8

Clustering with Intelligent Techniques

Cluster analysis is a technique for grouping data and finding structures in data. The most common application of clustering methods is to partition a data set into clusters or classes, where similar data are assigned to the same cluster whereas dissimilar data should belong to different clusters. In real-world applications there is very often no clear boundary between clusters so that fuzzy clustering is often a good alternative to use. Membership degrees between zero and one are used in fuzzy clustering instead of crisp assignments of the data to clusters.

Pattern recognition techniques can be classified into two broad categories: *unsupervised* techniques and *supervised* techniques. An unsupervised technique does not use a given set of unclassified data points, whereas a supervised technique uses a data set with known classifications. These two types of techniques are complementary. For example, unsupervised clustering can be used to produce classification information needed by a supervised pattern recognition technique. In this chapter, we first give the basics of unsupervised clustering. The Fuzzy C-Means algorithm (FCM), which is the best known unsupervised fuzzy clustering algorithm is then described in detail. Supervised pattern recognition using fuzzy logic will also be mentioned. Finally, we will describe the use of neural networks for unsupervised clustering and hybrid approaches.

8.1 Unsupervised Clustering

Unsupervised clustering is motivated by the need of finding interesting patterns or groupings in a given data set. For example, in voting analysis one may want to collect data about a group of voters (e.g. through a survey or interviews) and analyze these data to find interesting groupings of voters. The result of such an analysis can be used to plan the strategies of a particular candidate in an election.

Patricia Melin and Oscar Castillo: *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing*, StudFuzz **172**, 169–184 (2005)
www.springerlink.com

In the area of pattern recognition and image processing, unsupervised clustering is often used to perform the task of "segmenting" the images (i.e., partitioning pixels of an image into regions that correspond to different objects or different faces of objects in the images). This is because image segmentation can be considered as a kind of data clustering problem where each datum is described by a set of image features of a pixel.

Conventional clustering algorithms find what is called a "hard partition" of a given data set based on certain criteria that evaluate the goodness of a partition. By a "hard partition" we mean that each datum belongs to exactly one cluster of the partition. We can define the concept of a "hard partition", more formally, as follows.

Definition 8.1. *Let X be a data set, and x_i be an element of X. A partition $P = \{C_1, C_2, \ldots, C_l\}$ of X is "hard" if and only if the two following conditions are satisfied:*

(1) *For all $x_i \in X$, there exists a $Cj \in P$ such that $x_i \in Cj$*
(2) *For all $x_i \in X, x_i \in Cj \rightarrow x_i \notin Ci$ where $i \neq j, Ci, Cj \in P$.*

The first condition in this definition states that the partition has to cover all data points in X, and the second condition states that all the clusters in the partition are mutually exclusive (in other words, there can not be intersection between clusters).

In many real-world clustering problems, however, some data points can partially belong to multiple clusters, rather than to a single cluster exclusively. For example, a pixel in a medical image, of the human brain of a patient, may correspond to a mixture of two different types of cells. On the other hand, in the voting analysis example, a particular voter maybe a "borderline case" between two groups of voters (e.g., between moderate conservatives and moderate liberals). These examples motivate the need for having "soft partitions" and "soft clustering algorithms".

A soft clustering algorithm finds a "soft partition" of a given data set based on certain criteria. In a soft partition, a datum can partially belong to multiple clusters. A formal definition of this concept is the following.

Definition 8.2. *Let X be a set of data, and x_i be an element of X. A partition $P = \{C_1, C_2, \ldots, C_l\}$ of X is a "soft partition" if and only if the following two conditions are satisfied:*

(1) *For all $x_i \in X$, for all $C_j \in P$, then $0 \leq \mu_{Cj}(x_i) \leq 1$*
(2) *For all $x_i \in X$, there exists $C_j \in P$ such that $\mu_{Cj}(x_i) > 0$.*

Where $\mu_{Cj}(x_i)$ denotes the degree of membership to which x_i belongs to cluster C_j.

A particular type of soft clustering is one that ensures that the membership degree of a point x in all clusters add up to one. More formally,

$$\Sigma_j \mu_{Cj}(x_i) = 1 \quad \text{For all } x_i \in X$$

A soft partition that satisfies this additional condition is called a called a constrained soft partition. The fuzzy c-means clustering algorithm produces a constrained soft partition, as we will see later in this chapter.

A constrained soft partition can also be generated by what it is called a probabilistic clustering algorithm. Even though both fuzzy c-means and probabilistic clustering produce a partition with similar properties, the clustering criteria underlying these algorithms are very different. We will concentrate in this chapter on fuzzy clustering, but this does not mean that probabilistic clustering is not a good method. The fuzzy c-means algorithm generalizes a hard clustering algorithm called the c-means algorithm. The hard c-means algorithm aims to identify compact well-separated clusters. Informally, a "compact" cluster has a "ball-like" shape. The center of the ball is called "the center" or "the prototype" of the cluster. A set of clusters are "well separated" when any two points in a cluster are closer than the shortest distance between two points in different clusters.

