80, Cann 86, Hath 86, Hath 89, Isma 86]).

- Let us now examine the case in which Minkowski distances are in use. In the sequel, we consider only the case where p is even and $p < +\infty$. In this case, we can guarantee the differentiability of $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ with respect to $\boldsymbol{\theta}_j$. Equation (14.30) then gives

$$\frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \theta_{jr}} = \frac{(\theta_{jr} - x_{ir})^{p-1}}{\left(\sum_{k=1}^{l} |x_{ik} - \theta_{jk}|^p\right)^{1-\frac{1}{p}}}, \quad r = 1, \ldots, l \tag{14.33}$$

Substituting Eq. (14.33) into Eq. (14.28), we obtain

$$\sum_{i=1}^{N} u_{ij}^q(t-1) \frac{(\theta_{jr} - x_{ir})^{p-1}}{\left(\sum_{k=1}^{l} |x_{ik} - \theta_{jk}|^p\right)^{1-\frac{1}{p}}} = 0, \quad r = 1, \ldots, l \tag{14.34}$$

Hence, we end up with a system of l nonlinear equations and l unknowns, that is, the coordinates of $\boldsymbol{\theta}_j$. This can be solved by an iterative technique, such as the Gauss–Newton or the Levenberg–Marquardt (L-M) method (e.g., [Luen 84]).

The resulting algorithms are also known as *pFCM*, where p indicates the employed Minkowski distance ([Bobr 91]).

In the iterative technique, the initial estimates at step t can be the estimates obtained from the previous iteration step $t-1$.

Example 14.5

(a) Consider the setup of Example 14.1(a). We run the GFAS first for the distance defined in Eq. (14.29), when (i) A is the identity 2×2 matrix, and (ii) $A = \begin{bmatrix} 2 & 1.5 \\ 1.5 & 2 \end{bmatrix}$, and (iii) the Minkowski distance with $p = 4$ is used. The algorithm is initialized as in the Example 14.1, with $\boldsymbol{\theta}_j$ in the place of $\boldsymbol{\mu}_j$. The fuzzifier q was set equal to 2.

The estimates for $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, and $\boldsymbol{\theta}_3$ are $\boldsymbol{\theta}_1 = [1.37, 0.71]^T$, $\boldsymbol{\theta}_2 = [3.14, 3.12]^T$, and $\boldsymbol{\theta}_3 = [5.08, 1.21]^T$ for case (i), $\boldsymbol{\theta}_1 = [1.47, 0.56]^T$, $\boldsymbol{\theta}_2 = [3.54, 1.97]^T$, and $\boldsymbol{\theta}_3 = [5.21,\ 2.97]^T$ for

[6] A variant of the FCM tailored to a specific medical application is discussed in [Siya 05].

case (ii), and $\boldsymbol{\theta}_1 = [1.13,\ 0.74]^T$, $\boldsymbol{\theta}_2 = [2.99, 3.16]^T$, and $\boldsymbol{\theta}_3 = [5.21, 3.16]^T$ for case (iii). The corresponding confusion matrices (see Example 14.1) are

$$A_i = \begin{bmatrix} 98 & 2 & 0 \\ 14 & 84 & 2 \\ 11 & 0 & 89 \end{bmatrix}, \quad A_{ii} = \begin{bmatrix} 63 & 11 & 26 \\ 5 & 95 & 0 \\ 39 & 23 & 38 \end{bmatrix}, \quad A_{iii} = \begin{bmatrix} 96 & 0 & 4 \\ 11 & 89 & 0 \\ 13 & 2 & 85 \end{bmatrix}$$

Notice that in the cases of A_i and A_{iii}, almost all vectors from the same distribution are assigned to the same cluster. Note that for the construction of the confusion matrices we took the liberty to assign each point $\boldsymbol{x}_i$ to the cluster, for which the respective u_{ij} has the maximum value.

(b) Let us now consider the setup of Example 14.1(b). We run the GFAS algorithm for the three cases described in (a). The estimates for $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, and $\boldsymbol{\theta}_3$ are $\boldsymbol{\theta}_1 = [1.60,\ 0.12]^T$, $\boldsymbol{\theta}_2 = [1.15, 1.67]^T$, and $\boldsymbol{\theta}_3 = [3.37, 2.10]^T$ for case (i), $\boldsymbol{\theta}_1 = [1.01, 0.38]^T$, $\boldsymbol{\theta}_2 = [2.25,\ 1.49]^T$, $\boldsymbol{\theta}_3 = [3.75, 2.68]^T$ for case (ii), and $\boldsymbol{\theta}_1 = [1.50, -0.13]^T$, $\boldsymbol{\theta}_2 = [1.25, 1.77]^T$, $\boldsymbol{\theta}_3 = [3.54, 1.74]^T$ for case (iii). The corresponding confusion matrices are

$$A_i = \begin{bmatrix} 51 & 46 & 3 \\ 14 & 47 & 39 \\ 43 & 0 & 57 \end{bmatrix}, \quad A_{ii} = \begin{bmatrix} 79 & 21 & 0 \\ 19 & 58 & 23 \\ 28 & 41 & 31 \end{bmatrix}, \quad A_{iii} = \begin{bmatrix} 51 & 3 & 46 \\ 37 & 62 & 1 \\ 11 & 36 & 53 \end{bmatrix}$$

Let us now comment on these results. First, as expected, the closer the clusters are, the worse the performance of all the algorithms. Also, when the distance given in Eq. (14.29) is employed, the choice of the matrix A is critical. For the case of our example, when $A = I$, the GFAS identifies almost perfectly the clusters underlying X, when they are not too close to each other. The same holds true for the Minkowski distance with $p = 4$.

Remarks

- The choice of the fuzzifier q in significant for the fuzzy clustering algorithms. Especially for the FCM, heuristic guidelines for the choice of q are given in [Bezd 81], while in [Gao 00] a method for selecting q based on fuzzy decision theory concepts is discussed.
- Several generalized FCM schemes have been proposed in the literature. These are derived from the minimization of cost functions that result from the basic one given in eq. (14.19) by adding suitable terms (see e.g. [Yang 93, Lin 96, Pedr 96, Ozde 02, Yu 03]).
- Kernelized versions of the FCM are discussed in [Chia 03, Shen 06, Zeyu 01, Zhan 03, Zhou 04]. Also, a comparative study of the kernelized verisons of FCM and the FCM itself is reported in [Grav 07].

14.3.2 Quadric Surfaces as Representatives

In this section we consider the case of clusters of quadric shape, such as hyperellipsoids and hyperparaboloids. In the sequel, we present four algorithms of this type, out of a large number that have appeared in the literature.

Our first concern is to define the distance between a point and a quadric surface, as Eq. (14.19) demands. The next section is devoted to the definition and the physical explanation of some commonly used distances of this kind.

Distances between a Point and a Quadric Surface

In this section we introduce definitions in addition to those discussed in Chapter 11 concerning the distance between a point and a quadric surface.

We recall that the general equation of a quadric surface, Q, is

$$\boldsymbol{x}^T A\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{x} + c = 0 \tag{14.35}$$

where A is an $l \times l$ symmetric matrix, $\boldsymbol{b}$ is an $l \times 1$ vector, c is a scalar, and $\boldsymbol{x} = [x_1, \ldots, x_l]^T$. The $A, \boldsymbol{b}$ and c quantities are the parameters defining Q. For various choices of these quantities we obtain hyperellipses, hyperparabolas, and so on. An alternative to the Eq. (14.35) form is easily verified to be (see Problem 14.8)

$$\boldsymbol{q}^T\boldsymbol{p} = 0 \tag{14.36}$$

where

$$\boldsymbol{q} = \Big[\overbrace{x_1^2, x_2^2, \ldots, x_l^2}^{l}, \overbrace{x_1x_2, \ldots, x_{l-1}x_l}^{l(l-1)/2}, \overbrace{x_1, x_2, \ldots, x_l, 1}^{l+1}\Big]^T \tag{14.37}$$

and

$$\boldsymbol{p} = [p_1, p_2, \ldots, p_l, p_{l+1}, \ldots, p_r, p_{r+1}, \ldots, p_s]^T \tag{14.38}$$

with $r = l(l+1)/2$ and $s = r + l + 1$. Vector $\boldsymbol{p}$ is easily derived from $A, \boldsymbol{b}$, and c so that Eq. (14.36) is satisfied.

Algebraic Distance

The *squared algebraic distance* between a point $\boldsymbol{x}$ and a quadric surface Q is defined as

$$d_a^2(\boldsymbol{x}, Q) = (\boldsymbol{x}^T A\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{x} + c)^2 \tag{14.39}$$

Using the alternative formulation in Eq. (14.36), $d_a^2(\boldsymbol{x}, Q)$ can be written as

$$d_a^2(\boldsymbol{x}, Q) = \boldsymbol{p}^T M\boldsymbol{p} \tag{14.40}$$

where $M = \boldsymbol{q}\boldsymbol{q}^T$. The algebraic distance could be seen as a generalization of the distance of a point from a hyperplane (see Chapter 11). Its physical meaning will become clear later on. For the derivation of the GFAS algorithm, based on the squared algebraic distance, it is more convenient to use the last formulation in (14.40).

Perpendicular Distance

Another distance between a point $\boldsymbol{x}$ and a quadric surface Q is the *squared perpendicular distance* defined as

$$d_p^2(\boldsymbol{x}, Q) = \min_{\boldsymbol{z}} \|\boldsymbol{x} - \boldsymbol{z}\|^2 \tag{14.41}$$

subject to the constraint that

$$\boldsymbol{z}^T A \boldsymbol{z} + \boldsymbol{b}^T \boldsymbol{z} + c = 0 \tag{14.42}$$

In words, this definition states that the distance between $\boldsymbol{x}$ and Q is defined as the squared Euclidean distance between $\boldsymbol{x}$ and the point $\boldsymbol{z}$ of Q closest to $\boldsymbol{x}$. $d_p(\boldsymbol{x}, Q)$ is the length of the perpendicular line segment from $\boldsymbol{x}$ to Q. Although this definition seems to be the most reasonable one from an intuitive point of view, its computation is not straightforward. More precisely, it involves Lagrangian formalization. Specifically, we define

$$\mathcal{D}(\boldsymbol{x}, Q) = \|\boldsymbol{x} - \boldsymbol{z}\|^2 - \lambda(\boldsymbol{z}^T A \boldsymbol{z} + \boldsymbol{b}^T \boldsymbol{z} + c) \tag{14.43}$$

Taking into account that A is symmetric, the gradient of $\mathcal{D}(\boldsymbol{x}, Q)$ with respect to $\boldsymbol{z}$ is

$$\frac{\partial \mathcal{D}(\boldsymbol{x}, Q)}{\partial \boldsymbol{z}} = 2(\boldsymbol{x} - \boldsymbol{z}) - 2\lambda A \boldsymbol{z} - \lambda \boldsymbol{b}$$

Setting $\partial \mathcal{D}(\boldsymbol{x}, Q)/\partial \boldsymbol{z}$ equal to $\mathbf{0}$ and after some algebra, we obtain

$$\boldsymbol{z} = \frac{1}{2}(I + \lambda A)^{-1}(2\boldsymbol{x} - \lambda \boldsymbol{b}) \tag{14.44}$$

To compute λ, we substitute $\boldsymbol{z}$ in Eq. (14.42), and we obtain a polynomial of λ of degree $2l$. For each of the real roots, λ_k, of this polynomial, we determine the corresponding $\boldsymbol{z}_k$. Then, $d_p(\boldsymbol{x}, Q)$ is defined as

$$d_p^2(\boldsymbol{x}, Q) = \min_{\boldsymbol{z}_k} \|\boldsymbol{x} - \boldsymbol{z}_k\|^2$$

Radial Distance

This distance is suitable when Q is a hyperellipsoidal. Then Eq. (14.35) can be brought into the form

$$(\boldsymbol{x} - \boldsymbol{c})^T A (\boldsymbol{x} - \boldsymbol{c}) = 1 \tag{14.45}$$

where $\boldsymbol{c}$ is the center of the ellipse and A is a symmetric positive definite matrix,[7] which determines the major and the minor axis of the ellipse as well as its orientation.

[7] Obviously, this matrix is in general different (yet related) from the A matrix used in Eq. (14.35). We use the same symbol for notational convenience.

The *squared radial distance* [Frig 96] between a point $\boldsymbol{x}$ and Q is defined as

$$d_r^2(\boldsymbol{x}, Q) = \|\boldsymbol{x} - \boldsymbol{z}\|^2 \tag{14.46}$$

subject to the constraints

$$(\boldsymbol{z} - \boldsymbol{c})^T A(\boldsymbol{z} - \boldsymbol{c}) = 1 \tag{14.47}$$

and

$$(\boldsymbol{z} - \boldsymbol{c}) = a(\boldsymbol{x} - \boldsymbol{c}) \tag{14.48}$$

In words, we first determine the intersection point, $\boldsymbol{z}$, between the line segment $\boldsymbol{x} - \boldsymbol{c}$ and Q, and then we compute $d_r(\boldsymbol{x}, Q)$ as the squared Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{z}$ (see Figure 14.8).

Normalized Radial Distance

The *squared normalized radial distance* between $\boldsymbol{x}$ and Q is also appropriate for hyperellipsoids and is defined as

$$d_{nr}^2(\boldsymbol{x}, Q) = \left(\left((\boldsymbol{x} - \boldsymbol{c})^T A(\boldsymbol{x} - \boldsymbol{c}) \right)^{1/2} - 1 \right)^2 \tag{14.49}$$

It can be shown (Problem 14.10) that

$$d_r^2(\boldsymbol{x}, Q) = d_{nr}^2(\boldsymbol{x}, Q)\|\boldsymbol{z} - \boldsymbol{c}\|^2 \tag{14.50}$$

where $\boldsymbol{z}$ is the intersection of the line segment $\boldsymbol{x} - \boldsymbol{c}$ with Q. This justifies the term "*normalized*."

The following examples give some insight into the distances that have been defined.

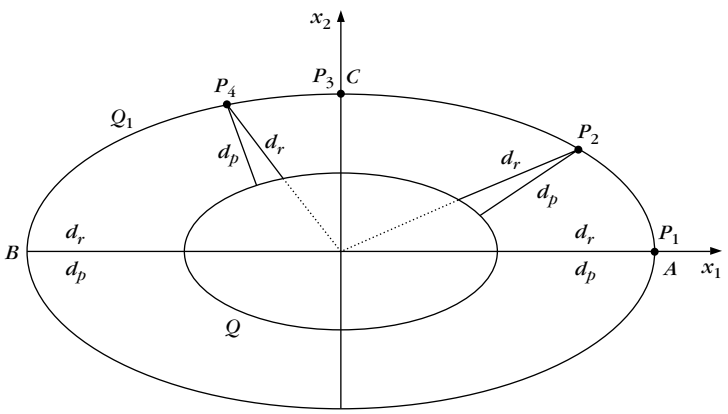

FIGURE 14.8

Graphical representation of the perpendicular and radial distances.

Example 14.6

Consider an ellipse Q centered at $\boldsymbol{c} = [0, 0]^T$, with

$$A = \begin{bmatrix} 0.25 & 0 \\ 0 & 1 \end{bmatrix}$$

and an ellipse Q_1 centered at $\boldsymbol{c} = [0, 0]^T$, with

$$A_1 = \begin{bmatrix} 1/16 & 0 \\ 0 & 1/4 \end{bmatrix}$$

Let $P(x_1, x_2)$ be a point in Q_1 moving from $A(4, 0)$ to $B(-4, 0)$ and always having its x_2 coordinate positive (Figure 14.8). Figure 14.9 illustrates how the four distances vary as P moves from A to B. One can easily observe that d_a and d_{nr} do not vary as P moves. This means that all points lying on an ellipse sharing the same center as Q and, having the same orientation as it, have the same d_a and d_{nr} distances from Q. However, this is not the case with the other two distances. Figure 14.8 shows graphically the d_p and d_r distances for various instances of P. As expected, the closer P is to the point $C(2, 0)$, the smaller the d_p and d_r distances. Also, Figure 14.8 indicates that d_r can be used as an approximation of d_p, since, as we saw earlier, it is hard to compute d_p. However, it should be recalled that d_p is applicable when general quadric surfaces are considered, whereas d_r is used only when hyperellipsoids are considered.

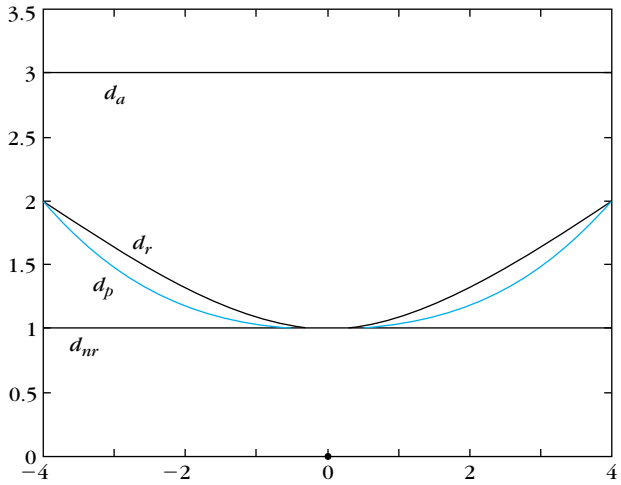

FIGURE 14.9

Variation of the distances d_p, d_a, d_{nr}, and d_r as P moves from $A(4, 0)$ to $B(-4, 0)$, with its x_2 coordinate being positive. The horizontal axis corresponds to the x_1 coordinate of the various points considered.

