

Chapter 2

Discovery of Intrinsic Clustering in Spatial Data

2.1 A Brief Background About Clustering

A fundamental task in knowledge discovery is the unraveling of clusters intrinsically formed in spatial databases. These clusters can be natural groups of variables, data-points or objects that are similar to each other in terms of a concept of similarity. They render a general and high-level scrutiny of the databases that can serve as an end in itself or a means to further data mining activities. Segmentation of spatial data into homogenous or interconnected groups, identification of regions with varying levels of information granularity, detection of spatial group structures of specific characteristics, and visualization of spatial phenomena under natural groupings are typical purpose of clustering with very little or no prior knowledge about the data. Often, clustering is employed as an initial exploration of the data that might form natural structures or relationships. It usually sets the stage for further data analysis or mining of structures and processes.

Clustering has long been a main concern in statistical investigations and other data-heavy researches (Duda and Hart 1974; Jain and Dubes 1988; Everitt 1993). It is essentially an unsupervised learning, a terminology used in the field of pattern recognition and artificial intelligence, which aims at the discovery from data a class structure or classes that are unknown a priori. It has found its applications in fields such as pattern recognition, image processing, micro array data analysis, data storage, data transmission, machine learning, computer vision, remote sensing, geographical information science, and geographical research. Novel algorithms have also been developed arising from these applications. The advancement of data mining applications and the associated data sets have however posed new challenges to clustering, and it in turn intensifies the interest in clustering research. Catering for very large databases, particularly spatial databases, some new methods have also been developed over the years (Murray and Estivilli-Castro 1998; Miller and Han 2001; Li et al. 2006). To facilitate our discussion, a brief review of the clustering methods is first made in this section.

These are two basic approaches to perform clustering: hierarchical clustering and partitioning clustering. With reference to some criteria for merging or splitting clusters on the basis of a similarity or dissimilarity/distance measure, hierarchical clustering algorithms produce, via an agglomerative or divisive manner, a dendrogram which is a tree showing a sequence of clustering with each being a partition of the data set. According to the structure adopted, hierarchical clustering can be further categorized into nested hierarchical clustering and non-nested hierarchical clustering. In nested hierarchical clustering, each small cluster fits itself in whole inside a larger cluster at a merging scale (or threshold) and every datum is not permitted to change cluster membership once an assignment has been made. In non-nested hierarchical clustering, a cluster obtained at small scale may divide itself into several small parts and fits these parts into different clusters at the merging scale and, therefore, each datum is permitted to change its cluster membership as the scale varies. The single-link (nearest-neighbor) algorithms (Hubert 1974; Dubes and Jain 1976), the complete-link (farthest-neighbor) algorithms (Johnson 1967; Hubert 1974), and the average-link (average-neighbor) algorithms (Ward 1963) are typical nested hierarchical clustering algorithms. The single-link method is more efficient but is sensitive to noise and tends to generate elongated clusters. Complete link and average link methods give more compact clusters but are computationally more expensive. On the other hand, the algorithms proposed in (Taven et al. 1990; Wilson and Spann 1990; Miller and Rose 1996; Blatt et al. 1997; Roberts 1997; Waldemark 1997) generate non-nested hierarchical clusterings.

Early hierarchical clustering algorithms such as AGENS (agglomerative nesting) and DIANA (divisa analysis) (Kaufman and Rousseeuw 1990) are under the curse of dimensionality and do not scale well for large data sets because of the difficulties in deciding on the merge or split points. To handle large data sets, BIRCH (balanced iterative reducing and clustering using hierarchies) obtains clusters by compressing data into smaller sub-clusters (Zhang et al. 1996). The algorithm appears to be linearly scalable and gives reasonably good-quality clustering. Clusters are spherical in shape but they may not be natural clusters. By combining random sampling and partitioning, CURE (clustering using representatives) merges clusters via the concepts of representative objects and shirking factor (Guha et al. 1998). It is relatively robust to outliers (objects in non-dense regions) and can identify clusters with non-spherical shapes and large variance. Somewhat similar to CURE, CHAMELEON employs the concepts of interconnectivity and closeness to merge clusters (Karypis et al. 1999). The algorithm appears to be more effective than CURE in identifying clusters with arbitrary shapes and varying density. The advantage of hierarchical clustering algorithms is that it is more versatile. They give a series of clusterings along some scales. The time complexity for agglomerative algorithms is $O(n^2 \log n)$ and the space complexity is $O(n^2)$, where n is the number of objects. The disadvantage of hierarchical clustering is that it is often difficult to determine at which level the clustering gives the optimal clusters essential to an investigation.

Differing from the hierarchical approach, partitioning algorithms give only a single partition of a data set. The majority of such algorithms partition a data set

into clusters through the minimization of some suitable measures such as a cost function. The K-means method, FORGY, ISODATA, WISH (MacQueen 1967; Anderberg 1973; Ball and Hall 1976; Dubes and Jain 1976), and Fuzzy ISODATA (Bezdek 1980), for examples, are essentially based on the minimization of a squared-error function. The K-means methods use the mean value of the objects in a cluster as the cluster center. Its time complexity is $O(nkt)$, where n is the number of objects, k is the number of clusters, and t is the number of iterations. That is, for fixed k and t , the time complexity is $O(n)$. Thus, it is essentially linear in the number of objects and this becomes its advantage. However, the K-means method is sensitive to initial partition, noise, and outliers (objects whose removal improves significantly the tightness of the clusters), and it cannot discover clusters of arbitrary shapes. By using the most centrally located object (medoid) in a cluster as the cluster center, the K-medoid is less sensitive to noise and outliers but in the expense of a higher computational cost. PAM (partitioning around medoids) is an earlier K-medoid method that uses a complex iterative procedure to replace k cluster centers (Kaufman and Rousseeuw 1990). The computational complexity in a single iteration is $O(k(n-k)^2)$. Thus, the algorithm is very costly for large data sets. To deal with large volume of data, CLARA (clustering large application) takes multiple samples of the whole data set and applies PAM to each sample to give the best clustering as the output (Kaufman and Rousseeuw 1990). The computational complexity for each iteration becomes $O(ks^2+k(n-k))$, where s is the sample size. So, the success of CLARA depends on the sample chosen. Good-quality clustering will not be achieved if the samples are biased. To better combine PAM and CLARA, CLARANS (clustering large applications based upon randomized search) is constructed to search only the subset of a data set but not confining itself to any sample at any time (Ng and Han 1994). The process is similar to searching a graph as if every one of its nodes are potential solutions. The algorithm attempts to search for a better solution by replacing the current one with a better neighbor in an iterative manner. Though CLARANS appears to be more effective than PAM and CLARA, its computational complexity is roughly $O(n^2)$. Furthermore, it assumes that all objects to be clustered are stored in the main memory. It should be noted that most of the partitioning methods cluster objects on the basis of the distance between them. It actually constitutes the expensive step of the algorithms. Since the minimization problems involved are generally NP-hard and combinatorial in nature, techniques such as simulated annealing (Kirpatrick et al. 1983), deterministic annealing (Rose et al. 1990), and EM (expectation maximization) algorithms (Celeux and Govaert 1992) are often utilized to lower the computational overhead. Moreover, most of the existing algorithms can only find clusters which are spherical in shape.

In addition to the hierarchical and partitioning approaches, there are other clustering methods such as the graph theoretic methods (Leung 1984; Karypis et al. 1999), the density-based methods (Banfield and Raftery 1993), the grid-based methods (Wang et al. 1997; Sheikholeslami et al. 1998), the neural network methods (Kohonen 1982), the fuzzy sets methods (Bezdek 1980; Leung 1984), and the evolutionary methods (Al-Sultan and Khan 1996). The graph theoretic methods

often convert the clustering problem into a combinatorial optimization problem that is solved by graph algorithms or heuristic procedures. The density-based methods generally assume a mixture of distributions, with each cluster belonging to a specific distribution, for the data. Their purpose is to identify the clusters and the associated parameters. The grid-based methods impose a grid data structure on the data space in order to make density-based clustering more efficient. They however suffer from the curse of dimensionality as the number of cells in the grid increases. Neural network models generally perform clustering through a learning process. The self-organizing map, for example, can be treated as an on-line version of k-means with competitive learning. The fuzzy sets methods solve clustering problems where an object can belong to multiple clusters with different degrees of membership. The fuzzy c-means algorithm and fuzzy graph method are typical examples. The evolutionary methods are stochastic multi-point search algorithms that can be employed to solve clustering problems involving optimization. The basic principle is to devise an evolutionary strategy so that global optimal clustering can be obtained by evolving a population of clustering structures with some evolutionary operators. To achieve good quality clustering, hybrid approaches are often used in applications. In any case, all of these methods generate either the hierarchical or partitioning clustering. They can, in a sense, be fitted under either one of the frameworks.

Due to the complexity and size of the spatial databases, clustering methods should be efficient in high dimensional space (though spatial clustering is often of low dimensions), explicit in the consideration of scale, insensitive to large amount of noise, capable of identifying useful outliers, insensitive to initialization, effective in handling multiple data types, independent to a priori or domain specific knowledge (except for application specific data mining), and able to detect structures of irregular shapes. Conventional clustering algorithms often fail to fulfill these requirements. Whilst it is difficult to develop an ideal method that can meet all of these requirements, it is important to construct algorithms so that they can entertain them as much as possible. Since each method has certain assumptions about the data, it is generally impossible to determine the best clustering algorithm across all circumstances. An algorithm may be best for one problem or data set but may not perform as well for another problem or data set. A thorough understanding of the problem that needs to be solved is the first step towards the selection of the appropriate algorithm.

In the remaining part of this chapter, a detailed examination of some clustering methods that we, with the view of satisfying some of the requirements specified above, have developed to solve particular classes of clustering problems over the years. In Sect. 2.2, scale space filtering is introduced as a method of hierarchical clustering for the discovery of natural clusters in spatial data. Incorporation of scale and treatment of noise, which are essential in spatial data analysis, are explicitly dealt with in the discussion. In Sect. 2.3, fuzzy relational data clustering is described as a method of partitioning clustering. The emphasis is again on the introduction of scale and robustness against noise. Similar to scale space filtering in hierarchical clustering, unidimensional scaling examined in Sect. 2.4 attempts to provide an

answer to the issues of sensitivity to initialization, presupposition of a cluster number, and difficulty of solving global optimization problem commonly encountered in partitioning clustering. To solve the problem of mixture distributions in highly noisy environment, a method of mixture decomposition clustering is introduced in Sect. 2.5 to discover natural clusters in spatial data. In Sect. 2.6, the concept of convex hull is introduced to detect clusters in exploratory spatial data analysis.

2.2 Discovery of Clustering in Space by Scale Space Filtering

In pattern recognition and image processing, human eyes seem to possess a singular aptitude to group objects and find important structures in an efficient and effective way. Coding of continuities that occur in natural images was a main research area of the Gestalt school in psychology in the early twentieth century. With respect to spatial data mining, one can argue that continuity in scale/resolution in natural images is analogous to continuity in space. Partitioning of spatial structures in scale is a fundamental property of our visual system. Thus a clustering algorithm simulating our visual processing may facilitate the discovery of natural clusters in spatial databases in general and images in particular.

Based on this view, Leung et al. (2000a) propose a scale space filtering approach to clustering. In this approach, a data set is considered as an image with each datum being a light point attached with a uniform luminous flux. As the image is blurred, each datum becomes a light blob. Throughout the blurring process, smaller blobs merge into larger ones until the whole image contains only one light blob at a low enough level of resolution. If each blob is equated to a cluster, the above blurring process will generate a hierarchical clustering with resolution being the height of a dendrogram. The blurring process is described by scale space filtering which models the blurring effect of lateral retinal interconnection through the Gaussian filtering of a digital image (Witkin 1983, 1984; Koenderink 1984; Babaud et al. 1986; Hummel and Moniot 1989). The theory in fact sheds light on the way we cluster data, regardless of whether they are digital images or raw data. It also renders a biological perspective on data clustering.

The proposed approach has several advantages. (1) The algorithms thus derived are computationally stable and insensitive to initialization. They are totally free from solving difficult global optimization problems. (2) It facilitates the formulation of new cluster validity checks and gives the final clustering a significant degree of robustness to noise in the data and change in scale. (3) It is more robust where hyper-ellipsoidal partitions may not be assumed. (4) It is suitable for the preservation of the structure and integrity of the outliers, peculiarities in space, which should not be filtered out as noise in the clustering process. (5) The patterns of clustering are highly consistent with the perception of human eyes. (6) It provides a unified generalization of the scale-related clustering algorithms derived in various fields.

Scale space theory is first described in brief in the discussion to follow. It is then extended to solve problems in data clustering.

2.2.1 On Scale Space Theory for Hierarchical Clustering

Consider a two-dimensional image given by a continuous mapping $p(\mathbf{x}) : \mathbf{R}^2 \rightarrow \mathbf{R}$. In scale space theory, $p(\mathbf{x})$ is embedded into a continuous family $P(\mathbf{x}, \sigma)$ of gradually smoother versions of it. The original image corresponds to the scale $\sigma = 0$ and increasing the scale should simplify the image without creating spurious structures. If there are no prior assumptions which are specific to the scene, then it is proven that one can blur the image in a unique and sensible way in which $P(\mathbf{x}, \sigma)$ is the convolution of $p(\mathbf{x})$ with the Gaussian kernel, i.e.,

$$P(\mathbf{x}, \sigma) = p(\mathbf{x}) * g(\mathbf{x}, \sigma) = \int p(\mathbf{x} - \mathbf{y}) \frac{1}{(\sigma^2 2\pi)} e^{-\frac{\|\mathbf{y}\|^2}{2\sigma^2}} d\mathbf{y}, \quad (2.1)$$

where $g(\mathbf{x}, \sigma)$ is the Gaussian function $g(\mathbf{x}, \sigma) = \frac{1}{(\sigma\sqrt{2\pi})^2} e^{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}}$, σ is the scale parameter, (\mathbf{x}, σ) -plane is the scale space and $P(\mathbf{x}, \sigma)$ is the scale space image.

For each maximum $\mathbf{y} \in \mathbf{R}^2$ of $p(\mathbf{x})$, we define the corresponding light blob being a region specified as follows:

$$B_{\mathbf{y}} = \left\{ \mathbf{x}_0 \in \mathbf{R}^2 : \lim_{t \rightarrow \infty} \mathbf{x}(t, \mathbf{x}_0) = \mathbf{y} \right\}, \quad (2.2)$$

where $\mathbf{x}(t, \mathbf{x}_0)$ is the solution of the gradient dynamic system

$$\begin{cases} \frac{d\mathbf{x}}{dt} = \nabla_{\mathbf{x}} p(\mathbf{x}) \\ \mathbf{x}(0) = \mathbf{x}_0. \end{cases} \quad (2.3)$$

In what follows, \mathbf{y} is referred to as the *blob center* of $B_{\mathbf{y}}$. All blobs in an image produce a partition of \mathbf{R}^2 with each point belonging to a unique blob except the boundary points.

Let $p(\mathbf{x}) = g(\mathbf{x}, \sigma)$, which contains only one blob for $\sigma > 0$. As $\sigma \rightarrow 0$, this blob concentrates on a light point defined as

$$\delta(\mathbf{x}) = \lim_{\sigma \rightarrow 0} g(\mathbf{x}, \sigma) = \frac{1}{(\sigma\sqrt{2\pi})^2} e^{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}}. \quad (2.4)$$

Mathematically, such a function is called a δ function or a generalized function.

A light point at $\mathbf{x}_0 \in \mathbf{R}^2$ in an image is defined as a δ function situated at \mathbf{x}_0 , i.e., $\delta(\mathbf{x} - \mathbf{x}_0)$, which satisfies

$$g(\mathbf{x}, \sigma) * \delta(\mathbf{x} - \mathbf{x}_0) = g(\mathbf{x} - \mathbf{x}_0, \sigma), \quad (2.5)$$

where g is the Gaussian function. From (2.5) we can see that if we blur a light point, it becomes a light blob again.

In our everyday visual experience, blurring of an image leads to the erosion of structure: small blobs always merge into large ones and new ones are never created. Therefore, the blobs obtained for images $P(\mathbf{x}, \sigma)$ at different scales form a hierarchical structure: each blob has its own survival range of scale, and large blobs are made up of small blobs. The survival range for a blob is characterized by the scale at which the blob is formed and the scale at which the blob merges with others. Each blob manifests itself purely as a simple blob within its survival range of scale.

Such blurring process can be related with the process of clustering. If $p(\mathbf{x})$ is a probability density function from which the data set is generated, then each blob is a connected region containing a relatively high density probability separated from other blobs by a boundary with relatively low density probability. Therefore, each blob is a cluster, and all blobs together produce a partition of a data space which provides a clustering for the data set with known distribution $p(\mathbf{x})$.

For a given data set $\mathbf{X} = \{\mathbf{x}_i \in \mathbf{R}^2 : i = 1, \dots, N\}$, the empirical distribution for the data set \mathbf{X} can be expressed as

$$\hat{p}_{emp}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{x} - \mathbf{x}_i). \quad (2.6)$$

The image corresponding to $\hat{p}_{emp}(\mathbf{x})$ consists of a set of light points situated at the data set, just like a scattergram of the data set. When we blur this image, we get a family of smooth images $P(\mathbf{x}, \sigma)$ represented as follows:

$$P(\mathbf{x}, \sigma) = \frac{1}{N} \sum_{i=1}^N \frac{1}{(\sigma\sqrt{2\pi})^2} e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}}. \quad (2.7)$$

The family $P(\mathbf{x}, \sigma)$ can be considered as the Parzen estimation with Gaussian window function. At each given scale σ , the scale space image $P(\mathbf{x}, \sigma)$ is a smooth distribution function so that the blobs and their centers can be determined by analyzing the limit of the solution $\mathbf{x}(t, \mathbf{x}_0)$ of the following differential equation:

$$\begin{cases} \frac{d\mathbf{x}}{dt} = \nabla_{\mathbf{x}} P(\mathbf{x}, \sigma) = \frac{1}{\sigma^2 N} \sum_{i=1}^N \frac{(\mathbf{x}_i - \mathbf{x})}{(\sigma\sqrt{2\pi})^2} e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}} \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \quad (2.8)$$

Remark 2.1. Treatment of Noise. When a distribution $p(\mathbf{x})$ is known but contains noise or is indifferntiable, we can also use scale space filtering method to erase the spurious maxima generated by the noise. In this case, the scale-space image is

$$P(\mathbf{x}, \sigma) = p(\mathbf{x}) * g(\mathbf{x}, \sigma) = \int \frac{p(\mathbf{y})}{(\sigma\sqrt{2\pi})^2} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} d\mathbf{y}, \quad (2.9)$$

and, the corresponding gradient dynamical system is given by

$$\begin{cases} \frac{d\mathbf{x}}{dt} = \nabla_{\mathbf{x}} P(\mathbf{x}, \sigma) = \int \frac{p(\mathbf{y})(\mathbf{y} - \mathbf{x})}{(\sigma\sqrt{2\pi})^2 \sigma^2} e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} d\mathbf{y} \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases}. \quad (2.10)$$

When the noise in $p(\mathbf{x})$ is an independent white noise process, (2.9) provides an optimal estimate of the real distribution.

Thus, instead of clustering the data by the underlying distribution $p(\mathbf{x})$, the scale space method clusters data according to a gradient dynamic system generated by $P(\mathbf{x}, \sigma)$ for each $\sigma > 0$. By considering the data points falling into the same blob as a cluster, the blobs of $P(\mathbf{x}, \sigma)$ at a given scale produce a pattern of clustering. In this way, each data point is deterministically assigned to a cluster via the differential gradient dynamical equation in (2.8) or (2.10), and the method thus renders a hard clustering result. As we change the scale, we get a hierarchical clustering. A detailed description of the clustering procedure and the corresponding numerical implementations are given in the discussion to follow.

2.2.2 Hierarchical Clustering in Scale Space

In scale space clustering, we use the maxima of $P(\mathbf{x}, \sigma)$ with respect to x as the description primitives. Our discussion is based on the following theorem:

Theorem 2.1. *For almost all data sets, we have: (1) 0 is a regular value of $\nabla_{\mathbf{x}} P(\mathbf{x}, \sigma)$, (2) as $\sigma \rightarrow 0$, the clustering obtained for $P(\mathbf{x}, \sigma)$ with $\sigma > 0$ induces a clustering at $\sigma = 0$ in which each datum is a cluster and the corresponding partition is a Voronoi tessellation, i.e., each point in the scale space belongs to its nearest-neighbor datum, and (3) as σ increases from $\sigma = 0$, there are N maximal curves in the scale space with each of them starting from a datum of the data set.*

We know that the maxima of $P(\mathbf{x}, \sigma)$ are the points satisfying

$$\nabla_{\mathbf{x}} P(\mathbf{x}, \sigma) = 0. \quad (2.11)$$

Therefore, 0 being a regular value of $\nabla_{\mathbf{x}}P(\mathbf{x}, \sigma)$ means that: (1) all maxima form simple curves in the scale space, and (2) we can follow these curves by numerical continuation method (Allgower and Georg 1990).

Remark 2.2. Initialization. In terms of the criterion for cluster centers (i.e., maximizing $P(\mathbf{x}, \sigma)$), there is a unique solution at small scale with N centers (each maximum is the blob center of the corresponding cluster) and hence the method is independent of initialization.

2.2.2.1 Nested Hierarchical Clustering

The construction procedure of a nested hierarchical clustering based on the scale-space image is as follows:

1. At scale $\sigma = 0$, each datum is considered as a blob center whose associated data point is itself.
2. As σ increases continuously, if the blob center of a cluster moves continuously along the maximal curve and no other blob center is siphoned into its blob, then we consider that the cluster has not changed and only its blob center moves along the maximal curve. If an existing blob center disappears at a singular scale and falls into another blob, then the two blobs merge into one blob and a new cluster is formed with the associated data points being the union of those of the original clusters.
3. Increase the scale until the whole data set becomes one single cluster. This stopping rule is well-defined because we have only one blob in the data space when scale is large enough.

A hierarchical clustering dendrogram can thus be constructed with scale as height. Such a hierarchical clustering dendrogram may be viewed as a regional tree with each of its node being a region so that data falling within the same region form a cluster. Therefore, the nested hierarchical clustering thus constructed provides a partition of the data space. In one dimensional case, such a regional tree is in fact an interval tree.

2.2.2.2 Non-Nested Hierarchical Clustering

Nested hierarchical clustering has been criticized for the fact that once a cluster is formed, its members cannot be separated subsequently. Nevertheless, we can construct a non-nested hierarchical clustering which removes such a problem. In a non-nested hierarchical clustering, we partition the data set $\mathbf{X} = \{\mathbf{x}\}$ at a given scale by assigning a membership to each datum $\mathbf{x}_0 \in \mathbf{X}$ according to (2.2). This process is similar to the way we perceive the data set at a given distance or a given resolution. Clusters obtained at different scales are related to each other by the cluster center lines. As σ changes, a non-nested hierarchical clustering is obtained

since each datum may change its membership under such a scheme. The evolution of the cluster centers in the scale-space image may be considered as a form of dendrogram. By Theorem 2.1 we know that 0 is a regular value of $\nabla_{\mathbf{x}}P(\mathbf{x}, \sigma)$ for almost all data sets. This means that cluster centers form simple curves in the scale space which can be computed through the path which follows the solutions of the equation $\nabla_{\mathbf{x}}P(\mathbf{x}, \sigma) = 0$ by the numerical continuation method.

Non-nested hierarchical clustering is more consistent with that obtained by human eyes at different distances or different resolutions, while nested hierarchical clustering has more elegant hierarchical structure.

2.2.2.3 Numerical Solution for Gradient Dynamic System

In the proposed clustering method, clusters are characterized by the maxima of $P(\mathbf{x}, \sigma)$ and the membership of each datum is determined by the gradient dynamical system in (2.8) or (2.10). Since the solution of the initial value problem of either equation cannot be found analytically, some numerical methods must be used. If the Euler difference method is used, the solution of (2.8) or (2.10), $\mathbf{x}(t, \mathbf{x}_0)$, is then approximated by the sequence $\{\mathbf{x}(n)\}$ generated in one of the following difference equations:

$$\begin{cases} \mathbf{x}(n+1) = \mathbf{x}(n) + h \nabla_{\mathbf{x}} p(\mathbf{x}(n), \sigma) = \mathbf{x}(n) + \frac{h}{\sigma^2 N} \sum_{i=1}^N \frac{(\mathbf{x}_i - \mathbf{x}(n))}{(\sigma \sqrt{2\pi})^2} e^{-\frac{\|\mathbf{x}(n) - \mathbf{x}_i\|^2}{2\sigma^2}}, \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases}, \quad (2.12)$$

or,

$$\begin{cases} \mathbf{x}(n+1) = \mathbf{x}(n) + \frac{h}{\sigma^2} \int p(\mathbf{y})(\mathbf{y} - \mathbf{x}(n)) e^{-\frac{\|\mathbf{x}(n) - \mathbf{y}\|^2}{2\sigma^2}} d\mathbf{y} \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases} \quad (2.13)$$

where h is the step length.