Assuming that a data set contains$c$compact, well-separated clusters, the goals of the hard c-means algorithm are the following:

(1) To find the centers of these clusters, and
(2) To determine the clusters (i.e., labels) of each point in the data set.

In fact, the second goal can easily be achieved once we accomplish the first goal, based on the assumption that clusters are compact and well separated. Given the cluster centers, a point in the data set belongs to the cluster whose center is the closest.

In order to achieve the first goal, we need to establish a criterion that can be used to search for these cluster centers. One of the criteria that are used is the sum of the distances between points in each cluster and their centers. This can stated formally as follows:

$$J(P,V) = \Sigma_j \Sigma_{xi} ||x_i - v_j||^2 \tag{8.1}$$

Where $j = 1$ to $j = c, x_i \in C_j$ and V is a vector of cluster centers to be identified. This criterion is useful because a set of true cluster centers will give a minimal value of J for a given data set. Based on these observations, the hard c-means algorithm tries to find the cluster centers that minimize the value of the objective function J. However, J is also a function of the partition P, which is determined by the cluster centers V. As a consequence, the hard c-means algorithm searches for the true cluster centers by iterating the following two steps:

(1) Calculating the current partition based on the current clusters,
(2) Modifying the current clusters using gradient descent method to minimize the objective function J.

The iterations terminate when the difference between cluster centers in two consecutive iterations is smaller than a specified threshold. This means that the clustering algorithm has converged to a local minimum of J.

8.2 Fuzzy C-Means Algorithm

The Fuzzy C-Means algorithm (FCM) generalizes the hard c-means algorithm to allow points to partially belong to multiple clusters. Therefore, it produces a soft partition for a given data set. In fact, it generates a constrained soft partition. To achieve this, the objective function of hard c-means has to be extended in two ways:

(1) the fuzzy membership degrees in clusters were incorporated into formula, and
(2) an additional parameter m was introduced as a weight exponent in the fuzzy membership.

The extended objective function, denoted J_m, is defined as follows:

$$J_m(P,V) = \Sigma_j \Sigma_{xi} \mu_{Cj}(x_k)^m ||x_i - v_j||^2 \tag{8.2}$$

where P is a fuzzy partition of the data set X formed by C_1, $C_2, \ldots, C_k$. The parameter m is a weight that determines the degree to which partial members of a cluster affect the clustering result.

Like hard c-means, the fuzzy c-means algorithm tries to find a good partition by searching prototypes vi that minimize the objective function J_m. Unlike hard c-means, however, the fuzzy c-means algorithm also needs to search for membership functions μ_{Cj} that minimize J_m. To accomplish these two objectives, a necessary condition for the local minimum of J_m has to be obtained from J_m. Basically, the partial derivatives of J_m have to be zero and then solved simultaneously. This condition, which is formally stated below, serves as the foundation for the fuzzy c-means algorithm.

Theorem 8.1. ***Fuzzy C-Means Theorem***
A constrained fuzzy partition $\{C_1, C_2, \ldots, C_k\}$ *is a local minimum of the objective function* J_m *only if the following conditions are satisfied:*

$$\mu_{Ci}(x) = 1/\left[\Sigma_j (||x - v_j||^2 / ||x - v_j||^2)^{(1/m-1)}\right] \quad 1 \le i \le k, x \in X \tag{8.3}$$

$$v_i = [\Sigma_x (\mu_{Ci}(x))^m (x)] / [\Sigma_x (\mu_{Ci}(x))^m] \quad 1 \le i \le k \tag{8.4}$$

Based on the equations of this theorem, FCM updates the prototypes and the membership functions iteratively using (8.3) and (8.4) until a termination criterion is satisfied. We can specify the FCM algorithm as follows:

Step 1: Initialize prototype $V = \{v_1, v_2, \ldots, v_c\}$.
Step 2: Make $V^{\text{old}} \leftarrow V$.
Step 3: Calculate membership functions with (8.3).
Step 4: Update the prototype v_i in V using (8.4).
Step 5: Calculate $E = \Sigma_i ||v^{\text{old}} - v_i||$
Step 6: If $E > \varepsilon$ then go to Step 2.
Step 7: If $E \le \varepsilon$ then output the final result.

In the previous algorithm, c is the number of clusters to form, m is the parameter of the objective function, and ε is a threshold value (usually very small) for the convergence criteria.

We have to say that the FCM algorithm is guaranteed to converge for $m > 1$. This important convergence theorem was established by Bezdek in 1990. FCM finds a local minimum of the objective function J_m. This is because the FCM theorem is derived from the condition that the gradient of J_m should be zero for a FCM solution, which is satisfied by all the local minimums. Finally, we have to say that the result of applying FCM to a given data set depends not only on the choice of the parameters m and c, but also on the choice of the initial prototypes.