Example 14.7

Consider the two ellipses Q_1 and Q_2 shown in Figure 14.10, with equations

$$(\boldsymbol{x} - \boldsymbol{c}_j)^T A_j (\boldsymbol{x} - \boldsymbol{c}_j) = 1, \quad j = 1, 2$$

where $\boldsymbol{c}_1 = [0, 0]^T$, $\boldsymbol{c}_2 = [8,\ 0]^T$ and

$$A_1 = \begin{bmatrix} 1/16 & 0 \\ 0 & 1/4 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 1/4 & 0 \\ 0 & 1 \end{bmatrix}$$

Also consider the points $A(5, 0)$, $B(3, 0)$, $C(0, 2)$, and $D(5.25, 1.45)$. The distances d_a, d_p, d_{nr}, d_r between each of these points and Q_1 and Q_2 are shown in Table 14.2. From this table we observe that the d_p distances from A, B, and C to Q_1 are equal. Moreover, as expected, the d_r distance is always greater than or equal to d_p (when the equality holds?). Also, d_p is unbiased toward the size of the ellipses ($d_p(A, Q_1) = d_p(A, Q_2)$). Finally, d_a and d_{nr} are biased toward larger ellipses ($d_a(A, Q_1) < d_a(A, Q_2)$ and $d_{nr}(A, Q_1) < d_{nr}(A, Q_2)$).

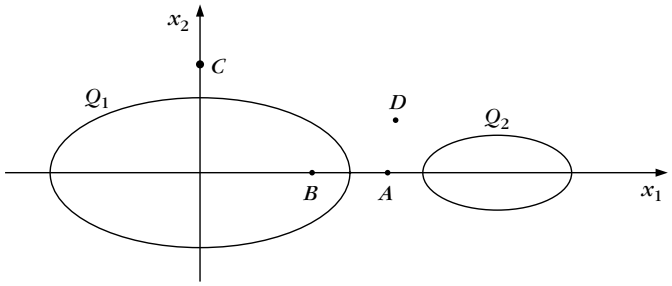

FIGURE 14.10

The setup of Example 14.7.

Table 14.2 Comparison of the various distances between points and hyperellipsoids

	d_a		d_p		d_{nr}		d_r	
	Q_1	Q_2	Q_1	Q_2	Q_1	Q_2	Q_1	Q_2
A	0.32	1.56	1	1	0.06	0.25	1	1
B	0.19	27.56	1	9	0.06	2.25	1	9
C	1.56	576	1	44.32	0.25	16	1	46.72
D	1.56	9.00	2.78	1.93	0.25	1	3.30	2.42

In the sequel, we derive some well-known algorithms suitable for shell-shaped clusters. These algorithms are usually called *fuzzy shell clustering algorithms*, and the representatives of the clusters are (in most cases) hyperquadrics.

Fuzzy Shell Clustering Algorithms

The first two algorithms that are examined are suitable for hyperellipsoid-shaped clusters. The first of them [Dave 92a, Dave 92b] is called the *adaptive fuzzy C-shells (AFCS) clustering algorithm*, and the second one is known as the *fuzzy C ellipsoidal shells (FCES) algorithm* [Kris 95a].

The Adaptive Fuzzy C-Shells (AFCS) Algorithm

The AFCS uses the squared distance d_{nr} between a point and a hyperellipsoidal (Eq. 14.49). Thus, Eq. (14.19) becomes

$$J_{nr}(\boldsymbol{\theta}, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^{q} d_{nr}^{2}(\boldsymbol{x}_i, Q_j) \tag{14.51}$$

It is clear that in this case the parameters used to identify a representative (an ellipse) are its center, $\boldsymbol{c}_j$, and the symmetric, positive definite matrix, A_j. Thus, the parameter vector $\boldsymbol{\theta}_j$ of the jth cluster contains the l parameters of $\boldsymbol{c}_j$ plus the $l(l+1)/2$ independent parameters of A_j, $j = 1, \ldots, m$. In the sequel, we write $d_{nr}(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ instead of $d_{nr}(\boldsymbol{x}_i, Q_j)$, in order to show explicitly the dependence on the parameter vector.

As in the case of point representatives, our first step is the computation of the gradient of $d_{nr}^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ with respect to $\boldsymbol{c}_j$ and A_j. The gradient $\partial d_{nr}^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j)/\partial \boldsymbol{c}_j$ after some algebra becomes

$$\frac{\partial d_{nr}^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{c}_j} = -2\frac{d_{nr}(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\phi(\boldsymbol{x}_i, \boldsymbol{\theta}_j)} A_j(\boldsymbol{x}_i - \boldsymbol{c}_j) \tag{14.52}$$

where

$$\phi^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = (\boldsymbol{x}_i - \boldsymbol{c}_j)^T A_j(\boldsymbol{x}_i - \boldsymbol{c}_j) \tag{14.53}$$

Let a_{rs}^j be the (r, s) element of A_j and x_{ir}, c_{jr} the rth coordinates of $\boldsymbol{x}_i$ and $\boldsymbol{c}_j$, respectively. Then, the partial derivative of $d_{nr}^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ with respect to a_{rs}^j, after some elementary manipulations, becomes

$$\frac{\partial d_{nr}^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial a_{rs}^j} = \frac{d_{nr}(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\phi(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}(x_{ir} - c_{jr})(x_{is} - c_{js})$$

Thus,

$$\frac{\partial d_{nr}^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial A_j} = \frac{d_{nr}(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\phi(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}(\boldsymbol{x}_i - \boldsymbol{c}_j)(\boldsymbol{x}_i - \boldsymbol{c}_j)^T \tag{14.54}$$

Substituting Eqs. (14.52) and (14.54) in (14.28), and after some minor manipulations, the parameter updating part of GFAS becomes

- Parameter updating:
  - Solve with respect to $\boldsymbol{c}_j$ and A_j the following equations.

$$\sum_{i=1}^{N} u_{ij}^{q}(t-1)\frac{d_{nr}(\boldsymbol{x}_i,\boldsymbol{\theta}_j)}{\phi(\boldsymbol{x}_i,\boldsymbol{\theta}_j)}(\boldsymbol{x}_i-\boldsymbol{c}_j)=\boldsymbol{0}$$

and

$$\sum_{i=1}^{N} u_{ij}^{q}(t-1)\frac{d_{nr}(\boldsymbol{x}_i,\boldsymbol{\theta}_j)}{\phi(\boldsymbol{x}_i,\boldsymbol{\theta}_j)}(\boldsymbol{x}_i-\boldsymbol{c}_j)(\boldsymbol{x}_i-\boldsymbol{c}_j)^T=O$$

where

$$\phi^2(\boldsymbol{x}_i,\boldsymbol{\theta}_j)=(\boldsymbol{x}_i-\boldsymbol{c}_j)^T A_j(\boldsymbol{x}_i-\boldsymbol{c}_j)$$

and

$$d_{nr}^2(\boldsymbol{x}_i,\boldsymbol{\theta}_j)=(\phi(\boldsymbol{x}_i,\boldsymbol{\theta}_j)-1)^2$$

  - Set $\boldsymbol{c}_j(t)$ and $A_j(t), j=1,\ldots,m$, equal to the resulting solutions.
- End parameter updating.

Once more, the above system of equations can be solved by employing iterative techniques.

Variants of the algorithm, imposing certain constraints on $A_j, j=1,\ldots,m$, have also been proposed in [Dave 92a] and [Dave 92b].

Example 14.8

Consider the three ellipses in Figure 14.11a, with centers $\boldsymbol{c}_1=[0,0]^T$, $\boldsymbol{c}_2=[8,\ 0]^T$, and $\boldsymbol{c}_3=[1,1]^T$, respectively. The corresponding matrices that specify their major and minor axes, as well as their orientation, are

$$A_1=\begin{bmatrix}\frac{1}{16} & 0\\ 0 & \frac{1}{4}\end{bmatrix},\quad A_2=\begin{bmatrix}\frac{1}{4} & 0\\ 0 & 1\end{bmatrix},\quad A_3=\begin{bmatrix}\frac{1}{8} & 0\\ 0 & \frac{1}{4}\end{bmatrix}$$

respectively. We generate 100 points, $\boldsymbol{x}_i$, from each ellipse and we add to each of these points a random vector whose coordinates stem from the uniform distribution in $[-0.5, 0.5]$. The initial values for the $\boldsymbol{c}_i$'s and the A_i's, $i=1,2,3$, are $\boldsymbol{c}_i(0)=\boldsymbol{c}_i+\boldsymbol{z}, i=1,2,3$, with $\boldsymbol{z}$ taken to be $\boldsymbol{z}=[0.3,0.3]^T$ and $A_i(0)=A_i+Z$, $i=1,2,3$, where all the elements of Z are equal to 0.2. The fuzzifier q was also set equal to 2. Application of the AFCS algorithm to this data set gives, after four iterations, the results shown in Figure 14.11b. Thus, the algorithm has identified the ellipses to a good approximation.

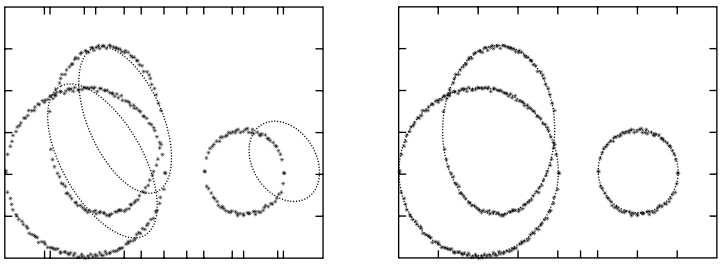

FIGURE 14.11

The setup of Example 14.8. Thick dots represent the points of the data set. Thin dots represent (a) the initial estimates and (b) the final estimates of the ellipses.

The Fuzzy C Ellipsoidal Shells (FCES) Algorithm

This algorithm uses the squared radial distance between a point and a hyperellipsoidal. Equation (14.19) now becomes

$$J_r(\boldsymbol{\theta}, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d_r^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j) \tag{14.55}$$

Defining the $\boldsymbol{\theta}_j$'s as in the previous case and carrying out the steps followed for the derivation of the AFCS, we end up with the following equations for $\boldsymbol{c}_j$ and A_j (see Problem 14.11):

$$\sum_{i=1}^{N} u_{ij}^q(t-1)\left[\frac{\|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2(1-\phi(\boldsymbol{x}_i,\boldsymbol{\theta}_j))}{\phi^4(\boldsymbol{x}_i,\boldsymbol{\theta}_j)}A_j - \left(1-\frac{1}{\phi(\boldsymbol{x}_i,\boldsymbol{\theta}_j)}\right)^2 I\right](\boldsymbol{x}_i - \boldsymbol{c}_j) = \mathbf{0} \tag{14.56}$$

and

$$\sum_{i=1}^{N} u_{ij}^q(t-1)\frac{\phi(\boldsymbol{x}_i,\boldsymbol{\theta}_j)-1}{\phi^4(\boldsymbol{x}_i,\boldsymbol{\theta}_j)}\|\boldsymbol{x}_i - \boldsymbol{c}_j\|^2(\boldsymbol{x}_i - \boldsymbol{c}_j)(\boldsymbol{x}_i - \boldsymbol{c}_j)^T = O \tag{14.57}$$

where $\phi(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ is defined as in the case of the AFCS algorithm.

The following two algorithms are proposed in [Kris 95a] and [Frig 96]. In contrast to the previous algorithms, they may fit quadrics of any shape to the data set. They are called the *fuzzy C quadric shells (FCQS) algorithm* and *modified fuzzy C quadric shells (MFCQS) algorithm*, respectively.

Fuzzy C Quadric Shells (FCQS) Algorithm

The FCQS algorithm is suitable for recovering general hyperquadric shapes. It uses the squared algebraic distance. Equation (14.19) now becomes

$$J_a(\boldsymbol{\theta}, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q d_a^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij}^q \boldsymbol{p}_j{}^T M_i \boldsymbol{p}_j \tag{14.58}$$

where $\boldsymbol{p}_j$ is defined in Eq. (14.38) and $M_i = \boldsymbol{q}_i \boldsymbol{q}_i{}^T$, with $\boldsymbol{q}_i$ defined in Eq. (14.37).

We recall that $\boldsymbol{p}_j$ incorporates all the parameters of the jth quadric surface (see Eq. 14.40), that is, $\boldsymbol{\theta}_j = \boldsymbol{p}_j$. Direct minimization of $J_a(\boldsymbol{\theta}, U)$ with respect to $\boldsymbol{p}_j$ would lead to the trivial zero solution for $\boldsymbol{p}_j$. Thus, constraints on $\boldsymbol{p}_j$ must be imposed, and a number of those have been proposed in the literature. Different constraints lead to different algorithms. Examples of such constraints are [Kris 95a] (i) $\|\boldsymbol{p}_j\|^2 = 1$, (ii) $\sum_{k=1}^{r+l} p_{jk}^2 = 1$, (iii) $p_{j1} = 1$, (iv) $p_{js}^2 = 1$, and (v) $\| \sum_{k=1}^{l} p_{jk}^2 + 0.5 \sum_{k=l+1}^{r} p_{jk}^2 \|^2 = 1$ (Problem 14.12). Each of these constraints has its advantages and disadvantages. For example, constraints (i) and (ii) [Gnan 77, Pato 70] do not preserve the invariance under translation and rotation of d_a. However, they are able to identify planar clusters. Also, constraint (iii) [Chen 89] precludes linear clusters and can lead to poor results if the points in X are approximately coplanar.

Modified Fuzzy C Quadric Shells (MFCQS) Algorithm

A different C-shells quadric algorithm is obtained if we employ the squared perpendicular distance d_p between a point and a quadric surface. However, because of the difficulty of its estimation, the resulting problem is much more difficult than those examined before. In this case, due to the complex nature of d_p, minimization of the $J_p(\boldsymbol{\theta}, U)$ with respect to the parameter vector $\boldsymbol{\theta}_j$ becomes very complex [Kris 95a].

One way to simplify things is to use the following alternative scheme. For the computation of u_{ij}'s the perpendicular distance d_p is used, and for the estimation of the parameters $\boldsymbol{\theta}_j, j = 1, \ldots, m$, the updating scheme of FCQS is employed (recall that in FCQS, $\boldsymbol{\theta}_i = \boldsymbol{p}_i$). In other words, the grade of membership of a vector $\boldsymbol{x}_i$ in a cluster is determined using the perpendicular distance, and the updating of the parameters of the representatives is carried out using the parameter updating part of the FCQS algorithm. However, this simplification implies that the algebraic and the perpendicular distances should be close to each other (see also Problem 14.13). This modification leads to the so called modified FCQS (MFCQS) algorithm.

Another algorithm, discussed in [Kris 95a] and [Frig 96], is the fuzzy C planoquadric shells (FCPQS) algorithm. This algorithm uses a first-order approximation of the algebraic distance, and it is derived, as are all the others, by taking derivatives of the resulting cost function with respect to the parameter vector, $\boldsymbol{\theta}_j$, and setting them equal to zero.

Finally, fuzzy clustering algorithms that are able to detect spherical clusters are discussed in [Dave 92a, Kris 92a, Kris 92b, Man 94]. However, most of these may be viewed as special cases of the algorithms developed to fit ellipses.

14.3.3 Hyperplane Representatives

In this section, we discuss algorithms that suitable for the recovery of hyperplanar clusters [Kris 92a]. Algorithms of this kind can be applied to the surface-fitting problem, which is one of the most important tasks in computer vision. In this problem, the surfaces of an object depicted in the image are approximated by planar surfaces. Successful identification of the surfaces is a prerequisite for the identification of the objects depicted in an image.

Some of these algorithms, such as the fuzzy c-varieties (FCV) algorithm [Ande 85], are based on minimization of the distances of the vectors in X from hyperplanes (see Chapter 11). However, FCV tends to recover very long clusters, and thus, collinear distinct clusters may be merged into a single cluster.

