1 Clustering and Fuzzy Clustering

This chapter provides a comprehensive, focused introduction to clustering, viewed as a fundamental means of exploratory data analysis, unsupervised learning, data granulation, and information compression. We discuss the underlying principles, elaborate on the basic taxonomy of numerous clustering algorithms (including such essential classes as hierarchical, objective function-based algorithms), and review the main interpretation mechanisms associated with various clustering algorithms.

1.1. INTRODUCTION

Making sense of data is an ongoing task of researchers and professionals in almost every practical endeavor. The age of information technology, characterized by a vast array of data, has enormously amplified this quest and made it even more challenging. Data collection anytime and everywhere has become the reality of our lives. Understanding the data, revealing underlying phenomena, and visualizing major tendencies are major undertakings pursued in intelligent data analysis (IDA), data mining (DM), and system modeling.

Clustering is a general methodology and a remarkably rich conceptual and algorithmic framework for data analysis and interpretation (Anderberg, 1973; Bezdek, 1981; Bezdek et al., 1999; Devijver and Kittler, 1987; Dubes, 1987; Duda et al., 2001; Fukunaga, 1990; Hoppner et al., 1999; Jain et al., 1999, 2000; Kaufmann and Rousseeuw, 1990; Babu and Murthy, 1994; Dave, 1990; Dave and Bhaswan, 1992; Kersten, 1999; Klawonn and Keller, 1998; Mali and Mitra, 2002; Webb, 2002). In this chapter, we introduce basic notions, explain the functional components essential to the formulation of clustering problems, and discuss the main classes of clustering algorithms. These algorithms are accompanied by the formalisms of granular computing, including sets, fuzzy sets, shadowed sets, and rough sets.

1.2. BASIC NOTIONS AND NOTATION

To establish a formal setting in which clustering can be carried out, we start with basic notions such as data types, distance, and similarity/resemblance.

Knowledge-Based Clustering, by Witold Pedrycz
ISBN 0-471-46966-1 Copyright © 2005 John Wiley & Sons, Inc.

1.2.1. Types of Data

The world surrounding us generates various types of data in abundance. The richness of data formats is impressive. The formal representation and organization of patterns reflect the way in which we intend to process the data. The most general taxonomy being in common use distinguishes among numeric (continuous), ordinal, and nominal variables. A numeric variable can assume any value in $\mathbf{R}$. An ordinal variable assumes a small number of discrete states, and these states can be compared. For instance, there are four states, denoted a_1, a_2, a_3, and a_4, and we can say that a_1 and a_2 are closer (in some sense of similarity that we define in the next section) than a_1 and a_3. A nominal variable assumes a small number of states, but nothing can be said about their closeness. Regardless of this distinction, nominal and ordinal variables are represented as discrete variables. For computing purposes, we usually have several coding schemes, such as binary coding or binary coding with various options.

The variables can be organized into internal structures that reflect the specificity of the problem. If each pattern is described by a number of features, intuitively we arrange them into vectors—say, $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$. Depending upon the character of the variables involved, the entries can be real or binary. Obviously, this can give rise to a variety of vectors, including both types of entries. Vectors and matrices are "flat" structures in the sense that all variables are at the same level as individual entries of the feature vector, and they have no structure. Hierarchical structures like trees are used to visualize the relationship between objects (patterns) we are interested in when dealing with clustering or classification.

1.2.2. Distance and Similarity

The concept of dissimilarity (or distance) or dual similarity is the essential component of any form of clustering that helps us navigate through the data space and form clusters. By computing dissimilarity, we can sense and articulate how close together two patterns are and, based on this closeness, allocate them to the same cluster. Formally, the dissimilarity $d(\mathbf{x}, \mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$ is considered to be a two-argument function satisfying the following conditions:

$$\begin{aligned} &d(\mathbf{x},\mathbf{y}) \geq 0 \quad \text{for every } \mathbf{x} \text{ and } \mathbf{y} \\ &d(\mathbf{x},\mathbf{x}) = 0 \quad \text{for every } \mathbf{x} \\ &d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x}) \end{aligned} \tag{1.1}$$

This list of requirements is intuitively appealing. We require a nonnegative character of the dissimilarity. The symmetry is also an obvious requirement. The dissimilarity attains a global minimum when dealing with two identical patterns, that is $d(\mathbf{x},\mathbf{x}) = 0$.

Distance, (metric) is a more restrictive concept, as we require the triangular inequality to be satisfied; that is, for any pattern $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ we have

$$d(\mathbf{x},\mathbf{y}) + d(\mathbf{y},\mathbf{z}) \geq d(\mathbf{x},\mathbf{z}) \tag{1.2}$$

TABLE 1.1. Selected Distance Functions Between Patterns x and y

Distance Function	Formula and Comments
Euclidean distance	$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
Hamming (city block) distance	$d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} \lvert x_i - y_i \rvert$
Tchebyschev distance	$d(\mathbf{x},\mathbf{y}) = \max_{i=1,2,\ldots,n} \lvert x_i - y_i \rvert$
Minkowski distance	$d(\mathbf{x},\mathbf{y}) = \sqrt[p]{\sum_{i=1}^{n}(x_i - y_i)^p}, \; p > 0$
Canberra distance	$d(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} \frac{\lvert x_i - y_i \rvert}{x_i + y_i}$, x_i and y_i are positive
Angular separation	$d(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{\left[\sum_{i=1}^{n} \mathrm{x}_\mathrm{i}^2 \sum_{i=1}^{n} y_i^2\right]^{1/2}}$ Note: this is a similarity measure that expresses the angle between the unit vectors in the direction of **x** and **y**

In the case of continuous features (variables), we have a long list of distance functions (see (Table 1.1)). Each of these functions implies a different view of the data because of their geometry. The geometry is easily illustrated when we consider only two features ($\mathbf{x} = [x_1 x_2]^T$) and compute the distance of **x** from the origin. The contours of the constant distance (Figure 1.1) show what type of geometric construct becomes a focus of the search for structure. Here we become aware that the Euclidean distance favors circular shapes of data clusters. With the distance functions come some taxonomy; the Minkowski distance comprises an infinite family of distances, including well-known and commonly used ones such as the Hamming, Tchebyschev, and Euclidean distances.

The same effect shown in Figure 1.1*d* can be achieved when the value of the power in the Minkowski distance is changed; see Figure 1.2.

One commonly used generalization is the Mahalanobis distance

$$d(\mathbf{x},\mathbf{y}) = \mathbf{x}^T A^{-1}\mathbf{y} \tag{1.3}$$

where A is a positive definite matrix. By choosing this matrix, we can control the geometry of potential clusters by rotating the ellipsoid (off diagonal entries of

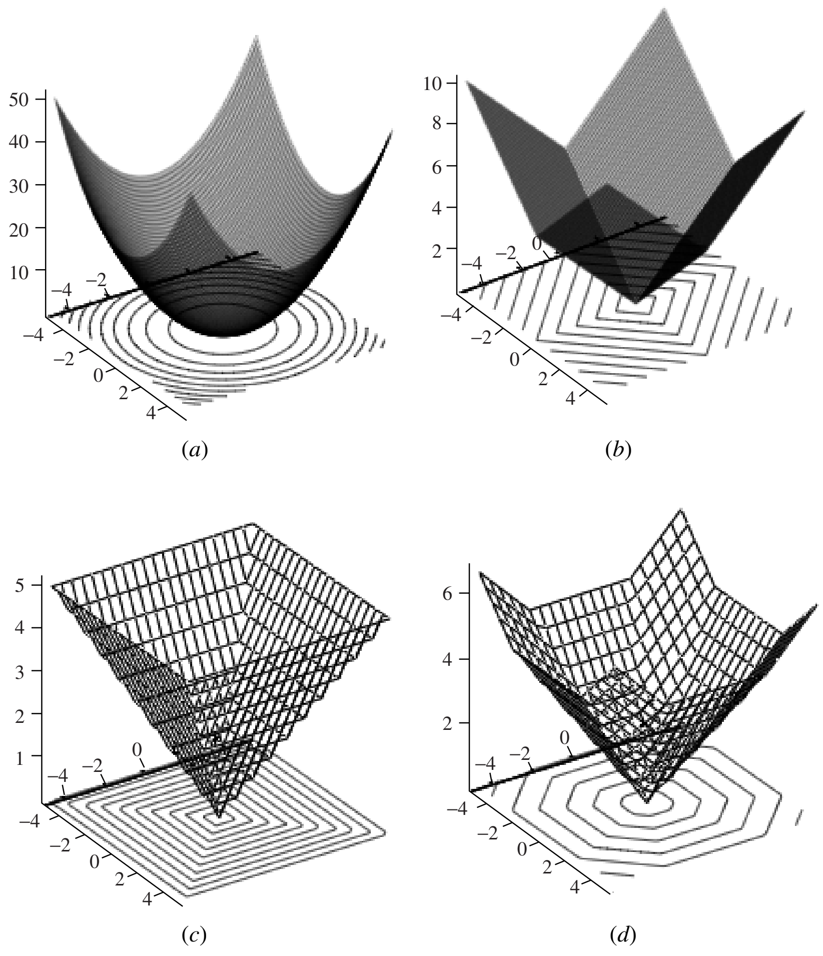

Figure 1.1. Examples of distance functions—three-dimensional and contour plots: (*a*) Euclidean, (*b*) Hamming (city block), (*c*) Tchebyschev, (*d*) "combined" type of distance max (2/3 Hamming, Tchebyschev).