We now consider a simple example to illustrate the concepts and ideas of the FCM algorithm. We consider a set of randomly generated points, and the application of the FCM algorithm to find four clusters. In Fig. 8.1 we can appreciate a simulation of the FCM algorithm in which the initial cluster centers (at the origin) move toward the "right" positions, which shows that the algorithm finds the optimal cluster centers.

After the clustering process stops, the final cluster centers are found. These cluster centers are the ones that minimize the objective function J_m. We show in Fig. 8.2 the final result of the FCM algorithm in which the final cluster center positions are indicated as black circles.

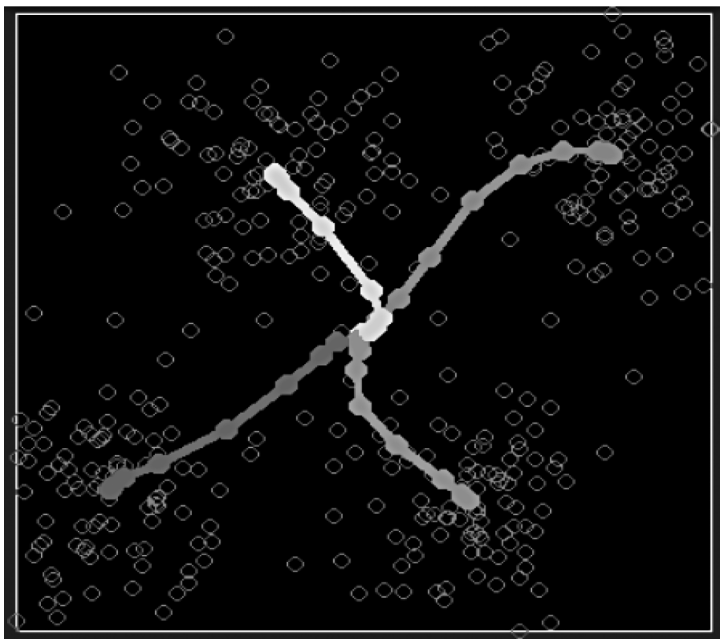

Fig. 8.1. Evolution of the FCM algorithm

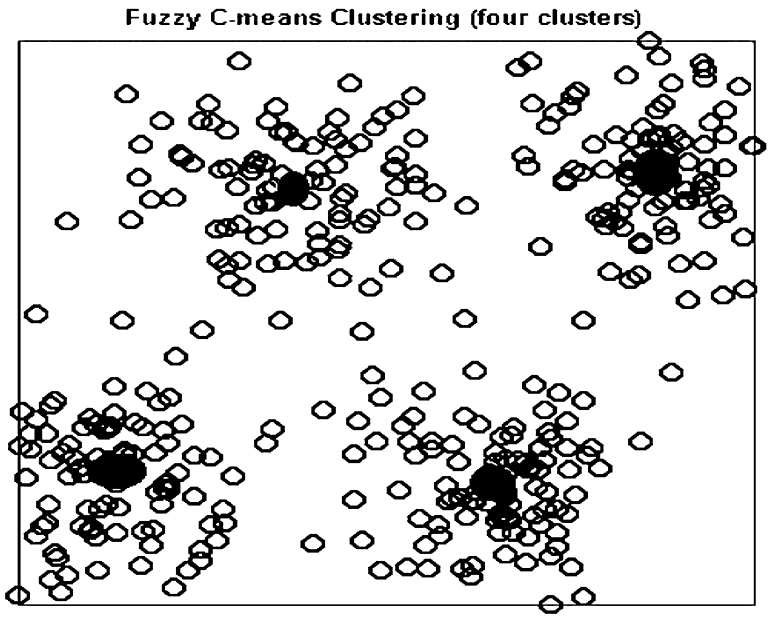

Fig. 8.2. Final cluster centers after the application of FCM

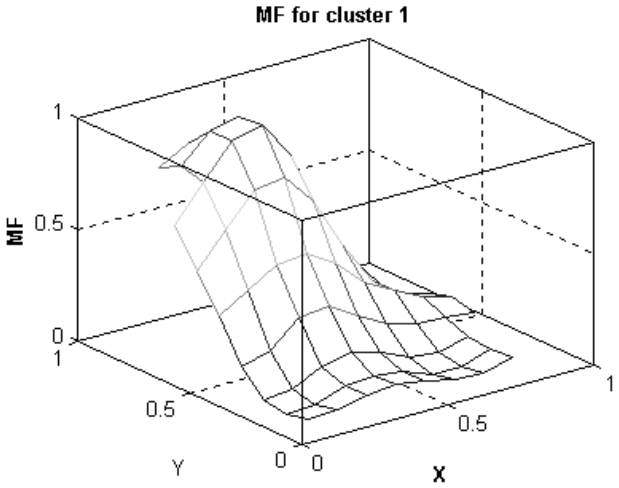

Fig. 8.3. Membership function for cluster one

We can also show the membership functions for each of the clusters that were generated by the FCM algorithm. We show in Fig. 8.3 the membership function for cluster one.