If the magnitude of P is scaled by the logarithmic function, the corresponding gradient dynamical system of (2.8) and (2.10) becomes

$$\frac{d\mathbf{x}}{dt} = \frac{1}{\sigma^2} \frac{\sum_{i=1}^N (\mathbf{x}_i - \mathbf{x}) e^{-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}}}{\sum_{i=1}^N e^{-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}}}, \quad (2.14)$$

and,

$$\frac{d\mathbf{x}}{dt} = \frac{1}{\sigma^2} \frac{\int p(\mathbf{y})(\mathbf{y} - \mathbf{x}) e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} d\mathbf{y}}{\int p(\mathbf{y}) e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} d\mathbf{y}}, \quad (2.15)$$

and the discrete approximations to (2.12) and (2.13) then become

$$\mathbf{x}(n+1) = \mathbf{x}(n) + \frac{h}{\sigma^2} \frac{\sum_{i=1}^N (\mathbf{x}_i - \mathbf{x}(n)) e^{-\frac{\|\mathbf{x}(n)-\mathbf{x}_i\|^2}{2\sigma^2}}}{\sum_{i=1}^N e^{-\frac{\|\mathbf{x}(n)-\mathbf{x}_i\|^2}{2\sigma^2}}}, \quad (2.16)$$

or,

$$\mathbf{x}(n+1) = \mathbf{x}(n) + \frac{h}{\sigma^2} \frac{\int p(\mathbf{y})(\mathbf{y} - \mathbf{x}(n)) e^{-\frac{\|\mathbf{x}(n)-\mathbf{y}\|^2}{2\sigma^2}} d\mathbf{y}}{\int p(\mathbf{y}) e^{-\frac{\|\mathbf{x}(n)-\mathbf{y}\|^2}{2\sigma^2}} d\mathbf{y}}. \quad (2.17)$$

Setting the step length $h = \sigma^2$ in (2.17), we get

$$\mathbf{x}(n+1) = \frac{\sum_{i=1}^N \mathbf{x}_i e^{-\frac{\|\mathbf{x}(n)-\mathbf{x}_i\|^2}{2\sigma^2}}}{\sum_{i=1}^N e^{-\frac{\|\mathbf{x}(n)-\mathbf{x}_i\|^2}{2\sigma^2}}}. \quad (2.18)$$

Such iteration can be interpreted as iterative local centroid estimation (Wilson and Spann 1990; Linderberg 1990).

When the size of the data set is large or the data are given in a serial form, we can use the stochastic gradient descent algorithm to search the blob center and determine the memberships of the data. The purpose is to find the maximum of $P(\mathbf{x}, \sigma)$ which can be represented as

$$P(\mathbf{x}, \sigma) = E \left[e^{-\frac{\|\mathbf{x}(n)-\mathbf{x}_j\|^2}{2\sigma^2}} \right], \quad (2.19)$$

where $E[\cdot]$ is the expectation of the density of the data set \mathbf{y} . By the theory of stochastic gradient descent algorithm, the blob center of a datum \mathbf{x}_0 can be obtained by the following iteration initialized at \mathbf{x}_0 :

$$\mathbf{x}(n+1) = \mathbf{x}(n) + h^{(n)} (\mathbf{x}^{(n)} - \mathbf{x}(n)) e^{-\frac{\|\mathbf{x}(n)-\mathbf{x}^{(n)}\|^2}{2\sigma^2}}, \quad (2.20)$$

where $\mathbf{x}^{(n)}$ is the n th randomly chosen member of X or the n th datum generated according to the distribution $p(\mathbf{x})$ to be presented to the algorithm, and $h^{(n)}$ is the adaptive step length chosen as:

$$h^{(n)} = \frac{1}{1+n}. \quad (2.21)$$

The datum \mathbf{x}_0 is then associated with a center \mathbf{x}^* if $\mathbf{x}(n)$ initialized from \mathbf{x}_0 converges to \mathbf{x}^* . In practice, $\mathbf{x}(n+1)$ is defined as a blob center if $\|\mathbf{x}(n+1) - \mathbf{x}(n)\| < \varepsilon$ or $\|\nabla_{\mathbf{x}}p(\mathbf{x}(n+1))\| < \varepsilon$, where ε is a small positive value which may vary with problems. If two centers \mathbf{x}_1 and \mathbf{x}_2 satisfy the condition $\|\mathbf{x}_1 - \mathbf{x}_2\| < \varepsilon$, then they are considered as one blob center.

To implement the proposed hierarchical clustering, we can use the path-following algorithm to trace the blob centers along the maximal curves. When a singular scale at which a blob center disappears is encountered, the new blob center is obtained by solving (2.8) or (2.10) with initial value $\mathbf{x}_0 = \mathbf{x}^*$. The new blob center is then followed by the path-following algorithm again. Alternatively, we can use the discretization of scale and an iterative scheme which works as follows:

2.2.2.4 Nested Hierarchical Algorithm

- Step 1. Given a sequence of scales $\sigma_0, \sigma_1, \dots$ with $\sigma_0 = 0$. At $\sigma_0 = 0$ each datum is a cluster and its blob center is itself. Let $i = 1$.
- Step 2. Find the new blob center at σ_i for each blob center obtained at scale σ_{i-1} by one of the iterative schemes in (2.12) to (2.18). Merge the clusters whose blob centers arrive at the same blob center into a new cluster.
- Step 3. If there are more than two clusters, let $i := i+1$, go to Step 2.
- Step 4. Stop when there is only one cluster.

2.2.2.5 Non-Nested Hierarchical Algorithm

- Step 1. Given a sequence of scales $\sigma_0, \sigma_1, \dots$ with $\sigma_0 = 0$. At $\sigma_0 = 0$ each datum is a cluster and its blob center is itself. Let $i = 1$.
- Step 2. Cluster the data at σ_i . Find the new blob center at σ_i for each blob center obtained at scale σ_{i-1} by one of the iterative schemes in (2.12) to (2.18). If two new blob centers arrive at the same point, then the old clusters disappear and a new cluster is formed.
- Step 3. If there are more than two clusters, let $i := i + 1$, go to Step 2.
- Step 4. Stop when there is only one cluster.

Remark 2.3. Computation for Large Data Sets. When the size of the data set is very large, we can substitute each datum in the iterative scheme in (2.12)–(2.18) with its

blob center and σ_i with $\sigma_i - \sigma_{i-1}$ in step 2 to reduce the computational cost of the above algorithm. In this case, (2.18) becomes

$$\mathbf{x}(n+1) = \frac{\sum_{j=1}^{N_i} k_j \mathbf{p}_j e^{-\frac{\|\mathbf{x}(n) - \mathbf{p}_j\|^2}{2\sigma^2}}}{\sum_{j=1}^{N_i} k_j e^{-\frac{\|\mathbf{x}(n) - \mathbf{p}_j\|^2}{2\sigma^2}}}, \quad (2.22)$$

where \mathbf{p}_j is blob center j obtained at scale σ_i , N_i is the number of \mathbf{p}_j , k_j is the number of data points in the blob whose center is \mathbf{p}_j and $\sigma = \sigma_i - \sigma_{i-1}$. Since N_i is usually much smaller than N , so the computational cost can be reduced significantly. In practical applications, σ_i should increase according to

$$\sigma_i - \sigma_{i-1} = k\sigma_{i-1}. \quad (2.23)$$

This comes from the requirement of accuracy and stability of the representation, as proved in Koenderink (1984). In psychophysics, Weber's law says that the minimal size of the difference ΔI in stimulus intensity which can be sensed is related to the magnitude of standard stimulus intensity I by $\Delta I = kI$, where k is a constant called Weber fraction. Therefore, psychophysical experimental results may be used to propose a low bound for k in the algorithms since we cannot sense the difference between two images $p(\mathbf{x}, \sigma_{i-1})$ and $p(\mathbf{x}, \sigma_i)$ when k is less than its Weber fraction. For instance, $k = 0.029$ in (2.23) is enough in one dimensional applications because scale σ is the window length in the scale space and the Weber fraction for line length is 0.029 (Coren et al. 1994).

2.2.3 Cluster Validity Check

Cluster validity is a vexing but very important problem in cluster analysis because each clustering algorithm always finds clusters, no matter they are genuine or not, even if the data set is entirely random. While many clustering algorithms can be applied to a given problem, there is in general no guarantee that two different algorithms will produce consistent answers. They particularly do not provide answers to the following questions: (1) Do the data exhibit a predisposition to cluster? (2) How many clusters are present in the data? (3) Are the clusters real or merely artifacts of the algorithms? (4) Which partition or which individual cluster is valid? Therefore, cluster validity check should be an essential requirement of any algorithm. Besides some procedures in statistics (Theodoridis and Koutroubas 1999), one widely used strategy is to employ visual processing to examine distributions on each separate variable by ways such as histograms, nonparametric density estimates or plots of each pair of variables using scattergram. However, there is no

theoretical basis for such visualization. Another strategy is to produce clustering algorithms based directly on the laws of psychology of form perception. Zahn (1971) has proposed a clustering algorithm based on the laws of Gestalt psychology of form perception. The algorithm is a graphical one which is based on the minimal spanning tree and attempts to mechanize the Gestalt law of *proximity* which says that perceptual organization favors groupings representing smaller inter-point distance. Zahn's algorithm has a strong influence on cluster analysis. Many algorithms have been developed on the basis of similar ideas. However, Zahn's algorithm is derived from Gestalt psychology laws in a heuristic way since Gestalt laws cannot be represented in an accurate computational model. This inaccuracy makes it difficult to establish a formal and efficient cluster validity check.

In scale space filtering, the questions are tackled on the basis of human visual experience: the real cluster should be perceivable over a wide range of scales. Thus, the notion of *lifetime* of a cluster is employed as its validity criterion: A cluster with longer lifetime is more valid than a cluster with shorter lifetime.

In Leung et al. (2000a), the lifetime of a cluster is used to test the "goodness" of a cluster, and the lifetime of a clustering is used to determine the number of clusters in a specific pattern of clustering.

Definition 2.1. *Lifetime of a cluster is defined as the range of logarithmic scales over which the cluster survives, i.e., the logarithmic difference between the point when the cluster is formed and the point when the cluster is absorbed into or merged with other clusters.*

Each pattern of clustering in a non-nested hierarchical clustering only consists of clusters which are formed at the same scale. A pattern of clustering in a nested hierarchical clustering, however, is a partition of the data set \mathbf{X} which may consist of clusters obtained at the same scale or at different scales. In what follows, we define the lifetime for these two kinds of clustering's.

Definition 2.2. *Let $\pi(\sigma)$ be the number of clusters in a clustering achieved at a given scale σ . Suppose \mathbf{C}_σ is a clustering obtained at σ with $\pi(\sigma) = m$. The σ -lifetime of \mathbf{C}_σ is defined as the supremum of the logarithmic difference between two scales within which $\pi(\sigma) = m$.*

Definition 2.3. *Suppose a clustering \mathbf{C} in a hierarchical clustering contains K clusters $\{\mathbf{C}_1, \dots, \mathbf{C}_K\}$. Denote the number of data points in \mathbf{C}_i by $|\mathbf{C}_i|$ and the lifetime of \mathbf{C}_i by l_i . Then the mean lifetime of all clusters in clustering \mathbf{C} is defined as*

$$\sum_{i=1}^K l_i \frac{|\mathbf{C}_i|}{|\mathbf{X}|}. \quad (2.24)$$

The lifetime of clustering \mathbf{C} is the mean lifetime of all of its clusters. If a cluster \mathbf{C}_i is further divided into K_i sub-clusters $\{\mathbf{C}_{i_1}, \dots, \mathbf{C}_{i_{K_i}}\}$, and the lifetime of \mathbf{C}_{i_j} is denoted by l_{i_j} , then the mean lifetime of all its sub-clusters is defined as

$$\sum_{j=1}^{K_i} l_{ij} \frac{|\mathbf{C}_{ij}|}{|\mathbf{C}_i|}. \quad (2.25)$$

The use of logarithmic scale in the above definitions is based on the experimental tests in Roberts (1997) which show that (σ) decays with scale σ according to

$$\pi(\sigma) = ce^{-\beta\sigma} \quad (2.26)$$

if the data are uniformly distributed, where β is a positive constant related to the dimensionality of the data space. If a data structure exists, then $\pi(\sigma)$ is a constant over a range of scales. So the stability of $\pi(\sigma)$ can be used as a criterion to test whether the data tend to cluster, i.e., have a structure. However, β is unknown and $\pi(\sigma)$ is only allowed to take integers. From (2.26) we can see that even for a uniformly distributed data set, if β is small, $\pi(\sigma)$ will then be a constant over a wide range of scales for a small $\pi(\sigma)$. If β is large, then $\pi(\sigma)$ will also be a constant over a wide range of scales for a large σ . This makes it difficult to find the structure in the $\pi(\sigma)$ plot. However, if the data are uniformly distributed and we rescale σ by a new parameter k such that the number of clusters in the clustering obtained at the new parameter k , denoted by $\pi(k)$, decays linearly with respect to k , i.e.,

$$\pi(k) = \pi(0) - k, \quad (2.27)$$

we can easily find the structure in the plot of $\pi(k)$. The reason is that it is much simpler to test whether $\pi(k)$ decays linearly with respect to k than to test whether $\pi(k)$ decays according to (2.26) in which an unknown parameter β is involved.

Under the assumption that $\pi(k)$ decays linearly with respect to k , the relationship of k and σ can be derived as follows:

Suppose σ relates to k through a function $\sigma(k)$. Then we have

$$\pi(k) = \pi(\sigma(k)) = ce^{-\beta\sigma(k)}. \quad (2.28)$$

Under the assumption that $\pi(k)$ decays linearly with respect to k , see (2.27), we have

$$\frac{d\pi(k)}{dk} = -1. \quad (2.29)$$

From (2.26), we obtain

$$\frac{d\pi(k)}{dk} = -c\beta e^{-\beta\sigma\frac{d\sigma}{dk}}. \quad (2.30)$$

Equations (2.29) and (2.30) imply that the new parameter k should satisfy

$$\frac{d\sigma}{dk} = \frac{1}{c\beta} e^{\beta\sigma}. \quad (2.31)$$

Solving this differential equation, we get

$$k = c(1 - e^{-\beta\sigma}). \quad (2.32)$$

Such a scaling is an ideal one, but it contains a parameter β which is usually unknown. In practice, we take the approximation $\beta/e^{\beta\sigma} = \beta/(1 + \beta\sigma + \dots) \approx 1/\sigma$ in (2.30) which does not contain the unknown parameter β , and this leads to the logarithmic scale

$$k = c \log \frac{\sigma}{\varepsilon}, \quad (2.33)$$

where ε is a positive constant.

The term k defined in (2.33) is called the *sensation intensity* under the Fechner's Law (Coren et al. 1994). In terms of the new parameter k , lifetime should be measured by the logarithmic scale of σ .

Once a partition has been established to be valid, a natural question that follows is “How good are the individual clusters?” The first measure of “goodness” of a cluster is naturally its lifetime: a good cluster should have a long lifetime. Associated measures are compactness and isolation of a cluster. Intuitively, a cluster is good if the distances between the data inside the cluster are small and those outside are large. *Compactness* and *isolation* of a cluster are two measures suggested for the identification of good clusters (Leung et al. 2000a). For a cluster C_i , the measures are defined as follows:

$$isolation = \frac{\sum_{\mathbf{x} \in C_i} e^{-\|\mathbf{x} - \mathbf{p}_i\|^2 / 2\sigma^2}}{\sum_{\mathbf{x}} e^{-\|\mathbf{x} - \mathbf{p}_i\|^2 / 2\sigma^2}}, \quad (2.34)$$

$$compactness = \frac{\sum_{\mathbf{x} \in C_i} e^{-\|\mathbf{x} - \mathbf{p}_i\|^2 / 2\sigma^2}}{\sum_{\mathbf{x} \in C_i} \sum_j e^{-\|\mathbf{x} - \mathbf{p}_j\|^2 / 2\sigma^2}}, \quad (2.35)$$

where \mathbf{p}_i is the blob center of cluster C_i . For a good cluster, the compactness and isolation are close to one. This measure is dependent on the scale and will be used to find the optimal scale at which the clustering achieved by non-nested hierarchical clustering is good.

Therefore, *lifetime*, *compactness* and *isolation* are three measures that can be employed to check the validity of a good cluster. A genuine cluster should be compact, isolated and have a relatively long life time. A natural clustering should be the one which contains a certain number of good clusters with high overall isolation and compactness, and stays relatively long in the scale space.

Remark 2.4. A data set invariably contains noisy data points which may be genuine outliers that carry crucial information. How to detect observations which appear to be markedly different from the rest of a data set is an important problem in many diagnostic or monitoring systems (Hawkins 1980; Barnett and Lewis 1994). Successful detection of spatial outliers is important in the discovery of peculiar patterns with significant spatial implications. In scale space clustering, we can use the number of data points in a cluster C_i and the lifetime of C_i to decide whether or not C_i is a genuine outlier. If C_i contains a small number of data and survives a long time, then we say that C_i is an outlier, otherwise, C_i is a normal cluster. Therefore, we can use the measure

$$outlierness_i = \frac{\text{life time of } C_i}{\text{number of data in } C_i} \quad (2.36)$$

to test for outliers. It means that an outlier is a well isolated group with small number of data in a large scale range. Since the method treats the data point as light point, each outlier should be a stable cluster in quite a large scale range. That is to say, an outlier generally exhibits a high degree of “outlierness.” A threshold may be used to exclude outliers that are non-essential in data clustering.

2.2.4 Clustering Selection Rules

Hierarchical clustering provides us with a sequence of clustering’s. Several selection rules are proposed in Leung et al. (2000a) to choose a good clustering from the sequence of clustering’s in the hierarchy.

The first rule is based on the σ -lifetime of a clustering and it tries to find a scale at which the clustering achieved has long lifetime and high degree of compactness or isolation.

2.2.4.1 Rule I

1. Find the integer m such that the clustering obtained at σ with $\pi(\sigma) = m$ has the longest σ -lifetime.
2. (a) In nested hierarchical clustering, clusterings which satisfy $\pi(\sigma) = m$ are identical to each other, so we can get a unique clustering when m is obtained.
- (b) In non-nested hierarchical clustering, clusterings obtained at two scales σ_1 and σ_2 are usually different from each other even though the result $\pi(\sigma_1) = \pi(\sigma_2) = m$ is obtained. Therefore, we still need a method to find the right scale at which a good clustering can be achieved when m is fixed.

Define respectively the overall isolation and overall compactness for a clustering achieved at σ with $\pi(\sigma) = m$ as follows:

$$F^{(i)}(\sigma) = \left(\sum_i^m i^{th} \text{isolation} - m \right) \quad (2.37)$$

$$F^{(c)}(\sigma) = \left(\sum_i^m i^{th} \text{compactness} - m \right) \quad (2.38)$$

where the i -th *isolation* and i -th *compactness* are the isolation and compactness of the i -th cluster respectively. By maximizing $F^{(i)}$ or $F^{(c)}$ under the condition that $\pi(\sigma) = m$, we can get a σ at which a partition with maximal isolation or maximal compactness is achieved. In the general case, $\pi(\sigma) = m$ is held in an interval $[\sigma_1, \sigma_2]$. Therefore we can use the gradient descent method to optimize $F^{(i)}$ or $F^{(c)}$. The gradient is given by

$$\frac{df}{d\sigma} = \sum_{i=1}^m \nabla_{\mathbf{x}_i} F \frac{d\mathbf{x}_i}{d\sigma} \quad (2.39)$$

where F is $F^{(i)}$ or $F^{(c)}$, and \mathbf{x}_i is the center of the i -th cluster. Knowing that each cluster center \mathbf{x} is a maximal point of $p(\mathbf{x}, \sigma)$, the term $d\mathbf{x}_i/d\sigma$ can be obtained as

$$\frac{d\mathbf{x}}{d\sigma} = -[\nabla_{\mathbf{x}\mathbf{x}}P(\mathbf{x}, \sigma)]^{-1} \nabla_{\mathbf{x}\sigma}P(\mathbf{x}, \sigma). \quad (2.40)$$

Finally, we obtain a σ which is a minimal point of $F^{(i)}$ or $F^{(c)}$ and we consider that the clustering obtained at this scale is good.

The second selection rule is constructed to search for a clustering with the longest lifetime in nested hierarchical clustering. Let Ω be the set of all clustering's in a nested hierarchical clustering. For each clustering $\mathbf{p}_i \in \Omega$, its lifetime is denoted by $l_{\mathbf{p}_i}$. The aim of the second rule is to find a clustering \mathbf{p}_j such that

$$l_{\mathbf{p}_j} = \max_{\mathbf{p}_i \in \Omega} l_{\mathbf{p}_i}. \quad (2.41)$$

Since such problem is usually difficult to solve, several heuristic procedures may be used to obtain a solution. Leung et al (2000a) propose two greedy methods Rule II.1 (depth-first search) and Rule II.2 (breadth-first search) for such purpose.

The first procedure is similar to Witkin's "top-level description." It works as follows:

2.2.4.2 Rule II.1 (Maximization with Depth-First Search)

1. Initially, let \mathbf{P} be a clustering with the whole data set as a cluster. Assign 0 as the lifetime of this unique cluster.
2. Find a cluster C_k in \mathbf{P} whose lifetime is shorter than the mean lifetime of its children, and delete the cluster C_k from \mathbf{P} and add all children clusters of C_k into \mathbf{P} , i.e., the new clustering \mathbf{P} consists of the children clusters of C_k and other clusters except C_k . Repeat this process until the lifetime of each cluster in \mathbf{P} is longer than the mean lifetime of its own children.

Clustering obtained by this procedure is usually less complex, i.e., with small number of clusters.

The second procedure can also be considered as a ‘longest-lifetime-first’ procedure. It works as follows:

2.2.4.3 Rule II.2 (Maximization with Breadth-First Search)

1. Initialize \mathbf{U} to be an empty set. Let $\mathbf{C} = \{C_1, C_2, \dots, C_K\}$ be the set of all clusters in the hierarchical clustering.
2. Pick the element C_k in \mathbf{C} with the longest lifetime and put it into \mathbf{U} . Remove C_k and the clusters in \mathbf{C} that are either contained in or contain C_k until \mathbf{C} is empty. The number of elements in \mathbf{U} is the number of clusters and \mathbf{U} is the corresponding clustering.

2.2.5 Some Numerical Examples

The first example involves a two-dimensional data set with 250 data points generated by a five cluster Gaussian mixture model with different shapes. Figure 2.1a is the data plot and Fig. 2.1b is the $\pi(k)$ plot. From Fig. 2.1b, we can observe that $\pi(k)$ has an approximately linear decrease with scale k between $0 < k < 60$, where $k = c \log(\sigma/\varepsilon)$ with $\varepsilon = 0.1$ and $c = 1/\log(1.05)$. For $k > 60$, the hidden data structure appears and $\pi(k) = 5$ has the longest σ -lifetime. Figure 2.1c, d are respectively the overall isolation and overall compactness plots. $F^{(t)}$ and $F^{(c)}$ achieve their maxima at about $k = 67$ ($\sigma = 2.628$). At this scale, the clustering obtained by the non-nested hierarchical clustering algorithm is consistent with that obtained by the nested-hierarchical clustering algorithm (the corresponding clustering is shown in Fig. 2.2b). Figure 2.2a is the evolutionary plot of the blob centers obtained by the nested hierarchical algorithm. Figure 2.2b is the data partition obtained at different scales. It can be observed that the results obtained via the concept of σ -lifetime, isolation and compactness are consistent.

This is actually the solution for the cluster discovery problem (Fig. 1.1) raised in Chapter 1, section 1.5.

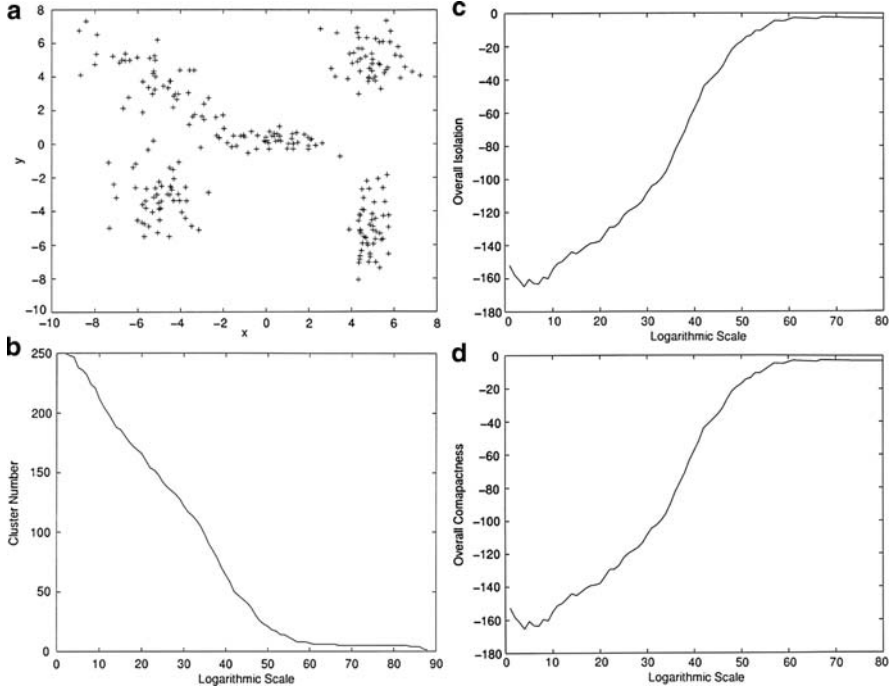


Fig. 2.1 A numerical example of scale space clustering (a) Plot of the data set. (b) Logarithmic-scale plot of the cluster number $\pi(k)$. (c) Logarithmic-scale plot of overall isolation. (d) Logarithmic-scale plot of overall compactness

For the sake of visualization, Fig. 2.3 depicts another two-dimensional data set with a hidden structure of $\pi(k) = 5$. At each scale we can generate the pseudo-color plot, the mesh plot and the contour plot of the scale space image. For example, Fig. 2.4a–c are respectively the pseudo-color plot, the mesh plot and the contour plot for $\sigma = 0.163$, and Fig. 2.5a–c are that for $\sigma = 1.868$. Apparently, the five clusters naturally settle in and form the natural clustering of the data at the appropriate scale.