A) and changing the length of its axes (the elements lying on the main diagonal of the matrix).

With binary variables, we traditionally focus on the notion of similarity rather than distance (or dissimilarity). Consider two binary vectors **x** and **y** that consist of two strings $[x_k]$, $[y_k]$ of binary data; compare them coordinatewise and do the simple counting of occurrences:

number of occurrences when x_k and y_k are both equal to 1
number of occurrences when $x_k = 0$ and $y_k = 1$
number of occurrences when $x_k = 1$ and $y_k = 0$
number of occurrences when x_k and y_k are both equal to 0

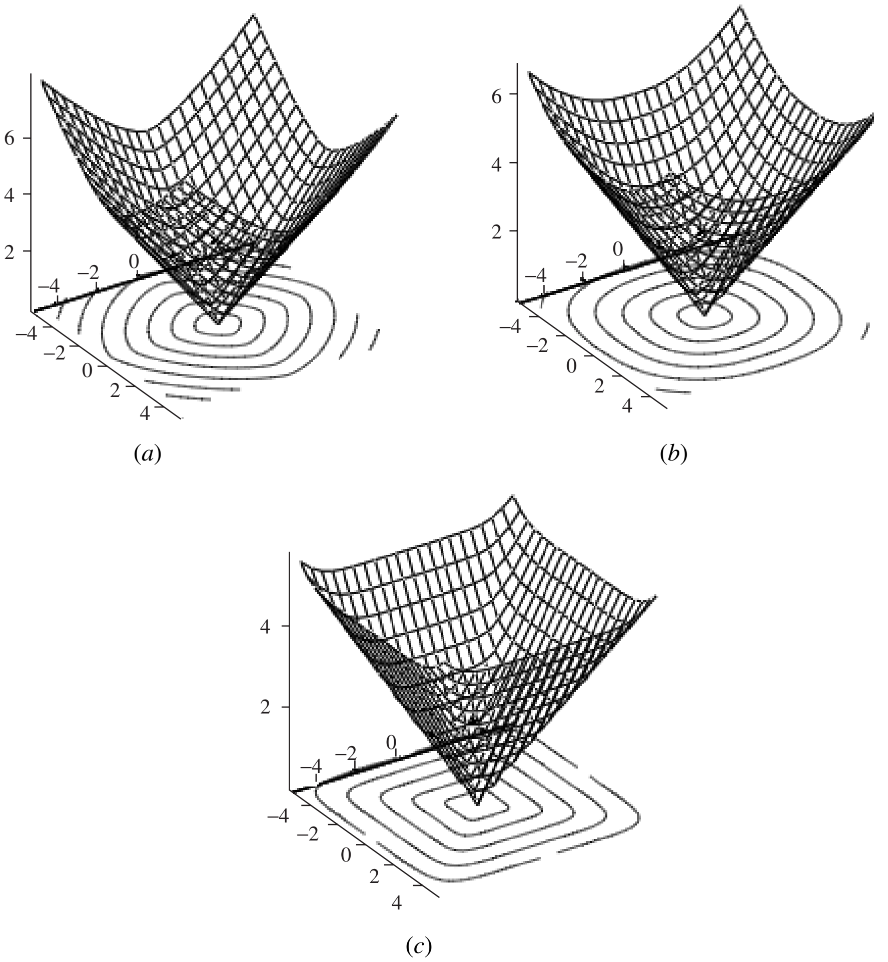

Figure 1.2. Examples of the Minkowski distance function for selected values of the power: (*a*) 1.5 (*b*) 2.5, and (*c*) 7.0.

These four numbers can be organized in a 2 by 2 co-occurrence matrix (contingency table) that visualizes how “close” these two strings are to each other.

	1	0
1	a	b
0	c	d

Evidently the zero nondiagonal entries of this matrix point at the ideal matching (the highest similarity). Based on these four entries, there are several commonly encountered measure of similarity of binary vectors **x** and **y**. The simplest matching coefficient computes as the following ratio:

$$\frac{a+d}{a+b+c+d} \tag{1.4}$$

The Russell and Rao measure of similarity consists of the quotient

$$\frac{a}{a+b+c+d} \tag{1.5}$$

The Jacard index involves the case when both inputs assume values equal to 1:

$$\frac{a}{a+b+c} \tag{1.6}$$

The Czekanowski index is practically the same as the Jacard index, but by adding the weight factor of 2, it emphasizes the coincidence of situations where entries of **x** and **y** both assume values equal to 1:

$$\frac{2a}{2a+b+c} \tag{1.7}$$

1.3. MAIN CATEGORIES OF CLUSTERING ALGORITHMS

Clustering techniques are rich and diversified. They have been continuously developing for over a half century following a number of trends, depending upon the emerging optimization techniques, main methodology (system modeling, DM, signal processing), and application areas. At the very high end of the overall taxonomy we envision two main categories of clustering, known as hierarchical and objective function-based clustering.

1.3.1. Hierarchical Clustering

The clustering techniques in this category produce a graphic representation of data (Duda et al., 2001). The construction of graphs (as these methods reveal the structure by considering each individual pattern) is done in two ways: bottom-up and top-down. The other names used reflect the way a structure is revealed. In the bottom-up mode known as an agglomerative approach, we treat each pattern as a single-element cluster and then successively merge the closest clusters. At each pass of the algorithm, we merge the two closest clusters. The process is repeated until we get to a single data set or reach a certain predefined threshold value. The top-down approach, known as a divisive approach, works in the opposite direction: we start with the entire set treated as a single cluster and keep splitting it into smaller clusters. Considering the nature of the process, these methods are often computationally inefficient, with the possible exception of patterns with binary variables.

The results of hierarchical clustering are usually represented in the form of dendrograms (Figure 1.3). Dendrograms are visually appealing graphical constructs: they show how difficult it is to merge two clusters. The distance scale shown at the right-hand side of the graph helps us quantify the distance between the clusters. This implies a simple stopping criterion: given a certain threshold value of the distance, we stop merging the clusters once the distance between them exceeds this threshold, meaning that merging two distinct structures does not seem to be feasible.

An important issue is how to measure the distance between two clusters. Note that we have discussed how to express the distance between two patterns. Here, as each cluster may contain many patterns, computation of the distance is neither obvious nor unique. Consider two clusters, A and B, illustrated in Figure 1.4. Let us describe the distance by $d(A, B)$ and denote the number of patterns in A and B by n_1 and n_2, respectively. Intuitively, we can easily envision three typical ways of computing the distance between the two clusters.

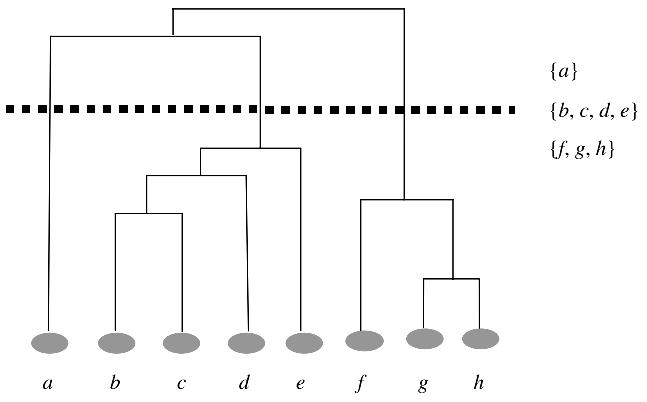

Figure 1.3. A dendrogram as a visualization of the structure of patterns; also shown are the distance values guiding the process of successive merging of the clusters.