We now show in the following figures the membership functions for the other clusters in this example. In Fig. 8.4 we can appreciate the membership

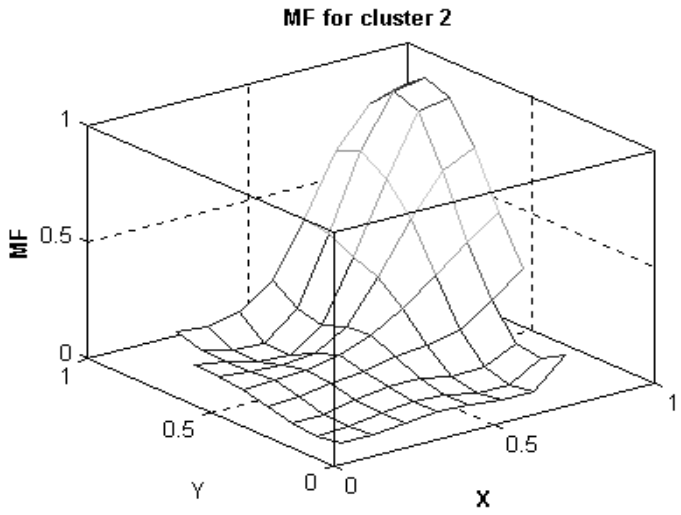

Fig. 8.4. Membership function of cluster two as a result of the FCM

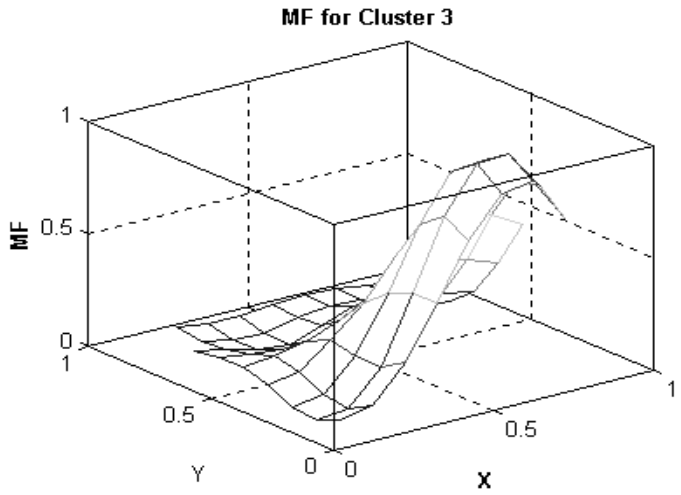

Fig. 8.5. Membership function of cluster three as a result of the FCM

function formed by FCM for cluster two. On the other hand, in Fig. 8.5, we can appreciate the membership function of cluster three. Finally, in Fig. 8.6, the membership function of cluster four is shown.

8.2.1 Clustering Validity

One of the main problems in clustering is how to evaluate the clustering result of a particular algorithm. The problem is called "clustering validity"

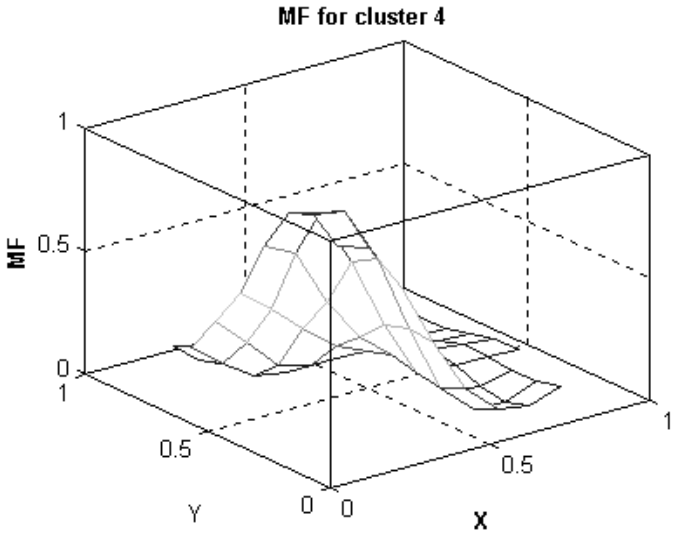

Fig. 8.6. Membership function of cluster four as a result of the FCM

in the literature. More precisely, the problem of clustering validity is to find an objective criterion for determining how good a partition generated by a clustering algorithm is. This type of criterion is important because it enables us to achieve three objectives:

(1) To compare the output of alternative clustering algorithms for a particular data set.
(2) To determine the best number of clusters for a particular data set (for example, the choice of parameter c for the FCM).
(3) To determine if a given data set contains any structure (i.e., whether there exists a natural grouping of the data set).