2.2.6 Discovering Land Covers in Remotely Sensed Images

Leung et al. (2000a) apply the scale-space clustering algorithm to a real-life Landsat TM image to discover natural clusters (land covers) in multidimensional data.

It should be noted that if the data set $\mathbf{X} = \{\mathbf{x}_i \in \mathbf{R}^n : i = 1, \dots, N\}$ is in the space \mathbf{R}^n , then its empirical distribution is expressed as $\hat{p}_{emp}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{x} - \mathbf{x}_i)$. The scale space image of $\hat{p}_{emp}(\mathbf{x})$, $P(\mathbf{x}, \sigma)$, can be written as

$$P_{\mathbf{x}}(\mathbf{x}, \sigma) = \frac{1}{N} \sum_{i=1}^k \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^N e^{-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}}, \text{ which is the convolution of } \hat{p}_{emp}(\mathbf{x}) \text{ with}$$

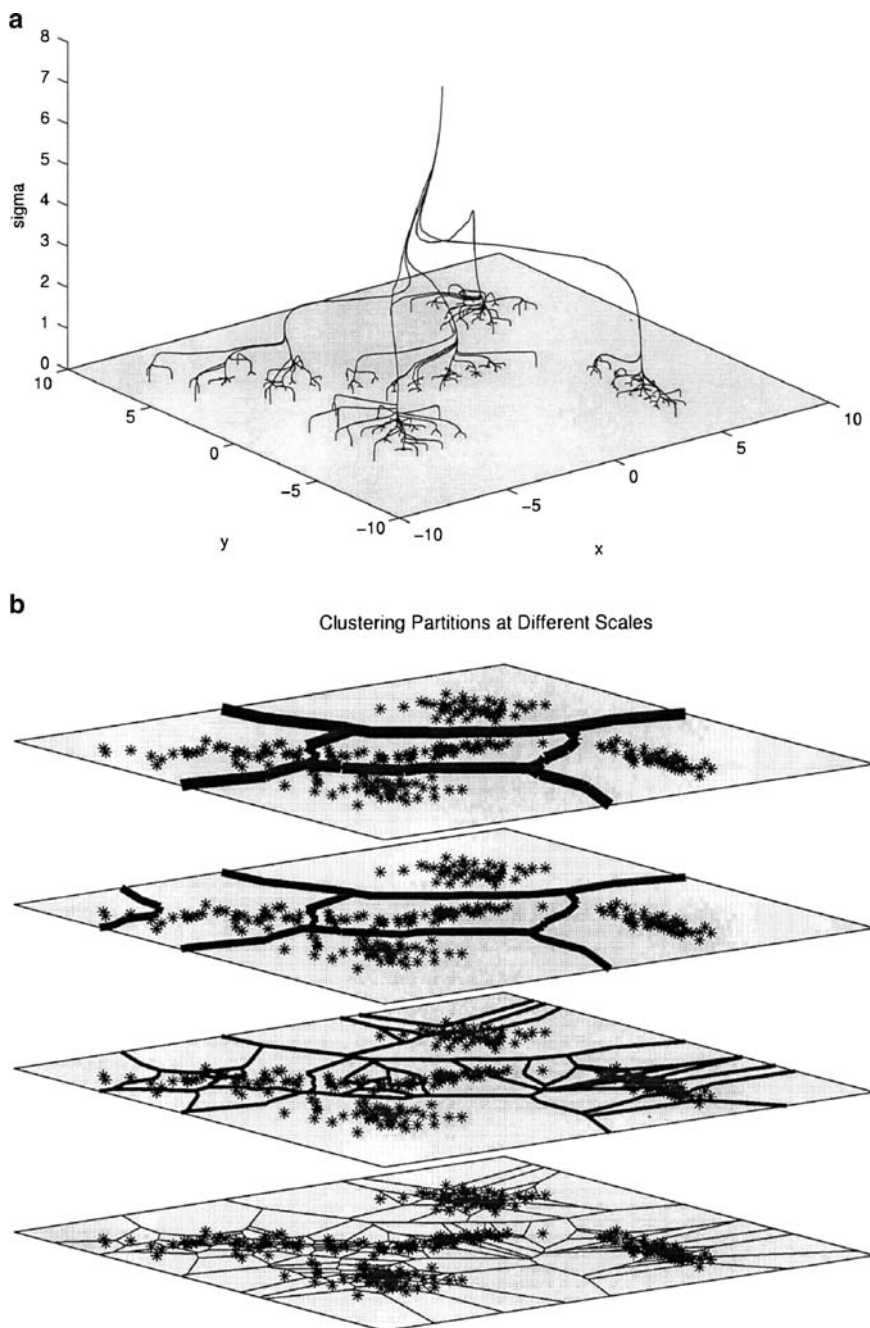


Fig. 2.2 Evolution plot of the scale space clustering in Fig. 2.1 (a) Evolutionary tree of cluster centers obtained by the algorithm. (b) The partition of the data space obtained by the nested hierarchical clustering algorithm at scales $\sigma_0 = 0$, $\sigma_1 = 0.99$, $\sigma_2 = 2.38$ and $\sigma_3 = 2.628$ (from bottom to top)

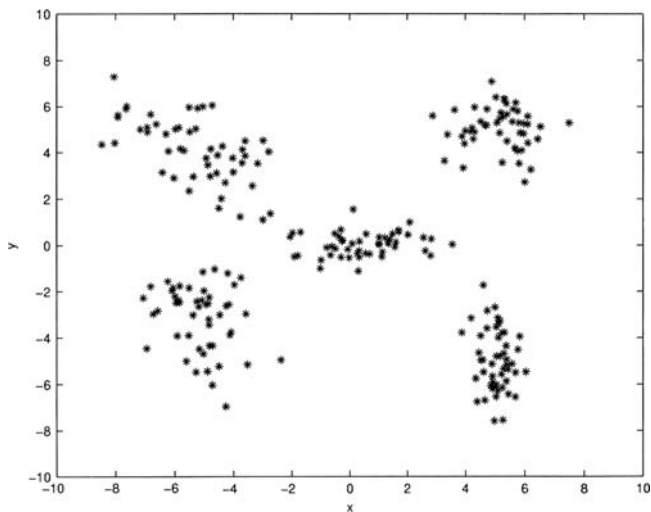


Fig. 2.3 Scatter plot of a two-dimensional data set

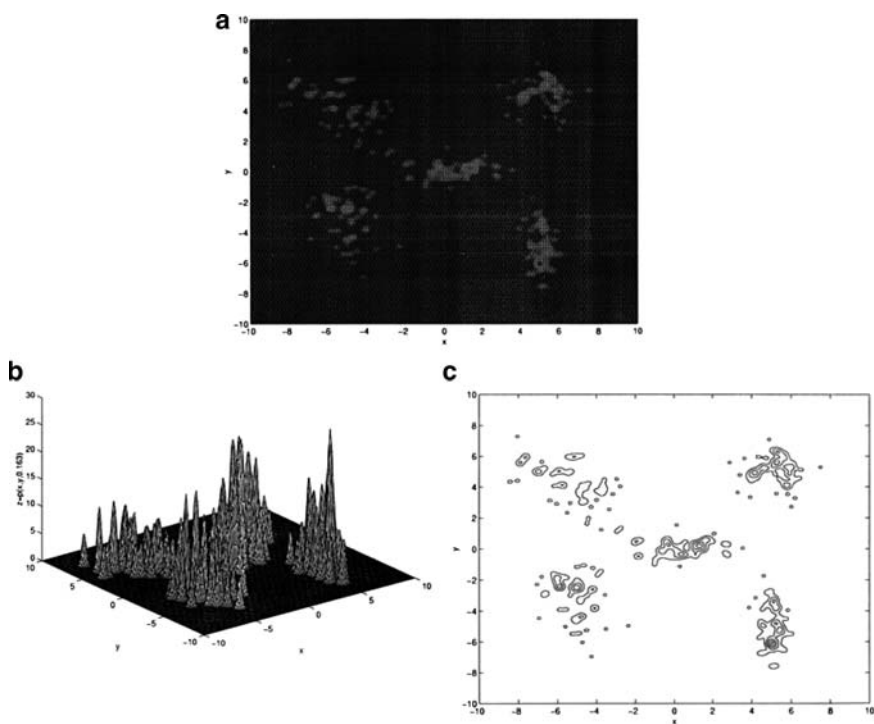


Fig. 2.4 Visualization of the scale-space image obtained from data set in Fig. 2.3 at $\sigma = 0.163$ (a) Scale-space image pseudo-color plot for $\sigma = 0.163$. (b) Mesh plot of scale-space image for $\sigma = 0.163$. (c) Scale-space image contour plot for $\sigma = 0.163$

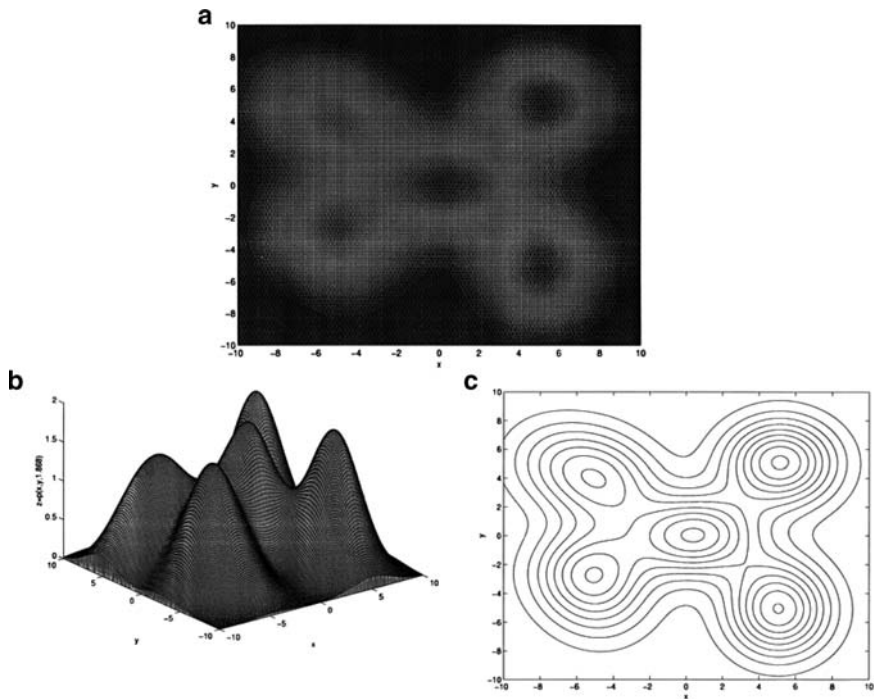


Fig. 2.5 Visualization of the scale-space image obtained from data set in Fig. 2.3 at $\sigma=1.868$ (a) Scale-space image pseudo-color plot for $\sigma = 1.868$. (b) Mesh plot of scale-space image for $\sigma = 1.868$. (c) Scale-space image contour plot for $\sigma = 1.868$

the Gaussian kernel $G(\mathbf{x}, \sigma) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N e^{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}}$. Each maximum of $P(\mathbf{x}, \sigma)$ is considered as a cluster center and a point in X is assigned to a cluster via the gradient dynamic equation for $P(\mathbf{x}, \sigma)$. Since Theorem 2.1 holds in any dimension, then the scale space filtering algorithms can straightforwardly be extended to n -dimensional data with slight adaptation.

The study area is Yuen Long, located in the northwest of Hong Kong, corresponding to an area of 230KM^2 on the Hong Kong topographic maps with geographical coordinates ($113^\circ 58'E$ – $114^\circ 07'E$ to $22^\circ 21'N$ – $22^\circ 31'N$). The main land covers include forest, grass, rock, water, build-up area, trees, marshland, shoals, etc. They are distributed in a complex way. The Landsat TM10 image used is from 3 March 1996 with fine weather. The image size is 455×568 pixels. In the experiment, six bands, TM1, 2, 3, 4, 5 and 7, are utilized, i.e., the clustering is done in six dimensions.

The experiment first clusters a data set consisting of 800 pixels randomly sampled from the image and then assigns each pixel to its nearest cluster center. Figure 2.6 is the Landsat image of Yuen Long, Hong Kong, and Fig. 2.7 shows the 15 cluster solution obtained by applying the scale space clustering algorithm to the image. The 15 clusters are obtained from Rule II.2 and the outliers are deleted according to their outlieriness defined in (2.36). Compared with the ground truth, the

Fig. 2.6 Landsat Image of Yuen Long, Hong Kong

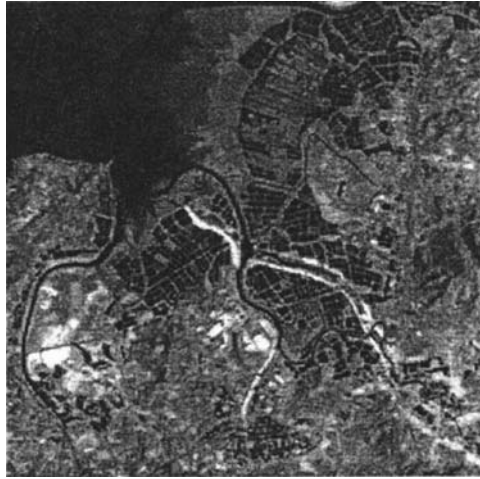
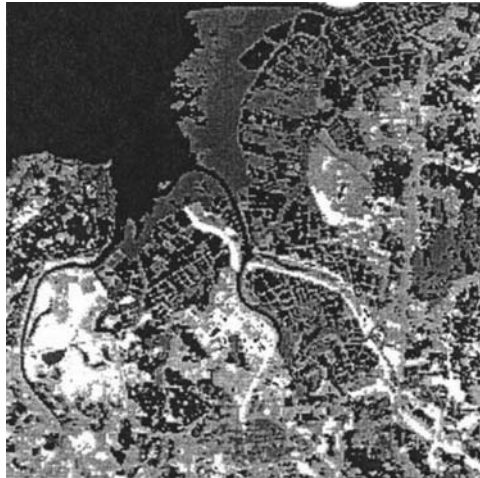


Fig. 2.7 Land covers revealed by the scale space clustering algorithm



scale space clustering is capable of finding the fine land covers. For example, three classes of water bodies corresponding to deep sea water, shallow seawater and fresh-water of the study area have respectively been identified, while they cannot be distinguished by ISODATA method. In the experiments, it is discovered that 150–1,000 sample points are usually large enough to find the land covers contained in the image.

2.2.7 Mining of Seismic Belts in Vector-Based Databases

In seismology, the identification of active faults is crucial to the understanding of the tectonic pattern and the assessment of seismic risk of a specific region. In areas of strong seismic activity, major seismic faults are usually tracked by the epicenters of the

seismic events. Seismic belts, by definition, are belts with dense and zonal distribution of earthquakes controlled by the tectonic belts or the geotectonic aberrance.

Seismic belts are often linear in shape because faults usually exist as wide linear features (Amorese et al. 1999). Due to the complexity of tectonic structures, perfectly linear seismic belts can hardly be found. So, methods for the discovery of seismic belts should be able to recognize features with less-than-perfect linear shape. Since seismic belts often cluster as non-spherical (ellipsoid) shape, spatial clustering algorithms need to identify such irregularly shaped structures.

Detecting all possible digital line components contained in a given binary edge image is one of the most fundamental problems in pattern recognition. Hough transform (Asano and Katoh 1996), for example, is a classical method which basically maps each point in the image space to a line in the parameter space, and counts the intersections to get the parameters of the lines in the image space. The Hough transform is, however, not suitable for detecting wide linear features such as the seismic belts (Amorese et al. 1999). Another conventional algorithm to clustering linear features is the Fuzzy C-Lines (Bezdek et al. 1981). Its basic idea is similar to ISODATA (Ball and Hall 1965), which minimizes some objective function to achieve optimal partitioning of a data set in terms of pre-specified clusters. The difference is that the centers of the clusters in Fuzzy C-Lines change from points to straight lines. The method, nevertheless, is affected by outliers (Honda et al. 2002). Seismologists have also developed several methods to search for seismic belts in databases. The collapsing method (Jones and Steward 1997), the strip method (Zhang and Lutz 1989), and the blade method (Amorese et al. 1999) are typical examples.

Though scale plays an important role in clustering, particularly for spatial databases, all of the above methods have not taken scale into consideration. Since seismic belts are natural structures which can only be detected or observed within a certain scale range, methods for the mining of such linear clusters should take scale into consideration. We particularly need to determine the appropriate spatial scale for the discovery of seismic belts, and to observe their behavior along the scale.

Mathematical morphology provides mathematical tools to analyze the geometry and structure of objects. To take advantage of such method, scale space can be constructed with several morphology filtering operators for data mining. Many attempts have been made to combine mathematical morphology with the concept of scale space or clustering. Postaire et al. (1993), for example, attempt to find the “core” of clusters with the opening and closing operators, and allocate the remainder points by the nearest neighbor method. Maragos (1989) use standard morphological opening and closing with structuring elements of varying shape and size to generate a scale space for shape representation. With increasing or decreasing scale, specific binary patterns are self-dilated or eroded and are subsequently used in the open or close operations. The scale parameter is governed by the degree of self dilation or erosion of a given pattern. In the study by Acton and Mukherjee (2000), scale space is constructed with the opening and closing operators of area morphology and the “scale space vectors” are used to perform image classification. Park and Lee (1996) have also studied the property of scale space using mathematical morphology. They point out that the scale space of one dimensional gray-scale

signals based on morphological filtering satisfies causality (no new feature points are created as scale gets larger), and with the generalized concept of zero-crossing, opening and closing based morphological filtering will construct a scale space satisfying causality. Di et al. (1998), on the other hand, propose a clustering algorithm using the closing operator with structuring elements increasing iteratively in size, and use the heuristic method to find the best number of clusters. They, however, do not describe their algorithm from the viewpoint of scale space, and they do not give thorough analysis on how to specify the precision of the raster image and how to remove noise to prevent it from disturbing the subsequent morphological operations.

With special reference to the work of Di et al. (1998) but adopting the scale space point of view (Leung et al. 2000a), Wang et al. (2006) propose a scale space clustering method, called Multi-scale Clustering Algorithm with Mathematical Morphology Operators (MCAMMO), for the mining of seismic belts in spatial databases. To extract linear or semi-linear features, the algorithm is further enhanced by some more morphological operations, and the algorithm is called Linear MCAMMO (L_MCAMMO). The idea of MCAMMO is to use mathematical morphology to obtain the most suitable scale to re-segment the seismic belts first. The final belts are then obtained with further processing. The procedure of MCAMMO can in brief be summarized as follows: the vector data set is first converted into a binary image data set with a grid whose precision is specified by the sorted k -dist graph (Ester et al. 1996). A pair of closing and opening operators is used to remove the noise. A scale space is then constructed by using the closing operator with structuring elements of increasing size. Through that, the connected components (the set of cells with neighborhood relationships, i.e., clusters) in the image will gradually merge into each other and become a single cluster in the end. This is essentially a binary image segmentation process, and can also be treated as a hierarchical clustering if the points under each connected component are viewed as one cluster.

The main enhancement of MCAMMO to the work of Di et al. (1998) is that it lucidly gives an effective and easy to follow solution to specify the precision of the raster data set. Based on that, noise removing becomes easier and it makes MCAMMO a robust clustering method.

To make it more effective in the mining of near linear belts such as the seismic belts, Wang et al. (2005) perform further segmentation on the data. In brief, the procedure obtains the skeletons of the segmented image at the most suitable scale with the thinning operator. It then obtains the nodes, extracts and classifies the linear (or near linear) axes, and uses such information to re-segment the image in order to obtain the final linear belts. The procedure is a specialized MCAMMO and is called the Linear MCAMMO (L_MCAMMO). Though it intends to mine linear or near linear seismic belts, it is also suitable for the mining of other linear or semi-linear features such as roads in a remote sensed image contaminated with noise.

The advantages of MCAMMO are: (1) the number of clusters does not need to be specified a priori, (2) only a few simple inputs are required, (3) capable of extracting clusters with arbitrary shapes, and (4) robust to noise.

2.2.7.1 Experiment 2.1

The data set in this experiment comes from real-life earthquake data collected in China by the Seismic Analysis and Forecasting Center (1980, 1989). The objective is to mine seismic belts from this data set. A total of 3,201 seismic events with magnitude ≥ 2.2 in the area of $[34^{\circ}\text{--}42^{\circ}N, 106^{\circ}\text{--}115^{\circ}E]$ are extracted. Figure 2.9a shows two nearly parallel seismic belts (in broken lines) corresponding to the north segment of the North–South seismic belt (on the left) and the Shanxi seismic belt (on the right) (Fu 1997). The difficulty in mining the belts lies on the discontinuity of the dense areas in one single belt, which is hard to pick up by the single-scale clustering algorithms such as DBSCAN (Ester et al. 1996). The task can, however, be accomplished effectively and efficiently by MCAMMO.

The lifetime of the clusterings along the scale is depicted in Fig. 2.8, and the connected components, clusters, at selected scales are shown in Fig. 2.9. From Figs. 2.8 and 2.9, we can observe that the lifetime of the 2-clusters clustering is the longest, while that of the 3-clusters clustering is the second longest. By comparing the images at scale 18 which starts the 2 clusters and scale 14 which starts the 3 clusters, we can observe that the connected components in the latter image are actually closer to the true seismic belts. It indicates that 3 is the most suitable number of clusters.

This experiment indicates that clustering of the longest lifetime may not always be the best solution to every problem. We should also pay attention to clustering’s whose lifetimes are relatively long, but not the longest. The scale space approach does provide such valid patterns unraveled by the concept of lifetime. Although the seismic belts can be extracted by MCAMMO, their shapes are still not very close to the “near linear” shape of the real seismic belts. In more complex situations (see Experiments 2.2 and 2.3), the differences would even be greater. To have better performance, some specializations on MCAMMO need to be made.

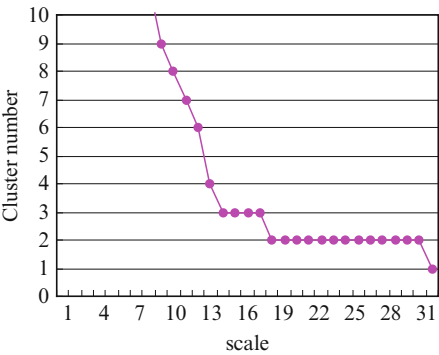


Fig. 2.8 Lifetime of the clusterings in Fig. 2.9

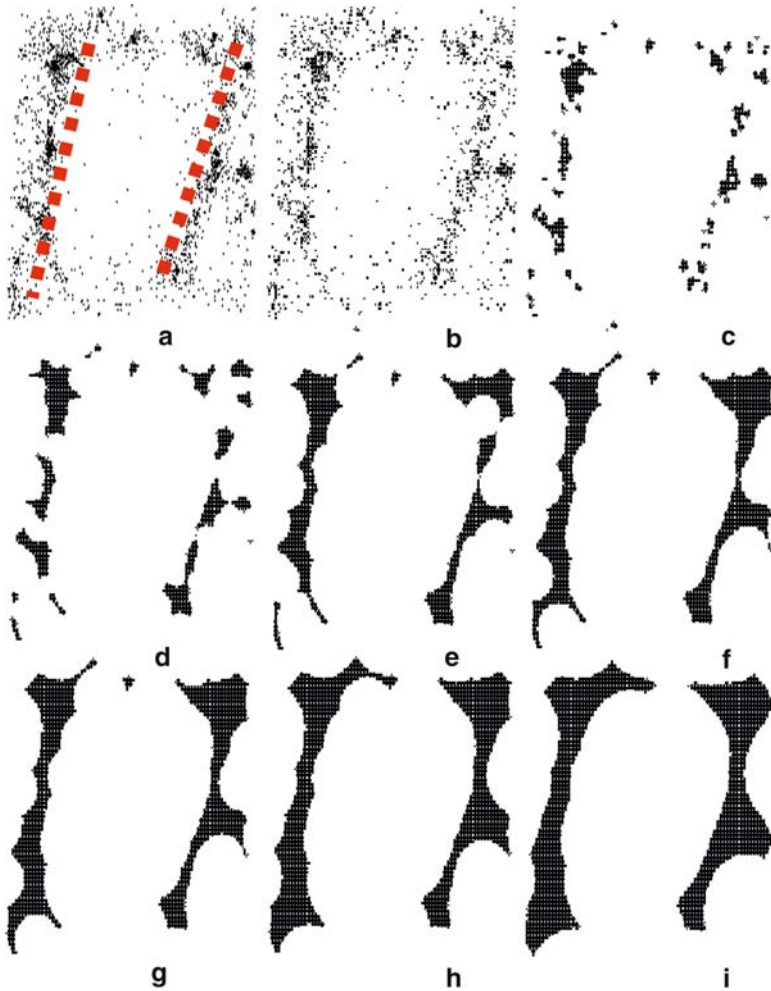


Fig. 2.9 Mining of seismic belts with MCAMMO (a) Original vector-based data set. (b) Rasterized image. (c) First scale with noises removed. (d) Scale 5. (e) Scale 10. (f) Scale 13. (g) Scale 14. (h) Scale 18. (i) Scale 25

2.2.7.2 Experiment 2.2

The image of scale 14 in Experiment 1 is re-processed with the strategy of L_MCAMMO. The skeletons of the segmented image are extracted at the most suitable scale. The nodes of the skeletons are obtained with the hit-or-miss transform. They are “smashed” to split the skeletons into the arcs which are recombined into several groups of “the longer the better” and “the straighter the better” linear (or near linear) axes. Using the information of nodes, skeletons and axes, the image is re-segmented into several linear (or near linear) belts. The belts such obtained

will be very close to the true seismic belts. As a result, two linear belts are obtained which are very close to the actual seismic belts (see Fig. 2.10). This actually provides the answer to the discovery of seismic belts problem (Fig. 1.2) posed in Chapter 1, section 1.5.