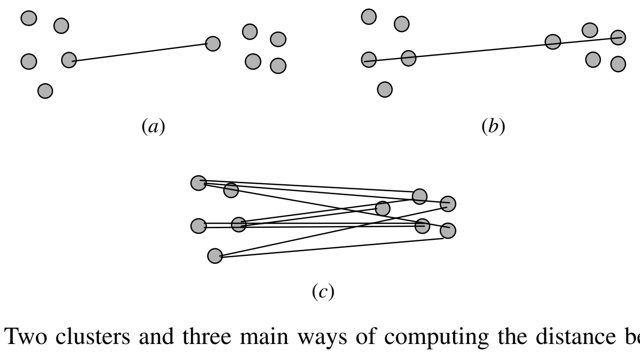

Figure 1.4. Two clusters and three main ways of computing the distance between them: (a) single link, (b) complete link, and (c) group average link.

Single-Link Method. The distance $d(A,B)$ is based on the minimal distance between the patterns belonging to A and B. It is computed in the form

$$d(A, B) = \min_{\mathbf{x}\in A, \mathbf{y}\in B} d(\mathbf{x}, \mathbf{y}) \tag{1.8}$$

In essence, the distance supports a sort of radically "optimistic" mode of expressing vicinity between clusters where we get involved the closest patterns located in different clusters. Clustering based on this distance is one of the most commonly used methods.

Complete-Link Method. This method is at the opposite end of the spectrum, as it is based on the distance between the two farthest patterns belonging to two clusters:

$$d(A, B) = \max_{\mathbf{x}\in A, \mathbf{y}\in B} d(\mathbf{x}, \mathbf{y}) \tag{1.9}$$

Group Average Link Method. In contrast to the two previous approaches, where the distance is determined on the basis of extreme values of the distance function, this method considers the average between the distances computed between all pairs of patterns, one from each cluster. We have

$$d(A, B) = \frac{1}{\text{card}(A)\text{card}(B)} \sum_{\mathbf{x}\in A, \mathbf{y}\in B} d(\mathbf{x}, \mathbf{y}) \tag{1.10}$$

Obviously, these computations are more intensive. However, they reflect a general tendency between the distances computed for individual pairs of patterns.

Obviously, we can develop other ways of expressing the distance between A and B. For instance, the Hausdorff method of computing the distance between two sets of patterns could be an attractive alternative.

There is an interesting general expression for describing various agglomerative clustering approaches known as the Lance-Williams recurrence formula. It expresses the distance between clusters A and B and the cluster formed by merging them (C)

$$d_{A\cup B,C} = \alpha_{\text{A}} d_{A,C} + \alpha_B d_{B,C} + \beta d_{A,B} + \gamma |d_{A,C} - d_{B,C}| \tag{1.11}$$

with the adjustable values of the parameters $\alpha_A(\alpha_B)$, β, and γ. This is shown in Table 1.2, where the choice of values implies a certain clustering method.

1.3.2. Objective Function-Based Clustering

The second general category of clustering is concerned with building partitions (clusters) of data sets on the basis of some performance index known also as an objective function. In essence, partitioning N patterns into c clusters (groups)

TABLE 1.2. Values of the Parameters in the Lance-Williams Recurrence Formula and the Resulting Agglomerative Clustering; n_A, n_B, and n_C Denote the Number of Patterns in the Corresponding Clusters

Clustering Method	α_A (α_B)	β	γ
Single link	1/2	0	$-1/2$
Complete link	1/2	0	1/2
Centroid	$\frac{n_A}{n_A + n_B}$	$-\frac{n_A n_B}{(n_A + n_B)^2}$	0
Median	1/2	$-1/4$	0

is a nontrivial problem. First, the number of the partitions is expressed in the following form (Webb, 2002):

$$\frac{1}{c!}\sum_{i=1}^{c}(-1)^{c-i}\binom{c}{i}i^N \tag{1.12}$$

This number increases very quickly, making any attempt to enumerate all of the partitions unfeasible. The minimization of a certain objective function can be treated as an optimization approach leading to some suboptimal configuration of the clusters (which, in practice, is an appealing solution). The main design challenge lies in formulating an objective function that is capable of reflecting the nature of the problem so that its minimization reveals a meaningful structure in the data set. The minimum variance criterion is one of the most common options. Having N patterns in $\mathbf{R}^n$, and assuming that we are interested in forming c clusters, we compute a sum of dispersions between the patterns and a set of prototypes $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c$

$$Q = \sum_{i=1}^{c}\sum_{k=1}^{N} u_{ik}\|\mathbf{x}_k - \mathbf{v}_i\|^2 \tag{1.13}$$

with $\| \quad \|^2$ being a certain distance between $\mathbf{x}_k$ and $\mathbf{v}_i$. The important component in the above sum is a partition matrix $U = [u_{ik}], i = 1, 2, \ldots, c, k = 1, 2, \ldots, N$ whose role is to allocate the patterns to the clusters. The entries of U are binary. Pattern k belongs to cluster i when $u_{ik} = 1$. The same pattern is excluded from the cluster when $u_{ik} = 0$. Partition matrices satisfy the following conditions:

Each cluster is nontrivial, that is, it does not include all patterns and is nonempty:

$$0 < \sum_{k=1}^{N} u_{ik} < N, \quad i = 1, 2, \ldots, c$$

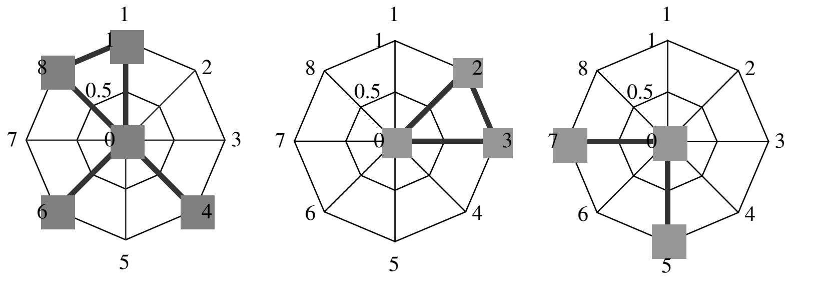

Figure 1.5. Star diagram as a graphical representation of the partition matrix for three clusters.

Each pattern belongs to a single cluster:

$$\sum_{i=1}^{c} u_{ik} = 1, \quad k = 1, 2, \ldots, N$$

The family of partition matrices (viz., binary matrices satisfying these two conditions) will be denoted by $\mathbf{U}$. As a result of minimization of Q, we construct the partition matrix and a set of prototypes. Formally we express this in the following way, which is just an optimization problem with constraints:

$$\text{Min } Q \text{ with respect to } \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c \text{ and } U \in \mathbf{U} \tag{1.14}$$

Several methods are used to achieve this optimization. The most common one, named C-Means (Duda et al., 2001; Webb, 2002), is a well-established way of clustering data.

Partition matrices are an intuitively appealing form in which to illustrate the structure of the patterns. For instance, the matrix formed for $N = 8$ patterns split into $c = 3$ clusters is

$$U = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Each row describes a single cluster. Thus we have the following arrangement: the first cluster consists of patterns {1, 4, 6, 8}, the second involves patterns {2, 3}, and the third covers the remaining patterns, {5, 7}.

Graphically, the partition matrix (or, equivalently, the structure of the data set) can be shown in the form of a so-called star or radar diagram (Figure 1.5).

1.4. CLUSTERING AND CLASSIFICATION

The structure revealed through the clustering process allows us to set up a classifier. The "anchor" points of the classifier are the prototypes of the clusters. Each

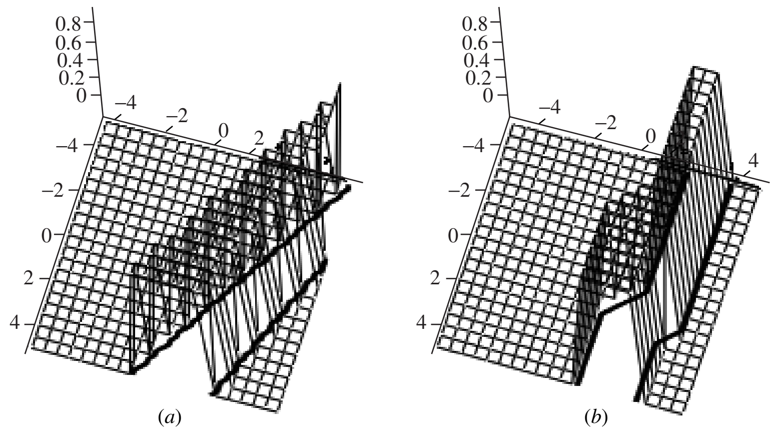

Figure 1.6. Plots of the classification regions of the first class with prototype $\mathbf{v}_1 = [1.7\ 2.5]^T$. The two other prototypes are $\mathbf{v}_2 = [0.4\ 0.3]^T$ and $\mathbf{v}_3 = [2.5\ 4.5]^T$ for two distance functions: (*a*) Euclidean and (*b*) Hamming.

cluster forms a single class $\omega_1, \omega_2, \ldots, \omega_c$. Using the clusters, we develop a nearest neighbor classification rule by stating that $\mathbf{x}$ belongs to class ω_j if it is the closest to the prototype $\mathbf{v}_j$:

$$j = \arg\ \min_i \|\mathbf{x} - \mathbf{v}_i\|^2 \tag{1.15}$$

This classification rule generates the corresponding decision regions in the feature space. Depending on the form of the distance function, we end up with different geometry of the classification boundaries, as shown in Figure 1.6.