In this section we describe an algorithm, known as the Gustafson–Kessel (G-K) algorithm (see, e.g., [Kris 92a, Kris 99a]). According to this algorithm, planar clusters are represented by centers $\boldsymbol{c}_j$ and covariance matrices Σ_j. Defining $\boldsymbol{\theta}_j$ as in previous cases, we define the squared distance between a vector $\boldsymbol{x}$ and the jth cluster as the scaled Mahalanobis distance

$$d^2_{GK}(\boldsymbol{x}, \boldsymbol{\theta}_j) = |\Sigma_j|^{1/l}(\boldsymbol{x} - \boldsymbol{c}_j)^T \Sigma_j^{-1}(\boldsymbol{x} - \boldsymbol{c}_j) \tag{14.59}$$

Let us now gain some insight into the behavior of this distance. A well-known property that characterizes the distance d_H of a point from a hyperplane, as defined in Chapter 11, is that all points lying on a hyperplane H_1 parallel to a given hyperplane H, have the same d_H distance from H. This will be our starting point for the investigation of d^2_{GK}.

Example 14.9

Consider the setup of Figure 14.12a, where a single cluster C is present, and let $\boldsymbol{\theta}$ be its parameter vector. The points of C are of the form $[x_{i1}, x_{i2}]^T$ where $x_{i1} = -2 + 0.1i$, $i = 0, 1, 2, \ldots, 40$, and the corresponding x_{i2}'s are random numbers following the uniform distribution in $[-0.1, 0.1]$.

Consider also the points of the line segment u connecting the points $(-2, 2)$ and $(2, 2)$. Figure 14.12b shows the distances $d_{GK}(\boldsymbol{x}, \boldsymbol{\theta})$ of the points $\boldsymbol{x} \in u$ from C. As we can see, all these distances are almost the same. Indeed, the relative difference $(d_{\max} - d_{\min})/d_{\max}$ between the maximum $d_{\max}$ and the minimum $d_{\min}$ values is approximately equal to 0.02.

Now consider the larger line segment $v_1(v_2)$ that connects $(-8, 2)((-8, -2))$ and $(8, 2)((8, -2))$. The distances $d_{GK}(\boldsymbol{x}, \boldsymbol{\theta})$ of the points $\boldsymbol{x} \in v_1(v_2)$ from C are shown in Figure 14.12b. Note that although we have larger variations compared to the previous case, they still remain relatively small (the relative difference $(d_{\max} - d_{\min})/d_{\max}$ is approximately 0.12).

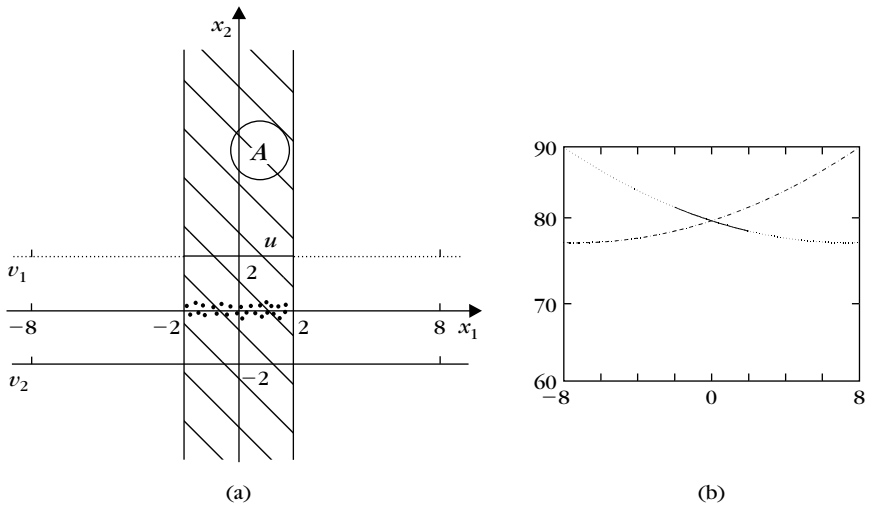

FIGURE 14.12

(a) The setup of the Example 14.9. (b) The solid line corresponds to the distances of the points of $\boldsymbol{u}$ from C. The dashed line corresponds to the distances of the points of the line segment $\boldsymbol{v}_1$ from C (the solid line is part of the dashed line). Also, the dash-dotted line corresponds to the distances of the points of the line segment $\boldsymbol{v}_2$ from C.

The G-K algorithm can be derived via the minimization of

$$J_{GK}(\boldsymbol{\theta}, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^{q} d_{GK}^{2}(\boldsymbol{x}_i, \boldsymbol{\theta}_j) \tag{14.60}$$

Taking the gradient of $J_{GK}(\boldsymbol{\theta}, U)$ with respect to $\boldsymbol{c}_j$, we obtain

$$\frac{\partial J_{GK}(\boldsymbol{\theta}, U)}{\partial \boldsymbol{c}_j} = \sum_{i=1}^{N} u_{ij}^{q} \frac{\partial d_{GK}^{2}(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{c}_j} \tag{14.61}$$

The gradient of the distance, after a bit of algebra, becomes

$$\frac{\partial d_{GK}^{2}(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{c}_j} = -2|\Sigma_j|^{1/l} \Sigma_j^{-1} (\boldsymbol{x}_i - \boldsymbol{c}_j) \tag{14.62}$$

Substituting $\partial d^2(\boldsymbol{x}_i, \boldsymbol{\theta}_j)/\partial \boldsymbol{c}_j$ from Eq. (14.62) into (14.61) and setting $\partial J_{GK}(\boldsymbol{\theta}, U)/\partial \boldsymbol{c}_j$ equal to zero, we obtain[8]

$$\boldsymbol{c}_j = \frac{\sum_{i=1}^{N} u_{ij}^{q} \boldsymbol{x}_i}{\sum_{i=1}^{N} u_{ij}^{q}} \tag{14.63}$$

[8] We also make the mild assumption that the covariance matrix is invertible.

Now taking the derivative of $J_{GK}(\boldsymbol{\theta}, U)$ with respect to the elements of the covariance matrix, Σ_j results (Problem 14.16) in

$$\Sigma_j = \frac{\sum_{i=1}^{N} u_{ij}^q (\boldsymbol{x}_i - \boldsymbol{c}_j)(\boldsymbol{x}_i - \boldsymbol{c}_j)^T}{\sum_{i=1}^{N} u_{ij}^q} \tag{14.64}$$

Having derived Eqs. (14.63) and (14.64), the parameter updating part of GFAS for the G-K algorithm becomes

- $$\boldsymbol{c}_j(t) = \frac{\sum_{i=1}^{N} u_{ij}^q (t-1)\boldsymbol{x}_i}{\sum_{i=1}^{N} u_{ij}^q (t-1)}$$

- $$\Sigma_j(t) = \frac{\sum_{i=1}^{N} u_{ij}^q (t-1)\big(\boldsymbol{x}_i - \boldsymbol{c}_j(t-1)\big)\big(\boldsymbol{x}_i - \boldsymbol{c}_j(t-1)\big)^T}{\sum_{i=1}^{N} u_{ij}^q (t-1)}$$

Example 14.10

(a) Consider Figure 14.13a. It consists of three linear clusters. Each cluster contains 41 points. The points of the first cluster lie around the line $x_2 = x_1 + 1$, while the points of the second and the third clusters lie around the lines $x_2 = 0$ and $x_2 = -x_1 + 1$, respectively. The $\boldsymbol{c}_j$'s, $j = 1, 2, 3$, are randomly initialized and the threshold of the termination criterion, ε, is set to 0.01. The G-K algorithm converges after 26 iterations. As shown in Figure 14.13b, the G-K identifies correctly the clusters underlying X.

(b) Now consider Figure 14.14a. We also have three clusters, each consisting of 41 points. The first and the third clusters are the same as in Figure 14.13a, while the points of the second cluster lie around the line $x_2 = 0.5$. Note that in this case the three intersection points between

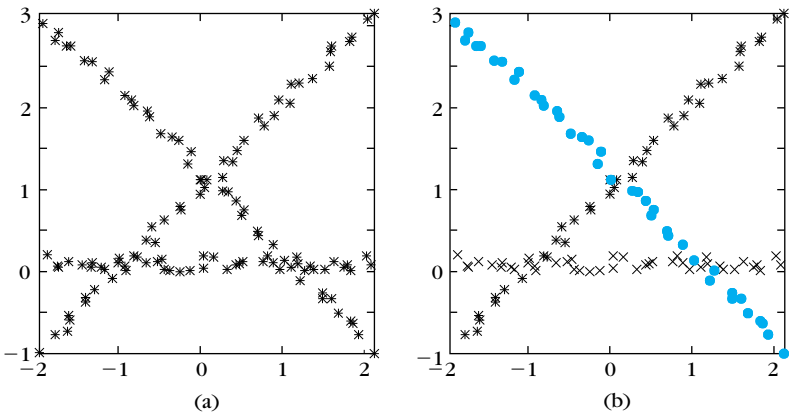

FIGURE 14.13

(a) The data set X for Example 14.10(a). (b) The results of the G-K algorithm.

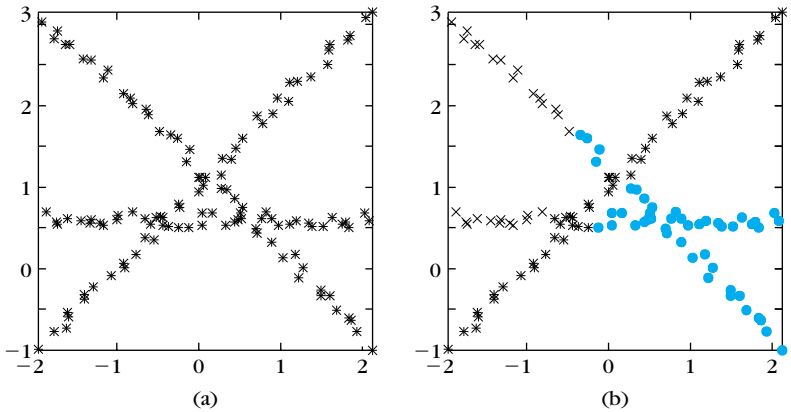

FIGURE 14.14

(a) The data set X for Example 14.10(b). (b) The results of the G-K algorithm.

any pair of lines lie very close to each other. The G-K algorithm terminates after 38 iterations. The results obtained are shown in Figure 14.14b. In this case, the G-K algorithm fails to identify the clusters correctly.

14.3.4 Combining Quadric and Hyperplane Representatives

In this section, we assume that $l = 2$. Consider the case in which X contains quadric-shaped clusters as well as linear clusters. How can we accurately identify both kinds of clusters? If we run an algorithm that fits quadric curves to the clusters, the linear clusters will not be properly represented. On the other hand, if we run an algorithm that fits lines to the clusters, the ellipsoidally and hyperbolically shaped clusters will be poorly represented. A way out of this problem is discussed in [Kris 95a]. The idea is to run the FCQS algorithm first on the whole data set X. This algorithm can be used to detect linear clusters, even though the adopted constraints force all representatives to be of second degree. This happens since, in practice, FCQS fits a pair of coincident lines for a single line, a hyperbola for two intersecting lines and a very "flat" hyperbola or a very elongated ellipse or a pair of lines for two parallel lines [Kris 95a]. The identification of "extreme" quadric curves (i.e., extremely elongated ellipses, "flat" hyperbolas, a set of lines) after the termination of the algorithm is a strong indication that X contains linear clusters. In order to represent these extreme clusters more accurately, we can run the G-K algorithm on the set X', which contains only the vectors that belong to them (with a high grade of membership). However, different actions have to be carried out depending on the shape of each extreme quadric curve. Let Q_j be the representative curve of the jth cluster, $j = 1, \ldots, m$, identified by FCQS and Q'_j's the representative curves

of the linear clusters that are identified by the G-K algorithm. Specifically, we have

- If Q_j is a pair of coincident lines, then initialize Q'_j using one of the two lines.
- If Q_j is a nonflat hyperbola *or* a pair of intersecting lines *or* a pair of parallel lines, then initialize two representatives Q'_{j1} and Q'_{j2} using the asymptotes of the hyperbola (for the first case) or using each of the lines (for the last two cases).
- If Q_j is an ellipse with a very large ratio of major to minor axis, then initialize two representatives Q'_{j1} and Q'_{j2} using the tangents of the ellipse at the two ends of the minor axis.
- If Q_j is a hyperbola with a very large ratio of conjugate axis to transverse axis, then initialize two representatives Q'_{j1} and Q'_{j2} using the two tangents of the hyperbola at its two vertices.

Since the initialization of the Q'_j representatives is very good, it is expected that the G-K algorithm will converge in a few iterations to a satisfactory solution.

14.3.5 A Geometrical Interpretation

Arguments similar to those given in Section 14.2.2 can also be repeated here. Now u_{ij} takes the place of $P(C_j|\boldsymbol{x}_i)$. The constraint equation in this case is

$$\sum_{j=1}^{m} u_{ij} = 1, \quad i = 1, \ldots, N. \tag{14.65}$$

The vector $\boldsymbol{y}$ associated with vector $\boldsymbol{x}_i$ becomes $\boldsymbol{y} = [u_{i1}, u_{i2}, \ldots, u_{im}]$ and it is also restricted on the hyperplane defined by the constraint (14.65) in the H_m hypercube. If we carry out the experiments discussed in Section 14.2.2, we will draw similar conclusions with respect to the effect of the outliers on the performance of the fuzzy algorithms.

In [Mena 00] an algorithm called *fuzzy c + 2 means* is introduced. This is an extension of GFAS for point representatives, where the outliers as well as the points that lie near the cluster boundaries are treated so as to control their effect on the estimates of the cluster representatives.

14.3.6 Convergence Aspects of the Fuzzy Clustering Algorithms

Although fuzzy clustering algorithms are obtained by minimizing a cost function of the form of Eq. (14.19), little is known about their convergence behavior. More specifically, it has been proved [Bezd 80, Hath 86], using the global convergence theorem of Zangwill [Luen 84], that when a Mahalanobis distance is used (or other distances satisfying certain conditions discussed in [Bezd 80]), the iteration sequence produced by the fuzzy c-means (FCM) algorithm either converges to a stationary point of the cost function in a finite number of iteration steps or it has at

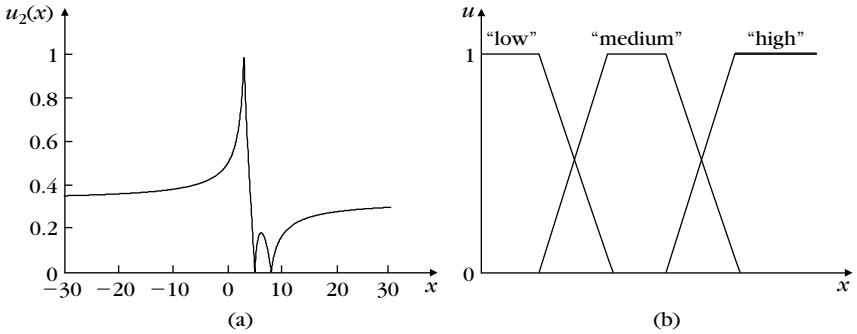

FIGURE 14.15

(a) The membership function $u_2(x)$ given by Eq. (14.26), for the one-dimensional case, with $\theta_1 = 5$, $\theta_2 = 3$, $\theta_3 = 8$, $q = 2$ and $d(x, \theta_i) = |x - \theta_i|$. (b) Examples of membership functions characterizing "low,", "medium," and "high" for a specific quantity.

least one subsequence that converges to a stationary point of the cost function. This point may be a local (or global) optimum or a saddle point. Tests for the identification of the nature of the convergence point are discussed in [Isma 86, Hath 86, Kim 88]. More recently in [Grol 05] it is shown that sequence produced by the FCM converges to a stationary point of the cost function. Issues concerning numerical convergence aspects of the FCM algorithms are discussed in [Bezd 92].

14.3.7 Alternating Cluster Estimation

It is not difficult to notice that the membership functions $u_j(\boldsymbol{x}_i)$, associated with the u_{ij}'s used in GFAS (Eq. (14.26)), are neither convex nor monotonous (see, for example, Figure 14.15a). However, in fuzzy rule-based systems convexity of the membership functions is an important requirement. For example, linguistic characterizations such as "low," "medium," or "high" require convex membership functions of the form shown in Figure 14.15b. In such cases it may be preferable to adopt a specific membership function and use the alternating updating philosophy used in GFAS to estimate u_{ij}'s and $\boldsymbol{\theta}_j$. The resulting algorithmic scheme is known as *alternating cluster estimation (ACE)* ([Runk 99, Hopp 99]) and GFAS may be viewed as a special case of it. Obviously, in this case, the solution obtained is not necessarily related to an optimizing criterion.