2.2.7.3 Experiment 2.3

In this experiment, the test area is moved to $[40\text{--}50^\circ\text{N}, 106\text{--}115^\circ\text{E}]$ to further validate the effectiveness of L_MCAMMO. There are three main seismic belts which are conglutinated with each other, with the upper one in near arch shape (Fig. 2.11a). MCAMMO is first employed to extract the most suitable image (see Fig. 2.12). It can be observed that the clustering stabilizes at scale ***9 with two clusters. Apparently, the segmented image (Fig. 2.11b) is very different from the actual seismic belts. So, by applying the L_MCAMMO, the image at scale 9 is employed to extract the skeletons (Fig. 2.11c), obtain the axes (Fig. 2.11d) and then extract the linear belts (Fig. 2.11e). Subsequently, the three longest linear belts obtained are very close to the actual seismic belts.

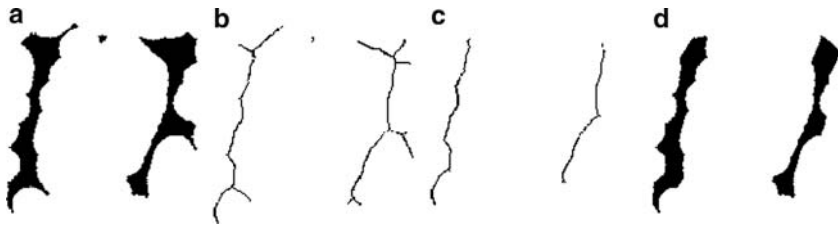


Fig. 2.10 Segmentation after specialization (a) Image with the longest lifetime. (b) Skeletons. (c) Axes of the two longest linear belts. (d) Two belts extracted

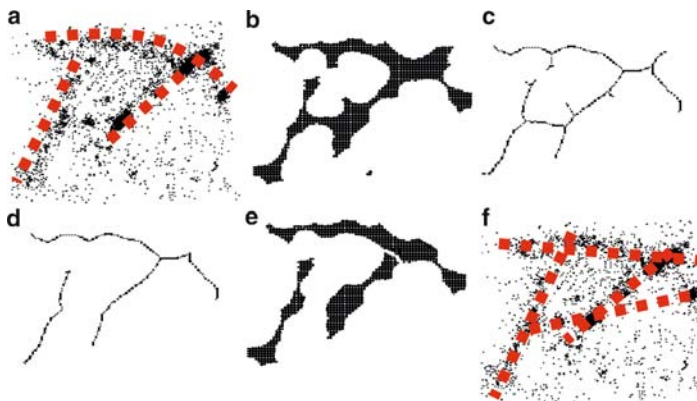


Fig. 2.11 Another seismic area (a) Original data set. (b) Image at the most suitable scale. (c) Skeletons. (d) Axes. (e) Linear belts. (f) Clustering result of Fuzzy C- Lines

As a comparison, Fuzzy C-Lines is employed to extract the belts with the same data set (Wang et al. 2003). Fuzzy C-Lines turns out to be very sensitive to noise. So, noise removal needs to be performed first. The inputs of Fuzzy C-Lines are: $m = 2$, the number of clusters = 4 (taking into account the short linear belts in the center of the image), and 100 iterations. The central lines of the final clusters and the points distributed around them are depicted in Fig. 2.11f. From this image, we find that the upper seismic belt is split apart, where as L_MCAMMO is robust to the “not very linear” clusters. Furthermore, a cluster composed by the points with very large space in-between is obtained by Fuzzy C-Lines (see the bottom-right in Fig. 2.11f), which is not very reasonable. This shows that L_MCAMMO does a better job on this data set. It should also be noted that L_MCAMMO, unlike fuzzy C-Lines, does not require the number of lines ($m = 2$) and the number of clusters = 4, to be pre-specified as inputs. That is what makes scale space clustering, L_MCAMMO in particular, more natural and spontaneous.

To recapitulate, MCAMMO, with the L_MCAMMO enhancement, can obtain the most suitable scale to re-segment an image, and the mining of the linear belts is completed by the re-segmentation procedure.

2.2.8 Visualization of Temporal Seismic Activities via Scale Space Filtering

In seismology, the identification of seismic active periods and episodes in the temporal domain, the seismic belts in the spatial domain, the seismic sequence and the seismic anomaly in the spatio-temporal domain can all be treated as a clustering problem. I have shown in Sect. 2.2.7 how scale space clustering can be employed to mine seismic belts in spatial data. I will show in this subsection how the clustering algorithm, together with its visualization, can be used to identify seismic active periods and episodes in temporal data.

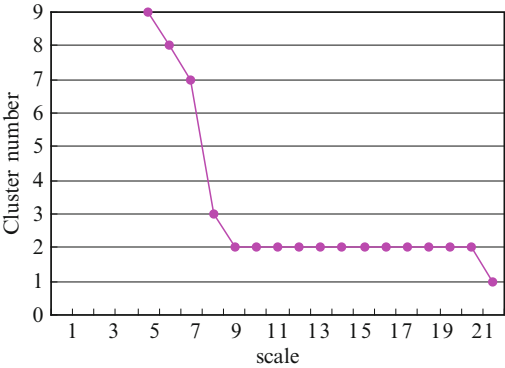


Fig. 2.12 Lifetime of the clusterings in Fig. 2.11

In a larger spatial context, the temporal sequence of strong earthquakes exhibits a certain pattern of clustering with interspersed quiescence and active periods, i.e., quasi-periodicity (Ma and Jiang 1987; Kagan and Jackson 1991; Fu and Jiang 1994). Accordingly, the regional seismic activity in the temporal domain can be segmented into the *seismic active periods* and the *seismic active episodes* on the finer temporal scale (Cao and Fu 1999). Exact and quantitative analysis of seismic active periods and episodes has important implications to the understanding and forecasting of long- and medium-term earthquakes.

Due to the complexity and unpredictability of earthquakes, as well as the difficulty in analyzing the seismic active periods and episodes, the study of seismic activities often rely on the seismologists' expertise and judgments with simple statistical indices (Matthews and Reasenberg 1988). To make the analysis more rigorous and results easier to evaluate, quantitative methods are often needed in conjunction with domain specific expertise (Kagan 1999). Cluster analysis has thus become a common approach to study seismic activities.

As discussed, clustering by scale space filtering has an intrinsic relationship with our visual system. The visualization of clustering by scale space filtering includes two phases: namely visual representation and interactive analysis.

In the first phase, the construction process of scale space clustering can naturally be visualized via a top-to-bottom tree-growing animation in two-dimensional/three-dimensional (2D/3D) views. Animation facilitates the generation of the original qualitative cognition about the clustering in the whole scale space. We can interactively set the visual properties of animation and navigate the scale space in 2D/3D view, including the rotation of a view and the one-dimensional or all-dimensional zooming of a view. This phase suits the visual representation of the scale space.

After the construction of the scale space, visualization based on the scale space and the indices for cluster validity check can assist us to interactively construct, verify and revise at any scale our cognition of the optimal clustering until the final result is obtained. The visualization techniques include the 2D/3D graphs and diagrams of indices which provide the interaction with the concrete numeric indices and customization of the visual properties. Based on the information conveyed by the indices, we can use the slider technique to select the scale of interest in free-style. The corresponding clustering result is shown by both the view of the scale space and the map or time sequence graph. Obviously this phase enables interactive analysis for obtaining the optimal result.

For illustration, I give in the following a brief description of a study on the visualization of seismic activities by scale space clustering (Qin et al. 2006).

2.2.8.1 Experimental Data

In this application, periodic seismic activity of strong earthquakes in Northern China (34–42°N, 109–124°E) is identified via the visualization of the clustering process of scale space filtering. Considering the completeness of the strong earthquake catalog (Huang et al., 1994a, b), two datasets are chosen: (1) the strong

earthquakes ($M_s \geq 6.0$) of 1290–2000 AD which have 71 records, and (2) the strong earthquakes ($M_s \geq 4.7$) of 1484–2000 AD which have 670 records. In seismology, both $M_s 6.0$ and $M_s 4.7$ are lower bounds of strong seismic meanings.

2.2.8.2 Temporal Segmentation of Strong Earthquakes ($M_s \geq 6.0$) of 1290–2000 AD

The scale space for the time sequence of earthquakes in this period is depicted in 2D in Fig. 2.13. The number of clusters and the indices including *lifetime*, *isolation* and *compactness* of the clustering are shown in Fig. 2.14.

The scale-space graph and indices call for special attention to the patterns appearing in both the 59–95th and the 6th scale steps (Fig. 2.14). In the 59–95th scale range, there are three clusters in the clustering with the longest lifetime, isolation and compactness. It is the seismic active period recognized through the visualization of the clustering algorithm (Fig. 2.15a). It actually corresponds to the Second, Third, and Fourth Seismic Active Periods singled out by the seismologists (Jiang and Ma 1985). The correspondence between the clustering and seismologists' results is summarized in Table 2.1.

In the 6th scale step, the number of clusters changes dramatically. The number of clusters decreases rapidly for scales preceding the 6th. After the 6th step, however, the change in clustering becomes comparatively smooth. This clustering process shows that the earthquakes, which are comparatively frequent in the time dimension preceding the 6th step, merge rapidly into clusters when the observation scale increases in this scale range. When the time scale is larger than six and seven, however, clusters are formed in more apparent isolations. Fewer clusters are formed in a relatively long scale range. The clustering result in the 6th scale step in fact corresponds to what is recognized by the seismologists as the seismic active episodes (Fig. 2.15b).

2.2.8.3 Temporal Segmentation of Strong Earthquakes ($M_s \geq 4.7$) of 1484–2000 AD

Similar analysis and visualization are applied to the time sequence of strong earthquakes ($M_s \geq 4.7$) of 1484–2000 AD. Based on the indices shown in Fig. 2.16, two

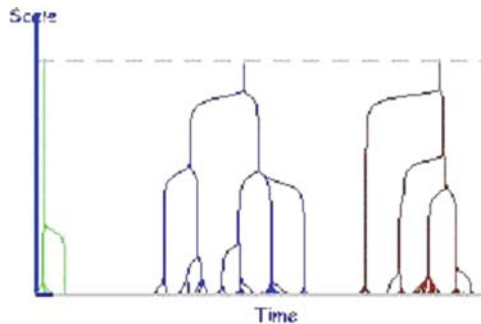


Fig. 2.13 Scale-space clustering for earthquakes ($M_s \geq 6$)

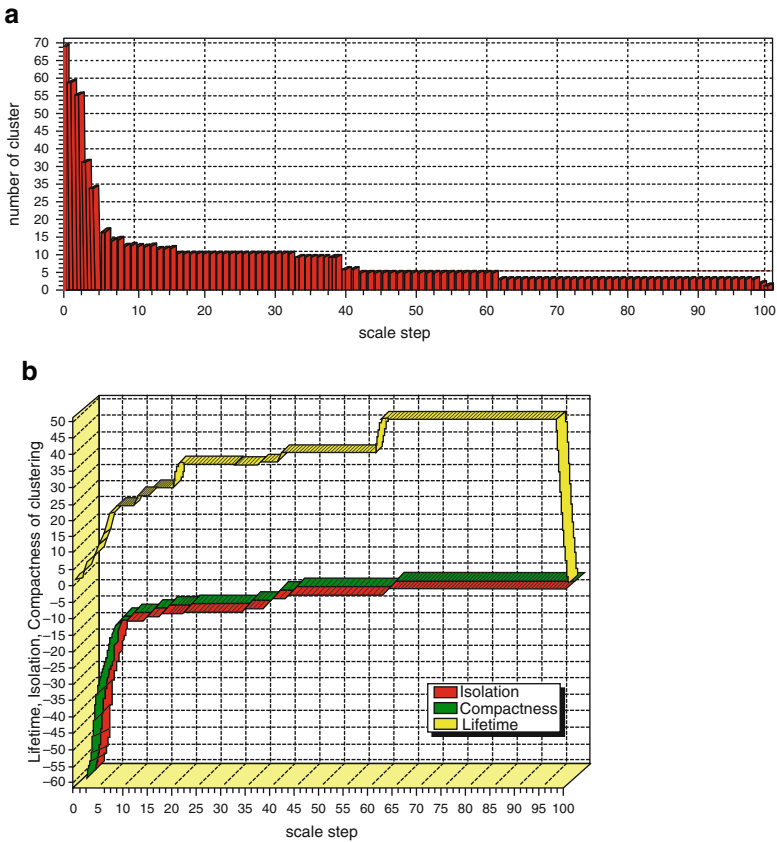


Fig. 2.14 Indices of clustering along the time scale for earthquakes ($M_s \geq 6.0$) (a) number of clusters. (b) lifetime, isolation and compactness of the clustering

clusters are deciphered in the 74–112th scale range. They correspond well with the Third and Fourth Seismic Active Periods identified by the seismologists (Fig. 2.17a). Similar to the 1290–2000 AD situation, in the 10th scale step of this time period, we discover 18 clusters which match well with the seismic active episodes identified by the seismologists (Fig. 2.17b).

2.2.8.4 An Overall Interpretation of the Clustering Results

Table 2.1 tabulates the seismic active periods and episodes unraveled by the scale space clustering algorithm versus that of the seismologists.

It can be observed that the periods and episodes of earthquakes ($M_s \geq 6$) and ($M_s \geq 4.7$) obtained by scale space clustering are consistent with the results identified by the seismologists' domain specific expertise, with the exception that the episodes of

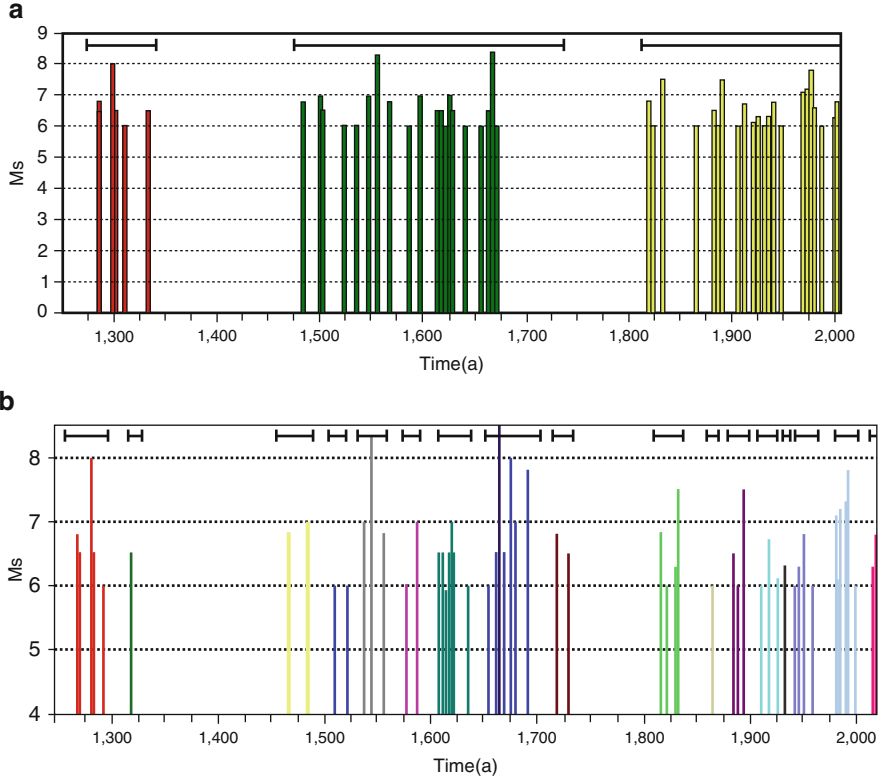


Fig. 2.15 M_s -time plot of clustering results for earthquakes ($M_s \geq 6$) (a) 3 clusters in the 59–95th scale range. (b) 17 clusters at the 6th scale step

the Fourth Seismic Active Period recognized by the clustering algorithm is not as consistent. It seems that there is a quasi-periodicity of about 10–15 years for active episodes.

2.2.9 Summarizing Remarks on Clustering by Scale Space Filtering

1. *Lifetime* is a suitable cluster-validity criterion. This can be observed in Fig. 2.2.
2. The algorithms are robust to the variation of cluster shape which can even be non-Gaussian. This is mainly because the objective function in (2.7) is the density distribution estimate and the algorithm is a “mode-seeking” one which tries to find the dense regions. If the data consist of long and thin clusters, we can make use of the Mahalanobis distance instead of the Euclidean distance in the algorithms, and the covariance matrices can be estimated iteratively with a

Table 2.1 Seismic active periods and episodes obtained by the clustering algorithm and the seismologists

Seismic active period	Seismologists' results		Clustering result	
	Seismic active episode	(Jiang and Ma 1985)	(Gu et al. 1995)	Ms \geq 6 Ms \geq 4.7
II				1290–1340 (6)
III		1484–1730	1481–1730	1484–1730 (31) 1484–1772 (200)
IV		1815–	1812–	1815–(34) 1789–(470)
II	1 (?)			1290–1314 (5)
	2 (?)			1337 (1)
III	1	1484–1487	1481–1487	1484–1502 (3) 1484–1494 (12)
	2	1497–1506	1501–1506	1495–1533 (37)
	3	1522–1538	1520–1539	1524–136 (2)
	4	1548–1569	1548–1569	1548–1568 (4) 1536–1569 (30)
	5	1578–1597	1580–1599	1587–1597 (2) 1576–1599 (28)
	6	1614–1642	1614–1642	1614–1642 (8) 1610–1633 (31)
				1638–1649 (10)
	7	1658–1683	1658–1695	1658–1695 (10) 1654–1695 (38)
	8	1695–1708		1698–1708 (3)
	9	1720–1730	1720–1730	1720–1730 (2) 1720–1746 (7)
Quiescent Period				1754–1772 (4)
				1789–1798 (6)
V	1	1815–1820	1812–1820	1815–1830 (4) 1805–1835 (26)
	2	1829–1835	1827–1835	
	3	1855–1862	1846–1863	1861 (1) 1851–1862 (11)
	4	1880–1898	1880–1893	1879–1888 (3) 1879–1893 (13)
	5	1909–1923	1909–1918	1903–1918 (4) 1898–1924 (28)
	6	1929–1952	1921–1952	1922 (1) 1929 (2)
				1929–1945 (6) 1931–1948 (15)
	7	1966–1978	1965–1976	1966–1983 (13) 1952– (369)
				1998– (2)

(The number in parentheses is the number of earthquakes in the cluster)

particular regulation technique if too few a data is contained in a given cluster. This phenomenon can also be seen in Fig. 2.1 and the other experiments where data are of different shapes.

3. The algorithms are insensitive to outliers because outliers can easily be detected in these algorithms. From (2.7) and (2.8), we can see that the influence of one point on a given cluster center is proportional to $O\left(de^{-d^2/\sigma^2}\right)$ with d being the distance between them. When d is large, $O\left(de^{-d^2/\sigma^2}\right)$ is very small. An outlier is usually very far from the cluster centers, so it has little influence on the estimation of the cluster center. On the other hand, the normal data points are usually far away from the outlier, so they have little influence on an outlier. That is to say, an outlier can survive for a long time as a cluster. Therefore, it has a high degree of outlierness (see (2.36)) and can easily be detected.

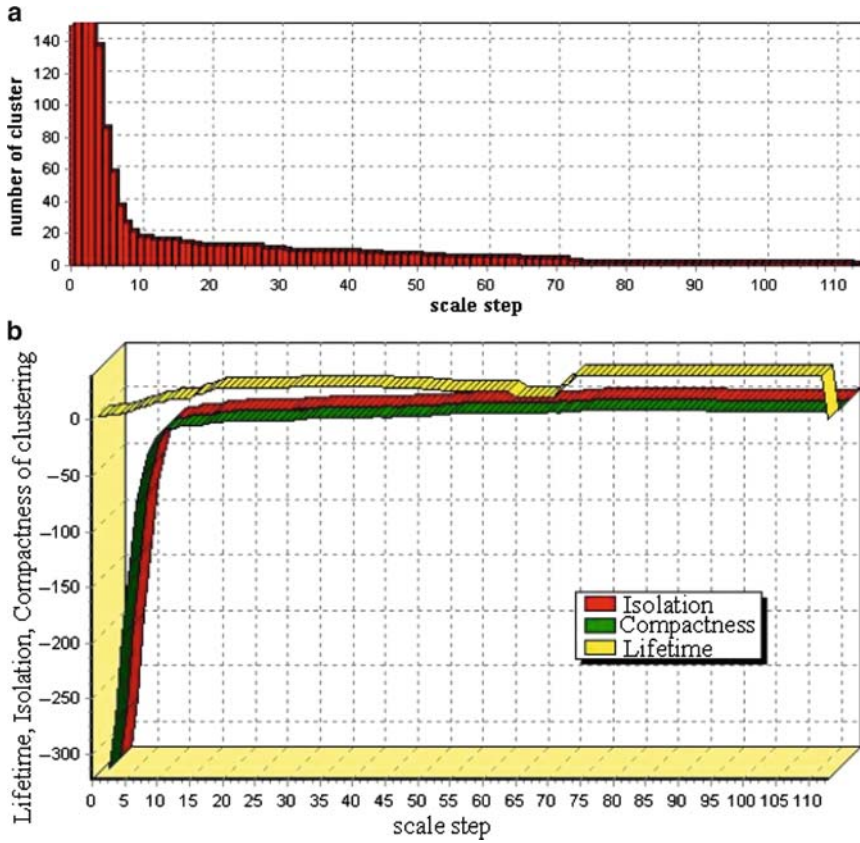


Fig. 2.16 Indices of clustering along the time scale for earthquakes ($M_s \geq 4.7$) (a) Number of clusters (The vertical axis just shows the part no larger than 150). (b) Lifetime, isolation and compactness of the clustering

4. Since the proposed algorithm allows cluster in a partition to be obtained at different scales, more subtle clustering, such as the discovery of land covers, can be obtained.
5. The algorithms work equally well in small and large data sets with low and high dimensions.
6. The proposed clustering method can also be applied to the clustering of data with known distribution containing noise or being indifferentiable.
7. Several scale-based clustering algorithms have been proposed in recent years (Taven et al. 1990; Wilson and Span 1990; Wong, 1993; Chakravarthy and Ghosh 1996; Miller and Rose 1996; Waldemark 1997; Roberts 1997; Blatt et al. 1997). They are derived from very different approaches, such as estimation theory, self-organization feature mapping, information theory, statistical mechanics, and radial basis function networks. One, however, can show that these algorithms are closely related to each other, and in fact, each of these algorithms is equivalent to a special implementation of the proposed algorithm in Leung et al. (2000a).