The classifier formed in this way is one of the simplest architectures focusing exclusively on the prototypes of the clusters.

1.5. FUZZY CLUSTERING

The binary character of partitions described so far may not always be a convincing representation of the structure of data. Consider the set of two-dimensional patterns illustrated in Figure 1.7. While we can easily detect three clusters, their character is different. The first one is quite compact, with highly concentrated patterns. The other two exhibit completely different structures. They are far less condensed, with several patterns whose allocation to a given cluster may be far less certain. In fact, we may be tempted to allocate them to two clusters with varying degrees of membership. This simple and appealing idea forms a cornerstone of fuzzy sets—collections of elements with partial membership in several categories. As illustrated in Figure 1.7, the two identified patterns could easily belong to several clusters.

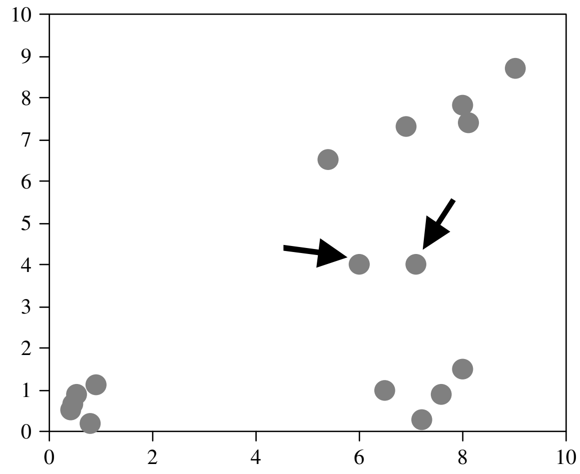

Figure 1.7. Three clusters with patterns of partial membership (belongingness) in the clusters. The patterns of borderline character are pointed to by the arrows.

These situations of partial membership occur quite often. Structures (clusters) may not be well separated for a variety of reasons. There may be noise or lack of discriminatory power of the feature space in which the patterns are represented. Some patterns could be genuine outliers. Some of them could be borderline cases and thus are difficult to classify. As a result, they may require far greater attention. A clustering algorithm that could easily provide detailed insight into the membership grades of the patterns could be a genuine asset. Let us assume that this is true and that the partition matrix now consists of grades of membership distributed in the unit interval. For the data in Figure 1.8, the partition matrix comes with the entries shown. The results are highly appealing, and they fully reflect our intuitive observations: patterns 6 and 7 have a borderline character, with membership grades in one of the clusters at the 0.5 level. The values in the partition matrix quantify the effect of partial membership.

Because we have allowed for partial membership, the clustering algorithm leading to such conceptual augmentation is regarded as a generalization of the standard FCM and is named of fuzzy C-Means or FCM, for short This generalization was introduced by Dunn (1974) and generalized by Bezdek (1981). The performance index (objective function) guiding the search through the data space assumes the form

$$Q = \sum_{i=1}^{c} \sum_{k=1}^{N} u_{ik}^{m} \|\mathbf{x}_k - \mathbf{v}_i\|^2 \tag{1.16}$$

It seems similar to the one guiding the optimization of the Boolean (two-valued) partition matrix with only one significant exception: we consider U to be a fuzzy partition, viz., a matrix with the entries confined to the unit interval that satisfies two important requirements:

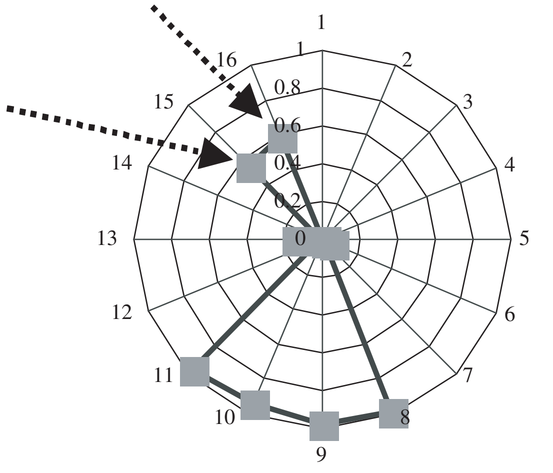

Figure 1.8. Star diagram of the fuzzy partition matrix.

- The clusters are nontrivial. For each cluster ($i = 1, 2, \ldots, c$) we end up with a nonempty construct that does not include all patterns.
- The total membership grades sum to 1, so the distribution of belongingness is equal to 1.

In the above objective function, we can rewrite the distance in a quadratic form by noting that $\|\mathbf{x}_k - \mathbf{v}_i\|^2 = (\mathbf{x}_k - \mathbf{v}_i)^T(\mathbf{x}_k - \mathbf{v}_i) = \mathbf{x}_k^T\mathbf{x}_k - 2\mathbf{x}_k^T\mathbf{v}_i + \mathbf{v}_i^T\mathbf{v}_i$. More formally, determining the structure is equivalent to the optimization task of the form given by (1.13) in the case of the binary C-Means. As before, the minimization is completed with respect to the partition matrix and the prototypes. The fuzzification factor (m), $m > 1$, helps control the shapes of the clusters and produces a balance between the membership grades close to 0 or 1 and those with intermediate values.

The derivations of the solution are completed in two steps. The first one involves the constraints accompanying the requirements imposed on the partition matrix. We incorporate the constraints with the aid of Lagrange multipliers. Then for each pattern $t = 1, 2, \ldots, N$, we formulate the augmented functional

$$V = \sum_{i=1}^{c} u_{it}^m d_{it}^2 - \lambda\left(\sum_{i=1}^{c} u_{it} - 1\right) \tag{1.17}$$

with λ denoting a Lagrange multiplier. Computing the derivative of V with respect to u_{st} and making it equal to 0, we obtain

$$\frac{\partial V}{\partial u_{st}} = m u_{st}^{m-1} d_{st}^2 - \lambda = 0 \tag{1.18}$$

and

$$u_{st} = \left(\frac{\lambda}{m}\right)^{1/(m-1)} \frac{1}{(d_{st})^{\frac{2}{m-1}}} \tag{1.19}$$

Taking into account the identity constraint $\sum_{j=1}^{c} u_{jt} = 1$, we have

$$\left(\frac{\lambda}{m}\right)^{1/(m-1)} \sum_{j=1}^{c} \frac{1}{(d_{jt})^{\frac{2}{m-1}}} = 1 \tag{1.20}$$

This allows us to determine the Lagrange multiplier λ:

$$\left(\frac{\lambda}{m}\right)^{1/(m-1)} = \frac{1}{\sum_{j=1}^{c} \frac{1}{(d_{jt})^{\frac{2}{m-1}}}} \tag{1.21}$$

Next, we insert the above expression into (1.19), which yields

$$u_{st} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d_{st}}{d_{jt}}\right)^{\frac{2}{m-1}}} \tag{1.22}$$

The computations of the prototypes are straightforward, as no constraints are imposed on them. The minimum of Q computed with respect to $\mathbf{v}_s$ yields

$$\nabla_{\mathbf{v}_s} Q = \mathbf{0}$$

The detailed solution depends on the distance function. In the case of the Euclidean distance, this leads to the expression

$$2\sum_{k=1}^{N} u_{sk}^{m}(\mathbf{x}_k - \mathbf{v}_s) = \mathbf{0} \tag{1.23}$$

We immediately obtain

$$\mathbf{v}_s = \frac{\sum_{k=1}^{N} u_{sk}^{m}\mathbf{x}_k}{\sum_{k=1}^{N} u_{sk}^{m}} \tag{1.24}$$

Other forms of the distance function, such as the Hamming or Tchebyschev distances do not lead to an immediate solution and require more optimization effort. The clustering using these options has been reported in the literature. We will discuss this extension later on.