14.4 POSSIBILISTIC CLUSTERING

The algorithms of this section are relaxed from constraints such as in (14.17) and (14.65) [Kris 93, Kris 96]. Speaking in the terms of Section 14.3.5, this means that the vector $\boldsymbol{y}$, with coordinates the u_{ij}'s, will be allowed to move anywhere in

the H_m hypercube, that is,

$$u_{ij} \in [0, 1]$$
$$\max_{j=1,\ldots,m} u_{ij} > 0, \quad i = 1, \ldots, N$$

and

$$0 < \sum_{i=1}^{N} u_{ij} \leq N, \quad i = 1, \ldots, N \tag{14.66}$$

This change in the constraints has an important impact on the interpretation of the u_{ij}'s. In the fuzzy framework, u_{ij} denotes the grade of membership of $\boldsymbol{x}_i$ in the jth cluster. Here, u_{ij} may be interpreted as the degree of compatibility of $\boldsymbol{x}_i$ with the jth cluster representative, or, following [Zade 78], the possibility that $\boldsymbol{x}_i$ belongs to the jth cluster. Note that *the possibility that $\boldsymbol{x}_i$ belongs to the jth cluster depends exclusively on $\boldsymbol{x}_i$ and the cluster representative of the jth cluster; that is, it is independent of the possibilities that $\boldsymbol{x}_i$ belongs to any other cluster.*

For convenience, let us recall here that the cost function to be minimized is

$$J(\boldsymbol{\theta}, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^{q} d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) \tag{14.67}$$

Obviously, direct minimization with respect to U will lead to the trivial zero solution. In order to avoid this situation, we must insert an additional term in $J(\boldsymbol{\theta}, U)$. This term, $f(U)$, will be a function of u_{ij}'s only. Motivated by the discussion in Section 14.2.2, it will be chosen in such a way so as to minimize the effects of outliers. As will become apparent soon, one such choice of $f(U)$ is

$$f(U) = \sum_{j=1}^{m} \eta_j \sum_{i=1}^{N} (1 - u_{ij})^q \tag{14.68}$$

Then, the cost function becomes

$$J(\boldsymbol{\theta}, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij}^{q} d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) + \sum_{j=1}^{m} \eta_j \sum_{i=1}^{N} (1 - u_{ij})^q \tag{14.69}$$

where η_j are suitably chosen positive constants.

The minimum of $J(\boldsymbol{\theta}, U)$, with respect to u_{ij}, is obtained by

$$\frac{\partial J(\boldsymbol{\theta}, U)}{\partial u_{ij}} = q u_{ij}^{q-1} d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) - q \eta_j (1 - u_{ij})^{q-1} = 0$$

or

$$u_{ij} = \frac{1}{1 + \left(\frac{d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\eta_j} \right)^{\frac{1}{q-1}}} \tag{14.70}$$

In words, u_{ij} is inversely proportional to the dissimilarity between $\boldsymbol{x}_i$, and the representative of the jth cluster. Loosely speaking, u_{ij} denotes the degree to which the representative of the jth cluster *should be "stretched"* in order to match $\boldsymbol{x}_i$. Large (small) values of u_{ij} indicate little (large) stretch for the jth representative. *It is clear that for a specific vector $\boldsymbol{x}_i$, this "stretching" action can be carried out independently for each cluster*.

The meaning of the second term in Eq. (14.69) is clearer now. Its effect is to minimize the influence of outliers in the estimation of the $\boldsymbol{\theta}_j$'s. Indeed, large dissimilarity levels correspond to small u_{ij}'s and they have little effect on the first term in the cost function, which controls the estimation of $\boldsymbol{\theta}_j$'s.

Since the second term does not involve the representatives of the clusters, one may easily conclude that in possibilistic clustering schemes, the updating of the parameters of each cluster is carried out in exactly the same way as in the case of their fuzzy counterparts.

Generalized Possibilistic Algorithmic Scheme (GPAS)

- Fix $\eta_j, j = 1, \ldots, m$.
- Choose $\boldsymbol{\theta}_j(0)$ as the initial estimates of $\boldsymbol{\theta}_j, j = 1, \ldots, m$.
- $t = 0$.
- Repeat
  - For $i = 1$ to N
    - For $j = 1$ to m

$$u_{ij}(t) = \frac{1}{1 + \left(\frac{d(\boldsymbol{x}_i, \boldsymbol{\theta}_j(t))}{\eta_j}\right)^{\frac{1}{q-1}}}$$

    - End {For-j}
  - End {For-i}
  - $t = t + 1$
  - For $j = 1$ to m
    - *Parameter updating*: Solve

$$\sum_{i=1}^{N} u_{ij}^{q}(t-1) \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} = \mathbf{0} \tag{14.71}$$

with respect to $\boldsymbol{\theta}_j$ and set $\boldsymbol{\theta}_j(t)$ equal to the computed solution.

  - End {For-j}.
- Until a termination criterion is met.

As usual, we may employ $\|\boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1)\| < \varepsilon$ as a termination criterion. Based on the preceding generalized scheme, for each of the fuzzy clustering algorithms, defined in the previous section, we can derive a corresponding possibilistic one.

An interesting observation is that, since for each vector $\boldsymbol{x}_i$, u_{ij}'s, $j = 1, \ldots, m$, are independent of each other, we can write $J(\boldsymbol{\theta}, U)$ as

$$J(\boldsymbol{\theta}, U) = \sum_{j=1}^{m} J_j$$

where

$$J_j = \sum_{i=1}^{N} u_{ij}^q d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) + \eta_j \sum_{i=1}^{N} (1 - u_{ij})^q \tag{14.72}$$

Each J_j corresponds to a different cluster and the minimization of $J(\boldsymbol{\theta}, U)$ with respect to the u_{ij}'s can be carried out separately for each J_j.

The value of η_j determines the relative significance of the two terms in (14.72) and it is related to the size and "shape" of the jth cluster, $j = 1, \ldots, m$. More specifically, as can be seen from Figure 14.16, η_j determines the dissimilarity level between a vector $\boldsymbol{x}_i$ and the representative $\boldsymbol{\theta}_j$ at which u_{ij} becomes equal to 0.5. Thus, η_j determines the influence of a specific point on the estimation of the jth cluster representative.

In general, the size of η_j is assumed constant during the execution of the algorithm. One way to estimate its value, under the assumption that X does not contain many outliers, is to run the generalized fuzzy algorithmic scheme (GFAS) and after its convergence to estimate η_j as [Kris 96]

$$\eta_j = \frac{\sum_{i=1}^{N} u_{ij}^q d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\sum_{i=1}^{N} u_{ij}^q} \tag{14.73}$$

or

$$\eta_j = \frac{\sum_{u_{ij} > a} d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\sum_{u_{ij} > a} 1} \tag{14.74}$$

where a is an appropriate threshold. In words, η_j is defined as a weighted average of the dissimilarities between the vectors $\boldsymbol{x}_i$ and $\boldsymbol{\theta}_j$. Once η_j's have been fixed, the GPAS algorithm can be applied.

In Figure 14.16, u_{ij} versus $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)/\eta_j$ is plotted for various choices of q (see Eq. (14.70)). From this diagram, it can be seen that q determines the rate of decrease of u_{ij} with respect to $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$. For $q = 1$, all points $\boldsymbol{x}_i$ with $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) > \eta_j$ have $u_{ij} = 0$. On the other hand, as $q \to +\infty$, u_{ij} tends to a constant and all the vectors of X contribute equally to the estimation of the representative of the jth cluster.

It is worth noting here that q has different meanings in the possibilistic and the fuzzy framework. In the first case, high values of q imply almost equal contributions of *all* feature vectors to *all* clusters, whereas in the second case, high values of q imply increased *sharing* of the vectors among all clusters [Kris 96]. This implies that, in general, different values of q are required to provide satisfactory results for the two cases.

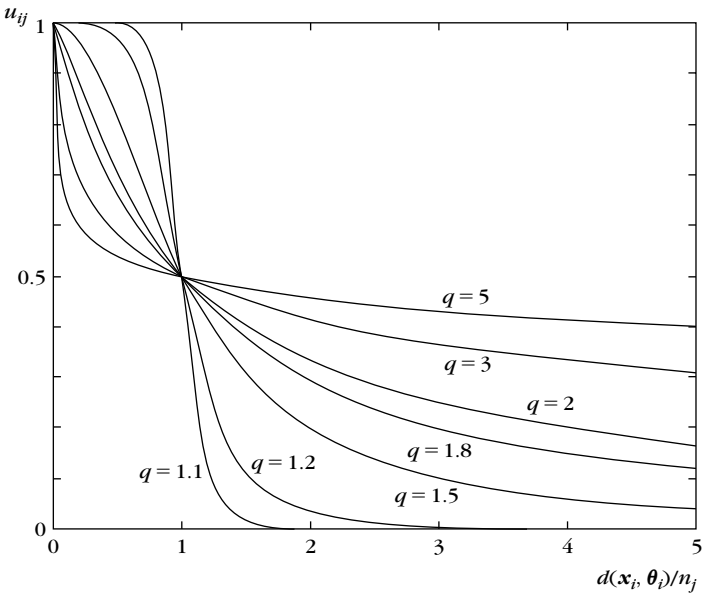

FIGURE 14.16

Plots of the membership function for various values of q.

14.4.1 The Mode-Seeking Property

The generalized mixture decomposition algorithmic scheme (GMDAS) and the generalized fuzzy algorithmic scheme (GFAS) are *partition algorithmic schemes*—that is, schemes that always end up with the predetermined number of clusters m, no matter how many "naturally formed" clusters underlie X. If, for example, the data set X contains two clusters and we run GMDAS or GFAS with $m = 3$, these algorithms will split at least one natural cluster and will end up with three clusters.

This is not the case however with the generalized possibilistic algorithmic scheme (GPAS). Algorithms of this kind are known as *mode-seeking algorithms*—that is, algorithms searching for dense regions of vectors in X.[9] In order to see this, let us consider again the individual functions J_j. Solving Eq. (14.70) with respect to $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$, we obtain

$$d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = \eta_j \left(\frac{1 - u_{ij}}{u_{ij}} \right)^{q-1}$$

[9] Such algorithms are also considered in Chapter 15.

Substituting $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ from this equation into Eq. (14.72) results in

$$J_j = \eta_j \sum_{i=1}^{N} (1 - u_{ij})^{q-1} \tag{14.75}$$

For fixed η_j, minimization of J_j requires maximization of u_{ij}'s, which, in turn, requires minimization of $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$. The last requirement implies that $\boldsymbol{\theta}_j$ should be placed in a region dense in vectors of X.

The mode-seeking property of the GPAS implies that the number of clusters in X need not be known *a priori*. Indeed, if we run a possibilistic algorithm for m clusters while X contains k natural clusters, with $m > k$, then, after proper initialization, *some of the m clusters will coincide with others* [Kris 96]. It is hoped that the number of the noncoincident clusters will be equal to k. If, on the other hand, $m < k$, proper initialization of the possibilistic algorithm will potentially lead to m different clusters. Of course, these are not all the natural clusters formed in X, but at least they are some of them [Kris 96].

Example 14.11

This example demonstrates the mode-seeking property. Consider three two-dimensional Gaussian distributions with means $\boldsymbol{\mu}_1 = [1, 1]^T, \boldsymbol{\mu}_2 = [6, 1]^T, \boldsymbol{\mu}_3 = [6, 6]^T$ and covariance matrices $\Sigma_j = I, j = 1, 2, 3$. One hundred vectors are generated from each distribution. These constitute the data set X. We set $q = 1.5$ and, finally, we employ the squared Euclidean distance. It is not difficult to realize that under the above choice, Eq. (14.71) gives

$$\boldsymbol{\theta}_j(t) = \frac{\sum_{i=1}^{N} u_{ij}^q (t-1) \boldsymbol{x}_i}{\sum_{i=1}^{N} u_{ij}^q (t-1)} \tag{14.76}$$

(a) Let $m = 3$. The initial estimates of $\boldsymbol{\theta}_j$'s (which, in this case, are vectors in the two-dimensional space) in GPAS are $\boldsymbol{\theta}_j(0) = \boldsymbol{\mu}_j + \boldsymbol{z}_j, j = 1, 2, 3$, where the $\boldsymbol{z}_j$'s are two-dimensional vectors whose components are drawn from the uniform distribution in $[-2, 2]$. Also, we set $\eta_j = 1.5, j = 1, 2, 3$. Application of the GPAS causes the movement of each one of the $\boldsymbol{\theta}_j$'s toward the mean of each distribution (i.e., toward dense regions). Indeed, the final estimates for $\boldsymbol{\theta}_j$'s obtained after 12 iterations, are $\boldsymbol{\theta}_1 = [0.93, 0.60]^T, \boldsymbol{\theta}_2 = [5.88, 1.12]^T$, and $\boldsymbol{\theta}_3 = [6.25, 5.86]^T$, which compare very favorably to $\boldsymbol{\mu}_j$'s.

(b) Let $m = 4$. In this case, $\boldsymbol{\theta}_j$'s, $j = 1, 2, 3$ are initialized as in the previous example, while $\boldsymbol{\theta}_4$ is initialized as $\boldsymbol{\mu}_1 + \boldsymbol{z}_4$. Application of GPAS in this case causes the movement of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_4$ toward the dense region that corresponds to the first distribution. Also, $\boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_3$ move toward the dense regions that correspond to the second and the third distribution, respectively. The resulting values for $\boldsymbol{\theta}_j$'s, obtained after 12 iterations, are $\boldsymbol{\theta}_1 = [0.93, 0.60]^T, \boldsymbol{\theta}_2 = [5.88, 1.12]^T, \boldsymbol{\theta}_3 = [6.25, 5.86]^T$, and $\boldsymbol{\theta}_4 = [0.94, 0.60]^T$.

(c) Let $m = 2$. We initialize $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ as in (a). Application of the GPAS algorithm causes the movement of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ toward the dense regions corresponding to first

and the second distribution, respectively. The resulting values for $\boldsymbol{\theta}_j$'s, obtained after 11 iterations, are $\boldsymbol{\theta}_1 = [0.93, 0.60]^T$ and $\boldsymbol{\theta}_2 = [5.88, 1.12]^T$.

14.4.2 An Alternative Possibilistic Scheme

An alternative possibilistic algorithm may be derived from the function [Kris 96]

$$J_1(\boldsymbol{\theta}, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij} d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) + \sum_{j=1}^{m} \eta_j \sum_{i=1}^{N} (u_{ij} \ln u_{ij} - u_{ij}) \tag{14.77}$$

Note that q is not involved in the definition of $J_1(\boldsymbol{\theta}, U)$. Also, in this case the second term is negative. Setting the partial derivative of $J_1(\boldsymbol{\theta}, U)$ with respect to u_{ij} equal to 0 and solving for u_{ij}, we obtain the following necessary condition for each u_{ij} to be a minimum of $J_1(\boldsymbol{\theta}, U)$:

$$u_{ij} = \exp\left(-\frac{d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\eta_j}\right) \tag{14.78}$$

Hence, u_{ij} decreases more rapidly with $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)$ than in the previous case (Eq. 14.70). Let us consider a point $\boldsymbol{x}_i$ and a cluster representative $\boldsymbol{\theta}_j$. For the same distance d, (14.78) leads to smaller values of u_{ij} than those derived from (14.70). This means that increased "stretching" is demanded for the former case. *This is an indication that this algorithmic scheme may be used when the clusters are expected to lie close to each other.*

14.5 HARD CLUSTERING ALGORITHMS

In this section we return to the world where each vector belongs *exclusively* to a single cluster. This is why such schemes are called *hard* or *crisp* clustering algorithms. *It turns out that some of the most well-known and widely used clustering algorithms fall into this category.* Our starting point is the assumption that the membership coefficients u_{ij} are either 1 or 0. Moreover, they are 1 for one cluster, C_j, and zero for all the others, $C_k, k \neq j$, that is,

$$u_{ij} \in \{0, 1\}, \qquad j = 1, \ldots, m \tag{14.79}$$

and

$$\sum_{j=1}^{m} u_{ij} = 1 \tag{14.80}$$

This situation may be seen as a special case of the fuzzy algorithmic schemes. However, the cost function

$$J(\boldsymbol{\theta}, U) = \sum_{i=1}^{N}\sum_{j=1}^{m} u_{ij} d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) \tag{14.81}$$

is *no longer differentiable with respect to* $\boldsymbol{\theta}_j$. Despite that, the general framework of the generalized fuzzy algorithmic schemes, can be adopted for the special case of hard clustering. Such schemes have been used extensively in practice (e.g., [Duda 01]).