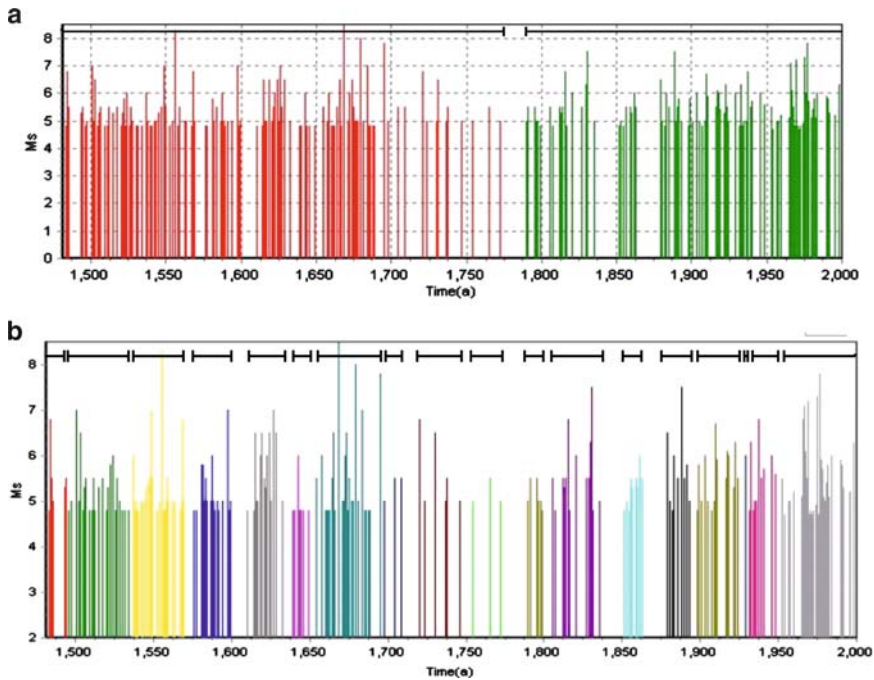


Fig. 2.17 Ms-time plot of clustering results for earthquakes ($M_s \geq 4.7$) (a) 2 clusters in the 74–112th scale range. (b) 18 clusters at the 10th scale step

8. For further research, mechanism should be devised to separate clusters which are close to each other. Furthermore, since Gaussian scale space theory is designed to be totally non-committal, it cannot take into account any a priori information on structures which are worthy of preserving. Such a deficiency may be improved by employing more sophisticated nonlinear scale space filters or by integrating appropriate methods, such as mathematical morphology in the seismic belt experiment.

2.3 Partitioning of Spatial Data by a Robust Fuzzy Relational Data Clustering Method

As discussed in Sect. 2.1, there are two basic approaches to discover clusters in data. Scale space filtering that has just been discussed in Sect. 2.2 belongs to hierarchical clustering. To make our discussion more complete, a method for partitioning clustering, called robust fuzzy relational data clustering, is introduced in this section. Similar to scale space filtering, special attention is again paid to the issue of scale and noise in the clustering of spatial data.

2.3.1 On Noise and Scale in Spatial Partitioning

In spatial clustering, data may be *object data* $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \in \mathbf{R}^s$, with feature vector \mathbf{x}_k corresponding to object k , or *relational data* represented by an $N \times N$ *relational data matrix* $\mathbf{D} = (D_{ij})_{N \times N}$, in which D_{ij} measures the relationship between object i and object j , and \mathbf{D} may be a similarity or dissimilarity relation (Leung 1984, 1988; Jain and Dubes 1988; Kaufmann and Rousseeuw 1990). The classical clustering algorithms for relational data can be found in Jain and Dubes (1998), and several fuzzy clustering algorithms for relational data can be found in (Hathaway et al. 1989; Bezdek et al. 1991; Hathaway and Bezdek 1994; Hathaway et al. 1994). In general, these methods are sensitive to noise and outliers in the data. However, data in real applications usually contain noise and outliers. Thus, clustering techniques need to be robust if they are to be effective under noise. Since fuzzy clustering, by showing the degree to which an object fits into each cluster (Bezdek et al. 1991, 1999), has the obvious advantage in conveying more information about the cluster structure, many robust fuzzy clustering algorithms have been developed in recent years (Ohashi 1984; Dave 1991; Dave and Krishnapuram 1997; Frigui and Krishnapuram 1999). While most of the existing robust clustering algorithms are designed to solve clustering problems involving object data only, a huge number of data sets collected in communication, transportation and other spatial analyses is however relational in nature. Therefore, it is essential to develop robust fuzzy relational data clustering algorithms for the analysis of such data type. By incorporating the concept of clustering against noise in the relational algorithms, Hathaway et al. (1994) and Sen and Dave (1998) have developed algorithms for clustering relational data contaminated by noise. Since the algorithms proposed by Ohashi (1984) and Dave (1991) are robust against noise in the object data, its relational versions are expected to be insensitive to noise in relational data. However, this approach is criticized for having only one “scale” parameter whilst in practical applications each cluster may have its own special scale. Another deficiency of the current clustering approach under noise is that a consistent method to find an appropriate value for the scale parameter is non-existent.

To be able to handle noise and scale, Zhang and Leung (2001) proposed a robust fuzzy relational clustering method by introducing multiple scale parameters into the objective function so that each cluster has its own scale space parameters. Without loss of generality, the method only considers dissimilarity relation, and the value of D_{ij} is arbitrary and no specific relations, such as positivity, reflexivity / anti-reflexivity or symmetry, are imposed on the dissimilarity matrix \mathbf{D} . (A fuzzy graph theoretic approach to clustering on the basis of a similarity or dissimilarity matrix resulting in hierarchical partitioning of spatial data can be found in Leung (1984)).

Based on Zhang and Leung (2001), noise clustering techniques are first briefly reviewed in this section, and a multiple-scale parameter clustering algorithm for object data containing noise is then proposed. Its relational versions are subsequently described and a new necessary condition for optimizing the corresponding

objective function is stipulated. The estimation of the scale parameters and detailed description of the proposed algorithm are then made and substantiated with examples.

2.3.2 Clustering Algorithm with Multiple Scale Parameters for Noisy Data

For an object data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, we denote its cluster centers by $\mathbf{P}_v, v = 1, \dots, k$. The fuzzy c-means algorithm (FCM) (Bezdek et al. 1999) assumes that the number of clusters c is known a priori and the goal is to minimize

$$J_{fcm} = \sum_{v=1}^k \sum_{i=1}^N (u_{iv})^m d_{iv}, \quad (2.42)$$

where $m > 1$ is fixed, d_{iv} is the squared distance from a feature point \mathbf{x}_i to the cluster center \mathbf{p}_v , and u_{iv} is the membership of \mathbf{x}_i in cluster v which satisfies:

$$v_{iv} \geq 0, \quad \text{for } i = 1, \dots, n; \quad v = 1, \dots, k, \quad (2.43)$$

$$\sum_{v=1}^k u_{iv} = 1, \quad \text{for } i = 1, \dots, n. \quad (2.44)$$

The necessary conditions for local extrema of the minimization of (2.42) subject to (2.43) and (2.44) are

$$u_{iv} = \left(\sum_{w=1}^k \left(\frac{d_{iw}}{d_{iv}} \right)^{1/(m-1)} \right)^{-1}, \quad i = 1, \dots, n; \quad v = 1, \dots, k, \quad (2.45)$$

and,

$$\mathbf{p}_v = \frac{\sum_{i=1}^N (u_{iv})^m \mathbf{x}_i}{\sum_{i=1}^N (u_{iv})^m}, \quad v = 1, \dots, k. \quad (2.46)$$

Similar to hard c-means algorithms, fuzzy c-means algorithm is sensitive to noise and outliers. Robust clustering technique is thus introduced to make FCM less sensitive to noise. The goal of such an algorithm is to minimize

$$J_{nc} = \sum_{v=1}^k \sum_{i=1}^N (u_{iv})^m d_{iv} + \sum_{i=1}^N \eta \left(1 - \sum_{v=1}^k u_{iv} \right)^m \quad (2.47)$$

subject to

$$u_{iv} \geq 0, \quad i = 1, \dots, n; \quad v = 1, \dots, k, \quad (2.48)$$

$$\sum_{v=1}^k u_{iv} \leq 1, \quad i = 1, \dots, n. \quad (2.49)$$

The necessary conditions for local extrema of the above optimization problem are

$$u_{iv} = \left(\sum_{w=1}^k \left(\frac{d_{iw}}{d_{iw}} \right)^{1/(m-1)} + \left(\frac{d_{iv}}{\eta} \right)^{1/(m-1)} \right)^{-1}, \quad i = 1, \dots, n; \quad (2.50)$$

$$v = 1, \dots, k,$$

and,

$$\mathbf{p}_v = \frac{\sum_{i=1}^N (u_{iv})^m \mathbf{x}_i}{\sum_{i=1}^N (u_{iv})^m}, \quad v = 1, \dots, k. \quad (2.51)$$

It should be noted that the clustering algorithm works satisfactorily provided that an appropriate value of the scale parameter, η , is known. However, a consistent method to find a good value of η is not available. Another deficiency of clustering under noise is that only one “scale” parameter is used while in practical applications, each cluster may have its own special scale. Zhang and Leung (2001) address these problems by letting each cluster have its own scale parameter. The proposed objective function becomes

$$J_{nc} = \sum_{v=1}^k \sum_{i=1}^N (u_{iv})^m \frac{d_{iv}}{\eta_v} + \sum_{i=1}^N \left(1 - \sum_{v=1}^k u_{iv} \right)^m, \quad (2.52)$$

where u_{iv} , $i = 1, \dots, n$; $v = 1, \dots, k$, are membership values that need to satisfy (2.48) and (2.49). The necessary conditions for local extrema of the minimization of (2.52) subject to (2.48) and (2.49) are

$$u_{iv} = \left(\sum_{w=1}^k \left(\frac{d_{iw}/\eta_v}{d_{iw}/\eta_v} \right)^{1/(m-1)} + \left(\frac{d_{iv}}{\eta_v} \right)^{1/(m-1)} \right)^{-1}, \quad (2.53)$$

$$i = 1, \dots, n; \quad v = 1, \dots, k,$$

and,

$$\mathbf{p}_v = \frac{\sum_{i=1}^N (u_{iv})^m \mathbf{x}_i}{\sum_{i=1}^N (u_{iv})^m \mathbf{x}_j}, \quad v = 1, \dots, k. \quad (2.54)$$

Since each cluster has its own scale parameters, we can use the techniques developed in the possibilistic c-means clustering approach (Dave and Krishnapuram 1997) to estimate the scale parameters as follows:

Obtain a pilot clustering by the FCM first and then estimate η_v by

$$\eta_v = K \frac{\sum_{i=1}^N (u_{iv})^m d_{iv}}{\sum_{i=1}^N (u_{iv})^m}, \quad v = 1, \dots, k, \quad (2.55)$$

where u_{iv} is the membership value obtained by the FCM, d_{iv} is the corresponding squared distance between \mathbf{x}_i and cluster center p_v , and K is typically chosen to be 1.

Another estimate of η_v is given by

$$\eta_v = \frac{\sum_{k=1}^N (u_{ik})^{\geq \alpha} d_{ik}}{\sum_{k=1}^N (u_{ik})^{\geq \alpha}}, \quad v = 1, \dots, k, \quad (2.56)$$

where $\alpha \in (0, 1)$ gives the crisp α -cut partition

$$(u_{ik})^{\geq \alpha} = \begin{cases} 0, & \text{if } u_{ik} < \alpha, \\ 1, & \text{if } u_{ik} \geq \alpha. \end{cases} \quad (2.57)$$

Based on the multiple-scale parametric objective function in (2.52), the multi-scale parametric clustering algorithm (MPCA) for noisy data is formulated as follows:

- Step 1. Execute a FCM algorithm to find an initial membership values u_{iv} .
- Step 2. Apply (2.55) to compute η_1, \dots, η_k based on the membership values and cluster centers obtained in step 1.
- Step 3. Repeat the following sub-steps: Apply (2.54) to update p_v , Apply (2.53) to compute u_{iv} , until $\max_{iv} |u_{iv}(i+1) - u_{iv}(i)| < \epsilon$.
- Step 4. Apply (2.55) or (2.56) to compute η_1, \dots, η_k based on the membership values obtained in step 3.
- Step 5. Repeat step 3 to improve d_{iv} and u_{iv} , and then stop.

In possibilistic c-means clustering, Krishnapuram and Keller (1993) have suggested the use of (2.55) in step 2 and (2.56) in step 4. However, there is no consistent method for finding an appropriate value of α for a given data set at present. Zhang and Leung (2001) propose to use (2.55) in steps 2 and 4 since the membership values obtained in step 3 are made robust by the noise clustering algorithm. Therefore, the outliers are of small membership values and they contribute very little to the estimates of η_v 's.

2.3.3 Robust Fuzzy Relational Data Clustering Algorithm

The clustering algorithm for relational data containing noise is perhaps first considered by Hathaway et al. (1994), and subsequent relational versions are developed by Sen and Dave (1998). These algorithms are the robust versions of fuzzy relational data clustering algorithms and their objective function is

$$J(U, D) = \sum_{v=1}^k \frac{\sum_{i,j=1}^n (u_{iv})^m (u_{jv})^m D_{ij}}{2 \sum_{j=1}^n (u_{jv})^m} + \eta \sum_{i=1}^N \left(1 - \sum_{v=1}^k u_{iv} \right)^m, \quad (2.58)$$

where the membership values u_{iv} are subjected to (2.48) and (2.49). The dissimilarity matrix D in these algorithms is assumed to have the following property:

$$D_{ij} \geq 0, D_{ij} = D_{ji}, \quad i \neq j \quad \text{and} \quad D_{jj} = 0. \quad (2.59)$$

It has been proved that the necessary conditions for minimizing (2.58) subject to (2.48) and (2.49) are as follows:

$$u_{iv} = \frac{(1/d_{iv})^{1/(m-1)}}{\sum_{w=1}^k (1/d_{iw})^{1/(m-1)} + 1/\eta^{1/(m-1)}}, \quad i = 1, \dots, n; \quad v = 1, \dots, k, \quad (2.60)$$

where

$$d_{iv} = \sum_{j=1}^n D_{ij} \left(\frac{(u_{jv})^m}{q_v} \right) - \frac{1}{2} \sum_{j,k=1}^n D_{jk} \left(\frac{(u_{jv})^m (u_{kv})^m}{(q_v)^2} \right), \quad i = 1, \dots, n; \quad (2.61)$$

$$v = 1, \dots, k,$$

and $q_v = \sum_{j=1}^n (u_{jv})^m$. When d_{iv} is negative, then u_{iv} may become negative. Therefore, there is no guarantee that the constraint in (2.48) will be satisfied. This problem can be solved by applying a “spreading” transformation proposed by Hathaway and Bezdek (1994). The spreading transformation adds a positive

number β to all off-diagonal elements of \mathbf{D} . In fact, Hathaway and Bezdek's algorithms are derived under the condition that the relational data \mathbf{D} is Euclidean, which means that $D_{ij} = \|\mathbf{x}_j - \mathbf{x}_i\|^2$ for some data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, and it has been proved that there exists a positive β_0 such that all dissimilarity matrices obtained by the spreading transformation with $\beta \geq \beta_0$ are Euclidean. When \mathbf{D} is Euclidean, d_{iv} is the squared Euclidean distance between \mathbf{x}_i and center of the cluster \mathbf{p}_v . Therefore, all d_{iv} 's are non-negative.

Zhang and Leung (2001) propose a new robust fuzzy relational data clustering algorithm with multiple-scale parameters and give an alternative approach to address the problem of negative d_{iv} . The algorithm aims at the minimization of the objective function

$$J(\mathbf{U}, \mathbf{D}) = \sum_{v=1}^k \frac{\sum_{i,j=1}^n (u_{iv})^m (u_{jv})^m D_{ij}}{2\eta_v \sum_{j=1}^n (u_{jv})^m} + \sum_{i=1}^N \left(1 - \sum_{v=1}^k u_{iv}\right)^m, \quad (2.62)$$

with the membership value u_{iv} , $i = 1, \dots, n$; $v = 1, \dots, k$, constrained by (2.48) and (2.49). In (2.62), η_v , $v \in \{1, \dots, k\}$, is a normalization constant, called the scale parameter (which is usually a threshold used to determine which object is an outlier), and k is the given cluster number. No restriction is imposed on the dissimilarity matrix \mathbf{D} .

The first term in the objective function is employed to reduce the u_{iv} when object i is with high dissimilarity with other object j in cluster v , and the second term is employed to guarantee that most data should be in the meaningful clusters.

For the object data clustering problem, if $\eta_v = 1$ and \mathbf{D}_{ij} is the Euclidean distance between two vectors representing object i and object j , the first term in the objective function is the general fuzzy c-means clustering objective function (Bezdek et al. 1999). Furthermore, if $\eta_1 = \dots = \eta_k = \eta$, then the objective function in (2.62) is equivalent to the objective function in Dave (1991).

If we denote

$$\begin{aligned} d_{iv} = & \frac{1}{2} \sum_{j=1}^n D_{ij} \left(\frac{(u_{jv})^m}{q_v} \right) + \frac{1}{2} \sum_{j=1}^n D_{ji} \left(\frac{(u_{jv})^m}{q_v} \right) \\ & - \frac{1}{2} \sum_{j,k=1}^n D_{jk} \left(\frac{(u_{jv})^m (u_{kv})^m}{(q_v)^2} \right), \quad i = 1, \dots, n; v = 1, \dots, k, \end{aligned} \quad (2.63)$$

where $q_v = \sum_{j=1}^n (u_{jv})^m$, then we can prove that

$$\begin{aligned} u_{iv} = & \frac{(1/|d_{iv}|)^{1/(m-1)}}{\sum_{w=1}^k (1/|d_{iw}|)^{1/(m-1)} + 1/\eta_v^{1/(m-1)}}, \quad i = 1, \dots, n; \\ & v = 1, \dots, k, \end{aligned} \quad (2.64)$$

satisfies the Karush–Kuhn–Tucker conditions for optimality of the problem in (2.62) when $m - 1 = r_1/2r_2$ with r_1 and r_2 being odd numbers. Since each $m - 1$ can be approximated by such numbers, (2.63) and (2.64) are used to estimate the membership value in the proposed algorithm for any $m \geq 1$.

If D_{ij} is the squared Euclidean distance between objects i and j , then d_{iv} is the squared distance from object i to the center of cluster v .

In the proposed algorithm, we must give the estimated value of η_v . In Zhang and Leung (2001), a fuzzy clustering is first obtained by minimizing the following objective function

$$J_1(\mathbf{U}, \mathbf{D}) = \sum_{v=1}^k \frac{\sum_{i,j=1}^N (u_{iv})^m (u_{jv})^m D_{ij}}{2 \sum_{j=1}^n (u_{jv})^m}, \quad (2.65)$$

with membership values u_{iv} , $i = 1, \dots, n$; $v = 1, \dots, k$, constrained by (2.43) and (2.44). This objective function is a natural extension of a fuzzy relational data clustering algorithm called *FANNY* (Kaufmann and Rousseeuw 1990) and is first proposed by Hathaway et al. (1989). As discussed in the above section, we can derive a necessary condition for the optimal membership variables:

$$u_{iv} = \frac{(1/|d_{iv}|)^{1/(m-1)}}{\sum_{w=1}^k (1/|d_{iw}|)^{1/(m-1)}}, \quad i = 1, \dots, n; \quad v = 1, \dots, k, \quad (2.66)$$

in which

$$\begin{aligned} d_{iv} = & \frac{1}{2} \sum_{j=1}^n D_{ij} \left(\frac{(u_{jv})^m}{q_v} \right) + \frac{1}{2} \sum_{j=1}^n D_{ji} \left(\frac{(u_{jv})^m}{q_v} \right) \\ & - \frac{1}{2} \sum_{j,k=1}^N D_{jk} \left(\frac{(u_{jv})^m (u_{kv})^m}{(q_v)^2} \right), \quad \text{for } i = 1, \dots, n; \quad v = 1, \dots, k. \end{aligned} \quad (2.67)$$

The fuzzy relational data clustering algorithm (FRDC) based on (2.66) and (2.67) is as follows (Zhang and Leung 2001):

- Step 1. Initialize the membership values $u_{iv}(0)$, taking into account constraints in (2.43) and (2.44). Let $i = 0$.
- Step 2. Compute d_{iv} by (2.67).
- Step 3. Compute $u_{iv}(i + 1)$ by (2.66).
- Step 4. If $\max_{i,v} |u_{iv}(i + 1) - u_{iv}(i)| < \varepsilon$, then stop. Otherwise, $i = i + 1$, then go to step 2.

When there is one $d_{iv} = 0$, we can update u_{iv} , as proposed in the fuzzy c-means algorithms, in step 3.

Compared with other fuzzy relational data clustering algorithms, the proposed algorithm has no restrictions on the fuzzy exponent m and the data type. Therefore, it is a more general fuzzy relational data clustering algorithm.

When a fuzzy clustering is obtained by the FRDC algorithm, the obtained membership value u_{iv} is employed to estimate η_v as follows:

$$\eta_v = \frac{\sum_{j=1}^n (u_{jv})^m |d_{jv}|}{\sum_{j=1}^n (u_{jv})^m}, \quad v = 1, \dots, k. \quad (2.68)$$

To formulate the robust fuzzy relational data clustering algorithm (RFRDC), the alternating optimization approach with three stages is employed to minimize the objective function in (2.62). In the first stage, we execute the FRDC algorithm to determine an initial membership value. In the second stage, (2.68) is applied to compute the scale parameters η_1, \dots, η_k based on the initial cluster membership values. Then (2.63) and (2.64) are employed to iteratively update the pseudo-distance and membership values until a given stopping criterion is satisfied (i.e., when the membership values u_{iv} cannot be significantly changed in two successive iterations). In the third stage, η_v is estimated on the basis of the membership values determined in the second stage. Then (2.63) and (2.64) are applied to refine d_{iv} and u_{iv} . Details of the robust fuzzy relational data clustering algorithm (RFRDC) are given as follows:

- Step 1. Execute the FRDC algorithm to find the initial membership values u_{iv} .
- Step 2. Apply (2.68) to compute η_1, \dots, η_k based on the membership values determined in step 1.
- Step 3. Repeat the following sub-steps:
 - Apply (2.63) to update d_{iv} ,
 - Apply (2.64) to compute u_{iv} ,
 - until $\max_{i,v} |u_{iv}(i+1) - u_{iv}(i)| < \varepsilon$.
- Step 4. Apply (2.68) to compute η_1, \dots, η_k based on the membership values determined in step 3.
- Step 5. Repeat step 3 to improve d_{iv} and u_{iv} , and then stop.

2.3.4 Numerical Experiments

2.3.4.1 A Pedagogic Example

This example involves two well separated clusters of seven points each and three noisy points (Fig. 2.18). We assume that the dissimilarity matrix \mathbf{D} is Euclidean with $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$. In this case, the sequence of partitioning membership value u_{iv} produced by the relational data clustering under noise is identical to the sequence produced by the corresponding clustering for object data under noise. The cluster centers in the experimental results can be computed by (2.46) which are listed in Table 2.2.

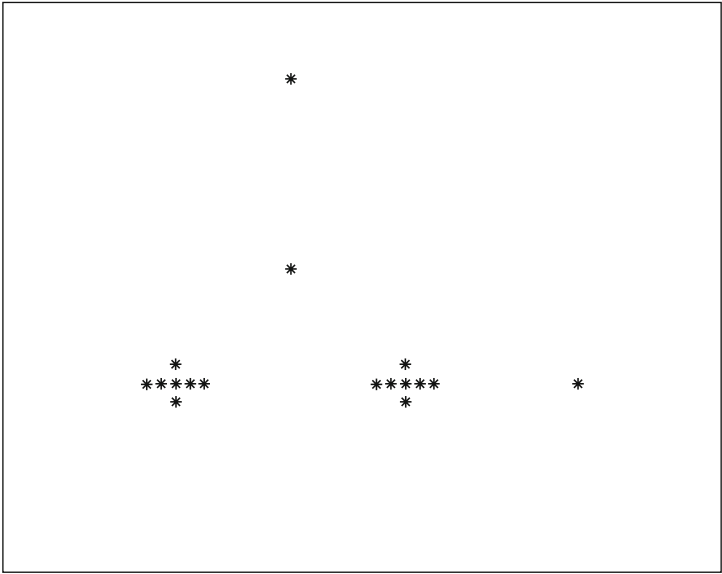


Fig. 2.18 Scatter plot of a noisy data set

Table 2.2 Cluster centers in the experiment

	cluster 1	cluster 2
real cluster centers	(60,150)	(140,150)
cluster centers obtained by noise clustering algorithm (Hathaway et al., 1994b)	(60.2724, 150.2078)	(140.3632, 150.1987)
cluster centers obtained by MPCA	(60.0002, 150.0001)	(140.0006, 150.0001)

From Table 2.2, we can see that the cluster centers found by the proposed algorithm are more precise than that of the relational noise clustering algorithm. Similar phenomena have also been observed in many other numerical experiments (Zhang and Leung 2001).