To summarize, we can regard the FCM algorithm as an iterative process involving successive computations (updates) of the prototypes and the partition matrix. The values of the parameters are set up in advance. They consist of the following items: the number of clusters (c), the distance function $\| \cdot \|$, the fuzzification factor (m), and the termination criterion (ε).

Initialization Phase. Select values of c, m, and ε. Choose the distance function. Initialize (randomly) the partition matrix:

```
Repeat // main iteration loop
Compute prototypes of the clusters
Compute the partition matrix
until a given stopping criterion quantified in terms of $e$ has been satisfied.
```

Let us review the design of the clustering process in more detail; in this context, it is useful to refer to the list of parameters we can choose up front the entire process. The number of clusters reflects the level of generality we are interested in setting up when dealing with the data. While this number is not known exactly, the feasible range of clusters we can establish is rather narrow and clearly reflects the domain knowledge we may have about the problem at hand and/or the objectives of the data analysis problem. For instance, when dealing with a huge customer database, it is reasonable to assume that we are interested in a few (say, five) groups of customers, which we want to characterize for further marketing of new products and services. There is no point in moving toward a very refined partition of the data into 20 or more clusters, as these could be difficult to interpret and take advantage of (perhaps we do not wish to be so specialized and launch a very narrow marketing campaign). The limit cases are trivial: $c = 1$ does not reveal any structure. The number of clusters set up to N, $c = N$, is meaningless: each pattern forms an individual cluster, and this does not make any sense. Overall, we observe that there is a strong monotonic character of the objective function treated as a function of c; when the number of clusters increases, the values of Q decrease. There may be a slight departure from this tendency, but very often it is limited and insignificant. The distance function helps focus our search for structure. As we demonstrated in Section 1.2.2 each distance function implies a certain geometry and navigates search for finding structure in data. Hopefully, the structure revealed in this way is compatible with the "internal" geometry of the data set itself. For instance, when dealing with the Euclidean distance, our favorite geometry consists of hyperballs (or hyperellipsoids in case we start weighting the features). The Hamming distance focuses clustering on the search for the structure that is compatible with diamond-like shapes of concentrations of data. The Tchebyschev distance favors a search focusing on hyperboxes. The

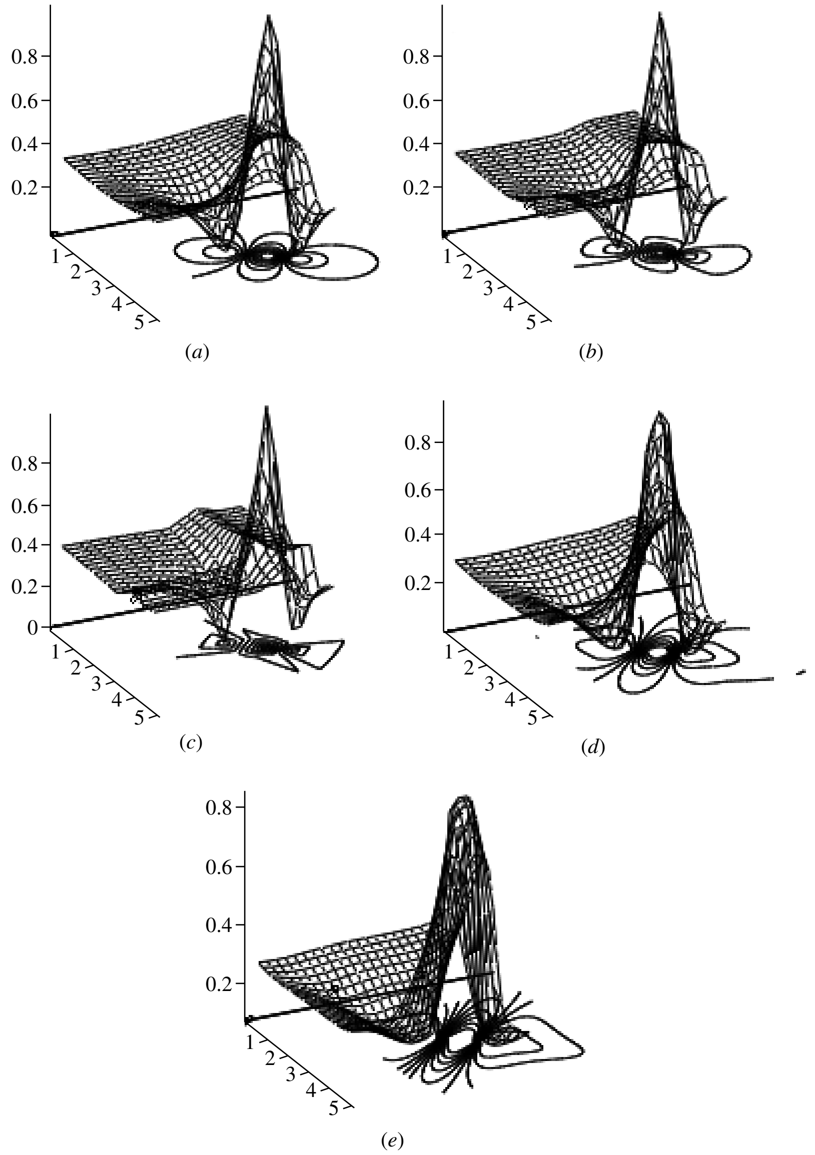

Figure 1.9. Shapes of the membership function for selected values of the fuzzification factor (m): (a) $m = 2$; (b) $m = 1.5$; (c) $m = 1.05$; (d) $m = 3$; (e) $m = 5$.

fuzzification factor influences the shape of the clusters. Typically, its value is set to 2. By changing the values of m, we can make the clusters (partition matrices) look more Boolean so that we see more membership grades close to 0 or 1. This happens when m approaches 1. On the other hand, when m increases (with values greater than 2), the resulting membership grades lead to spike-like functions. Illustrative examples are presented in Figure 1.9.

The stopping criterion quantifies the situation in which the clustering process can be stopped because it has reached a steady state and further computations are not justified. The typical approach used is to compare partition matrices produced in two successive iterations of the FCM, say $U(\text{iter}+1)$ and $U(\text{iter})$, and, if the distance between them $\|U(\text{iter}+1) - U(\text{iter})\|$ does not exceed a certain threshold (ε), stop the computing. The distance itself could be quantified by taking into account the biggest change in the partition matrix, that is,

$$\|U(\text{iter}+1) - U(\text{iter})\| = \max_{i,k} |u_{ik}(\text{iter}+1) - u_{ik}(\text{iter})| \tag{1.25}$$

The numeric value of ε is application oriented; usually it is confined to the range of 10^{-3}–10^{-5}.

Fuzzy partition matrices provide detailed insight into the structure of the data set that is thoroughly quantified through the membership values. Based on these values, we can easily separate the patterns that are typical of the cluster (as they have membership grades close to 1) from the data of borderline character. Obviously, the membership grades help quantify the effect of partial membership. Sometimes we are interested in cluster allocation without having any detailed information on the patterns. This is achieved by a transformation known as hardening (Bezdek, 1981). It produces a binary relation (partition matrix) based on the original partition matrix by choosing the largest membership value. More specifically, given u_{ik}, we produce Boolean entries of matrix $\tilde{u}_{ik}$ such that

$$\tilde{u}_{ik} = \begin{cases} 1 & \text{if } i = \arg\ \max_j u_{jk} \\ 0 & \text{otherwise} \end{cases} \tag{1.26}$$

Let us briefly note the use of the clustering results in the design of the classifier. As in the K-means algorithm, we use the nearest neighbor rule, which in this case takes into account the fact that the membership grades are continuous. We then anticipate that all prototypes contribute to some extent to the determination of the membership values. Consider that, given the set of prototypes, we now have a new pattern, $\mathbf{x}$. Its membership grades to the clusters $\mathbf{u} = [u_1 u_2 \ldots u_c]^T$ take the form

$$u_i = \frac{1}{\displaystyle\sum_{j=1}^{c} \left(\frac{\|\mathbf{x} - \mathbf{v}_i\|}{\|\mathbf{x} - \mathbf{v}_j\|} \right)^{2/(m-1)}} \tag{1.27}$$

It is easy to check that (1.27) is a solution to the optimization problem

$$\text{Min}_{\boldsymbol{u}} \sum_{i=1}^{c} u_i^m \|\mathbf{x} - \mathbf{v}_i\|^2 \tag{1.28}$$

with the unity constraint imposed on the membership grades, $\sum_{i=1}^{c} u_i = 1$.