Let us fix $\boldsymbol{\theta}_j, j = 1, \ldots, m$. Since for each vector $\boldsymbol{x}_i$ only one u_{ij} is 1 and all the others are 0, it is straightforward to see that $J(\boldsymbol{\theta}, U)$ in Eq. (14.81) is minimized if we assign each $\boldsymbol{x}_i$ to its closest cluster, that is,

$$u_{ij} = \begin{cases} 1, & \text{If } d(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = \min_{k=1,\ldots,m} d(\boldsymbol{x}_i, \boldsymbol{\theta}_k) \\ 0, & \text{otherwise} \end{cases} \quad i = 1, \ldots, N \tag{14.82}$$

Let us now fix u_{ij}s. Working as in the fuzzy algorithms case, the updating equations of the parameter vectors, $\boldsymbol{\theta}_j$, of the clusters are

$$\sum_{i=1}^{N} u_{ij} \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} = \mathbf{0}, \quad j = 1, \ldots, m \tag{14.83}$$

Having derived Eqs. (14.82) and (14.83), we are now in a position to write down the generalized hard clustering algorithmic scheme

Generalized Hard Algorithmic Scheme (GHAS)

- Choose $\boldsymbol{\theta}_j(0)$ as initial estimates for $\boldsymbol{\theta}_j, j = 1, \ldots, m$.
- $t = 0$
- Repeat
  - For $i = 1$ to N
    - For $j = 1$ to m
      - — *Determination of the partition:*[10]

$$u_{ij}(t) = \begin{cases} 1, & \text{if } d(\boldsymbol{x}_i, \boldsymbol{\theta}_j(t)) = \min_{k=1,\ldots,m} d(\boldsymbol{x}_i, \boldsymbol{\theta}_k(t)) \\ 0, & \text{otherwise,} \end{cases}$$

    - End {For-j}
  - End {For-i}
  - $t = t + 1$
  - For $j = 1$ to m
    - *Parameter updating*: Solve

$$\sum_{i=1}^{N} u_{ij}(t-1) \frac{\partial d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} = \mathbf{0} \tag{14.84}$$

[10] In the case in which two or more minima occur, an arbitrary choice is made.

with respect to $\boldsymbol{\theta}_j$ and set $\boldsymbol{\theta}_j(t)$ equal to the computed solution.

- End {For-j}.

- Until a termination criterion is met.

Note that in the update of each $\boldsymbol{\theta}_j$, only the vectors $\boldsymbol{x}_i$ closest to it (i.e., those $\boldsymbol{x}_i$'s for which $u_{ij}(t-1)=1$) are used. As usual, the termination criterion $\|\boldsymbol{\theta}(t)-\boldsymbol{\theta}(t-1)\|<\varepsilon$ can be used. Alternatively, GHAS may terminate if U remains unchanged for two successive iterations.

Each hard clustering algorithm has its corresponding fuzzy clustering algorithm. As with the fuzzy clustering algorithms, we may obtain hard clustering algorithms when $\boldsymbol{\theta}_j$s represent points, quadric surfaces, or hyperplanes. The updating equations for the parameter vectors $\boldsymbol{\theta}_j$ in the hard clustering algorithms are obtained from their fuzzy counterparts if we set $q=1$.

Remarks

- Hard clustering algorithms are not as robust as fuzzy clustering algorithms when other than point representatives are employed. If, for example, hyperplane representatives are used and the G-K algorithm is adopted, we must have an adequate number of vectors N from all underlying clusters in order to avoid degenerate cases where Σ_j is not invertible [Kris 92a].

- The determination of the partition part of the algorithms optimizes $J(\boldsymbol{\theta}, U)$ with respect to U given a set of representatives $\boldsymbol{\theta}_j$. On the other hand, the parameter updating phase optimizes $J(\boldsymbol{\theta}, U)$ with respect to $\boldsymbol{\theta}$ given a specific partition. Note that this procedure does not necessarily lead to a (local) optimum of $J(\boldsymbol{\theta}, U)$.

14.5.1 The Isodata or k-Means or c-Means Algorithm

This is one of the most popular and well-known clustering algorithms [Duda 01, Ball 67, Lloy 82]. It can be viewed as a special case of the generalized hard clustering algorithmic scheme when point representatives are used and the *squared* Euclidean distance is adopted to measure the dissimilarity between vectors $\boldsymbol{x}_i$ and cluster representatives $\boldsymbol{\theta}_j$. Before we state the algorithm explicitly, some further comments may be of interest. For this case Eq. (14.81) becomes

$$J(\boldsymbol{\theta}, U)=\sum_{i=1}^{N}\sum_{j=1}^{m}u_{ij}\|\boldsymbol{x}_i-\boldsymbol{\theta}_j\|^2 \tag{14.85}$$

This is nothing but the trace of the within scatter matrix S_w, defined in Chapter 5. That is,

$$J(\boldsymbol{\theta}, U)=\text{trace}\{S_w\} \tag{14.86}$$

For the above choice of distance, Eq. (14.83) gives that $\boldsymbol{\theta}_j$ is the mean vector of the jth cluster. *Applying the generalized hard algorithmic scheme for this specific choice, it turns out that the algorithm converges to a minimum of the cost function.* In other words, the isodata algorithm recovers clusters that are as compact as possible. It must be emphasized however, that this convergence result is not valid for other distances, including the Euclidean distance. For example, when Minkowski distances are used, the algorithm converges but not necessarily to a minimum of the corresponding cost function [Seli 84a].

The Isodata or k-Means or c-Means Algorithm

- Choose arbitrary initial estimates $\boldsymbol{\theta}_j(0)$ for the $\boldsymbol{\theta}_j$'s, $j = 1, \ldots, m$.
- Repeat
  - For $i = 1$ to N
    - Determine the closest representative, say $\boldsymbol{\theta}_j$, for $\boldsymbol{x}_i$.
    - Set $b(i) = j$.
  - End {For}
  - For $j = 1$ to m
    - Parameter updating: Determine $\boldsymbol{\theta}_j$ as the mean of the vectors $\boldsymbol{x}_i \in X$ with $b(i) = j$.
  - End {For}.
- Until no change in $\boldsymbol{\theta}_j$'s occurs between two successive iterations.

As with all the algorithms that use point representatives, isodata is suitable for recovering compact clusters. A sequential version of the k-means (see, for example, [Pena 99]) results if the updating of the representatives takes place immediately after determining the representative that lies closest to the currently considered vector $\boldsymbol{x}_i$. Clearly, the result of this version of the algorithm is dependent on the order in which the vectors are considered. A version of the k-means algorithm, where in each cluster C_i the number of vectors is constrained *a priori* to be n_i, is proposed in [Ng 00].

Example 14.12

(a) Consider the setup of Example 14.1(a). In this case $\boldsymbol{\theta}_j$s correspond to the μ_js. We set $m = 3$ and we initialize randomly $\boldsymbol{\theta}_j$s. After convergence, the isodata algorithm identifies successfully the underlying clusters in X, as indicated by the corresponding confusion matrix,

$$A = \begin{bmatrix} 94 & 3 & 3 \\ 0 & 100 & 0 \\ 9 & 0 & 91 \end{bmatrix}.$$

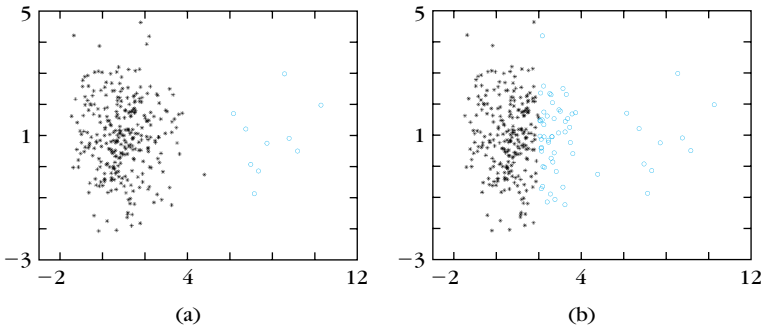

FIGURE 14.17

(a) The data set. (b) The results of the isodata algorithm.

The resulting values of $\boldsymbol{\theta}_j$s are $\boldsymbol{\theta}_1 = [1.19, 1.16]^T, \boldsymbol{\theta}_2 = [3.76, 3.63]^T$ and $\boldsymbol{\theta}_3 = [5.93, 0.55]^T$.

(b) Let us now consider two 2-dimensional Gaussian distributions with means $\boldsymbol{\mu}_1 = [1, 1]^T$ and $\boldsymbol{\mu}_2 = [8, 1]^T$ and covariance matrices $\Sigma_1 = 1.5I$ and $\Sigma_2 = I$, respectively. We generate 300 points from the first distribution and 10 points from the second distribution in order to form the data set X (Figure 14.17a). Also, we set $m = 2$ and we initialize randomly $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. After convergence, we observe that the large group has been split into two parts and, in addition, the right part joins the vectors of the second distribution in the same cluster (Figure 14.17b). Specifically, the results of the algorithm are $\boldsymbol{\theta}_1 = [0.54, 0.94]^T$ and $\boldsymbol{\theta}_2 = [3.53, 0.99]^T$ and 61 vectors from the first distribution are assigned to the same cluster with the ten vectors of the second distribution. The above situation reveals a weakness of the algorithm to deal accurately with clusters having significantly different sizes.

A major advantage of the *k*-means algorithm is its computational simplicity, which makes it an attractive candidate for a variety of applications. Its time complexity is $O(Nmq)$, where q is the number of iterations required for convergence. Because in practice m and q are significantly less than N, *k*-means becomes eligible for processing large data sets. Furthermore, its conceptual simplicity has been a source of inspiration to many authors, who have proposed a number of variants in order to remedy drawbacks associated with the algorithm. Some of them are summarized in the following

- As all optimization schemes considered in this chapter, the *k*-means algorithm cannot guarantee convergence to the global minimum of $J(\boldsymbol{\theta}, U)$. Equivalently, different initial partitions may lead *k*-means to produce different final clusterings, each corresponding to a different local minimum of $J(\boldsymbol{\theta}, U)$. To minimize or even overcome this drawback, a number of strategies have been suggested.
  - Instead of initializing $\boldsymbol{\theta}_j$s by m randomly chosen points (some suggest random initialization with points drawn from X, [Forg 65]), one can use any

of the sequential algorithms discussed in Chapter 12 to produce the initial estimates for $\boldsymbol{\theta}_j$s. Another way is to partition randomly the data set, X, into m subsets and use their mean values as initial estimates of the $\boldsymbol{\theta}_j$s. A number of variants based on different partition schemes of X and running the *k*-means algorithm many times have also been proposed. See, for example, [Kauf 90, Pena 99, Brad 98]. An alternative approach is discussed in [Lika 03], where the representatives are computed iteratively, one at a time, by running the algorithm mN times. The authors claim convergence that is independent of the initial estimates, at the cost, of course, of increased computational complexity.

- Another path is to adopt tools from stochastic optimization techniques, such as simulated annealing and genetic algorithms (see also Chapter 15), in that such techniques guarantee, in probability, the computation of the global minimum of $J(\boldsymbol{\theta}, U)$ at the cost of excessive computations. Extensions of the *k*-means in this spirit are discussed in [Kris 99] and [Pata 01].

- Although computing the optimal partition for the *k*-means, as well as the *k*-medoids algorithm to be discussed next, is an NP-hard problem, recent theoretical work shows that it is possible to find solutions that are provably good approximations. In addition, this can be achieved via reasonably efficient techniques, see, for example, [Indy 99, Kuma 04, Kanu 04].

- The number of clusters m in the data set, X, is required as an input parameter to the algorithm. Clearly, a poor estimate of m will prevent the algorithm to unravel the underlying clustering structure in X. To this end, a number of variants of the *k*-means algorithm have been suggested, employing various splitting, merging, and discarding operations among the resulting clusters [Ande 73] (based on suitably chosen user-defined parameters). No doubt, such *ad hoc* techniques are no more the result of an optimization process.

  An alternative method for estimating m is to apply the procedure described in Section 12.3, using a sequential clustering algorithm.

- *k*-means is sensitive to outliers and noise. The outliers, being points in X, are necessarily assigned to one of the clusters. Thus, they influence the respective means and, as a consequence, the final clustering. Taking into account that in general small clusters are likely to be formed by outliers, a version of the algorithm given in [Ball 67] deals with outliers by simply discarding "small" clusters.

  In addition, this drawback of the *k*-means gave rise to the so-called *k-medoids algorithms* (see next section), where each cluster is represented by one of its points. This way of representing clusters is less sensitive to outliers, at the cost of increased computational complexity.

- *k*-means is generally applicable to data sets with continuous valued feature vectors, and in principle it is not suitable for data with nominal (categorical) coordinates. Variants of the *k*-means that can deal with data sets consisting of data stemming from a finite discrete-valued domain are discussed in [Huan 98, Gupa 99]. The *k-medoids* algorithms, discussed next, are another possibility.
- As it is currently the trend kernelized versions of the *k*-means algorithm have also been proposed, see, for example, [Scho 98, Giro 02].

Other advances related to the *k*-means and other square-error-based clustering algorithms can be found in [Hans 01, Kanu 00, Pata 02, Su 01, Wags 01].

14.5.2 *k*-Medoids Algorithms

In the *k*-means algorithm described in the previous section, each cluster is represented by the mean of its vectors. In the *k-medoids* methods, discussed in this section, each cluster is represented by a vector selected *among the elements* of X, and we will refer to it as the *medoid*. Apart from its medoid, each cluster contains all vectors in X that (a) are not used as medoids in other clusters and (b) lie closer to its medoid than to the medoids representing the other clusters. Let Θ be the set of medoids for all clusters. We will denote by I_Θ the set of indices of the points in X that constitute Θ, and by $I_{X-\Theta}$ the set of indices of the points that are not medoids. Thus, if for example $\Theta = \{\boldsymbol{x}_1, \boldsymbol{x}_5, \boldsymbol{x}_{13}\}$ is the set of medoids for a three-cluster case then $I_\Theta = \{1, 5, 13\}$. The quality of the clustering associated with a given set Θ of medoids is assessed through the cost function

$$J(\Theta, U) = \sum_{i \in I_{X-\Theta}} \sum_{j \in I_\Theta} u_{ij} d(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{14.87}$$

and

$$u_{ij} = \begin{cases} 1, & \text{if } d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \min_{q \in I_\Theta} d(\boldsymbol{x}_i, \boldsymbol{x}_q) \\ 0, & \text{otherwise} \end{cases} \qquad i = 1, \ldots, N \tag{14.88}$$

Thus, obtaining the set of medoids Θ that best represents the data set, X, is equivalent to minimizing $J(\Theta, U)$. Note that Eqs. (14.87) and (14.81) are almost identical. The only difference is that $\boldsymbol{\theta}_j$ in Eq. (14.81) is replaced by $\boldsymbol{x}_j$, in that each cluster is now represented by a vector in X.

Representing clusters using medoids has two advantages over the *k*-means algorithm. First, it can be used with data sets originating from either continuous or discrete domains, whereas *k*-means is suited only for the case of continuous domains because in a discrete domain application the mean of a subset of the data vectors is not necessarily a point lying in the domain (see Figure 14.18a). Second, *k*-medoids algorithms tend to be less sensitive to outliers compared to the *k*-means algorithm (see Figure 14.18b). However, it should be noted that the mean of a cluster has a clear geometrical and statistical meaning, which is not necessarily the case with the

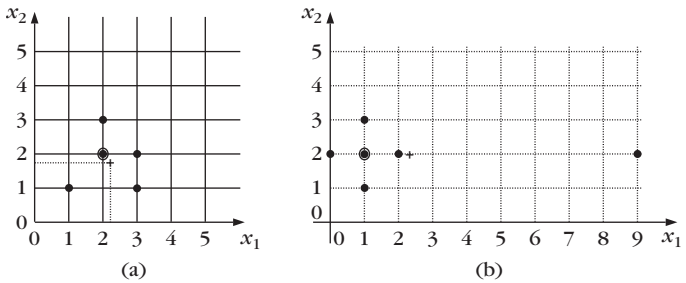

FIGURE 14.18

(a) The five-point two-dimensional set stems from the discrete domain $\mathcal{D} = \{1, 2, 3, 4, \ldots\} \times \{1, 2, 3, 4, \ldots\}$. Its medoid is the circled point. The mean of the vectors of the set is denoted by "+" and does not belong to $\mathcal{D}$. (b) In the six-point two-dimensional set, the point $(9, 2)$ can be considered an outlier. Clearly, the outlier affects significantly the position of the mean of the set, whereas it has no affect on the position of its medoid.

medoids. In addition, the algorithms for the estimation of the best set of medoids are computationally more demanding compared to the *k*-means algorithm.

In the sequel, we describe three *k*-medoids algorithms: PAM (Partitioning Around Medoids), CLARA (Clustering LARge Applications), and CLARANS (Clustering Large Applications based on RANdomized Search). Note that the last two algorithms are inspired by PAM but are suitable for dealing with large data sets more efficiently than PAM.

The PAM Algorithm

To determine the set Θ of the m medoids that best represent the data set, PAM uses a function optimization procedure that minimizes $J(\Theta, U)$, subject to the constraint that the representatives of the clusters are themselves elements of X. Before proceeding any further, some definitions are in order. Two sets of medoids Θ and Θ', each consisting of m elements, are called *neighbors* if they share $m - 1$ elements. Clearly, the number of neighbors a set $\Theta \subset X$ with m elements can have is $m(N - m)$. Also, let Θ_{ij} denote the neighbor of Θ that results if $\boldsymbol{x}_j$, $j \in I_{X-\Theta}$ replaces $\boldsymbol{x}_i$, $i \in I_\Theta$. Finally, let $\Delta J_{ij} = J(\Theta_{ij}, U_{ij}) - J(\Theta, U)$.