2.3.4.2 Concordance in Languages

This example is based on the real relational data from the study carried out by Johnson and Wichern (1992, Table 12.4), called “concordant first letters for numbers in eleven languages” which compares eleven European languages (English, Norwegian, Danish, Dutch, German, French, Spanish, Italian, Polish, Hungarian, and Finnish) by looking at the first letters of the first ten numbers. The words for the same number in two different languages are concordant if they have the same first

letters and discordant if they do not. The following matrix of discordant first letters for numbers is used as the dissimilarity matrix \mathbf{D} to cluster these languages.

$$\begin{pmatrix} & E & N & Da & Du & G & Fr & Sp & I & P & H & Fi \\ E & 0 & & & & & & & & & & \\ N & 2 & 0 & & & & & & & & & \\ Da & 2 & 1 & 0 & & & & & & & & \\ Du & 7 & 5 & 6 & 0 & & & & & & & \\ G & 6 & 4 & 5 & 5 & 0 & & & & & & \\ Fr & 6 & 6 & 6 & 9 & 7 & 0 & & & & & \\ Sp & 6 & 6 & 5 & 9 & 7 & 2 & 0 & & & & \\ I & 6 & 6 & 5 & 9 & 7 & 1 & 1 & 0 & & & \\ P & 7 & 7 & 6 & 10 & 8 & 5 & 3 & 4 & 0 & & \\ H & 9 & 8 & 8 & 8 & 9 & 10 & 10 & 10 & 10 & 0 & \\ Fi & 8 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 8 & 0 \end{pmatrix}$$

For $k = 2$, the results obtained by the proposed RFRDC algorithm and NERF (Hathaway and Bezdek 1994) are listed in Table 2.2, where u_v denotes the membership value of a language in cluster v obtained by the RFRDC algorithm, u_v^N denotes the membership value of a language in cluster v obtained by NERF, d_v denotes the distance value obtained from (2.63) by the RFRDC algorithm.

From Table 2.3, we can observe that English, Norwegian, Danish, Dutch and German form a group, French, Spanish, Italian, and Polish form another group, while Hungarian and Finnish appear to be standing alone. This clustering result can be checked by our visual impression of the dissimilarity matrix \mathbf{D} . The advantage of the proposed approach is that it is less subjective in creating clusters and it gives the extent to which a language is in a cluster. For example, from Table 2.3, we can see that English, Norwegian, and Danish are more typical than Dutch and German in cluster 2, and Dutch is less typical than German in this

Table 2.3 Experimental results of the concordance in languages

	u_1	u_2	u_1^N	u_2^N	d_1	d_2
E	0.0346	0.8694	0.1548	0.8452	5.2637	1.0948
N	0.0059	0.9794	0.0488	0.9512	5.2348	0.4095
Da	0.0136	0.9602	0.1041	0.8959	4.5612	0.5492
Du	0.0622	0.3488	0.1427	0.8573	8.1486	4.2307
G	0.0901	0.4447	0.1834	0.8166	6.1874	3.2897
Fr	0.9155	0.0168	0.9542	0.0458	0.6598	4.9528
Sp	0.9499	0.0109	0.9805	0.0195	0.4923	4.6468
I	0.9858	0.0030	0.9824	0.0176	0.2572	4.6619
P	0.3495	0.1405	0.8609	0.1391	3.0218	5.5482
H	0.0589	0.1781	0.2874	0.7126	9.0909	6.7481
Fi	0.0743	0.1491	0.4172	0.5828	8.1090	7.3931

cluster. In cluster 1, Italian is the most typical one and Polish is the least typical one. While these conclusions can be drawn from the clustering results produced by the proposed RFRDC algorithm, they are not obvious in the results obtained by NERF (see Table 2.3).

2.3.4.3 Clustering of Oil Types

This example employs a real data set from Gowda and Diday (1992) for eight different types of oil. The similarity matrix obtained from that study is given as follows:

Oil Type	o_1	o_2	o_3	o_4	o_5	o_6	o_7	o_8
o_1 : Linseedoil	—							
o_2 : Perillaoil	4.98	—						
o_3 : Coiion — seedoil	3.66	5.70	—					
o_4 : Sesmaeoil	3.77	5.88	7.00	—				
o_5 : Camelia	3.84	4.70	6.25	5.90	—			
o_6 : Oliveoil	3.24	5.30	6.68	6.37	6.24	—		
o_7 : Beef — tallow	0.86	2.78	4.11	3.61	3.48	4.28	—	
o_8 : Lard	1.22	3.08	4.44	3.97	3.89	4.68	6.74	—

The dissimilarity matrix can be generated from the similarity matrix in either of the following ways:

$$D_{ij} = 1/S_{ij} - \min_{r \neq t}(1/S_{rt}), i \neq j$$

or

$$D_{ij} = \max_{r \neq t}(S_{rt}) - S_{ij}, i \neq j$$

and $D_{ij} = 0$ for all i . The dissimilarity matrices \mathbf{D}_1 and \mathbf{D}_2 generated respectively by the above equations are as follows:

$$\mathbf{D}_1 = \begin{pmatrix} 0 & 0.0579 & 0.1304 & 0.1224 & 0.1176 & 0.1658 & 1.0199 & 0.6768 \\ 0.0579 & 0 & 0.0326 & 0.0272 & 0.0699 & 0.0458 & 0.2169 & 0.1818 \\ 0.1304 & 0.0326 & 0 & 0 & 0.0171 & 0.0068 & 0.1005 & 0.0824 \\ 0.1224 & 0.0272 & 0 & 0 & 0.0266 & 0.0141 & 0.1342 & 0.1090 \\ 0.1176 & 0.0699 & 0.0171 & 0.0266 & 0 & 0.0174 & 0.1445 & 0.1142 \\ 0.1658 & 0.0458 & 0.0068 & 0.0141 & 0.0174 & 0 & 0.0908 & 0.0708 \\ 1.0199 & 0.2169 & 0.1005 & 0.1342 & 0.1445 & 0.0908 & 0 & 0.0055 \\ 0.6768 & 0.1818 & 0.0824 & 0.1090 & 0.1142 & 0.0708 & 0.0055 & 0 \end{pmatrix}$$

and

$$\mathbf{D}_2 = \begin{pmatrix} 0 & 0.0579 & 0.1304 & 0.1224 & 0.1176 & 0.1658 & 1.0199 & 0.6768 \\ 0.0579 & 0 & 0.0326 & 0.0272 & 0.0699 & 0.0458 & 0.2169 & 0.1818 \\ 0.1304 & 0.0326 & 0 & 0 & 0.0171 & 0.0068 & 0.1005 & 0.0824 \\ 0.1224 & 0.0272 & 0 & 0 & 0.0266 & 0.0141 & 0.1342 & 0.1090 \\ 0.1176 & 0.0699 & 0.0171 & 0.0266 & 0 & 0.0174 & 0.1445 & 0.1142 \\ 0.1658 & 0.0458 & 0.0068 & 0.0141 & 0.0174 & 0 & 0.0908 & 0.0708 \\ 1.0199 & 0.2169 & 0.1005 & 0.1342 & 0.1445 & 0.0908 & 0 & 0.0055 \\ 0.6768 & 0.1818 & 0.0824 & 0.1090 & 0.1142 & 0.0708 & 0.0055 & 0 \end{pmatrix}$$

Table 2.4 exhibits the final memberships found by the RFRDC algorithm and NERF on the dissimilarity matrices \mathbf{D}_1 and \mathbf{D}_2 . The cluster number is $k = 2$. In Table 2.4, $u_{\cdot v}^{(1)}$ is the membership value produced by the proposed algorithms for dissimilarity matrix \mathbf{D}_1 , and $u_{\cdot 1}^{(2)}$ is the membership value for \mathbf{D}_2 ; $u_{\cdot v}^{N1}$ is the membership value produced by NERF for dissimilarity matrix \mathbf{D}_1 , and $u_{\cdot v}^{N2}$ is the membership value produced by NERF for dissimilarity matrix \mathbf{D}_2 (the membership value is taken from Hathaway and Bezdek 1994). It is interesting to see that in the results obtained by the RFRDC algorithm, o_2, o_3, o_4, o_5, o_6 form the first cluster; o_7, o_8 form another cluster; o_1 seems to be alone and o_2 seems to be less typical in the first cluster. However, we cannot observe these phenomena in the clustering results obtained by NERF.

2.4 Partitioning of Spatial Object Data by Unidimensional Scaling

2.4.1 A Note on the Use of Unidimensional Scaling

In Sect. 2.3, I have introduced an algorithm for the discovery of optimal partitioning of fuzzy relational data in noisy environment. The emphasis is on the robustness to noise and the multiplicity of scale for clusters. The method falls within the realm

Table 2.4 Experimental results of clustering of oil types

	$u_{\cdot 1}^{(1)}$	$u_{\cdot 2}^{(1)}$	$u_{\cdot 1}^{(2)}$	$u_{\cdot 2}^{(2)}$	$u_{\cdot 1}^{N1}$	$u_{\cdot 2}^{N1}$	$u_{\cdot 1}^{N2}$	$u_{\cdot 2}^{N2}$
o_1	0.0619	0.0000	0.0771	0.0019	0.888	0.112	0.704	0.296
o_2	0.4914	0.0002	0.3513	0.0039	0.811	0.189	0.818	0.182
o_3	0.9998	0.0000	0.9993	0.0001	0.631	0.369	0.935	0.065
o_4	0.9874	0.0004	0.9726	0.0016	0.696	0.304	0.924	0.076
o_5	0.8329	0.0006	0.7015	0.0051	0.663	0.337	0.816	0.184
o_6	0.9687	0.0011	0.9340	0.0046	0.539	0.461	0.834	0.166
o_7	0.0001	0.8275	0.0005	0.9391	0.087	0.913	0.036	0.964
o_8	0.0002	0.8475	0.0006	0.9395	0.096	0.904	0.028	0.972

of partitioning clustering. Though it is robust and scale-based, it, similar to other partitioning methods, is sensitive to initialization and is subjected to the presupposition of a class number k .

To circumvent the sensitivity to initial seed values (if handled appropriately), the presupposition of a cluster number, and the trapping by local minima, I introduce in this section the clustering of object data by unidimensional scaling (UDS). The method is mainly developed by Guttman (1968). It has been applied to social science and medical science research (Gorden 1977; McIver and Carmines 1981), and equipped with algorithms for solving the associated global optimization problem (Pliner 1984, 1996; Simantiraki 1996; Lau et al. 1998). Our discussion in this section is based on the study by Leung et al. (2004e) on the mining of natural clusters in remotely sensed data.

2.4.2 Basic Principle of Unidimensional Scaling in Data Clustering

The basic idea of UDS is to arrange n objects on the real line so that the inter-point distances/dissimilarities can best approximate the observed distances (McIver and Carmines 1981). UDS is a relatively simple but effective algorithm. Compared with multidimensional Scaling (MDS) methods such as K-means and ISODATA, UDS is easier to understand and implement, free from the presupposition of a cluster number, insensitive to initial seed values, independent of information structure, and not limited by the feature-space dimension.

In UDS, the basis of analysis is the dissimilarity matrix. Let there be n observed objects with p dimensions:

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ip})^T, \mathbf{x}_i \in \mathbf{R}^p, i = 1 \dots n. \quad (2.69)$$

Then we can establish a matrix of dissimilarities among these objects. As discussed in Leung (1984), dissimilarity between objects can be expressed by the distance between them as follows:

$$d_{ij} = \left\{ \sum_{k=1}^p |x_{ik} - x_{jk}|^q \right\}^{1/q}, \quad 1 \leq q \leq \infty. \quad (2.70)$$

Specifically, d_{ij} is the L1 distance or City block metric when $q = 1$, and the L2 distance or Euclidean distance when $q = 2$. Here we select the Euclidean distance as a basis of measurement.

Based on the distance measure, we can establish the $n \times n$ matrix of dissimilarity between objects as:

$$\mathbf{D} = (d_{ij}) \quad (2.71)$$

UDS attempts to map n objects $\mathbf{x}_i, i = 1 \dots n$, from the p -dimensional space into the one dimensional coordinates $y_i, i = 1 \dots n$, and arrange them on the real line so that

their inter-point distances are as close as possible to their observed distances. That is, it arranges these coordinates in ascending order. The objective is to find the real numbers y_1, \dots, y_n by minimizing the following objective function:

$$\sigma(\mathbf{y}) = \sum_{i < j} (d_{ij} - |y_i - y_j|)^2, \quad \mathbf{y} = (y_1, y_2, \dots, y_n)^T, \quad y_i \in \mathbf{R}. \quad (2.72)$$

The solution is however not unique. For example, the translation and reflection of \mathbf{y} also give the same minimum. To overcome this shortcoming, Guttman imposes a centering constraint: $\sum_{i=1}^n y_i = 0$ on the above function.

As an illustration, the Guttman algorithm and Pliner algorithm are outlined as follows:

2.4.2.1 Guttman Algorithm (1968)

First, we set the initial value, y^0 , for \mathbf{y} as:

$$\mathbf{y}^0 = (y_1^0, y_2^0, \dots, y_n^0)^T, \quad (2.73)$$

where y^0 starts with any random value, or $y_i^0 = \frac{1}{p} \sum_{k=1}^p x_{ik}$.

Then, the optimal estimator \mathbf{y} can be obtained with an iterative algorithm using the equation below:

$$y_i^{(r+1)} = \frac{1}{n} \sum_{j=1}^n d_{ij} \text{sign}(y_i^{(r)} - y_j^{(r)}), \quad i = 1 \dots n, \quad (2.74)$$

where $y_i^{(r)}$ is the coordinate of object i at the r -th iteration.

The iterative process stops when $|y_i^{(r+1)} - y_i^{(r)}| < \delta$ (a small number) is met. The algorithm is fast, simple and has the self-centering property. Nevertheless, the objective function $\sigma(\mathbf{y})$ has many local minima and they increases with n . To prevent trapping by local minima, Pliner (1996) proposes a smoothing algorithm for the UDS problem.

2.4.2.2 Pliner Algorithm (1996)

The smoothing technique is employed to obtain the minimum of $\sigma_\varepsilon(\mathbf{y})$:

$$\sigma_\varepsilon(\mathbf{y}) = (1/\varepsilon^n) \int_{D(\mathbf{y}, \varepsilon)} \sigma(\mathbf{x}) d\mathbf{x} \quad (2.75)$$

where $D(\mathbf{y}, \varepsilon)$ is a cube in \mathbf{R}^n with the center in \mathbf{y} and a side ε . Taking the integral in the above equation we obtain:

$$\sigma(\mathbf{y}) = \sum_{i < j} \left[(y_i - y_j)^2 - 2d_{ij}g_\varepsilon(y_i - y_j) \right] + c \quad (2.76)$$

where c is a constant and

$$g_\varepsilon(t) = \begin{cases} t^2(3\varepsilon - |t|)/3\varepsilon^2 + \varepsilon/3, & \text{if } |t| < \varepsilon, \\ |t|, & \text{if } |t| \geq \varepsilon. \end{cases} \quad (2.77)$$

It is easy to verify that $\sigma_\varepsilon(\mathbf{y})$ is twice continuously differentiable. Taking the partial derivatives of $\sigma_\varepsilon(\mathbf{y})$ and setting them to zero, we obtain the following equation:

$$y_i^{(r+1)} = \frac{1}{n} \sum_{j=1}^n d_{ij} u_\varepsilon(y_i^{(r)} - y_j^{(r)}), i = 1, \dots, n \quad (2.78)$$

where

$$u_\varepsilon(t) = \begin{cases} (t/\varepsilon)(2 - |t|/\varepsilon), & \text{if } |t| < \varepsilon, \\ \text{sign}(t), & \text{otherwise} \end{cases}. \quad (2.79)$$

The quality of solution of both methods however depends on the initial configuration. Leung et al. (2003) propose a method for finding a good starting configuration. In general, the UDS algorithm produces a curve with obvious break off points according to the number of natural clusters in a data set (Fig. 2.20). Generally, if the differences among classes are apparent, there will be distinct step changes in the UDS curve. Consequently, we can choose the corresponding y coordinates as the natural break off points demarcating the cluster. To make the identification of break off points less judgmental, Leung et al. (2004e), propose the UDS histogram method to assist us in determining more objectively the break points in the UDS curve for the discovery of land covers in remote sensing imagery (see Sect. 2.4.4).

2.4.3 Analysis of Simulated Data

To better understand the characteristics and the performance of the UDS method, Leung et al. (2004e) perform simulation studies on three sets of artificially generated data with specific data properties (Arbia 1989).

Data set G1 is of cirque shape and includes 160 sample points, with 60 samples evenly distributed on the circumference and the rest distributed randomly around the circle center. The spatial distribution is unusual but not too complicated (Fig. 2.19a1). Data set G2 includes 200 sample points distributed randomly around two cluster centers with 100 samples each. The distribution is relatively simple and common (Fig. 2.19b1). Data set G3 consists of 200 sample points splitting into two cincture lines, with each having 100 samples. The spatial distribution is complicated (Fig. 2.19c1). The three data sets are employed to test the UDS method against the K-means classifier.

Basing on the UDS curves obtained in the three experiments (Fig. 2.20a), we can observe that Data set G1 has two obvious step changes. The coordinates at which

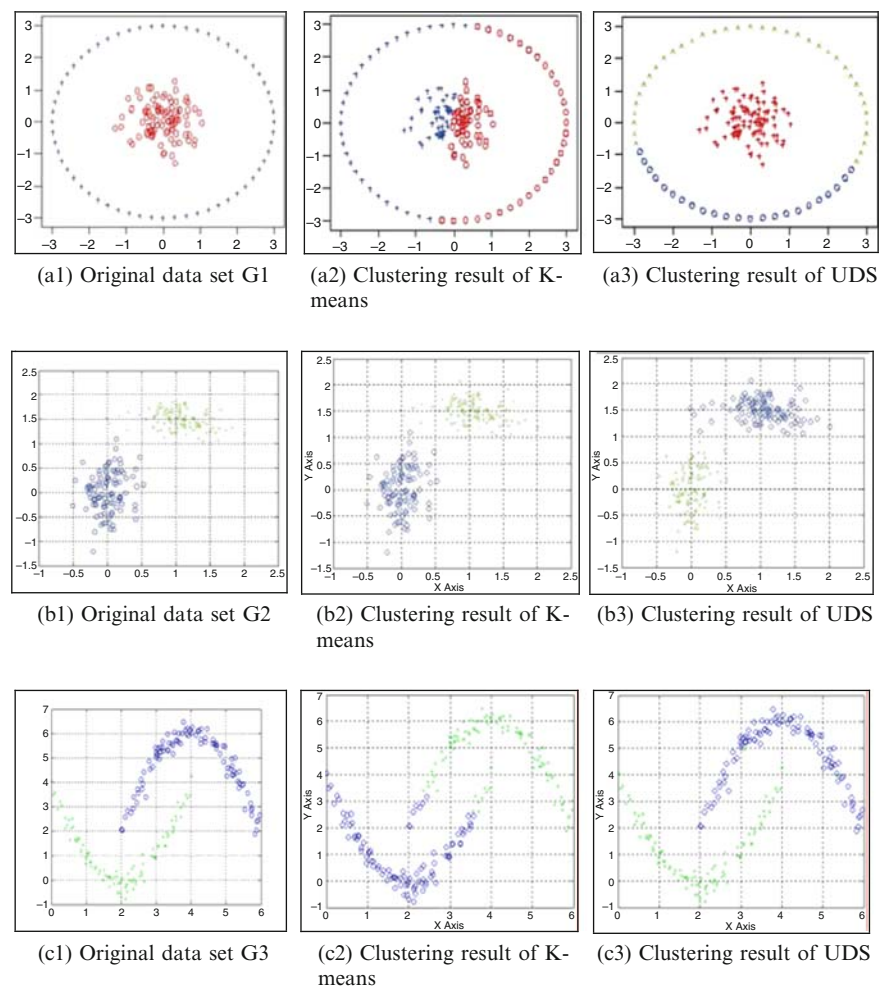
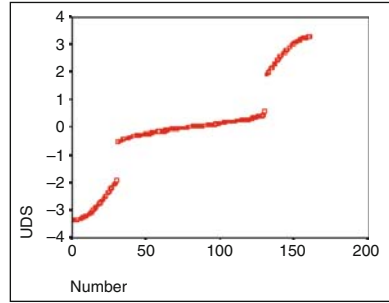
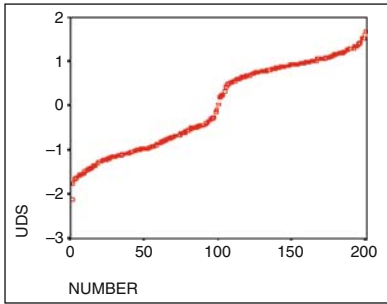


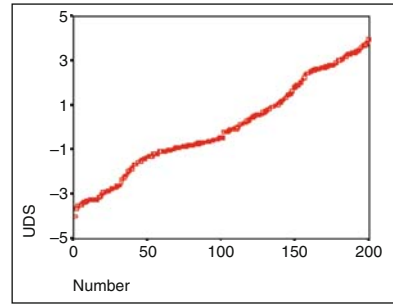
Fig. 2.19 Simulated Experiments of UDS clustering



(a) The UDS curves for data set G1



(b) The UDS curve for data set G2



(c) The UDS curve for data set G3

Fig. 2.20 The experimental UDS curves

step change occurs are the natural break off points between classes. So, we can say that Data set G1 can be clustered into three groups. With reference to the UDS curve for data set G2 (Fig. 2.20b), we can say that there are two natural clusters. As for data set G3, due to the interactive effect of the two cincture lines, the natural breaks in the UDS curve are not obvious (Fig. 2.20c). Nevertheless, it manages to bring forth the spatial features of the clusters involved. Apparently, UDS out-performs the K-means method in the clustering of data set G1 (Fig. 2.19a2, a3). Their performances are nearly the same for the simpler data set G2 (Fig. 2.19b2, b3). With some human interaction with the computer, UDS performs better for the more complicated data set G3 (Fig. 2.19c2, c3). For substantiation, the accuracy assessments are provided in Table 2.5.

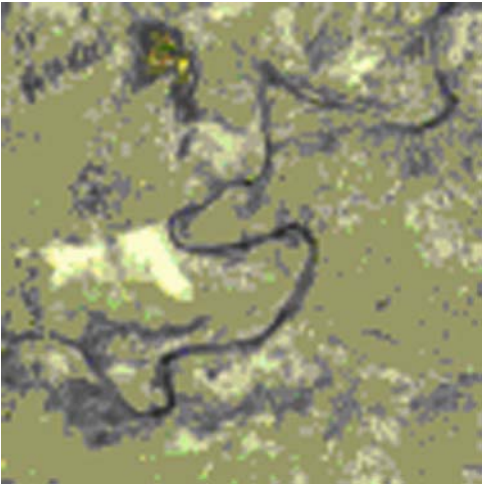
2.4.4 UDS Clustering of Remotely Sensed Data

The UDS method is customized by Leung et al. (2004e) in the analysis of a SPOT-HRV multispectral image acquired over Xinjing on August 30, 1986. The size of the original image is $3,000 \times 3000$ pixels with three spectral bands. It contains *sand*

Table 2.5 The error matrix of the numerical experiment

	Class	K-Means		UDS			Total	Accuracy	
		C1	C2	C1	C2	C3		K-Means	UDS
G1	C1	54	46	100	0	0	100	52.50%	76.90%
	C2	30	30	0	23	37	60		
	Total	84	76	100	23	37	160		
G2	C1	99	1	100	0		100	99.50%	100%
	C2	0	100	0	100				
	Total	99	101	100	100		200		
G3	C1	91	9	100	0		100	92%	98.50%
	C2	7	93	3	97		100		
	Total	98	102	103	97		200		

Fig. 2.21 SPOT multispectral image acquired over Xinjing



(c1) *water* (c2), and *saline area* (c3) as land covers. As usual, the image is preprocessed by filtering, stretching and geometric correction.

For pedagogy, a 100×100 small area (Fig. 2.21) is extracted from this image to evaluate the performance of the UDS method against the K-means and ISODATA methods.

Due to the spatial characteristics and continuity of remotely sensed data, it is necessary to take additional measures to facilitate the application of the UDS method in the clustering of remotely sensed data. First, similarity of pixels in remote sensing images should be analyzed in the multispectral space, for this case the space of 3-dimensional spectral bands. Second, the matrix of similarity D is constructed by calculating the distances between pixels in the multispectral space. Third, the ordinates $y_i, i = 1, \dots, 10000$, are calculated and sorted in ascending order. Four, due to the continuity of ground objects in a remotely sensed image, the derived UDS curve naturally has no obvious step changes (Fig. 2.22).