1.6. CLUSTER VALIDITY

As we indicated above, the number of clusters is application-driven and user-centric. As the user is in the middle of the process of data analysis, it is beneficial to consider a varying number of clusters and analyze the results produced. Obviously, it would be helpful to have some automation in this process. This automation should come with a synthetic measure with which we can assess the quality of the discovered structure. To state the problem in a different way, what is the optimal number of clusters? Or, even better, what is the preferred number of clusters given the underlying geometry imposed on the clustering process through the use of some objective function? The characteristics of the data could be quite different from those captured by the objective function. How easily could we detect elongated clusters when the objective function favors spherical shapes? What about two elongated and crossing clusters? What about a porcupine-like distribution of patterns? To address these questions, we need a certain cluster validity measure (Dubes, 1987; Windham, 1980; Windham, 1982; Xie and Beni, 1991). Now that the problem has been identified, a number of proposals have been made. Most of them define the validity functional on the partition matrix returning a certain real number. More formally, the validity index $v(c)$ is defined as the following mapping $V\colon U \rightarrow \mathbf{R}$. The extreme value (either minimum or maximum) of V treated as a function of c (as we are interested in changing this parameter) points at the feasible or most likely number of clusters in the data structure. The commonly used validity functionals include the partition index and partition entropy.

Partition Index. The partition index of U, denoted by $P(U)$, produces an average of the squared values of the membership grades encountered in the partition matrix:

$$P(U) = \frac{1}{N}\sum_{i=1}^{c}\sum_{k=1}^{N} u_{ik}^2 \tag{1.29}$$

If each pattern belongs to a single cluster (hard partition), then the partition index assumes its maximal value of 1. If patterns share their membership across all clusters, with the same membership grade equal to $1/c$, this gives rise to the lowest value of $P(U)$, which in this case equals $1/c$. In other words, the index quantifies the ambiguity of the partition matrices so that we can rank them and select the one with the lowest ambiguity.

Partition Entropy. We form an entropy function defined over the partition matrix in the following manner:

$$H(U) = -\frac{1}{N}\sum_{i=1}^{c}\sum_{k=1}^{N} u_{ik}\ln(u_{ik}) \tag{1.30}$$

(we assume that for $u = 0$, $\ln = 0$)

The values of the partition entropy range from 0 to $\ln(N)$. Again, if we consider Boolean entries of the partition matrix, the entropy is equal to 0. The highest value is obtained when there is a uniform distribution of membership grades (equal to $1/c$); here we have

$$H(U) = -\frac{1}{N}\sum_{i=1}^{c}\sum_{k=1}^{N}\frac{1}{c}\ln\left(\frac{1}{c}\right) = \ln(c) \tag{1.31}$$

If we are interested in the lowest entropy, the clusters leading to it are preferred over other structures discovered in the collection of patterns.

While the partition index and partition entropy exhibit interesting properties that are useful in quantifying the ambiguity of partition matrices and identifying those with the lowest values as the most preferred, their use in revealing the most plausible number of clusters may not be obvious. Even though they have been used for this purpose, both indexes tend to be quite monotonic, without a well-delineated minimum (which could have been used as a sound indicator of the plausible data structure). The literature provides other cluster validity criteria [proportion exponent (Windham, 1980, 1982), separation index, fuzzy hypervolume, average partition density], but to some extent, all of them are affected by monotonicity. Furthermore when applied together, they may lead to conflicting findings concerning the most preferred number of clusters. These indexes can be useful, but we should keep in mind their limited role and treat the findings implied by them as only useful guidelines. In any case, we should be concerned about the range of the plausible number of clusters rather than a single value.

1.7. EXTENSIONS OF OBJECTIVE FUNCTION-BASED FUZZY CLUSTERING

The objective function given by (1.16) is generic in terms of its composition and the distance function being used. There are several interesting extensions of this objective function that are helpful in exploring a broad range of geometry of the clusters.

1.7.1. Augmented Geometry of Fuzzy Clusters: Fuzzy C Varieties

The search for structure (clusters) is always biased by the selection of the distance function. Changing the distance function imposes different "computing lenses" of the algorithm and causes the clustering technique to favor some geometry of the clusters. There are many possible variations of the distance function. Instead of expressing the distance between two patterns (one of them being a prototype), we may imagine that the prototypes are more complex geometric constructs. This is the rationale behind constructs such as fuzzy c-lines, fuzzy c-ellipsoids, and so on. The concept of fuzzy c-varieties studied by Bezdek et al. (1981) addresses the geometry of clusters. So far, these are just points. Here we consider a variety of geometric constructs. Each cluster represents an r-dimensional variety, where $r \in \{0, 1, 2, \ldots, n-1\}$, and is described by a prototype $\mathbf{v}_i$ and the collection of orthogonal unit vectors $\mathbf{e}_{i1}, \mathbf{e}_{i2}, \ldots, \mathbf{e}_{ir}$ and in essence becomes an affine subspace of $\mathbf{R}^n$:

$$\{\mathbf{y} \in \mathbf{R}^n | \mathbf{y} = \mathbf{v}_i + \sum_{j=1}^{r} \mathbf{t}_j \mathbf{e}_{i1}, \mathbf{t} \in \mathbf{R}^r\} \tag{1.32}$$

In the case where $r = 0$, we end up with the generic version of the FCM confined to a single prototype. If $r = 1$, this variety represents a straight line. The value of r set to 2 returns a plane. In general, if $r = p - 1$, we end up with a hyperplane. The distance between any pattern $\mathbf{x}$ and the ith c-variety is computed in the form

$$d(\mathbf{x}; \mathbf{v}_i, \mathbf{e}_i) = \|\mathbf{x} - \mathbf{v}_i\|^2 - \sum_{j=1}^{r} ((\mathbf{x} - \mathbf{v}_i)^T e_{ij})^2 \tag{1.33}$$

These distances for $n = 2$ (here the c-variety reduces to a straight line) are illustrated in Figure 1.10.

The geometry of fuzzy clusters becomes of paramount importance when we use clustering for image processing, that is, processing of geometric objects such as circles, ovals, boxes, elongated edges of machine parts, and the like. While in this domain we are generally confined to two-dimensional spaces (digital images), it is apparent that the diversity of geometric shapes poses a serious challenge. If we are interested in searching for a specific geometric form, then this search should be driven by the geometry in question. For example, if we are interested in circular shapes, the distance function of the form $d(\mathbf{x}; \mathbf{v}, r) = |\|\mathbf{x} - \mathbf{v}\| - r|$ would reflect our interest in looking for these type of clusters (see Figure 1.11). Its minimal values are distributed on the circle we are interested in revealing in the data.

The number of clustering studies in this area is extensive. The research monograph by Hoppner et al. (1999) forms a comprehensive compendium.

1.7.2. Possibilistic Clustering

Possibilistic clustering arose as a challenge to the probabilistic-like character required for the membership grades in clusters. The limitation of the form does not allow us to distinguish between a situation in which a pattern is shared between clusters and one in which it is simply atypical and we would like to see this effect quantified in some way. The unity constraint does not allow this. Possibilistic clustering, as advocated by Krishnampuram and Keller (1993, 1996), drops the requirement that the sum of membership grades must equal 1. This is reflected in the form of the augmented objective function:

$$Q = \sum_{i=1}^{c} \sum_{k=1}^{N} u_{ik}^m \|\mathbf{x}_k - \mathbf{v}_i\|^2 + \sum_{i=1}^{c} \beta_i \sum_{k=1}^{N} (1 - u_{ik})^m \tag{1.34}$$