PAM starts with a set Θ of m medoids, which are *randomly* selected out of X. Then, among all $m(N - m)$ neighbors, Θ_{ij}, $i \in I_\Theta$, $j \in I_{X-\Theta}$, of the set Θ, we select Θ_{qr}, $q \in I_\Theta$, $r \in I_{X-\Theta}$, with $\Delta J_{qr} = \min_{ij} \Delta J_{ij}$. If ΔJ_{qr} is negative, then Θ is replaced by Θ_{qr} and the same procedure is repeated. Otherwise, if $\Delta J_{qr} \geq 0$ the algorithm has reached a local minimum and terminates. Once the set Θ that best represents the data has been determined, each $\boldsymbol{x} \in X - \Theta$ is assigned to the cluster represented by the closest to it medoid.

Let us focus now on the computation of ΔJ_{ij}. This quantity may be written as

$$\Delta J_{ij} = \sum_{b \in I_{X-\Theta}} C_{bij} \tag{14.89}$$

where C_{bij} is the difference in J resulting from the (possible) assignment of the vector $\boldsymbol{x}_b \in X - \Theta$ from the cluster it currently belongs to another, as a consequence of the replacement of $\boldsymbol{x}_i \in \Theta$ by $\boldsymbol{x}_j \in X - \Theta$. For the computation of C_{bij} we consider the following four cases.

- Suppose that $\boldsymbol{x}_b$ belongs to the cluster represented by $\boldsymbol{x}_i$. Also, let $\boldsymbol{x}_{b2} \in \Theta$ denote the second closest to $\boldsymbol{x}_b$ representative. If $d(\boldsymbol{x}_b, \boldsymbol{x}_j) \geq d(\boldsymbol{x}_b, \boldsymbol{x}_{b2})$ (see Figure 14.19a), then after the replacement of $\boldsymbol{x}_i$ by $\boldsymbol{x}_j$ in Θ, $\boldsymbol{x}_b$ will now be represented by $\boldsymbol{x}_{b2}$. Thus,

$$C_{bij} = d(\boldsymbol{x}_b, \boldsymbol{x}_{b2}) - d(\boldsymbol{x}_b, \boldsymbol{x}_i) \geq 0 \tag{14.90}$$

  The equality corresponds to the case of a tie, that is, $d(\boldsymbol{x}_b, \boldsymbol{x}_{b2}) = d(\boldsymbol{x}_b, \boldsymbol{x}_i)$.

- Suppose again that $\boldsymbol{x}_b$ belongs to the cluster represented by $\boldsymbol{x}_i$ and let $\boldsymbol{x}_{b2}$ denote the second closest to $\boldsymbol{x}_b$ representative. If $d(\boldsymbol{x}_b, \boldsymbol{x}_j) \leq d(\boldsymbol{x}_b, \boldsymbol{x}_{b2})$ (see Figure 14.19b–c), then after the replacement of $\boldsymbol{x}_i$ by $\boldsymbol{x}_j$ in Θ, $\boldsymbol{x}_b$ will now be represented by $\boldsymbol{x}_j$. Thus,

$$C_{bij} = d(\boldsymbol{x}_b, \boldsymbol{x}_j) - d(\boldsymbol{x}_b, \boldsymbol{x}_i) \tag{14.91}$$

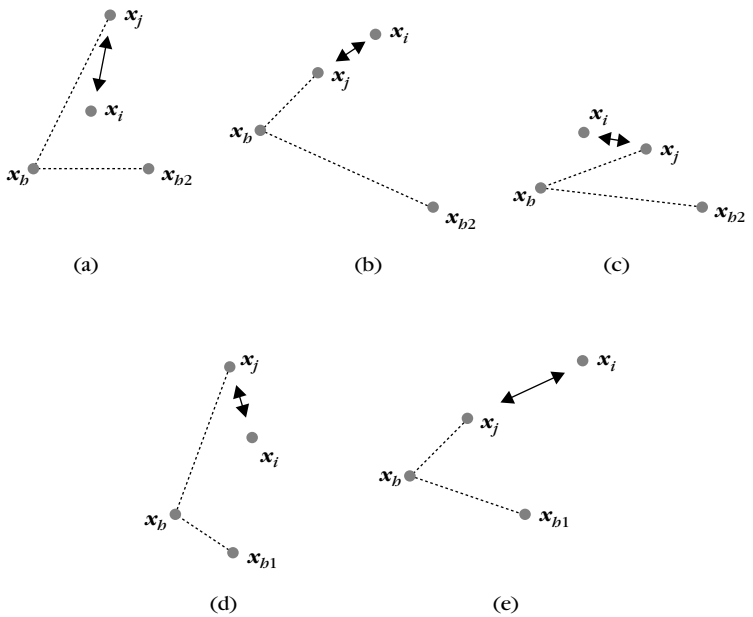

FIGURE 14.19

Different cases encountered in the computation of $C_{\hat{b}ij}$: (a) $C_{\hat{b}ij} > 0$, (b) $C_{\hat{b}ij} < 0$, (c) $C_{\hat{b}ij} > 0$, (d) $C_{\hat{b}ij} = 0$, (e) $C_{\hat{b}ij} < 0$.

In this case, C_{bij} may be either negative, zero, or positive (e.g., Figure 14.19b–c).

- Suppose now that $\boldsymbol{x}_b$ is not represented by $\boldsymbol{x}_i$ and let $\boldsymbol{x}_{b1}$ be the closest to $\boldsymbol{x}_b$ medoid. If $d(\boldsymbol{x}_b, \boldsymbol{x}_{b1}) \leq d(\boldsymbol{x}_b, \boldsymbol{x}_j)$ (see Figure 14.19d), then $\boldsymbol{x}_b$ will continue to be represented by $\boldsymbol{x}_{b1}$. Thus,

$$C_{bij} = 0 \tag{14.92}$$

- Finally, suppose that $\boldsymbol{x}_b$ is not represented by $\boldsymbol{x}_i$ and let $\boldsymbol{x}_{b1}$ be the closest to $\boldsymbol{x}_b$ medoid. If $d(\boldsymbol{x}_b, \boldsymbol{x}_{b1}) > d(\boldsymbol{x}_b, \boldsymbol{x}_j)$ (see Figure 14.19e), then

$$C_{bij} = d(\boldsymbol{x}_b, \boldsymbol{x}_j) - d(\boldsymbol{x}_b, \boldsymbol{x}_{b1}) < 0 \tag{14.93}$$

Experimental results ([Kauf 90]) show that PAM works satisfactorily for relatively small data sets. However, it becomes inefficient for large data sets because its time complexity per iteration increases quadratically with respect to N. This is easily verified because at each iteration the term ΔJ_{ij} for $m(N-m)$ pairs of vectors has to be calculated. In addition, for the computation of a single $\Delta J_{ij}, N-m$ of C_{bij} terms have to be considered (see Eq. (14.89)). Thus, the total complexity of the algorithm per iteration amounts to $O(m(N-m)^2)$.

CLARA and CLARANS: k-Medoids Algorithms for Large Data Sets

These algorithms are versions of the PAM algorithm, and they have been developed to cope with the high computational demands imposed by large data sets. Both algorithms exploit the idea of randomized sampling but each in a different way. Specifically, the idea underlying CLARA is to draw randomly a sample X' of size N' from the entire data set, X, and to determine the set Θ' of the medoids that best represents X' using the PAM algorithm. The rationale behind this algorithm is based on the assumption that if the sample X' is drawn in a way that is representative of the statistical distribution of the data points in X the set Θ' will be a satisfactory approximation of the set Θ of the medoids that would result if the PAM algorithm was run on the entire data set. To obtain better results, CLARA runs PAM on a number of sample subsets of X, denoted by $X'_1, \ldots, X'_s$. Each run returns a set of medoids denoted by $\Theta'_1, \ldots, \Theta'_s$. Then, the quality of the clustering associated with each of them is assessed through Eq. (14.87), where the entire data set, X, is taken into account. Experimental studies ([Kauf 90]) suggest that $s = 5$ and $N' = 40 + 2m$ lead to satisfactory results.

The philosophy behind CLARANS is different from that behind CLARA. According to CLARANS, PAM is applied on the entire data set, X, but with a slight modification. At each iteration, not all neighbors of the current set Θ of medoids are considered. Instead, only a randomly selected fraction $q < m(N-m)$ of them is utilized. The selected neighbors are considered in a sequential manner and if the currently considered neighbor Θ_{ij} of Θ is better than Θ (in terms of J) then Θ is replaced by Θ_{ij} and the procedure is repeated. When none of the q selected

neighbors of Θ is better than Θ, in terms of J, then Θ is considered to be "local minimum."[11] Then CLARANS starts from another arbitrarily chosen Θ, and the same procedure is followed in order to obtain a different "local minimum." This is repeated for a predetermined number of times, s, and the algorithm outputs the best among the s "local minima." In the sequel, based on this set of medoids each point $\boldsymbol{x} \in X - \Theta$ is assigned to the cluster whose representative is closest to $\boldsymbol{x}$.

Remarks

- The performance of CLARANS depends on the two parameters q and s. As q gets closer to $m(N - m)$, CLARANS approaches PAM and the time complexity increases. As suggested in [Ng 94a], a typical value for s is 2, whereas for q it is suggested to be chosen as the maximum between $0.12m\,(N - m)$ and 250.
- The CLARANS algorithm can also be described in terms of graph theory concepts ([Ng 94]).
- CLARANS unravels better-quality clusterings than CLARA. On the other hand, in some cases CLARA runs significantly faster than CLARANS ([Ng 94]). It must be pointed out that CLARANS retains its quadratic computational nature and is thus not appropriate for very large data sets.

14.6 VECTOR QUANTIZATION

An area that has close affinity with clustering is that of vector quantization (VQ), and it has been the focus of intensive research effort over the past years (e.g., [Gray 84, Gers 92]). Vector quantization techniques are used mainly for data compression, which is a prerequisite for achieving better computer storage utilization and better bandwidth utilization (in communications). Let T be the set of all possible vectors for the problem at hand. The task of VQ may be stated in the general case in which T is a continuous subset of $\mathcal{R}^l$. The idea is rather simple. We separate T into m distinct regions R_j that exhaust T, and we represent each of them with an l-dimensional vector, the so-called *code vector* or *reproduction vector*, $\boldsymbol{\theta}_j, j = 1, \ldots, m$. In the sequel, given an $\boldsymbol{x} \in T$, we determine the region where it belongs, say R_j, and we adopt the corresponding representative $\boldsymbol{\theta}_j$, instead of $\boldsymbol{x}$, for further use, that is, storage or transmission. This is obviously associated with some information loss, which is known as *distortion*. The major goal in VQ is to define the regions R_j and their representatives $\boldsymbol{\theta}_j$ so that distortion is minimized.

After stating the general idea, let us proceed now to some more formal definitions. A *vector quantizer* Q of dimension l and size m is a mapping of T to a finite set C,

[11] Note that Θ may not actually be a local minimum because it may have a nonselected neighbor that gives lower value of J.

which is called the *reproduction set* and contains m output reproduction points, the code vectors or *code words*. Thus

$$Q : T \rightarrow C$$

where $C = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_m\}$ with $\boldsymbol{\theta}_i \in T$. Each code vector $\boldsymbol{\theta}_i$ represents a specific region R_i of the vector space.

The next question that naturally arises is how one can select the code vectors $\boldsymbol{\theta}_j$ in such a way as to achieve the least possible distortion. A usual approach is to optimize an appropriate criterion function, which in this framework is also known as a *distortion function*, with respect to $\boldsymbol{\theta}_j$'s. Let $\boldsymbol{x}$ be a random vector that models T and $p(\boldsymbol{x})$ its corresponding pdf.

A commonly used distortion criterion is the *average expected quantization error*, which is defined as

$$D(Q) = \sum_{j=1}^{m} D_j(Q) \tag{14.94}$$

where

$$D_j(Q) = \int_{R_j} d(\boldsymbol{x}, \boldsymbol{\theta}_j) p(\boldsymbol{x})\, d\boldsymbol{x} \tag{14.95}$$

$D_j(Q)$ is known as the average quantization error for region R_j. The quantity d is a distance measure, for example, Euclidean, and it is also referred to as a *distortion measure*.

When a finite number of samples, $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$, of $\boldsymbol{x}$ is available, the distortion criterion becomes

$$D(Q) = \sum_{i=1}^{N} d(\boldsymbol{x}_i, Q(\boldsymbol{x}_i)) P(\boldsymbol{x}_i) \tag{14.96}$$

where $Q(\boldsymbol{x}_i) \in C$ is the code vector that represents $\boldsymbol{x}_i$ and $P(\boldsymbol{x}_i)(>0)$ $i = 1, \ldots, N$, the respective probabilities.

In [Gers 92] it is shown that the following conditions are *necessary* for a given quantizer to be optimal. The first refers to the *encoder* part of the vector quantizer, that is, the optimal way in which T is partitioned in the regions $R_j, j = 1, \ldots, m$, given the code vectors $\boldsymbol{\theta}_j$. It is known as the *nearest neighbor condition*, and it states that

- *For fixed C,*

$$Q(\boldsymbol{x}) = \boldsymbol{\theta}_j \quad \text{only if } d(\boldsymbol{x}, \boldsymbol{\theta}_j) \leq d(\boldsymbol{x}, \boldsymbol{\theta}_k), \quad k = 1, \ldots, m, k \neq j$$

The second condition refers to the *decoder* part of the VQ, that is, the optimal way the code words $\boldsymbol{\theta}_j$ are chosen, given the partition regions $R_j, j = 1, \ldots, m$. It is known as the *centroid condition,* and it is stated as

- *For a fixed partition $R_1, R_2, \ldots, R_m$, each $\boldsymbol{\theta}_j$ is chosen such that*

$$\int_{R_j} d(\boldsymbol{x}, \boldsymbol{\theta}_j) p(\boldsymbol{x})\, d\boldsymbol{x} = \min_{\boldsymbol{y}} \int_{R_j} d(\boldsymbol{x}, \boldsymbol{y}) p(\boldsymbol{x})\, d\boldsymbol{x}$$ [12]

In the case that T is finite, the integrals are replaced with summations. One way to compute the code vectors of the set C is to start with an arbitrary initial estimate of the code vectors and to iteratively apply the nearest neighbor condition and the centroid condition, interchangeably, until a termination criterion is satisfied.[13] This is the well-known Lloyd's algorithm [Lloy 82].[14] Note that if $P(\boldsymbol{x}_i) = 1/N$, $\forall \boldsymbol{x}_i \in T$, Lloyd's algorithm coincides with the generalized hard clustering algorithmic scheme. This is not surprising. Both algorithms try to place optimally point representatives in space. Note, however, that despite algorithmic similarities, the two tasks have different goals. The goal of VQ is to place points in space in a way that is representative of the data distribution. On the other hand, clustering focuses on revealing the underlying clusters in X, if they exist.

Finally, it is worth pointing out that many other models for vector quantization have been proposed in the literature. For example, hierarchical and fuzzy vector quantizers are discussed in [Lutt 89] and [Kara 96], respectively.

APPENDIX

Derivation of μ_j and Σ_j for the EM Algorithm (Section 14.2)

Equations (14.3) and (14.13) for $\boldsymbol{\mu}_j$ lead to

$$\boldsymbol{\mu}_j = \frac{\sum_{k=1}^{N} P(C_j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))\boldsymbol{x}_k}{\sum_{k=1}^{N} P(C_j|\boldsymbol{x}_k; \boldsymbol{\Theta}(t))} \tag{14.97}$$

for $j = 1, \ldots, m$.

Let us now turn our attention to Σ_j. Let σ_{rs} be the (r, s) element of Σ_j^{-1}. Then Eq. (14.13) gives

$$\frac{\partial}{\partial \sigma_{rs}} \ln p(\boldsymbol{x}|C_j; \boldsymbol{\theta}_j) = \frac{1}{2}|\Sigma_j| \frac{\partial}{\partial \sigma_{rs}} |\Sigma_j^{-1}| - \frac{1}{2}(x_r - \mu_{jr})(x_s - \mu_{js})$$

or

$$\frac{\partial}{\partial \sigma_{rs}} \ln p(\boldsymbol{x}|C_j; \boldsymbol{\theta}_j) = \frac{1}{2}|\Sigma_j| \boldsymbol{\sigma}_{rs} - \frac{1}{2}(x_r - \mu_{jr})(x_s - \mu_{js})$$

[12] An additional optimality condition, given in [Gers 92], is that no vector in T is equidistant from two (or more) code vectors.

[13] One such criterion is to have the same values for all $\boldsymbol{\theta}_j$'s, $j = 1, \ldots, m$, for two successive iterations.