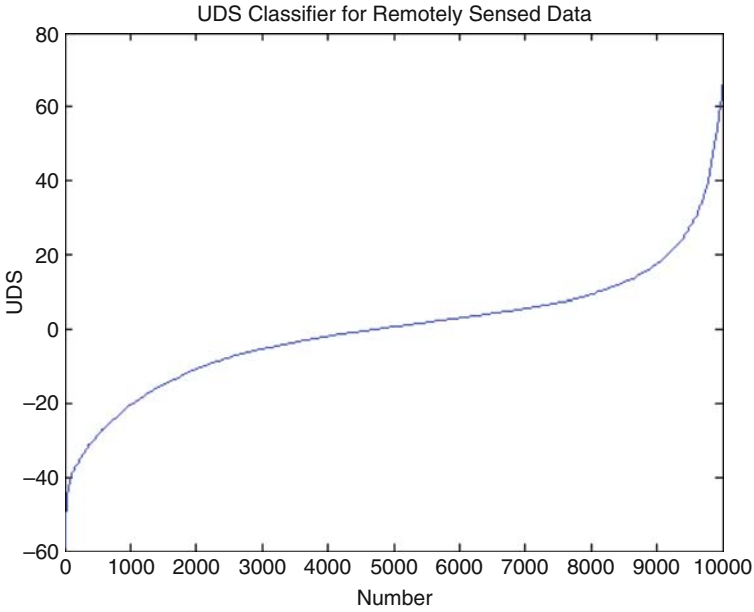


Fig. 2.22 The UDS curve obtained in the remote sensing experiment

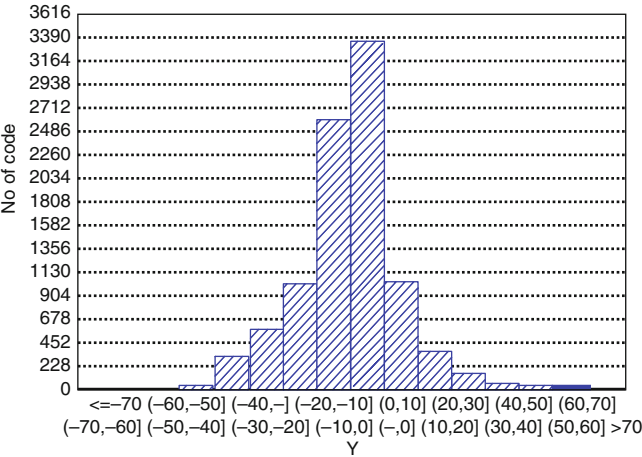


Fig. 2.23 The histogram of the UDS curve

However, we could observe an obvious ascending trend between different clusters in the UDS curve if there are natural clusters, such as this experiment. To facilitate the location of the break off points, Leung et al. (2004e) propose to plot the histogram of the UDS curve (Fig. 2.23). The abscissa is the intervals of the UDS curve and the ordinate is the number of objects in the interval.

In this experiment, $Y_i \in [-60, 70]$, and the interval length is 10. From Fig. 2.23, we can observe that the number of objects rises and drops when $Y_i = -10$ and 10 respectively. It indicates that abscissa -10 and 10 can be regarded as the natural break off points of the clusters. According to this principle, there are two break off points: (2,000, -10.7214) and (8,250, 10.7285) in the UDS curve. On that, we can partition the remotely sensed image into three clusters (Fig. 2.24), which are consistent with the real situation.

In case we cannot directly obtain the break off points, we can just adjust the interval value continually to find the optimal break off points. We can also simultaneously employ visual interpretation or other knowledge to facilitate the identification process.

As a comparison, the K-means and ISODATA methods are applied to the same image and the results are depicted in Figs. 2.25 and 2.26, respectively.

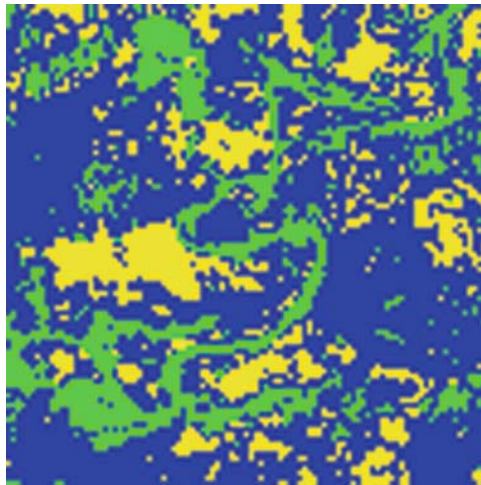


Fig. 2.24 Result obtained by the UDS method

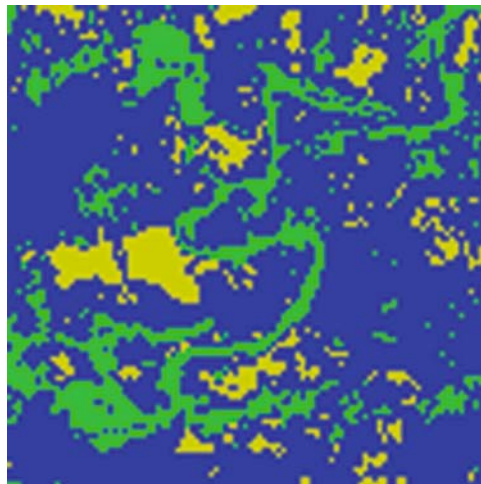


Fig. 2.25 Result obtained by the K-means method

Fig. 2.26 Result obtained by the ISODATA method

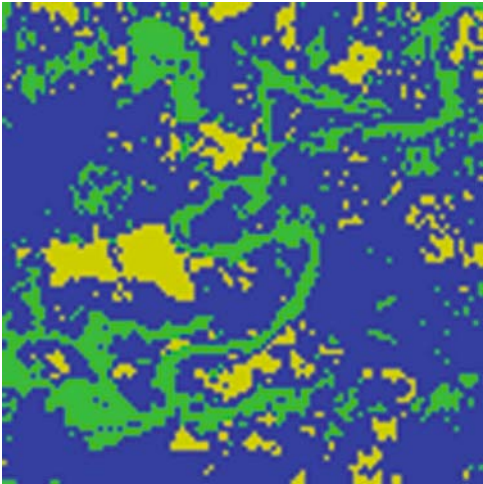


Table 2.6 The error matrix of the remote sensing experiment

True Class	K-Means			ISODATA			UDS			Total
	C1	C2	C3	C1	C2	C3	C1	C2	C3	
C1	39	1	2	39	0	3	40	0	2	42
C2	7	9	0	3	13	0	3	13	0	16
C3	9	0	8	6	0	11	3	0	14	17
Accuracy	74.50%			84%			89%			

The accuracies of the three methods are summarized in Table 2.6. We can observe that the UDS method is more sensitive to the spectral features of the ground objects.

Remark 2.5. The K-means and ISODATA methods calculate the distance of a pixel from the seed and discriminate pixels according to their distances. Thus, if the seed setting is not optimal and the objects actually do not belong to the cluster determined by the seed, it may lead to undesirable results. The UDS, on the other hand, calculates the spectral distance of a pixel to all pixels rather than the seed. Therefore, the classification result of the UDS is more objective and insensitive to seed initialization. It thus leads to higher accuracy than the K-means and ISODATA.

2.5 Unraveling Spatial Objects with Arbitrary Shapes Through Mixture Decomposition Clustering

2.5.1 On Noise and Mixture Distributions in Spatial Data

A major problem in the mining of spatial objects or natural features is the rampant existence of noise and mixture distributions in large spatial databases. Thus, being

able to describe distribution in the feature space with a high tolerance of noise is essential to detect successfully features in spatial data in general and remotely sensed images in particular.

Density-based method can often discover clusters of arbitrary shapes in databases consisting of noise/outliers. Unlike most partitioning methods that cluster features in terms of the distance between them, density-based methods identify clusters as dense regions interspersed by low density regions (often treated as noise/outliers) in the feature space. The DBSCAN algorithm (Ester et al. 1996) and OPTICS (Ankerst et al. 1999), for example, are early density-based methods that cluster (in a partitioning way) data on the basis of density and a set of user supplied parameters. Since both methods rely on some spatial index structures such as R*-tree and X-tree, they are not efficient for clustering high dimensional spatial data. By using the sum of influence functions of all data points and a grid-like structure to assist the calculation of the density function and the hill-climbing procedure that identify the density attractor of each data point, the DENCLUE algorithm (Hinneburg and Keim 1998) exhibits a much better performance in cluster discovery. However, the algorithm again requires a set of parameters that need to be carefully chosen.

Due to the noise level and feature inter-mixing or overlapping, spatial features often take on the form of mixture density distributions. Thus, conventional approaches for simple distributions are inadequate for feature representation and mining in such feature spaces. Mixture density models, on the other hand, become useful for such purpose.

In a mixture model, data are assumed to follow two or more common parametric distributions mixed in varying proportions. Thus, finite mixture density models provide an important means to describe complex phenomena in relatively simple ways (Derin 1987; McLachlan and Basford 1988; Dattereya and Kanal 1990). In practice, the most important class of finite mixture densities is Gaussian (or normal) mixtures. For parametric estimation of this class of mixtures, we can usually select the expectation maximization (EM) algorithm which enables us to compute the maximum likelihood (ML) estimates of the mean vectors and covariance matrices of a Gaussian mixture distribution in an iterative manner (McLachlan and Krishnan 1997). The EM algorithms have been employed to perform spatial feature extraction, data fusion, and data mining in remotely sensed images (Bruzzone et al. 1999; Tadjudin and Landgrebe 2000).

However, the EM algorithm is severely handicapped in the estimation of suitable number of mixtures, especially when the overlapping of features exists in very noisy feature space. To overcome such difficulty, the Gaussian mixture density decomposition (GMDD) algorithm has been proposed as an effective clustering approach for data sets with mixture densities (Zhuang et al. 1996; Dave and Krishnapuram 1997). As an extension of GMDD, an effective data mining method, called regression-class mixture decomposition (RCMD), for regression relations has been developed for large data sets (Leung et al. 2001a) (see Chapter 5 for a detailed description of the approach). Within the framework of RCMD, a data set is treated as a mixture population composed of many components. Each component

corresponds to a regression class defined as a subset of the data set that is subjected to a regression model. The RCMD method then extracts those regression classes in succession. In essence, a regression class reflects a kind of structure existing in the data set. Therefore, the RCMD method is more suitable for mining structured features in data sets. It can be extended to extract spatial features in remotely sensed images. Leung et al. (2006b) propose the RCMD-based feature mining model (RFMM) with genetic algorithms (GA). Through the RFMM-GA model, geometric features, represented by extended parametric models such as the linear structures, ellipsoidal structures, and more complicated parametric structures are extracted from noisy, complex and large spatial data sets. The main idea of the RFMM-GA model is to estimate effectively the parameters of the components of a mixture data set in order to find the components corresponding to the individual features. The GA is employed as a multi-point global optimization procedure to estimate efficiently the parameter sets of RFMM.

In a feature space, it is generally difficult to describe the distributions of feature sets with a common simple density distribution model since distributions of samples usually follow a mixture model. As depicted in Fig. 2.27, there are apparently three structured features in a two-dimensional space. However, the shapes of the features are so different and they overlap to some extent. Hence, the conventional density distribution model is too simple to successfully mine such features. They, however, can be appropriately unraveled by an extension of mixture density models.

As a flexible approach to density estimation, mixture density models have been applied to solve problems in a variety of disciplines. Mixture density modeling and decomposition (MDMD) (Zhuang et al. 1992, 1996; Dave and Krishnapuram 1997) can be viewed as a mixture clustering model that involves the use of robust statistics to identify individual densities more accurately and reliably. The basic flows of a MDMD algorithm are depicted in Fig. 2.28. Given a data set, the parametric distribution model corresponding to the feature to be mined is pre-specified at each step. After performing estimation procedure for the parameters of the distribution model, the data subset, which is fittest to the model, is mined and taken out from the data set. The iterative decomposition procedure is completed until the whole data set is decomposed into categories of features in the mixture.

Gaussian mixtures are commonly employed to model finite mixture densities. The widespread use of Gaussian mixture densities is due to the fact that a univariate Gaussian distribution has a simple and concise representation requiring only two parameters: mean and variance. The Gaussian density is symmetric, unimodal, and isotropic, and it assumes the least prior knowledge (as measured in terms of the uncertainty or entropy of the distribution) in estimating an unknown probability density with given mean and variance. These characteristics of the Gaussian distribution along with its well-studied properties give the Gaussian mixture density models the power and effectiveness that other mixture densities can hardly surpass.

Evolved from the MDMD algorithm and the GMDD algorithm, robust regression-class mixture decomposition (RCMD) is formulated as a composition of simple

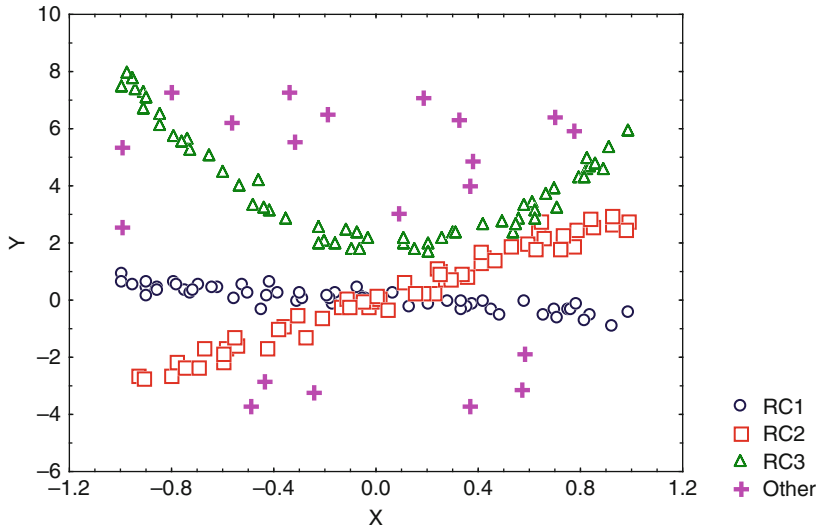
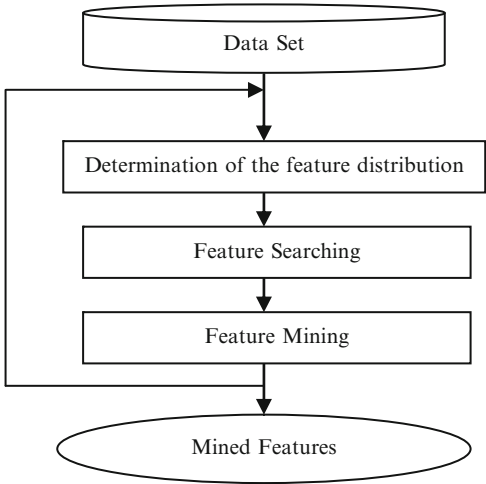


Fig. 2.27 Mixture population containing noise and genuine features

Fig. 2.28 Process of MDMD algorithm



structured regression classes. With respect to a particular regression class, all data points from the other regression classes can be classified as the outlier with different statistical characterization. Thus, a mixture population can be viewed as a contaminated regression class with respect to each class component in the mixture. When all of the observations fitting a single regression class are grouped together, the remaining observations can be considered as elements of an unknown outlier set. Each class component in the mixture population can be estimated separately one at a time in an iterative fashion by using the contaminated model. The iterative

estimation successively reduces the number of class components in the resulting mixture until all regression classes in the mixture are mined.

The output of the RCMD algorithm includes a number of extracted regression classes and possibly an unassigned set containing samples that do not belong to any of the detected classes. Compared to conventional statistical clustering methods, the scheme of RCMD has several distinct advantages:

1. The number of regression classes does not need to be specified a priori. Given a group of sample data sets, the number of classes can be determined one by one with the RCMD. Domain knowledge can even be integrated into the data mining process.
2. The mixtures can contain a large proportion of noise. In fact, a regression-class mixture population can be viewed as a contaminated distribution with respect to each class component in the mixture. Thus the proportion of outliers relative to a component may be large. Even in such situation, RCMD still can pick out each class component sequentially via the robust statistics approach. It is shown that RCMD can resist a large proportion of noise.
3. The estimation of parameters of each class component is virtually independent of each other. This property is derived from the search strategy adopted by the RCMD (Leung et al. 2001). However, for high dimensional feature space, it is more effective if a suitable search range can be pre-determined.
4. The variability in the shape and size of the components in the mixture is taken into consideration. In the search procedure, parameters of each component should be dynamically changed so that points identified by this component follow the corresponding distribution. Therefore, the distribution for the whole data set should be a variable mixture, but not single and fixed.

2.5.2 A Remark on the Mining of Spatial Features with Arbitrary Shapes

Spatial data mining should be built upon definite spatial analysis model and follows the true regularity of spatial distribution. The target is to discover spatial features from complicated spatial data sets. Due to complexity and uncertainty, overlapping and inter-disturbing phenomena among spatial data sets often occur. It is thus difficult to acquire substantial structure of the feature distribution which will directly affect the accuracy of the analysis and interpretability of the features unraveled. An effective way to describe complexity and uncertainty of data in statistics is mixture modeling, i.e., utilizing a mixture model of finite number of simple distributions (called components) to characterize a complicated data set.

However, features in spatial databases such as remotely sensed images may often not appear as a conventional mixture in which the distribution of each component is a density function with a fixed point as its “center” (mean), as in usual statistical distributions. The more likely situation is that they may take on a mixture whose

components may contain features such as roads and rivers. It is thus inappropriate to model them by a single conventional distribution model. Many of the conventional feature mining approaches, however, are based on such a conventional distribution (especially the multivariate Gaussian distribution model) and the number of the features usually needs to be known a priori. Since the number of significant features is generally not known a priori, then the conventional single-distribution approach is not suitable to model and optimally extract features in an image. As pointed out in Richards and Xia (1999, p. 261), information classes of interest often do not appear as single distributions but rather a set of constituent spectral classes or sub-classes. So we not only need a descriptive approach to characterize spatial features in remotely sensed images but also an effective method for mining those features.

An important application of the RCMD algorithm is to identify features or classes in multi-dimensional data sets. It can easily be observed that a regression class actually corresponds to a feature so that the variable y , called the response (dependent) variable, is a function of the other variables x_1, x_2, \dots, x_p , called the explanatory (independent) variables, i.e., $y = f(x_1, x_2, \dots, x_p)$. However, in many practical situations, we may not be able to explicitly express a variable by other relevant variables. Thus, a feature may only be characterized by an equation $F(z_1, z_2, \dots, z_p) = 0$ with respect to p variables z_1, z_2, \dots, z_p , where dependent and independent variables are indistinguishable. Obviously, such a representation of features generalizes that of the regression-class framework. It is more flexible and effective in the mining of features in remotely sensed images. Leung et al. (2006b) further extend the RCMD method into the RFMM method to perform feature mining in more general situations. The RFMM method is first described in the following subsection and then the version with a genetic algorithm for more efficient performance is discussed in Sect. 2.5.4.

2.5.3 A Spatial-Feature Mining Model (RFMM) Based on Regression-Class Mixture Decomposition (RCMD)

Within the RFMM framework, the spatial feature to be mined should first be determined. The shape distribution is extended on the regression-class concepts in RCMD. Let $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be a p -dimensional data set, $\mathbf{x}_k = (x_{k1}, \dots, x_{kp})^T \in \mathbf{R}^p$. Assume that \mathbf{x}_k follows a distribution $g(\mathbf{x}_k; \boldsymbol{\theta})$ with probability $1 - \varepsilon$ ($0 < \varepsilon < 1$) and another distribution $h(\mathbf{x}_k; \boldsymbol{\theta})$ with probability ε for the outlier, where $\boldsymbol{\theta}$ is the parameter vector of the feature to be estimated. Thus, data samples are identically distributed with the common density:

$$f(\mathbf{x}_k; \boldsymbol{\theta}) = (1 - \varepsilon) \cdot g(\mathbf{x}_k; \boldsymbol{\theta}) + \varepsilon \cdot h(\mathbf{x}_k; \boldsymbol{\theta}). \quad (2.80)$$

According to the strategy of RCMD, the search process is to maximize the model-fitting function:

$$Q(\boldsymbol{\theta}) = \sum_k \log(g(\mathbf{x}_k; \boldsymbol{\theta}) + t), \quad (2.81)$$

where $t > 0$ is called a *partial model* and is selected by the method suggested in RCMD. According to the derivation in Leung et al. (2001a), the partial model t actually corresponds to the partial information about the outlier distribution $h(\mathbf{x}_k; \boldsymbol{\theta})$. Although outliers with respect to an underlying model $g(\mathbf{x}_k; \boldsymbol{\theta})$ exist inevitably in reality and the knowledge on the whole shape of the outlier distribution is usually unknown, we can approximately represent their existence by introducing a positive number t and use its value as a reduction of the information about the outlier as a whole. If the partial model t does not appear in (2.81), that is $t = 0$, then the method determined by (2.81) is the ordinary maximum likelihood (ML) method, which is not robust. However, once we have $t > 0$, the resulting method is fairly robust. Here, the shape of the distribution, controlled by the parameter $\boldsymbol{\theta}$, represents the feature structures hidden in the mixture.

As depicted in Fig. 2.29, spatial features can generally be categorized into several basic shapes, such as the simple Gaussian classes, linear structures, curvilinear structures, ellipsoidal structures, and other complicated structures integrated with domain specific knowledge. Specifically we have:

1. Simple Gaussian class (Fig. 2.29a)

The density corresponding to the Gaussian feature in a data set \mathbf{X} can be expressed as:

$$g(\mathbf{x}_k; \boldsymbol{\theta}) = \frac{1}{(\sqrt{2\pi})^p \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}d^2(\mathbf{x}_k)\right), \quad (2.82)$$

$$d^2(\mathbf{x}_k) = (\mathbf{x}_k - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_k - \boldsymbol{\mu}), \quad (2.83)$$

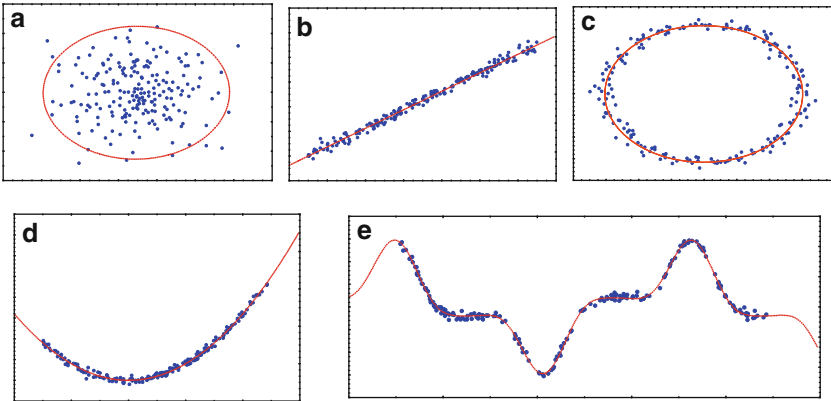


Fig. 2.29 The distributions of various spatial features (a) Simple Gaussian class. (b) Linear structure. (c) Ellipsoidal structure. (d) General curvilinear structure. (e) Complex structure

where $d^2(\mathbf{x}_k)$ is the square of the Mahalanobis distance, and Σ is the covariance matrix such that the parameter vector θ to be searched is the mean vector $\boldsymbol{\mu}$ and the covariance matrix Σ .

2. Linear structure (Fig. 2.29b)

In multi-dimensional space, linear features can be characterized by the following distribution with parameter vector $\boldsymbol{\theta} = (\boldsymbol{\beta}^T, \sigma)^T$:

$$g(\mathbf{x}^k; \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{r_k^2(\boldsymbol{\beta})}{2\sigma^2}\right), \quad (2.84)$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_p)^T$ is the coefficient vector of the following linear equation:

$$\beta_0 + (\beta_1, \dots, \beta_p)\mathbf{x} = 0, \quad (2.85)$$

and r_k denotes the residuals of data \mathbf{x}_k with respect to (2.85):

$$r_k = \beta_0 + (\beta_1, \dots, \beta_p)\mathbf{x}_k, \quad (2.86)$$

σ is such that at least 98% of the points constituting the feature are contained within 3σ from the line.

3. Ellipsoidal structure (Fig. 2.29c)

In multi-dimensional space, an ellipsoidal-like structure depicted by

$$F(\mathbf{x}; \boldsymbol{\theta}) \equiv 1 - \sum_{i=1}^p \frac{(x_i - \beta_i)^2}{\gamma_i^2} = 0, \quad (2.87)$$

can also be considered, where $\mathbf{x} = (x_1, \dots, x_p)^T$, $\boldsymbol{\theta} = (\boldsymbol{\beta}^T, \boldsymbol{\gamma}^T, \sigma)^T$ is the parameter vector, $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)^T$ is the location of the center point, $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_p)^T$, and γ_i is the i -th semimajor-like axes of length. In this situation, its features are still characterized by (2.84), but the residuals become

$$r_k = 1 - \sum_{i=1}^p \frac{(x_i^k - \beta_i)^2}{\gamma_i^2}. \quad (2.88)$$

For simplicity, we consider the ellipse feature (i.e., the case $p = 2$). In this situation, the major and minor axes of the ellipse depicted by (2.87) are parallel to the coordinate axes. For more general ellipses, their equations can be transferred into (2.87) by the rotation transformation. Then the expression in (2.88) still holds.

4. General curvilinear structure (Fig. 2.29d)

For more general curvilinear structure, (2.76) is still applicable. We only need to modify the residuals r_k . As the general curve can be described by the equation $f(\mathbf{x}; \boldsymbol{\beta}) = 0$, the corresponding residuals r_k are:

$$r_k = f(\mathbf{x}_k; \boldsymbol{\beta}), \quad (2.89)$$

where f is a known function for specifying a curve and $\boldsymbol{\beta}$ is the parameter vector of the curve.