It is important to point out that both hard clustering and soft clustering need to consider the issue of clustering validity, even though their methods may differ. We will concentrate in this chapter on describing validity measures of a fuzzy partition generated by FCM or, more generally, a constrained soft partition of a data set.

Validity measures of a constrained soft partition fall into three categories: (1) membership-based validity measures, (2) geometry-based validity measures, and (3) performance-based validity measures. The membership-based validity measures calculate certain properties of the membership functions in a constrained soft partition. The geometry-based validity measures consider geometrical properties of a cluster (like area or volume) as well as geometrical relationships between clusters (for example, the separation) in a soft partition. The performance-based validity measures evaluate a soft partition based on

its performance for a predefined goal (for example, minimum error rate for a classification task).

Jim Bezdek introduced the first validity measure of a soft partition, which is called "partition coefficient" (Yen & Langari, 1999). The partition coefficient has the goal of measuring the degree of fuzziness of clusters. The reasoning behind this validity measure is that the fuzzier the clusters are, the worse the partition is. The formula for this validity measure, denoted v_{pc}, is

$$v_{pc} = (1/n)\Sigma_i\Sigma_j\mu_{Ci}(x_j), \quad i = 1, \ldots, c, \quad j = 1, \ldots, n\ . \tag{8.5}$$

Subsequently, Bezdek also introduced another membership-based validity measure called "partition entropy", which is denoted as v_{pe} and defined formally as follows

$$v_{pe} = (-1/n)\Sigma_i\Sigma_j[\mu_{Ci}(x_j)\log_a(\mu_{Ci}(x_j))],\ i = 1, \ldots, c,\ j = 1, \ldots, n\ . \tag{8.6}$$

where $a \in (1, \infty)$ is the logarithmic base. The entropy measure increases as the fuzziness of the partition decreases. Therefore, a clustering achieved with lower partition entropy is preferred. The two membership-based validity measures are related in the following ways for a constrained soft partition:

(1) $v_{pc} = 1 \Leftrightarrow v_{pe} = 0 \Leftrightarrow$ the partition is hard.
(2) $v_{pc} = 1/c \Leftrightarrow v_{pe} = \log_a(c) \Leftrightarrow \mu_{Ci}(x_j) = 1/c$ for all i, j.

The two cases above correspond to the extreme situations. The first case is the least fuzzy and therefore is the one preferred (at least ideally). The second case is the fuzziest and therefore the least preferred by the validity measures.

X. Xie and G. Beni introduced a validity measure that considers both the compactness of clusters as well as the separation between clusters. The basic idea of this validity measure is that the more compact the clusters are and the further the separation between clusters, the more desirable the partition. To achieve this measure, the Xie-Beni validity index (denoted as v_{xB}) is defined formally as follows:

$$v_{xB} = [\Sigma_i\sigma_i/n]\left[1/d^2_{\min}\right] \tag{8.7}$$

where σ_i is the variation of cluster C_i defined as follows:

$$\sigma_i = \Sigma_j[\mu_{Ci}(x_j)]||x_j - v_i||^2 \tag{8.8}$$

n is the cardinality of the data set and $d_{\min}$ is the shortest distance between cluster centers defined as

$$d_{\min} = \min||v_j - v_i||, \quad i \neq j\ , \tag{8.9}$$

The first term in (8.7) is a measure of non-compactness, and the second term is a measure of non-separation. Hence, the product of the two terms reflects the degree to which the clusters in the partition are not compact and not well separated. Obviously, the lower the cluster index, the better the soft partition is.

8.2.2 Extensions to Fuzzy C-Means

A major extension to FCM is to generalize the distance measure between data x and prototype vj from the usual Euclidean distance:

$$\text{dist}(x_i, vj) = ||x_i - v_j||^2 \tag{8.10}$$

to the generalized distance

$$\text{dist}(x_i, vj) = (x_i - v_j)^T A_j (x_i - v_j) \tag{8.11}$$

where A_j is a symmetric $d \times d$ matrix (d is the dimensionality of x_i and v_j). This enables an extended FCM to adapt to different hyper-ellipsoidal shapes of different clusters by adjusting the matrix A_j.

D. Gustafsun and W. Kessel were the first to propose such an extension of the matrix Aj, they developed a modified FCM that dynamically adjusts $(A_1, A_2, \ldots, A_c)$ such that these matrices adapt to the different hyper-ellipsoidal shape of each cluster.

8.3 Mountain Clustering Method

The "mountain clustering method", as proposed by Yager & Filev (1994), is a relatively simple and effective approach to approximate the estimation of cluster centers on the basis of a density measure called the "mountain function". This method can be used to obtain initial cluster centers that are required by more sophisticated cluster algorithms, such as the FCM algorithm introduced in the previous section. It can also be used as a quick stand-alone method for approximate clustering. The method is based on what a human does in visually forming clusters of a data set.