[14] For the special case in which the squared Euclidean distance is considered, the centroid condition becomes $\boldsymbol{\theta}_j = (1/n_j) \sum_{\boldsymbol{x} \in R_j} \boldsymbol{x}$, where n_j is the number of vectors that lie in R_j. The corresponding algorithm is the isodata algorithm, which in this framework is also called the LBG algorithm [Lind 80].

where $\boldsymbol{\sigma}_{rs}$ is the cofactor of σ_{rs}[15] and x_r, μ_{jr} (x_s, μ_{js}) are the rth (sth) coordinates of $\boldsymbol{x}$ and $\boldsymbol{\mu}_j$, respectively. Thus,

$$\frac{\partial \ln p(\boldsymbol{x}|C_j;\boldsymbol{\theta}_j)}{\partial \Sigma_j^{-1}} = \frac{1}{2}|\Sigma_j|\boldsymbol{\sigma} - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)(\boldsymbol{x}-\boldsymbol{\mu}_j)^T \tag{14.98}$$

where $\boldsymbol{\sigma}$ is the matrix of the cofactors of Σ_j^{-1}. Since, in general, $|\Sigma^{-1}| \neq 0$, the following identity holds from linear algebra:

$$\Sigma_j^{-1}\boldsymbol{\sigma}^T = |\Sigma_j^{-1}|I$$

Premultiplying both sides of this equation by Σ_j and noting that σ is a symmetric matrix, we obtain

$$\boldsymbol{\sigma} = |\Sigma_j^{-1}|\Sigma_j$$

Substituting the last result into Eq. (14.98), we obtain

$$\frac{\partial \ln p(\boldsymbol{x}|C_j;\boldsymbol{\theta}_j)}{\partial \Sigma_j^{-1}} = \frac{1}{2}\Sigma_j - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)(\boldsymbol{x}-\boldsymbol{\mu}_j)^T \tag{14.99}$$

Substituting this result into Eq. (14.3) and after some manipulations, we finally obtain

$$\Sigma_j = \frac{\sum_{k=1}^{N} P(C_j|\boldsymbol{x}_k;\boldsymbol{\Theta}(t))(\boldsymbol{x}_k-\boldsymbol{\mu}_j)(\boldsymbol{x}_k-\boldsymbol{\mu}_j)^T}{\sum_{k=1}^{N} P(C_j|\boldsymbol{x}_k;\boldsymbol{\Theta}(t))} \tag{14.100}$$

for $j = 1, \ldots, m$.

14.7 PROBLEMS

14.1 Consider the case in which there exist m clusters in X, which are characterized by normal distributions of unknown means and known covariance matrices; that is, the parameter vector $\boldsymbol{\theta}$ consists only of the parameters of $\boldsymbol{\mu}_j$, $j = 1, \ldots, m$. State the corresponding generalized mixture decomposition algorithmic scheme (GMDAS).

14.2 Consider the case that there exist m clusters in X which are described by normal distributions of unknown means and unknown diagonal covariance matrices. Derive the corresponding GMDAS.

14.3 Consider the case that there exist m clusters in X which are described by normal distributions. Derive the maximum likelihood estimates of the means $\boldsymbol{\mu}_j$ and covariance matrices Σ_j when:

a. the means and the covariance matrices are unknown and

[15] Recall that the cofactor of the element a_{ij} of a matrix A is the determinant of the matrix that results from A if we delete its ith row and its jth column.

b. the means and the covariance matrices are unknown but $\Sigma_j = \Sigma$, $j = 1, \ldots, m$.

Compare the results of (a) with those of Section 14.2.1.

14.4 Consider the data set $X = \{\boldsymbol{x}_i \in \mathcal{R}^2, i = 1, \ldots, 16\}$, where $\boldsymbol{x}_1 = [2, 0]^T$, $\boldsymbol{x}_2 = [\sqrt{2}, \sqrt{2}]^T$, $\boldsymbol{x}_3 = [0, 2]^T$, $\boldsymbol{x}_4 = [-\sqrt{2}, \sqrt{2}]^T$, $\boldsymbol{x}_5 = [-2, 0]^T$, $\boldsymbol{x}_6 = [-\sqrt{2}, -\sqrt{2}]^T$, $\boldsymbol{x}_7 = [0, -2]^T$, $\boldsymbol{x}_8 = [\sqrt{2}, -\sqrt{2}]^T$. The remaining points $\boldsymbol{x}_i, i = 9, \ldots, 16$, are obtained from the first eight points as follows. The first coordinate of $\boldsymbol{x}_i, i = 9, \ldots, 16$, equals the first coordinate of $\boldsymbol{x}_{i-8}$ plus 6, while the second coordinate of $\boldsymbol{x}_i$, $i = 9, \ldots, 16$, equals the second coordinate of $\boldsymbol{x}_{i-8}$.

a. Run the GMDAS, with Gaussian pdf's, to obtain estimates of $\boldsymbol{\mu}_j$, and Σ_j, $j = 1, 2$.

b. Does the algorithm determine the clusters that underlie X correctly? Justify your answer.

Hint: In the rest problems, where possible, one may use the MATLAB codes given in the Computer Programs section of this chapter.

14.5 Consider the setup of Problem 14.4, with the difference that the points $\boldsymbol{x}_i$, $i = 9, \ldots, 16$, are derived as follows. The first coordinate of $\boldsymbol{x}_i, i = 9, \ldots, 16$, equals the first coordinate of $\boldsymbol{x}_{i-8}$ plus 2, while the second coordinate of $\boldsymbol{x}_i$, $i = 9, \ldots, 16$, equals to the second coordinate of $\boldsymbol{x}_{i-8}$.

a. Run the GMDAS, with Gaussian pdf's, to obtain estimates for $\boldsymbol{\mu}_j$, and Σ_j, $j = 1, 2$.

b. Does the algorithm determine the clusters that underlie X accurately? Justify your answer.

c. Compare the results obtained in this and the previous problem.

14.6 Consider four two-dimensional distributions with means $\boldsymbol{\mu}_1 = [0, 0]^T, \boldsymbol{\mu}_2 = [2, 2]^T, \boldsymbol{\mu}_3 = [4, 0]^T, \boldsymbol{\mu}_4 = [7, 0]^T$, respectively, and covariance matrices

$$\Sigma_1 = \begin{bmatrix} 1 & 0.3 \\ 0.3 & 1 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}, \quad \Sigma_4 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

respectively. Draw 80 points from each distribution and let X be the set that contains these 320 points. Initialize $\boldsymbol{\mu}_i$ and Σ_i, $i = 1, \ldots, 4$, as in Example 14.1. Set $m = 4$, $\varepsilon = 0.01$ and run the GMDAS, with Gaussian pdf's.

a. What are the estimates of $\boldsymbol{\mu}_j, j = 1, \ldots, 4$, and $\Sigma_j, j = 1, \ldots, 4$?

b. Assign each feature vector $\boldsymbol{x} \in X$ to a cluster C_j according to the Bayes decision rule.

c. Derive the respective confusion matrix.

d. Run the algorithm for $m = 3$ and $m = 2$ and repeat steps (a) and (b). Discuss the results.

14.7 In the framework of Example 14.4, prove that for $m = 2, q \in \{2, 3\}$ and for fixed $\boldsymbol{\theta}$, there are cases where the fuzzy clusterings are favored against the hard ones.

14.8 Find the relation between $\boldsymbol{p}$ and A, $\boldsymbol{b}$, and c so that Eqs. (14.35) and (14.36) are equivalent.
Hint: Consider each coordinate of $\boldsymbol{p}$ separately.

14.9 Let $l = 2$. Prove that the substitution of $\boldsymbol{z}$, as given by Eq. (14.44), into Eq. (14.42) leads to a fourth-degree polynomial, with respect to λ.

14.10 Prove Eq. (14.50).

14.11 **a.** Derive Eqs. (14.56) and (14.57) for the fuzzy C ellipsoidal shells (FCES) algorithm.

b. Write the parameter determination part of the fuzzy C ellipsoidal shells (FCES) algorithm.

14.12 Derive the fuzzy C quadric shells (FCQs) algorithm by minimizing Eq. (14.58) under constraint (v).

14.13 **a.** State explicitly the modified fuzzy C quadric shells (MFCQS) algorithm.

b. Under what general conditions are the algebraic and the perpendicular distances close to each other?

14.14 What is the relation between the perpendicular and the radial distance between a point $\boldsymbol{x}$ and a hyperspherical cluster?

14.15 **a.** Derive the AFCS algorithm [Dave 92b] for the case that spherical clusters are to be recovered. The distance between a point $\boldsymbol{x}$ and a hypersphere Q with center $\boldsymbol{c}$ and radius r is

$$d^2(\boldsymbol{x}, Q) = (\|\boldsymbol{x} - \boldsymbol{c}\| - r)^2$$

b. Derive the fuzzy C spherical shells (FCSS) algorithm [Kris 92b] for the case that spherical clusters are to be identified.

14.16 Prove Eq. (14.64).

14.17 Derive the possibilistic algorithm obtained via minimization of the function J_1 given in Eq. (14.77).

14.18 Plot u_{ij} versus $d(\boldsymbol{x}_i, \boldsymbol{\theta}_j)/\eta_j$, using Eq. (14.78). Compare this plot with the one shown in Figure 14.15.

14.19 Compare the isodata algorithm with the variant of the BSAS proposed in MACQ 67 and outlined in Section 12.6.

MATLAB PROGRAMS AND EXCERCISES

Computer Programs

14.1 *GMDAS algorithm.* Write a MATLAB function named *GMDAS* that implements the GMDAS algorithm when normal distributions are adopted for the representation of the clusters. The function takes as input (a) an $l \times N$ dimensional matrix X whose columns are the data vectors, (b) an $l \times m$ dimensional matrix *mv* whose ith column contains an initial estimate of the mean of the ith normal distribution, (c) an $l \times l \times m$ dimensional matrix *mc*, whose ith two-dimensional $l \times l$ component contains an initial estimate of the covariance matrix of the ith normal distribution, (d) a parameter e used in the termination condition of the algorithm, which is $||\boldsymbol{\Theta}(t) - \boldsymbol{\Theta}(t-1)|| < e$, (e) the maximum number of allowable iterations, *maxiter*, (f) a seed *sed* for the *rand* MATLAB function. The output consists of (a) an m dimensional row vector *ap* with the a priori probabilities, (b) an $N \times m$ dimensional matrix *cp*, whose ith row contains the conditional probabilities $P(C_j|\boldsymbol{x}_i), j = 1, \ldots, m$, (c)–(d) the final estimates *mv* and *mc* of the mean values and covariance matrices of the normal distributions, respectively, (e) the number of iterations required for convergence, (f) the vector *diffvec* that contains the differences between successive values of $\boldsymbol{\Theta}$, during the training phase.

Solution

For an implementation of this function see in http://www.di.uoa.gr/~stpatrec.

14.2 *Random initialization.* Write a MATLAB function named *rand_vec*, that selects randomly m vectors in the range of values of a given data set. The function takes as input (a) an $l \times N$ dimensional matrix X, whose columns are the data vectors, (b) the number m of column vectors that will be produced, (c) a seed (integer) for the initialization of the *rand* MATLAB function. It returns an $l \times m$ dimensional matrix consisting of the randomly selected column vectors, which will be used for initialization purposes.

Solution

```
function w=rand_vec(X,m,sed)
  rand('seed',sed)
  mini=min(X');
  maxi=max(X');
  w=rand(size(X,1),m);
  for i=1:m
    w(:,i)=w(:,i).*(maxi'-mini')+mini';
  end
```

14.3 *k-means algorithm.* Write a MATLAB function, named *k_means*, that implements the *k*-means algorithm. The function takes as input (a) an $l \times N$ dimensional matrix X, each column of which is a data vector, (b) an $l \times m$ dimensional matrix w, the ith column of which is the initial estimate of the ith representative. The output consists of (a) a matrix w similar to the previous one, which contains the final estimates of the representatives and (b) an N-dimensional row vector whose ith element contains the identity number of the cluster where the ith vector belongs (an integer in the set $\{1, 2, \ldots, m\}$).

Solution

```
function [w,bel]=k_means(X,w)
 [l,N]=size(X);
 [l,m]=size(w);
 e=1;
 iter=0;
 while(e~=0)
    iter=iter+1;
    w_old=w;
    dist_all=[];
    for j=1:m
      dist=sum(((ones(N,1)*w(:,j)'-X'). ^ 2)');
      dist_all=[dist_all; dist];
    end
    [q1,bel]=min(dist_all);
    for j=1:m
      if(sum(bel==j)~=0)
        w(:,j)=sum(X'.*((bel==j)'*ones(1,l))) / sum(bel==j);
      end
    end
    e=sum(abs(w-w_old));
 end
```

Computer Experiments

14.1 **a.** Generate $q = 50$ two-dimensional vectors from three normal distributions with mean values $[1,\ 1]^T$, $[5,\ 5]^T$, $[9,\ 1]^T$ and covariance matrices $\begin{bmatrix} 1 & 0.4 \\ 0.4 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & -0.6 \\ -0.6 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, respectively. Form the 2×150 dimensional matrix X, whose columns are the data vectors produced before.

b. Run the *GMDAS* algorithm on X setting $e = 0.01, maxiter = 300, sed = 110$ and initializing randomly the *mv* and *mc* using the *rand* MATLAB function.

c. Compute the sample mean and the sample covariance matrix for the vectors from each distribution and compare them with the corresponding estimates produced by the algorithm.

d. Comment on the conditional probabilities for each vector.

e. Repeat (b)–(d) five times for different initial estimates for mv and mc.

Hint: Assuming that the first group of q vectors in X is generated from the first distribution, the second group of q vectors in X is generated from the second distribution and so on, the sample mean and sample covariance matrix of the ith group are computed via $sum\,(X(:,(i-1)*q+1:i*q)')'/q$ and $cov(X(:,(i-1)*q+1:i*q)')$, respectively.

14.2 Repeat 14.1 when the mean values of the normal distributions are $[1,\ 1]^T$, $[3.5,\ 3.5]^T, [6,\ 1]^T$.

14.3 a. Repeat 14.1 when the mean values of the normal distributions are $[1,\ 1]^T$, $[2,\ 2]^T, [3,\ 1]^T$.

b. Compare the results obtained in (a) with those obtained in 14.1 and 14.2 and draw your conclusions.

14.4 a. Generate 100 two-dimensional vectors from each one of the three normal distributions with mean values $m1 = [2,\ 2]^T$, $m2 = [6,\ 6]^T$, $m3 = [10,\ 2]^T$ and covariance matrices $S1 = S2 = S3 = 0.5 * I$. Form the 2×300 dimensional matrix X using as columns the vectors generated previously from the three distributions.

b. Run the *k*-means algorithm on X for $m = 2, 3, 4$ representatives using 10 different (randomly selected) initial conditions for the representatives, for each value of m.

c. Comment on the results.
Hint: Use the *rand_vec* function for the initialization of the representatives.

14.5 In the data set of the previous experiment apply the *k*-means algorithm for $m = 3$ representatives, initializing the representatives to the vectors $[-100,\ -100]^T, [4.5,\ 6.5]^T, [3.5,\ 5.5]^T$. Comment on the results.

14.6 a. Generate 400 two-dimensional vectors from the normal distribution with mean $[0,\ 0]^T$ and covariance matrix $1.5 * I$ and another 15 two-dimensional vectors from the normal distribution with mean $[7,\ 0]^T$ and covariance matrix I. Form the 2×415 dimensional matrix X using as columns the vectors generated previously from both distributions.

b. Run the *k*-means algorithm on X for $m = 2$ representatives using 10 different randomly selected initial conditions for the representatives.

c. Comment on the results.
Hint: Use the *rand_vec* function for the initialization of the representatives.

14.7 **a.** Generate 20 two-dimensional vectors from each one of the two normal distributions with mean values $m1 = [0, 0]^T$, $m2 = [6, 6]^T$ and covariance matrices $S1 = S2 = 0.5 * I$. Form the 2×40 dimensional matrix using as columns the vectors generated previously from the two distributions.

b. Run the fuzzy *c*-means (FCM) algorithm on the above data set with $m = 2$ representatives, initialized randomly. Comment on the grade of memberships of the data vectors in the two resulting clusters.
Hint: Just type

```
[w,U,obj_fun]=fcm(X,m)
```

This function returns (a) the cluster representatives in the rows of w, (b) the grade of membership of each vector to each cluster in matrix U, and (c) the values of the objective function during iterations.

14.8 Repeat the previous experiment when $S1 = S2 = 6 * I$.

14.9 Run the FCM algorithm for $m = 3$ representatives on the data sets produced in 14.7 and 14.8 and comment on the results.

14.10 Run the *k*-means algorithm on the data sets of experiments 14.7 and 14.8 for $m = 2$ randomly initialized representatives and compare the final values of the representatives with those produced by the FCM algorithm on these data sets. Comment on the results.

REFERENCES

[Ande 73] Anderberg M.R. *Cluster analysis for applications*, Academic Press, 1973.

[Ande 85] Anderson I., Bezdek J.C. "An application of the c-varieties clustering algorithms to polygonal curve fitting," *IEEE Transactions on Systems Man and Cybernetics*, Vol. 15, pp. 637–639, 1985.

[Ball 67] Ball G.H., Hall D.J. "A clustering technique for summarizing multivariate data," *Beh- avioral Science*, Vol. 12, pp. 153–155, March 1967.