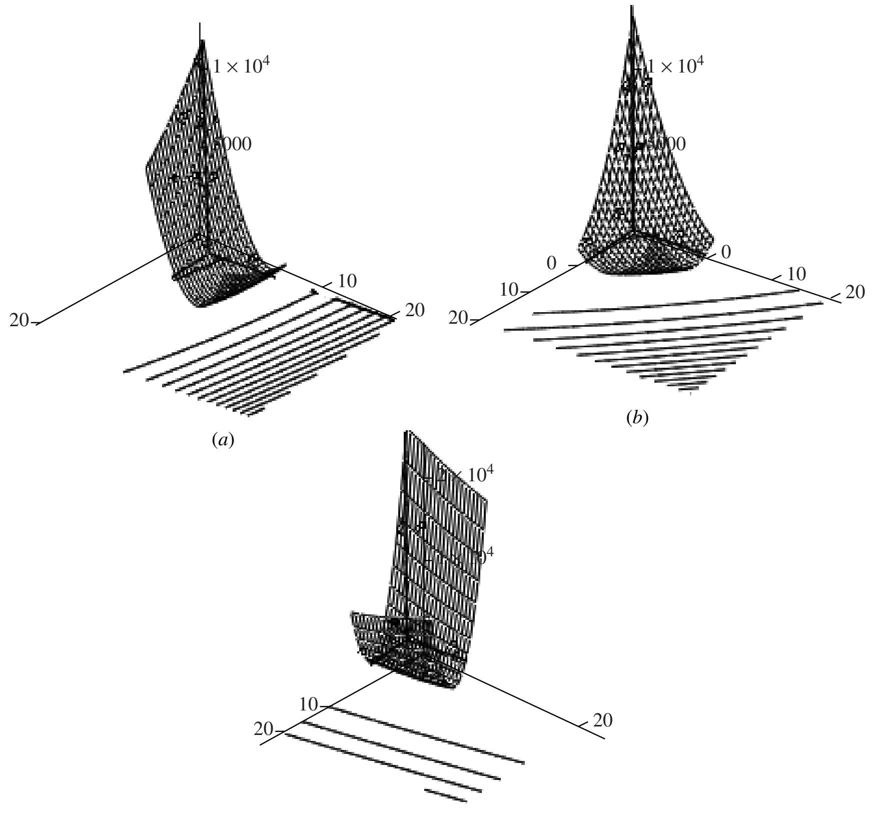

Figure 1.10. Three-dimensional and contour plots of the distance for the *c*-variety for selected values of **e**: (*a*) $\mathbf{e} = [1.0\ \ 5.0]^T$; (*b*) $\mathbf{e} = [3.0\ \ 3.0]^T$; (*c*) $\mathbf{e} = [7.0\ \ 0.5]^T$. In all cases $\mathbf{v} = [1\ \ 2]^T$.

Note that in addition to the standard term, the second sum expresses our desire to have the membership grades sum to 1 (but this is not the constraint articulated in the standard FCM algorithm). The positive weight factor β_i (which depends on a specific cluster) helps us set up a suitable balance between the structure-seeking term and the departures from the unity requirement of the membership grades. From the standpoint of the second term, we prefer the membership grades to be closer to 1; this may have led to very high values of the first sum. This weighted sum is critical when we want to articulate a sound balance. Because of the character of the membership constraint, we refer to the resulting algorithm as a possibilistic fuzzy C-Means (P-FCM). The derivations of the partition matrix lead to the following expression:

$$u_{ik} = \frac{1}{1 + \left(\dfrac{\|\mathbf{x}_k - \mathbf{v}_i\|^2}{\beta_i}\right)^{\frac{1}{m-1}}} \tag{1.35}$$

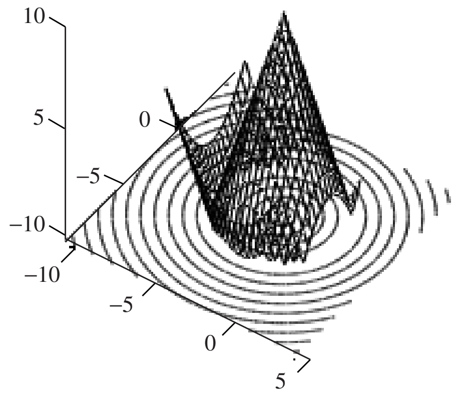

Figure 1.11. Expressing the distance between **x** and a circle with center **v** (= [1 1]^T) and radius $r(=5)$; $||.||$ is specified as the Euclidean distance.

The formula for the prototypes is the same as that used the FCM method. The choice of an optimal value of β_k impacts the results; a sound selection of these weight factors was proposed by Krishnampuram and Keller (1996) to be in the form

$$\beta_i = \frac{\sum_{k=1}^{N} u_{ik}^m \|\mathbf{x}_k - \mathbf{v}_i\|}{\sum_{k=1}^{N} u_{ik}^m} \tag{1.36}$$

1.7.3. Noise Clustering

The method used to cope with noisy data is another important application-driven issue in searching for structure in experimental data. The technique of Ohashi (1984) and Dave (1991) is to introduce a special cluster, the noise cluster, whose role is to "localize" the noise and place it in a single auxiliary cluster. By assigning patterns to this noise cluster, we declare them to be outliers in the data set. The objective function that helps capture this effect is represented in the form

$$Q = \sum\sum u_{ik}^m \|\mathbf{x}_k - \mathbf{v}_i\|^2 + \sum_{k=1}^{N} \delta^2 \left(1 - \sum_{i=1}^{c} u_{ik}\right)^m \tag{1.37}$$

The weight coefficient δ^2 reflects the distance between all data and the noise cluster. Note that we end up with $c+1$ clusters, with the extra cluster serving as the noise cluster. The difference in the second term of the objective function expresses the degree of membership of each pattern in the noise cluster. The sum over the first c is less than or equal to 1. The derivation of the partition matrix produces the following expression:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{\|\mathbf{x}_k - \mathbf{v}_i\|}{\|\mathbf{x}_k - \mathbf{v}_j|} \right)^{\frac{2}{m-1}} + \sum_{j=1}^{c} \left(\frac{\|\mathbf{x}_k - \mathbf{v}_i\|}{\delta} \right)^{\frac{2}{m-1}}} \tag{1.38}$$

1.8. SELF-ORGANIZING MAPS AND FUZZY OBJECTIVE FUNCTION-BASED CLUSTERING

Objective function-based clustering forms one of the main optimization paradigms of data discovery. To put this in a broader perspective, we elaborate on the alternative arising in neurocomputing, namely, self-organizing maps. This helps us contrast the underlying optimization mechanisms and look at various formats of results generated by different methods.

The term self-organizing map (SOM) was coined by Kohonen (1982, 1989, 1995 Kohonen et al, 1996). As usually emphasized in the literature, SOMs are regarded as regular neural structures (neural networks) composed of a grid of artificial neurons that attempt to visualize highly dimensional data in a low-dimensional structure, usually in the form of a two- or three-dimensional map. To make such visualization meaningful, this low-dimensional representation of the originally high-dimensional data has to preserve the *topological* properties of the data set. This means that two data points (patterns) that are close to each other in the original feature space should retain this similarity (or closeness) in their representation (mapping) in the reduced, low-dimensional space in which they are visualized. And, reciprocally, two distant patterns in the original feature space should retain their distant location in the low-dimensional space. Put it differently, we can state that the SOM acts as a *computer eye* that helps us gain insight into the structure of the data set and observe relationships occurring between the patterns originally located in a highly dimensional space. In this way, we can confine ourselves to the two-dimensional map that apparently reveals all essential relationships between the data, as well as dependencies between the software measures themselves. In spite of the existing variations, the generic SOM architecture (as well as the learning algorithm) remains basically the same. Below we summarize the essence of the underlying self-organization algorithm that achieves a certain form of unsupervised learning (Kohonen et al., 1996; Vesanto and Alhoniemi, 2000).

Before proceeding with the detailed computations, we introduce the necessary notation. Here n software measures are organized in a vector $\mathbf{X}$ of real numbers situated in the n-dimensional space of real numbers, $\mathbf{R}^n$. The SOM is a collection of linear neurons organized in the form of a regular two-dimensional grid (array) (Figure 1.12).

In general, the grid may consist of p rows and r columns; commonly we confine ourselves to the square array of $p \times p$ elements (neurons). Each neuron

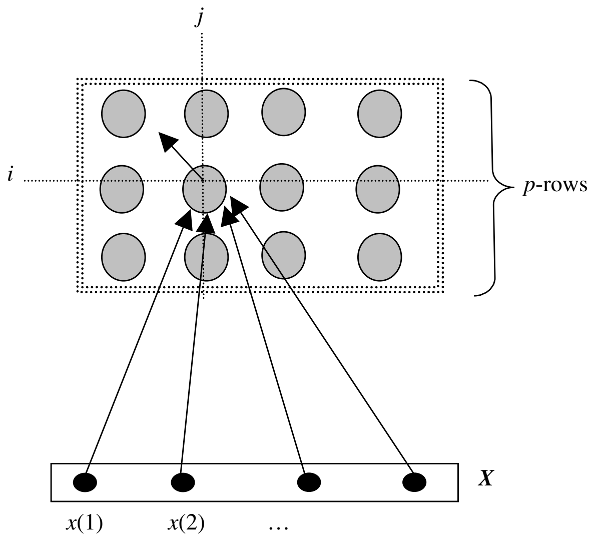

Figure 1.12. A basic topology of the SOM constructed as a grid of identical processing units (neurons).