The first step involves forming a grid on the data space, where the intersections of the grid lines constitute the candidates for cluster centers, denoted as a set V. A finer gridding increases the number of potential clustering centers, but it also increases the computation required. The grid is generally evenly spaced, but it not a requirement. We can have unevenly spaced grids to reflect "a priori" knowledge of data distribution. Moreover, if the data set itself (instead of the grid points) is used as the candidates for cluster centers, then we have a variant called subtractive clustering.

The second step entails building a mountain function representing a data density measure. The height of the mountain function at a point $\mathbf{v} \in V$ is equal to

$$m(\mathbf{v}) = \Sigma_i \exp[(-||\mathbf{v} - x_i||^2)/2\sigma^2] \tag{8.12}$$

where x_i is the ith data point and σ is a constant, which is application-specific. The preceding equation, implies that each data point x_i contributes to the height of the mountain function at $\mathbf{v}$, and the contribution is inversely

proportional to the distance between x_i and $\mathbf{v}$. The mountain function can be viewed as a measure of "data density", since it tends to be higher if more data points are located nearby, and lower if fewer data points are around. The constant σ determines the height as well as the smoothness of the resultant mountain function. The clustering results are normally insensitive to the value of σ, as long as the data set is of sufficient size and is well clustered.

The third step involves selecting the cluster centers by sequentially destroying the mountain function. We first find the point in the candidate centers V that has the greatest value for the mountain function; this becomes the first cluster center $\mathbf{c}_1$. Obtaining the next cluster center requires eliminating the effect of the recently found center, which is typically surrounded by a number of grid points that also have high-density scores. This is realized by revising the mountain function; a new mountain function is formed by subtracting a scaled Gaussian function centered at $\mathbf{c}_1$:

$$m_{\text{new}}(\mathbf{v}) = m(\mathbf{v}) - m(\mathbf{c}_1) \exp[(-||\mathbf{v} - \mathbf{c}_1||^2)/2\beta^2] \tag{8.13}$$

The subtracted amount is inversely proportional to the distance between $\mathbf{v}$ and the recently found center $\mathbf{c}_1$, as well as being proportional to the height $m(\mathbf{c}_1)$ at the center. Note that after subtraction, the new mountain function $m_{\text{new}}(\mathbf{v})$ reduces to zero at $\mathbf{v} = \mathbf{c}_1$.

After subtraction, the second cluster center is again selected as the point in V that has the largest value for the new mountain function. This process of revising the mountain function and finding the next cluster center continues until a sufficient number of cluster centers is obtained.

We now give a simple example to illustrate the ideas mentioned above. Figure 8.7(a) shows a set of two-dimensional data, in which three clusters can be very easily recognized. However, for higher dimensional data sets is very difficult to visualize the clusters, and for this reason, methods like this are very useful. In this example, the mountain method is used to find the three clusters. To show the effect of changing σ, Figs. 8.7(b) through 8.7(d) are the surface plots of the mountain functions with σ equal to 0.02, 0.1, and 0.2, respectively. Obviously, σ affects the mountain function's height as well as its smoothness; therefore, the value of σ should be selected carefully considering both the data size and input dimension.

Once the σ value is determined (0.1 in this example) and the mountain function is constructed, we begin to select clusters and revise the mountain functions sequentially. This is shown in Figs. 8.8(a), (b), (c), and (d), with β equal to 0.1 in (8.13).

8.4 Clustering of Real-World Data with Fuzzy Logic

Now we will consider a more realistic application of clustering techniques. We consider in this example clustering of World Bank data for 98 countries

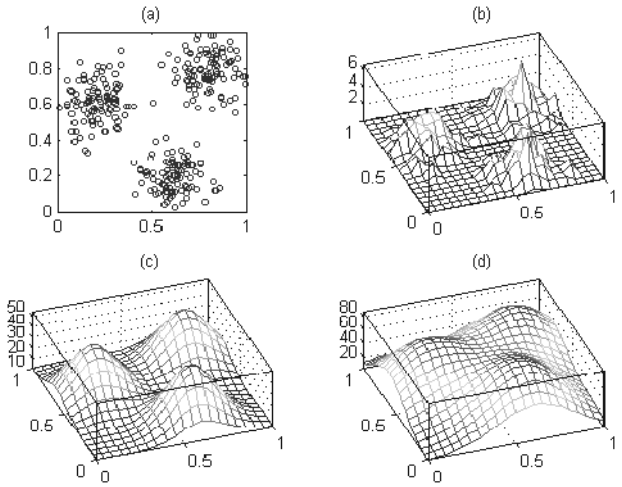

Fig. 8.7. Mountain construction: (**a**) original data set, (**b**) mountain function with $\sigma = 0.02$, (**c**) mountain function with $\sigma = 0.1$, and (**d**) mountain function with $\sigma = 0.2$