[Barn 96] Barni M., Cappellini V., Mecocci A. "Comments on 'A possibilistic approach to clustering'," *IEEE Transactions on Fuzzy Systems*, Vol. 4(3), pp. 393–396, August 1996.

[Berk 02] Berkhin P. "Survey of clustering data mining techniques," *Technical Report*, Accrue Software Inc., 2002.

[Bezd 80] Bezdek J.C. "A convergence theorem for the fuzzy Isodata clustering algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2(1), pp. 1–8, 1980.

[Bezd 81] Bezdek J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum, 1981.

[Bezd 92] Bezdek J.C., Hathaway R.J. "Numerical convergence and interpretation of the fuzzy c-shells clustering algorithm," *IEEE Transactions on Neural Networks*, Vol. 3(5), pp. 787–793, September 1992.

[Bezd 95] Bezdek J.C., Hathaway R.J., Pal N.R., "Norm-induced shell prototypes (NISP) clus- tering," *Neural, Parallel and Scientific Computations*, Vol. 3, pp. 431–450, 1995.

[Bobr 91] Bobrowski L., Bejdek J.C. "c-Means clustering with l_1 and l_∞ norms," *IEEE Transactions on Systems Man and Cybernetics*, Vol. 21(3), pp. 545–554, May/June 1991.

[Brad 98] Bradley P., Fayyad U. "Refining initial points for *K*-means clustering," *Proceedings of the 15th International Conference on Machine Learning*, pp. 91–99, 1998.

[Cann 86] Cannon R.L., Dave J.V., Bezdek J.C. "Efficient implementation of the fuzzy c-means clustering algorithms," *IEEE Transactions on PAMI*, Vol. 8(2), pp. 248–255, March 1986.

[Chen 89] Chen D.S. "A data-driven intermediate level feature extraction algorithm," *IEEE Transactions on PAMI*, Vol. 11(7), pp. 749–758, July 1989.

[Chia 03] Chiang J.H., Hao P.Y., "A new kernel-based fuzzy clustering approach: support vector clustering with cell growing," *IEEE Transactions of Fuzzy Systems*, Vol. 11(4), pp. 518–527, 2003.

[Dave 92a] Dave R.N., Bhaswan K. "Adaptive fuzzy c-shells clustering and detection of ellipses," *IEEE Transactions on Neural Networks*, Vol. 3(5), pp. 643–662, 1992.

[Dave 92b] Dave R.N. "Generalized fuzzy c-shells clustering and detection of circular and elliptical boundaries," *Pattern Recognition*, Vol. 25(7), pp. 713–721, 1992.

[Duda 01] Duda R.O., Hart P.E., Stork D. *Pattern Classification*, John Wiley & Sons, 2001.

[Figu 02] Figueiredo M., Jain A.K. "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24(3), pp. 381–396, 2002.

[Forg 65] Forgy E. "Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications," *Biometrics*, Vol. 21, pp. 768–780, 1965.

[Frig 96] Frigui H., Krishnapuram R. "A comparison of fuzzy shell clustering methods for the detection of ellipses," *IEEE Transactions on Fuzzy Systems*, Vol. 4(2), pp. 193–199, May 1996.

[Gao 00] Gao X., Li J., Xie W., "Parameter optimization in FCM clustering algorithms," *Proc. of the Int. Conf. on Signal Processing (ICSP) 2000*, pp. 1457–1461, 2000.

[Gers 79] Gersho A. "Asymptotically optimal block quantization," *IEEE Transactions on Information Theory*, Vol. 25(4), pp. 373–380, 1979.

[Gers 92] Gersho A., Gray R.M. *Vector Quantization and Signal Compression*, Kluwer Publishers, 1992.

[Giro 02] Girolami M. "Mercer kernel based clustering in feature space," *IEEE Transactions on Neural Networks*, Vol. 13(3), pp. 780–784, 2002.

[Gnan 77] Gnanadesikan R. *Methods for Statistical Data Analysis of Multivariate Observations*, John Wiley & Sons, 1977.

[Grav 07] Graves D., Pedrycz W., "Fuzzy c-means, Gustafson-Kessel FCM, and kernel-based FCM. A comparative study," in *Analysis and Design on Intelligent Systems using Soft Computing Techniques*, eds. Mellin P. *et al.*, Springer, pp. 140–149, 2007.

[Gray 84] Gray R.M. "Vector quantization," *IEEE ASSP Magazine*, Vol. 1, pp. 4–29, 1984.

[Grol 05] Groll L., Jakel J., "An new convergence proof of fuzzy c-means", *IEEE Transactions on Fuzzy Systems*, Vol. 13(5), pp. 717–720, 2005.

[Gupa 99] Gupata S., Rao K., Bhatnagar V. "*K*-means clustering algorithm for categorical attributes," *Proceedings of the 1st International Conference on Data Warehousing and Knowledge Discovery*, pp. 203–208, Florence, Italy, 1999.

[Hans 01] Hansen P., Mladenovic. "*J*-means: A new local search heuristic for minimum sum of squares clustering." *Pattern Recognition*, Vol. 34, pp. 405–413, 2001.

[Hath 86] Hathaway R.J., Bezdek J.C. "Local convergence of the fuzzy c-means algorithms," *Pattern Recognition* Vol. 19(6), pp. 477–480, 1986.

[Hath 89] Hathaway R.J., Davenport J.W., Bezdek J.C. "Relational duals of the c-means clustering algorithms," *Pattern Recognition*, Vol. 22(2), pp. 205–212, 1989.

[Hath 93] Hathaway R.J., Bezdek J.C. "Switching regression models and fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, Vol. 1(3), pp. 195–204, August 1993.

[Hath 95] Hathaway R.J., Bezdek J.C. "Optimization of clustering criteria by reformulation," *IEEE Transactions on Fuzzy Systems*, Vol. 3(2), pp. 241–245, 1995.

[Hopp 99] Hoppner F., Klawonn F., Kruse R., Runkler T. *Fuzzy Cluster Analysis*, John Wiley & Sons, 1999.

[Horn 86] Horn B.K.P. *Robot Vision*, MIT Press, 1986.

[Huan 98] Huang Z. "Extensions to the *K*-means algorithm for clustering large data sets with categorical values," *Data Mining Knowledge Discovery*, Vol. 2, pp. 283–304, 1998.

[Indy 99] Indyk P. "A sublinear time approximation scheme for clustering in metric spaces," *Foundations of Computer Science (FOCS)*, pp. 154–159, 1999.

[Isma 86] Ismail M.A., Selim S.Z. "Fuzzy c-means: Optimality of solutions and effective termination of the algorithm," *Pattern Recognition*, Vol. 19(6), pp. 481–485, 1986.

[Kanu 00] Kanungo T., Mount D.M., Netanyahu N., Piatko C., Silverman R., Wu A. "An efficient *k*-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24(7), pp. 881–892, 2000.

[Kanu 04] Kanungo T., Mount D.M., Netanyahu N., Piatko C., Silverman R., Wu A. "A local search approximation algorithm for *k*-means clustering," *Computational Geometry*, Vol. 28(2–3), pp. 89–112, 2004.

[Kara 96] Karayiannis N.B., Pai P.-I. "Fuzzy algorithms for learning vector quantization," *IEEE Transactions on Neural Networks*, Vol. 7(5), pp. 1196–1211, September 1996.

[Kare 94] Karen D., Cooper D., Subrahmonia J. "Describing complicated objects by implicit polynomials," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16(1), pp. 38–53, 1994.

[Kauf 90] Kaufman L., Rousseeuw P. *Finding groups in data: An introduction to cluster analysis*. John Wiley & Sons, 1990.

[Kim 88] Kim T., Bezdek J.C., Hathaway R.J. "Optimality tests for fixed points of the fuzzy c-means algorithm," *Pattern Recognition*, Vol. 21(6), pp. 651–663, 1988.

[Kris 92a] Krishnapuram R., Freg C.-P. "Fitting an unknown number of lines and planes to image data through compatible cluster merging," *Pattern Recognition*, Vol. 25(4), pp. 385–400, 1992.

[Kris 92b] Krishnapuram R., Nasraoui O., Frigui H. "The fuzzy c spherical shells algorithm: A new approach," *IEEE Transactions on Neural Networks*, Vol. 3(5), pp. 663–671, 1992.

[Kris 93] Krishnapuram R., Keller J.M. "A possibilistic approach to clustering," *IEEE Transactions on Fuzzy Systems*, Vol. 1(2), pp. 98–110, May 1993.

[Kris 95a] Krishnapuram R., Frigui H., Nasraoui O. "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Part I," *IEEE Transactions on Fuzzy Systems*, Vol. 3(1), pp. 29–43, February 1995.

[Kris 95b] Krishnapuram R., Frigui H., Nasraoui O. "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Part II," *IEEE Transactions on Fuzzy Systems*, Vol. 3(1), pp. 44–60, February 1995.

[Kris 96] Krishnapuram R., Keller J.M. "The possibilistic c-means algorithm: Insights and recommendations," *IEEE Transactions on Fuzzy Systems*, Vol. 4(3), pp. 385–393, August 1996.

[Kris 99] Krishna K., Muthy M. "Genetic *k*-means algorithm," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 29(3), pp. 433–439, 1999.

[Kris 99a] Krishnapuram R., Kim J., "A note on the Gustafson-Kessel and adaptive fuzzy clustering algorithms," *IEEE Transactions on Fuzzy Systems*, Vol. 7(4), pp. 453–461, 1999.

[Kuma 04] Kumar A., Sabharwal Y., Sen S. "A simple linear time (1+)-approximation algorithm for *k*-means clustering in any dimension," *Foundations of Computer Science (FOCS)*, pp. 454–462, 2004.

[Lika 03] Likas A., Vlassis N., Verbeek J. "The global *K*-means clustering algorithm," *Pattern Recognition*, Vol. 36(2), pp. 451–461, 2003.

[Lin 96] Lin J.S., Cheng K.S., Mao C.W., "Segmentation of multispectral magnetic resonance image using penalized fuzzy competitive learning network," *Computers and Biomedical Research*, Vol. 29, pp. 314–326, 1996.

[Lind 80] Linde Y., Buzo A., Gray R.M. "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, Vol. 28, pp. 84–95, 1980.

[Lloy 82] Lloyd S.P. "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, Vol. 28(2), pp. 129–137 March 1982.

[Luen 84] Luenberger D.G. *Linear and Nonlinear Programming*, Addison Wesley, 1984.

[Lutt 89] Luttrell S.P. "Hierarchical vector quantization," *IEE Proceedings (London)*, Vol. 136 (Part I), pp. 405–413, 1989.

[MacQ 67] MacQueen J.B. "Some methods for classification and analysis of multivariate observations," *Proceedings of the Symposium on Mathematical Statistics and Probability, 5th Berkeley*, Vol. 1, pp. 281–297, AD 669871, University of California Press, 1967.

[Man 94] Man Y., Gath I. "Detection and separation of ring-shaped clusters using fuzzy clustering," *IEEE Transactions on PAMI*, Vol. 16(8), pp. 855–861, August 1994.

[Mena 00] Menard M., Demko C., Loonis P. "The fuzzy c+2 means: solving the ambiguity rejection in clustering," *Pattern Recognition*, Vol. 33, pp. 1219–1237, 2000.

[Ng 94] Ng R., Han J. "Efficient and effective clustering methods for spatial data mining." *Proceedings of the 20th Conference on VLDB*, pp. 144–155, Santiago, Chile, 1994.

[Ng 94a] Ng R., Han J. "Efficient and effective clustering methods for spatial data mining." *Technical Report 94–13*, University of British Columbia.

[Ng 00] Ng M. K. "A note on constrained k-means algorithms," *Pattern Recognition*, Vol. 33, pp. 515–519, 2000.

[Ozde 01] Özdemir D., Akarun L., "Fuzzy algorithms for combined quantization and dithering," *IEEE Transactions on Image Processing*, Vol. 10(6), pp. 923–931, 2001.

[Ozde 02] Özdemir D., Akarun L., "A fuzzy algorithm for color quantization and images," *Pattern Recognition*, Vol. 35, pp. 1785–1791, 2002.

[Pata 01] Patane G., Russo M. "The enhanced-LBG algorithm," *Neural Networks*, Vol. 14(9), pp. 1219–1237, 2001.

[Pata 02] Patane G., Russo M. "Fully automatic clustering system," *IEEE Transactions on Neural Networks*, Vol. 13(6), pp. 1285–1298, 2002.

[Pato 70] Paton K. "Conic sections in chromosome analysis," *Pattern Recognition*, Vol. 2(1), pp. 39–51, January 1970.

[Pedr 96] Pedrycz W., "Conditional fuzzy c-means," *Pattern Recognition Letters*, Vol. 17, pp. 625–632, 1996.

[Pena 99] Pena J., Lozano J., Larranaga P. "An empirical comparison of four initialization methods for the *k*-means algorithm," *Pattern Recognition Letters*, Vol. 20, pp. 1027–1040, 1999.

[Runk 99] Runkler T.A., Bezdek J.C. "Alternating cluster estimation: A new tool for clustering and function approximation," *IEEE Trans. on Fuzzy Systems*, Vol. 7, No. 4, pp. 377–393, August 1999.

[Sabi 87] Sabin M.J. "Convergence and consistency of fuzzy c-means/Isodata algorithms," *IEEE Transactions on PAMI*, Vol. 9(5), pp. 661–668, September 1987.

[Scho 98] Schölkopf B., Smola A.J., Müller "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, Vol. 10(5), pp. 1299–1319, 1998.

[Seli 84a] Selim S.Z., Ismail M.A. "K-means type algorithms: A generalized convergence theorem and characterization of local optimality, "*IEEE Transactions on PAMI*, Vol. 6(1), pp. 81–87, 1984.

[Seli 84b] Selim S.Z., Ismail M.A. "Soft clustering of multidimensional data: A semifuzzy approach," *Pattern Recognition*, Vol. 17(5), pp. 559–568, 1984.

[Seli 86] Selim, S.Z., Ismail, M.A. "On the local optimality of the fuzzy Isodata clustering algorithm," *IEEE Transactions on PAMI*, Vol. 8(2), pp. 284–288, March 1986.

[Shen 06] Shen H., Yang J., Wang S., Liu X., "Attribute weighted Mercer kernel-based fuzzy clustering algorithm for general non-spherical data sets," *Soft Computing*, Vol. 10(11), pp. 1061–1073, 2006.

[Siya 05] Siyal M.Y., Yu L., "An intelligent modified fuzzy c-means based algorithm for bias estimation and segmentation of brain MRI," *Pattern Recognition Letters*, Vol. 26(13), pp. 2052–2062, 2005.

[Spra 66] Spragins J. "Learning without a teacher," *IEEE Transactions on Information Theory*, Vol. IT-12, pp. 223–230, April 1966.

[Su 01] Su M., Chou C. "A modified version of the *K*-means algorithm with a distance based on cluster symmetry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23(6), pp. 674–680, 2001.

[Wags 01] Wagstaff K., Rogers S., Schroedl S. "Constrained *k*-means clustering with background knowledge," *Proceedings of the 8th International Conference on Machine Learning*, pp. 577–584, 2001.

[Wei 94] Wei W., Mendel J.M. "Optimality tests for the fuzzy c-means algorithm," *Pattern Recognition*, Vol. 27(11), pp. 1567–1573, 1994.

[Yama 80] Yamada Y., Tazaki S., Gray R.M. "Asymptotic performance of block quantizers with difference distortion measures," *IEEE Transactions on Information Theory*, Vol. 26(1), pp. 6–14, 1980.

[Yang 93] Yang M.S., "On a class of fuzzy classification maximum likelihood procedures," *Fuzzy Sets and Systems*, Vol. 57, pp. 365–375, 1993.

[Yu 03] Yu J., Yang M., "A study on a generalized FCM," in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, eds. Wang G. *et al*, pp. 390–393, Springer, 2003.

[Zade 78] Zadeh L.A. "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, Vol. 1, pp. 3–28, 1978.

[Zeyu 01] Zeyu L., Shiwei T., Jing X., Jun J., "Modified FCM clustering based on kernel mapping," *Proc. of Int. Society of Optical Engineering*, Vol. 4554 pp. 241–245, 2001.

[Zhan 03] Zhang D.Q., Chen S.C., "Clustering incomplete data using kernel-based fuzzy c-means algorithm," *Neural Processing Letters*, Vol. 18(3), pp. 155–162, 2003.

[Zhou 04] Zhou S., Gan J., "Mercer kernel fuzzy c-means algorithm and prototypes of clusters," *Proc. Cong. on Int. Data Engineering and Automated Learning*, pp. 613–618, 2004.

[Zhua 96] Zhuang X., Huang Y., Palaniappan K., Zhao Y. "Gaussian mixture density modelling, decomposition and applications," *IEEE Transactions on Image Processing*, Vol. 5, pp. 1293–1302, September 1996.