is equipped with modifiable connections $\mathbf{w}(i,j)$ that form an n-dimensional vector of connections $\mathbf{w}(i, j) = [w_1(i, j)w_2(i, j) \ldots w_n(i, j)]$. It completes computing of the distance function $d(. , .)$ between its connections and the corresponding input $\mathbf{x}$

$$y(i, j) = d(\mathbf{w}(i, j), \mathbf{x}) \tag{1.39}$$

where the pair (i,j) denotes a certain (i,j) position of the neuron in the array. $\mathbf{x}$ is an input to all neurons. The distance can be chosen from the list of alternatives presented earlier. The same input $\mathbf{x}$ affects all neurons. The neuron with the shortest distance between the input and the connections becomes activated to the highest extent and is called the winning neuron. Let us denote its coordinates by $(i0,\ j0)$. More precisely, we have

$$(i0,\ j0) = \arg\ \min_{(i,j)}\ \mathrm{d}(\mathbf{w}(i, j), \mathbf{x}) \tag{1.40}$$

The winning neuron matches (responds to) $\mathbf{x}$. Because it is the winner of this competition, we reward the neuron and allow it to modify the connections so that they are even closer to the input data. The update mechanism is governed by the expression

$$\mathbf{w\_new}(i0,\ j0) = \mathbf{w}(i0,\ j0) + \alpha(\mathbf{x} - \mathbf{w}(i0,\ j0)) \tag{1.41}$$

where α denotes a learning rate, $\alpha > 0$. The higher the learning rate, the more intensive the updates of the connections. In addition to the changes of the connections of the winning node (neuron), we allow this neuron to affect its neighbors (viz., the neurons located at similar coordinates of the map). The way in which this influence is quantified is expressed via a neighbor function $\Phi(i, j, i0, j0)$. In general, this function satisfies two intuitively appealing conditions: (a) it attains a maximum equal to 1 for the winning node, $i = i0$, $j = j0$, and (b) when the node is apart from the winning node, the value of the function gets lower (in other words, the updates are less vigorous). Evidently, there are also nodes where the neighbor function equals 0. Considering the above, we rewrite (1.41) in the following form:

$$\mathbf{w\_new}(i, j) = \mathbf{w}(i0, j0) + \alpha\Phi(i, j, i0, j0)(\mathbf{x} - \mathbf{w}(i, j)) \tag{1.42}$$

The typical neighbor function comes in the form

$$\Phi(i, j, i0, j0) = \exp(-\beta((i - i0)^2 + (j - j0)^2)) \tag{1.43}$$

with the parameter β usually assuming small positive values.

The above update expression (1.42) applies to all the nodes (i,j) of the map. As we iterate (update) the connections, the neighbor function shrinks: at the beginning of updates, we start with a large region of updates, and when the learning settles down, we start reducing the size of the neighborhood. For instance, its size may decrease linearly.

The number of iterations is specified in advance or the learning terminates once there are no significant changes in the connections of the neurons.

SOM and FCM are complementary, and so are their advantages and shortcomings. FCM requires the number of groups (clusters) to be defined in advance. It is guided by a certain performance index (objective function), and the solution comes in the clear form of a certain partition matrix. In contrast, SOM is more user-oriented. No specific number of clusters (group) needs to be specified in advance.

1.9. CONCLUSIONS

We have reviewed the paradigm of clustering, and fuzzy clustering in particular, and have discussed its role in revealing structure in data sets. Fuzzy clustering leads to information granulation in terms of fuzzy sets or fuzzy relations. Membership grades are important indicators of the typicality of patterns or their borderline character. We discussed various categories of fuzzy clustering and extensions of the underlying objective functions. The cluster indexes are useful in identifying the most relevant number of clusters. Each objective function implies a search for structure in data driven by some superimposed geometry, and we have shown that different distance functions emphasize a certain geometry we intend to find in the data set. Stated differently, clustering predisposes the search toward a certain geometry that is favored when the clusters are built. For instance, the Euclidean distance focuses the search on spherical shapes of the clusters. The Mahalanobis distance expands this search by allowing hyperellipsoidal shapes in the data set. Having said that, we should interpret the notion of unsupervised learning (which is often synonymous with clustering) in a suitable manner. There is no direct supervision (as encountered, for example, in classifier design), but there is still a mechanism of implicit supervision in the form of the geometric bias of search accompanying the accepted distance function.

REFERENCES

M.R. Anderberg, *Cluster Analysis for Applications*, Academic Press, New York, 1973.

G.P. Babu, M.N. Murthy, Clustering with evolutionary strategies, *Pattern Recognition*, 27, 1994, 321–329.

J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

J.C. Bezdek, C. Coray, R. Guderson, J. Watson, Detection and characterization of cluster substructure, *SIAM Journal of Applied Mathematics*, 40, 1981, 339–372.

J.C. Bezdek, J. Keller, R. Krishnampuram, N.R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer Academic Publishers, Dordercht, 1999.

R.N. Dave, Fuzzy shell clustering and application to circle detection in digital images, *Int. J. General Systems*, 16, 1990, 343–355.

R.N. Dave, Characterization and detection of noise in clustering, *Pattern Recognition Letters*, 12, 1991, 657–664.

R.N. Dave, K. Bhaswan, Adaptive c-shells clustering and detection of ellipses, *IEEE Trans. on Neural Networks*, 3, 1992, 643–662.

P.A. Devijver, J. Kittler (eds.), *Pattern Recognition Theory and Applications*, Springer-Verlag, Berlin, 1987.

R. Dubes, How many clusters are the best?—an experiment, *Pattern Recognition*, 20, 6, 1987, 645–663.

R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, 2nd edition, John Wiley, New York, 2001.

J.C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, *J. of Cybernetics*, 3, 3, 1974, 32–57.

H. Frigui, R. Krishnapuram, A comparison of fuzzy shell clustering methods for the detection of ellipses, *IEEE Trans. on Fuzzy Systems*, 4, 1996, 193–199.

K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd edition, Academic Press, London, 1990.

F. Hoppner, F. Klawonn, R. Kruse, T. Runkler, *Fuzzy Cluster Analysis*, John Wiley, Chichester, England, 1999.

A.K. Jain, R.P.W. Duin, J. Mao, Statistical pattern recognition: a review, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22, 1, 2000, 4–37.

A.K. Jain, M.N. Murthy, P.J. Flynn, Data clustering: a review, *ACM Comput. Survey*, 31, 3, 1999, 264–323.

L. Kaufmann, P.J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley, New York, 1990.

P.R. Kersten, Fuzzy order statistics and their applications to fuzzy clustering, *IEEE Trans. on Fuzzy Systems*, 7, 7, 1999, 708–712.

F. Klawonn, A. Keller, Fuzzy clustering with evolutionary algorithms, *Int. J. of Intelligent Systems*, 13, 1998, 975–991.

T. Kohonen, Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43, 1982, 59–69.

T. Kohonen, *Self-organization and Associative Memory*, Springer-Verlag, Berlin, 1989.

T. Kohonen, *Self-organizing Maps*, Springer-Verlag, Berlin, 1995.

T. Kohonen, S. Kaski, K. Lagus, T. Honkela, Very large two-level SOM for the browsing of newsgroups, in: *Proceedings of ICANN96*, Lecture Notes in Computer Science, 1112, Springer-Verlag, Berlin, 1996, 269–274.

R. Krishnapuram, J. Keller, A possibilistic approach to clustering, *IEEE Trans. on Fuzzy Systems*, 1, 1993, 98–110.

R. Krishnapuram, J. Keller, The possibilistic C-Means algorithm: insights and recommendations, *IEEE Trans. on Fuzzy Systems*, 4, 1996, 385–393.

K. Mali, S. Mitra, Clustering of symbolic data and its validation, in: N.R. Pal and M. Sugeno (eds.), *Advances in Soft Computing—AFSS 2002*, Springer-Verlag, Heidelberg, 2002, 339–344.

Y. Ohashi, Fuzzy Clustering and robust estimation, Proceedings of 9th Meeting SAS User Group Int, Hollywood Beach, Florida, 1984.

J. Vesanto, A. Alhoniemi, Clustering of the self-organizing map, *IEEE Trans. on Neural Networks*, 11, 2000, 586–600.

A. Webb, *Statistical Pattern Recognition*, 2nd edition, John Wiley, Hoboken, NJ, 2002.

M.P. Windham, Cluster validity for fuzzy clustering algorithms, *Fuzzy Sets and Systems*, 3, 1980, 1–9.

M.P. Windham, Cluster validity for the fuzzy C-Means clustering algorithms, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11, 1982, 357–363.

X.L. Xie, G. Beni, A validity measure for fuzzy clustering, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13, 1991, 841–847.