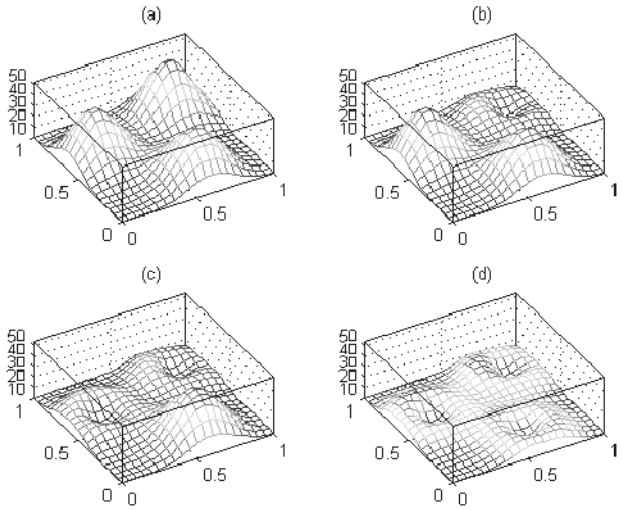

Fig. 8.8. Mountain destruction with $\beta = 0.1$: (**a**) original mountain function with $\sigma = 0.1$; (**b**) mountain function after the first reduction; (**c**) mountain function after the second reduction; (**d**) mountain function after the third reduction

with respect to two variables. The economic variables are the Electrical Power Consumption (x), and High Technology Exports (y) for 2001. The application of intelligent clustering to this data set has the goal of grouping similar countries with respect to these economic variables. The cluster will represent groups of countries with similarities in these variables. Of course, we expect that countries with low electrical power consumption will also have low high technology exports. However, other groups are more difficult to identify.

The results of the clustering will be useful to people working in Economics because a great part of their research consists in analyzing economic data and constructing models based on these studies. Of course, the clustering of countries can also be done using more variables or economic indicators that are also available in the web page of the World Bank.

We used the FCM algorithm with different number of clusters until the optimal number was found. The best results were obtained for a number $c = 6$ clusters. We show in Fig. 8.9 the final result of the FCM for the data set of the 98 countries. We also show in Fig. 8.10 the membership function obtained for cluster one.

The other membership functions for the other clusters are similar. The final clusters give the groups of countries that were grouped together by the FCM algorithm. The results were checked by the Economists and were find to be consistent with what they believe is true for these countries.

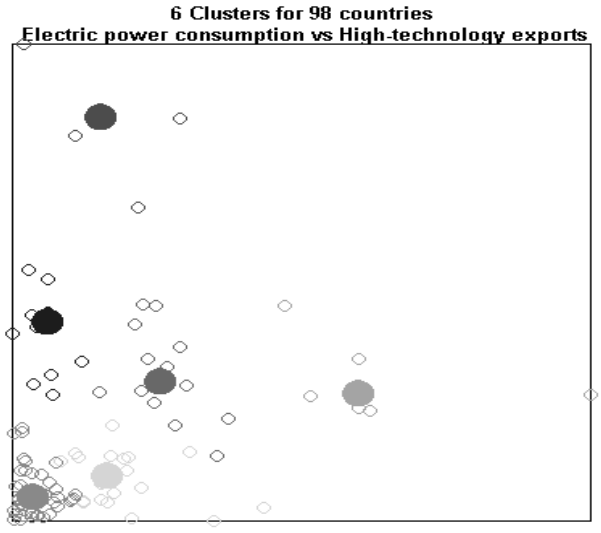

Fig. 8.9. Clustering of the data set for 98 countries with FCM

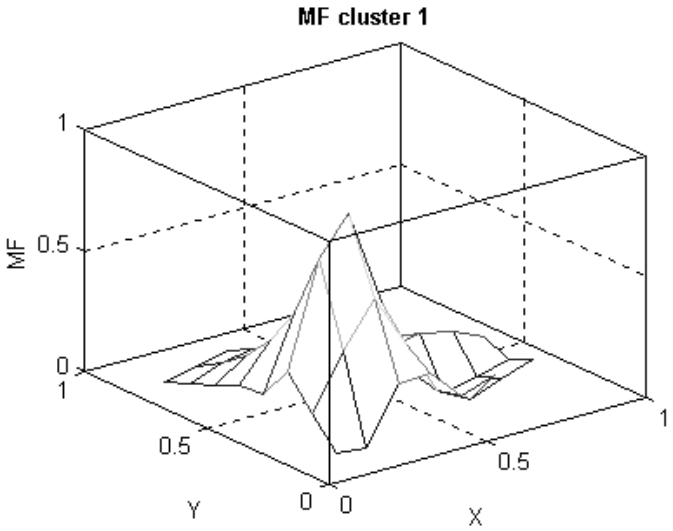

Fig. 8.10. Membership function for cluster number one of World Bank data set

8.5 Clustering of Real-World Data with Unsupervised Neural Networks

We will now consider the application of unsupervised neural networks (described in Chap. 5) for achieving intelligent clustering. This type of neural network can also be used to cluster a data set, but the main difference with respect to fuzzy clustering is that neural networks do not use membership functions. We can say that unsupervised neural network perform hard clustering on the data sets. However, neural network may have other advantages because their basis for performing clustering is based on models of learning.

We will consider again the World Bank data set of 98 countries with the two variables: electric power consumption, and high technology exports. We will apply a competitive learning neural network to this data set to form the groups of countries. These groups will form according to the similarities between the values of the countries in the data set. We show in Fig. 8.11 the results of applying a competitive neural network with 6000 epochs and six clusters.

In this case, we can appreciate that clustering has not worked well because there are points that were not considered. However, if we change the number of epochs of learning in the same neural network, we obtain the results shown in Fig. 8.12. These results are better because one of the clusters takes into account the upper points in the figure. Also, if we compare Fig. 8.12 with Fig. 8.9, we can appreciate an agreement with the results of fuzzy clustering.

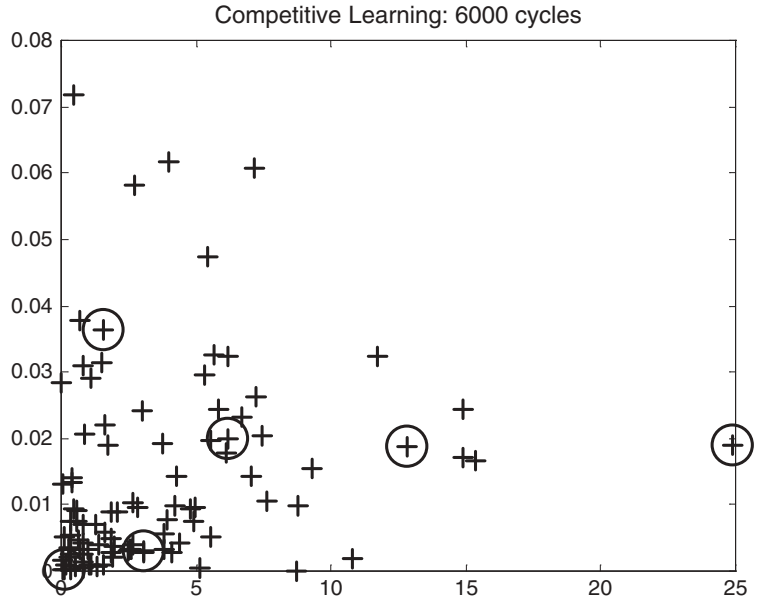

Fig. 8.11. Application of a competitive neural network for the World Bank data with 6000 epochs and a learning rate of 0.01

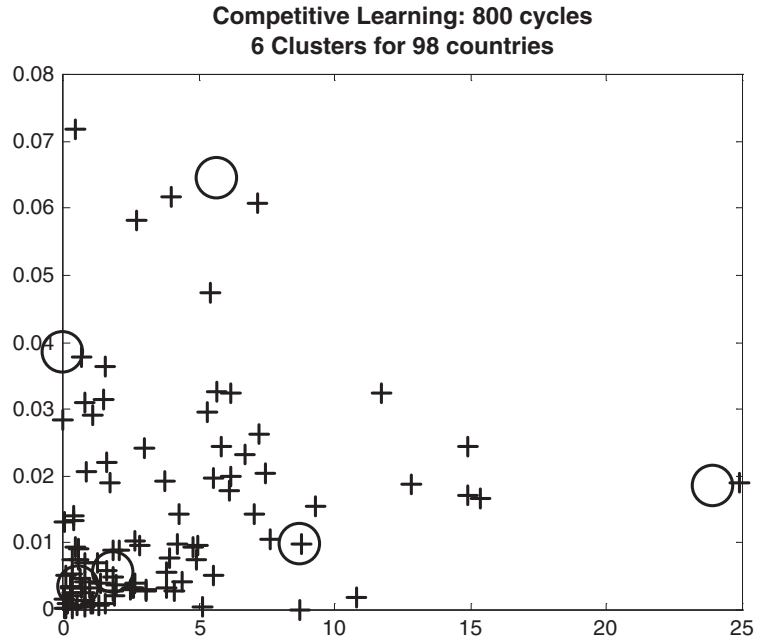

Fig. 8.12. Application of a competitive neural network for the World Bank data with 800 epochs and a learning rate of 1.3

Of course, other types of unsupervised neural networks can also be used for the same data set. In all the cases, it will require some experimental work to find the right parameters of the neural network.

8.6 Summary

We have presented in this chapter the basic concepts and theory of unsupervised clustering. We have described in some detail the hard c-means algorithm, the fuzzy c-means algorithm, the mountain clustering method, and clustering validity. Unsupervised clustering is useful for analyzing data without having desired outputs; the clustering algorithm will evolve to capture density characteristics of a data set. We have shown some examples of how to use this type of unsupervised clustering algorithms. We will describe in future chapters some applications of unsupervised clustering algorithms to real world problems related to pattern recognition.