5. Complex structure (Fig. 2.29e)

Spatial features often take on a more complex shape. A simple method to represent features with complex shape is to combine simpler feature structures into an integrative one with prior knowledge. For example, a production system can be employed to determine a complex structure as follows:

$$0 = \begin{cases} f_1(\mathbf{x}), \text{ if } (\mathbf{x} \in A_1), \\ f_2(\mathbf{x}), \text{ if } (\mathbf{x} \in A_2), \\ \dots \\ f_m(\mathbf{x}), \text{ if } (\mathbf{x} \in A_m) \end{cases} \quad (2.90)$$

Moreover, more complicated structures or irregular structures, seemingly not being able to be parametrically represented, can be simulated by appropriate combinations of these simple parametric structures.

2.5.4 The RFMM with Genetic Algorithm (RFMM-GA)

Finding solution for the RFMM is essentially an optimization process that estimates the parameter vector θ of the feature structures. Mean squared error (MSE) is frequently employed as an optimization criterion. The disadvantages of many of the conventional optimization methods are their computational complexities and their prone to local minima. It, in particular, becomes more difficult when complex distributions integrated with domain knowledge in symbolic forms are encountered in optimization. The use of more flexible methods such as genetic algorithms (GA) is often necessary.

Genetic algorithms (GA) are highly parallel and adaptive search processes based on the principles of natural selection (Holland 1975; Goldberg 1989; Zhang and Leung 2003) (see Chap. 3 for a more formal discussion of GA). Genetic operators (namely selection, crossover and mutation) are applied to evolve a population of

coded solutions (strings /chromosomes) in an iterative fashion until the optimal population is obtained. GA is thus a multi-point search algorithm which seeks the optimal solution with the highest value of a fitness function. For example, in order to solve the optimization problem in (2.81) in which $g(\mathbf{x}^k; \boldsymbol{\theta})$ is defined by (2.82) or (2.84), the function in (2.81) is selected as the fitness function to be maximized. The GA starts with a population of individuals (chromosomes) representing the parameter vector $\boldsymbol{\theta} = (\theta_1, \theta_2 \dots, \theta_l)^T$ which is encoded as a string of finite length. A chromosome is usually a binary string of 0's and 1's. For example, suppose the binary representation of $\theta_1, \theta_2 \dots, \theta_l$ for 5-bit strings are 10110, 00100, ..., 11001, respectively. Then the string $s = 10110\ 00100 \dots 11001$ is a binary representation of $\boldsymbol{\theta} = (\theta_1, \theta_2 \dots, \theta_l)^T$ and forms a one-to-one relation with $\boldsymbol{\theta}$. The q -tuple of individual strings (s_1, \dots, s_q) is said to be a population S in which each individual $s_i \in S$ represents a feasible solution of the problem in (2.81). The randomly generated binary strings then form the initial population to be evolved by the GA procedure, i.e., by the GA operators briefly outlined as follows:

1. Selection. It is the first operator by which individual strings are selected into an intermediate population (termed mating pool) according to their proportional fitness obtained from the fitness function. The *roulette wheel selection* technique is employed in such a way that strings with higher fitness would have higher probability to be selected for reproduction.
2. Crossover. After selection, two individuals can exchange materials at certain position(s) through the crossover operator. Crossover is a recombination mechanism to explore new solutions. The crossover operator is applied with some probability P_c . Single-point, multi-point, or uniform crossover may be employed. In practice, single-point crossover is simpler and more popular. First, individuals of the intermediate population are paired up randomly. Individuals of each pair (*parents*) are then combined, choosing one point in accordance with a uniformly distributed probability over the length of the individual strings and cutting them in two parts accordingly. The two new strings (*offspring*) are formed by the juxtaposition of the first part of one parent and the last part of the other parent.
3. Mutation. After crossover, the mutation operator is applied with uniform probability P_m . Mutation operates independently on each offspring by probabilistically perturbing each bit string. In other words, it alters the genetic code (e.g., from 0 to 1 or 1 to 0) of an individual at a certain randomly generated position. The mutation operator helps to prevent the irrecoverable loss of potentially important genetic material in an individual.

The basic procedure of the GA-based optimization for parameter estimation of the RFMM is depicted in Fig. 2.30. The GA search aims at the maximization of Q in (2.81). The parameter $\boldsymbol{\theta}$ is estimated while Q attains its maximum through the GA-based evolution. The spatial feature specified by $\boldsymbol{\theta}$ is thus successfully mined from the image.

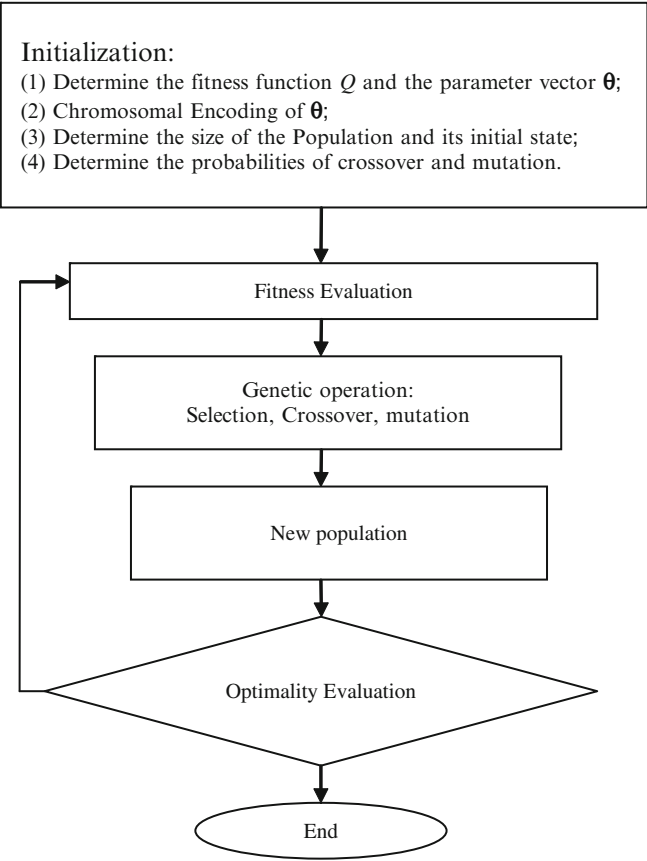


Fig. 2.30 RFMM-GA optimization algorithm

2.5.5 Applications of RFMM-GA in the Mining of Features in Remotely Sensed Images

For substantiation, the first two numerical experiments involve the extraction of one and two ellipsoidal features in simulated data sets contaminated with noise, and the third experiment deals with the automatic detection of linear features in a real-life remotely sensed image.

To simplify our discussion, the default set up of the RFMM-GA is specified as follows: the partial model level $t = 0.1$, $q = 300$, $P_c = 0.8$, $P_m = 0.5$.

2.5.5.1 Experiment 2.4 Ellipsoidal Feature Extraction from Simulated Data

In this experiment, the RFMM-GA is employed to extract features with ellipsoidal shape from simulated data sets contaminated with noise. It is actually a special clustering approach for estimating and extracting patterns. As shown in Fig. 2.31,

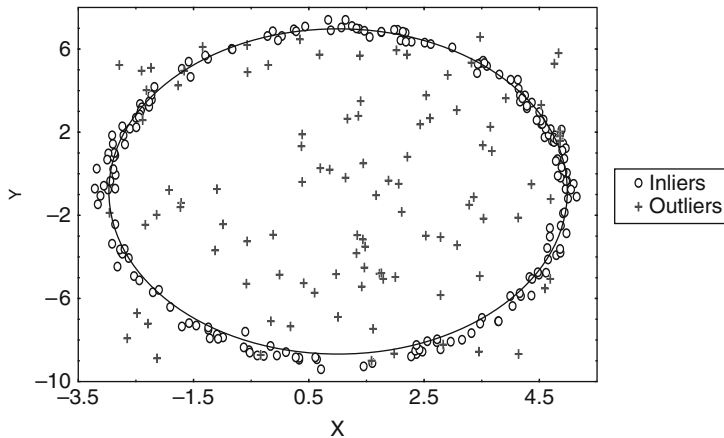


Fig. 2.31 Extraction of ellipsoidal feature

there is an ellipsoidal feature in a two-dimensional feature space with a lot of noisy points distributed randomly around it. The parameters of the true model in (2.87) are $p = 2$, $\beta_1 = 1 = -\beta_2$, $\gamma_2 = 8 = 2\gamma_1$, and σ in (2.84) is selected as 0.5. There are 300 points in Fig. 2.26, i.e., $n = 300$, in which 200 points (inliers) are generated randomly from the true model with ellipsoidal feature, and 100 points (outliers) are uniform noise. Applying RFMM-GA, the feature parameter θ can be acquired. In this experiment, the obtained parametric estimation includes the center point: $(\hat{\beta}_1, \hat{\beta}_2) = (0.999, -1.025)$; semi-major axes of length: $\hat{\gamma}_1 = 3.989$, $\hat{\gamma}_2 = 8.031$; and $\hat{\sigma} = 0.502$. With these unraveled parameters, the fitness value Q in (2.81) attains its maximum at -124.379 , and the feature is successfully mined.

2.5.5.2 Experiment 2.5 Extraction of Two Ellipsoidal Features from Simulated Data

To further illustrate the effectiveness of the RFMM-GA, this experiment is designed for the extraction of two ellipsoidal features in a data set contaminated with noise (Fig. 2.32). In the data set, 200 points come from the ellipsoidal feature characterized by the equation: $x^2/2^2 + y^2/1^2 = 1$; 200 points come from the ellipsoidal feature characterized by the equation: $(x-1)^2/1^2 + (y-5)^2/2^2 = 1$; and the other 100 points are noise. The first ellipsoidal feature in (2.87) unraveled by the RFMM-GA has the parameter estimates: $(\hat{\beta}_1, \hat{\beta}_2) = (-0.049, -0.009)$, $(\hat{\gamma}_1, \hat{\gamma}_2) = (2.023, 1.050)$, $\hat{\sigma} = 0.20$, and the fitness value Q in (2.81) attains its maximum -1952.813 at $t = 0.005$. The corresponding data points are then removed from the data set. The RFMM-GA is again applied to unravel the second ellipsoidal feature with parameters: $(\hat{\beta}_1, \hat{\beta}_2) = (1.040, 5.007)$, $(\hat{\gamma}_1, \hat{\gamma}_2) = (1.008, 2.013)$,

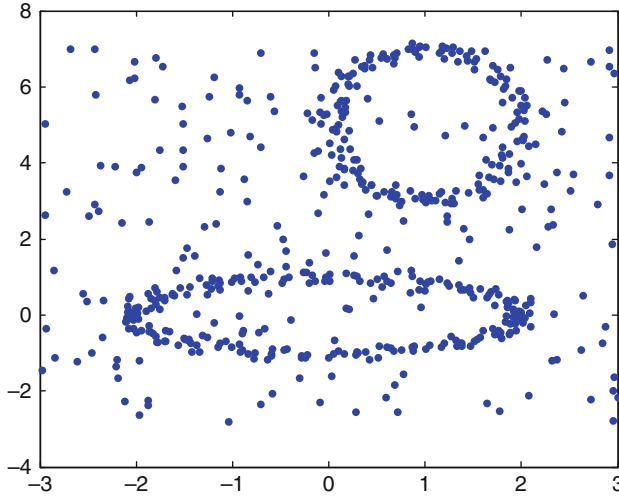


Fig. 2.32 Extraction of two ellipsoidal features

$\hat{\sigma} = 0.20$, and Q has its maximum -1277.918 at $t = 0.005$. This clearly shows that the RFMM-GA can effectively extract multiple features from noisy data sets.

2.5.5.3 Experiment 2.6 Linear Feature Extraction from a Satellite Image

A lineament in a feature space is defined as a simple or composite linear feature whose parts are aligned in a rectilinear or slightly curvilinear manner which might indicate the existence of some kind of spatial structures. Classical lineament detection methods are mainly based on gradient or Laplacian filtering which often generate a large amount of false edges and fail to link together missing occluding parts combined with the use of thresholds. Though some improvements have been achieved by applying a Hough transform to the threshold image, more recent approaches attempt to circumvent the problem by extracting the gray level in high variability through filtering techniques. Neural network models, such as adaptive resonance theory (ART), multilayer perceptron with back propagation (MLP-BP), and cellular neural networks (CNN), have also been proposed to extract connected edges (Basak and Mahata 2000; Lepage et al. 2000; Wong and Guan 2001). However, all of these approaches could only produce good results in detecting small scale edge features, but are of very limited use in the detection of linear or non-linear features, especially when lineaments have a fuzzy, gleaming, or broken appearance in aerial or satellite images (Man and Gath 1994).

Since features can be parametrically defined in the feature space under RFMM-GA, it can then provide a framework to parametrically extract spatial features from remotely sensed images. By the RFMM-GA method, the fittest linear features are successively searched and extracted from the feature space by stepwise

decomposition. The RFMM-GA is supported by robust statistical technique which could reduce the interference of adjacent features and noisy points, and enable a reliable discrimination of linear features without any a priori knowledge about the number involved. Finally, linear features are mined and characterized by the associated parameters.

Figure 2.33 depicts the result of an experiment on the extraction of lineaments, defined by (2.85), by the RFMM -GA from a real-life satellite image.

Figure 2.34a depicts the original imagery of TM band 5 in another experiment located in Guangzhou, China, acquired on January 2, 1999. Three lineaments are

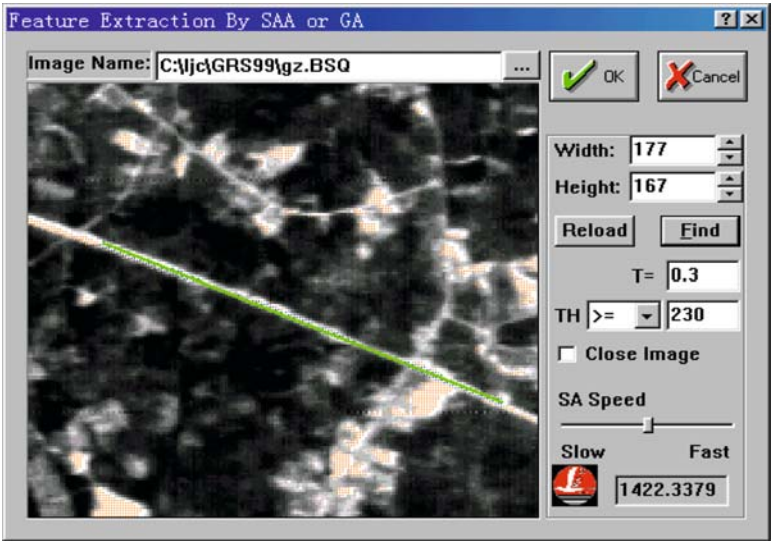


Fig. 2.33 Feature extraction system with RFMM

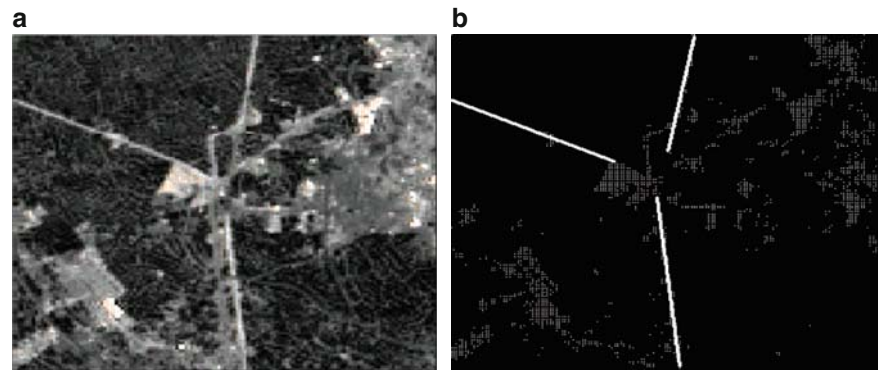


Fig. 2.34 Lineament extraction from satellite imagery (a) Original TM5 imagery (b) Results of lineament extraction

apparently three highways intersecting at a small town in the image. The features are first separated from the background with the threshold segmentation approach. Then according to the feature distribution of the lineaments, the targets are extracted by the stepwise search of the RFMM-GA. The three highways are successfully mined from the blurred imagery (Fig. 2.34b). These two experiments demonstrate that RFMM-GA provides a novel framework for feature extraction in remotely sensed images.

2.6 Cluster Characterization by the Concept of Convex Hull

2.6.1 A Note on Convex Hull and its Computation

In the search for spatial clusters, we sometimes may not have any idea about the exact location and size of a cluster, particularly in databases with undefined or ill-defined spatial boundaries. In some applications, we might just need to discover and delimit a cluster (a particular spatial concentration or hotspot) in real-time. The discovery of disease concentration, particularly the time varying concentration and spread of epidemics such as SARS and avian flu, is a typical example. Such study might not be interested in the partition of the whole data set but the discovery of localized incidence of excessive rate (Lawson 2001). Under some situations, we might need to compute the cluster diameter or to determine whether a point in space belong to a cluster. All of these tasks need a formal approach for cluster characterization and detection in spatial databases. It is proposed in here the method of convex hull computation formulated by Leung et al. (1997a) to detect spatial clusters. The basic idea is to encompass a cluster by a convex hull in high dimensional space.

To facilitate our discussion, I first give some notions of convex hulls and their computations. Let $\mathbf{S} = \{p^{(1)}, p^{(2)}, \dots, p^{(M)}\}$ be a set of M points in \mathbf{R}^N . The convex hull of \mathbf{S} , denoted as $C(\mathbf{S})$, is the smallest convex set that contains \mathbf{S} . Specifically, $C(\mathbf{S})$ is a polygon in the planar case, and a polyhedron in the three-dimensional (3-D) case. In general, $C(\mathbf{S})$ can be described in terms of one of the following characteristics:

1. The *faces* of $C(\mathbf{S})$, or equivalently, the boundary of $C(\mathbf{S})$ denoted as $\text{Bound}(\mathbf{S})$.
2. The *vertex* set, $\text{Ver}(\mathbf{S})$, which is the minimum subset of \mathbf{S} such that $C[\text{Ver}(\mathbf{S})] = C(\mathbf{S})$.
3. The *set of hyperplanes*, denoted as $\mathbf{H}^*(\mathbf{S})$, by which $C(\mathbf{S})$ becomes the intersection of the closed-half spaces bounded by $\mathbf{H}^*(\mathbf{S})$.

Thus, three types of convex hull computation problems with respect to the above-stated characteristics can be formally stated as:

Problem 1: to find the boundary set $\text{Bound}(\mathbf{S})$.

Problem 2: to determine the vertex set $\text{Ver}(\mathbf{S})$.

Problem 3: to specify the hyperplanes $\mathbf{H}^*(\mathbf{S})$.

These three problems are closely related to each other. Each of them, however, has its own concern and applications. Over the years, much effort has been devoted to develop algorithms for convex hull computation which can generally be classified into two different approaches: computing the exact convex hull (Atallah 1992; Bentley et al. 1993) and computing an approximate convex hull (Bern et al. 1992; Guibas et al. 1993).

For computing the exact convex hull by a serial computer, it has been shown that the problem can be solved in the planar and the 3-D cases with a time complexity of $O(M \log M)$ if all the points $p^{(i)}$ are given (the *off-line* problem) (Graham 1972; Preparata and Hong 1977), or with a complexity of $O(\log M)$ if the points are given one by one and the convex hull is updated after each point is added (the *on-line* problem) (Preparata 1979). Bentley et al. (1993) propose a novel algorithm that computes the convex hull in N -dimensional space in $2MN + O(M^{1-1/N} \log^{1/N} M)$ expected scalar comparisons, which represents a substantial improvement over the previous best result of $2^{N+1}NM$ (Golin and Sedgewick 1988). Wennmyr (1989) presents a neural network algorithm which computes an exact convex hull in $O(M)$ time *off-line*, and $O(\log M)$ time *on-line* in the planar case. The respective performances are $O(hM)$ and $O(M)$ in the 3-D case, where h is the number of faces in the convex hull.

There are essentially two kinds of algorithms for computing an approximate convex hull. The first kind can be classified as *robust algorithms* which compute the convex hull with imprecise computation. The basic geometric tests needed to compute the convex hull are considered unreliable or inconclusive when implemented with imprecise computations (e.g., ordinary floating-point arithmetic). Such algorithms aim at constructing a convex hull very close to containing all the points under consideration. The algorithms of this kind are often much more complicated than those for computing the exact convex hull.

The second kind of algorithms for computing an approximate convex hull is the *approximate algorithms*. The geometric tests are considered as reliable and conclusive. Such algorithms compute a convex hull that closely approximates the exact one (Bern et al. 1992). Despite of losing a certain degree of accuracy in computing the convex hull, approximate algorithms have in general very low complexity but very high computation efficiency. They would be particularly useful in applications where the speed rather than the accuracy of computing a convex hull is of major concern, or generating the exact convex hull is not necessary or impossible (e.g., when the data involved in the set of points are inherently not exact).

Leung et al. (1997a) employ a neural-network approach to develop an approximate algorithm for computing approximate convex hulls in the general N -dimensional space. It solves the *off-line* problem with a linear time complexity of $O(M)$, and the *on-line* problem with $O(1)$ time complexity. Its advantages are: First, unlike the known linear expected-time complexity algorithm (Bentley et al. 1993) which might not keep linear time complexity for the worst case (it could even be much worse), the convex hull computing neural network (CHCNN) always keep linear time

complexity for any case. Second, the massively parallel processing capability of the neural network makes the derived algorithm developed to be real-time in nature. This real-time processing capability is of particular importance and is required in a wide variety of applications related to adaptive and real-time processing. For example, in the collision avoidance applications (Hwang and Ahuja 1993), a robot is to be controlled to move automatically in an environment involving variable obstacles. Assume the obstacles are polyhedrons. The problem then can be deduced to determining in real-time the condition (the control strategy) under which the two convex hulls, one being the obstacle(s) and the other the range of the robot's motion, do not intersect. This then requires the real-time construction of the related convex hulls. Third, once the neural network is implemented as a physical device, it becomes extremely direct and handy to use it in various applications, such as judging if a given point (e.g., suspected outlier) belongs to a cluster, and computing cluster diameter (e.g., extent of spread of a contagious disease) when the given points constitutes a set of samples.

2.6.2 Basics of the Convex Hull Computing Neural Network (CHCNN) Model

Let $\mathbf{n} = (n_1, n_2, \dots, n_N)^T \in \mathbf{R}^N$ be a unit vector. For any real number α , the set H defined by

$$H = \{x \in \mathbf{R}^N : \langle \mathbf{n}, x \rangle = \alpha\} \quad (2.91)$$

is called a hyperplane and the set \bar{H} defined by

$$\bar{H} = \{x \in \mathbf{R}^N : \langle \mathbf{n}, x \rangle \leq \alpha\} \quad (2.92)$$

is called a closed half-space bounded by H . In the case, the vector \mathbf{n} is said to be the normal vector of H .

Given any set S of finite number of points in \mathbf{R}^N , $C(S)$ can be expressed as the intersection of a finite number of closed half-space bounded by certain hyperplanes.

A hyperplane H is said to be a *supporting hyperplane* of $C(S)$ if $S \subseteq \bar{H}$ and H itself contains at least one point of S . Therefore a supporting hyperplane supports $C(S)$ in a specific direction. Every point in $H \cap S$ is referred to as a *supporting point* of $C(S)$, and any intersection of $H \cap C(S)$ is referred to as a *face* of $C(S)$. Any supporting point of $C(S)$ clearly lies on the boundary of $C(S)$. A supporting point p is a vertex of $C(S)$ if there do not exist two different points a, b in $C(S)$ such that p lies on the open line segment $]a, b[$ (i.e., no $\beta \in (0, 1)$ exists such that $p = (1 - \beta)a + \beta b$).

The CHCNN generates two specific approximations of the convex hull $C(S)$: one is inscribed within $C(S)$ and the other circumscribes $C(S)$ in the geometric

sense. These two types of approximate convex hulls are specified by the following definitions:

Definition 2.4. A convex hull C_1 is said to be an inscribed approximation of $C(S)$ if $C_1 \subseteq C(S)$ and any vertices of C_1 are on the boundary of $C(S)$. A convex hull C_2 is said to be a circumscribed approximation of $C(S)$ if $C(S) \subseteq C_2$ and every face of C_2 contains at least a vertex of $C(S)$.

In Figs. 2.35 and 2.36, all convex hulls demarcated by thin lines are inscribed approximations of $C(S)$ and those demarcated by bold lines are circumscribed approximations of $C(S)$. The line with medium width represents the $C(S)$.

The CHCNN developed in Leung et al. (1997a) is motivated by the following observations: every vertex (say, p) of the convex hull $C(S)$ must be supporting point and therefore, there is a direction vector \mathbf{n} in which p will maximize the inner product $\langle \mathbf{n}, p^{(i)} \rangle$ among all the $p^{(i)}$ s in S . With finite points in S , all vertices of $C(S)$ can be uniquely recognized in terms of the maximization procedure with a finite number of direction vectors. The basic idea in developing the CHCNN then is to yield the vertices of $C(S)$ through the maximization process via a prespecified set of direction vectors.

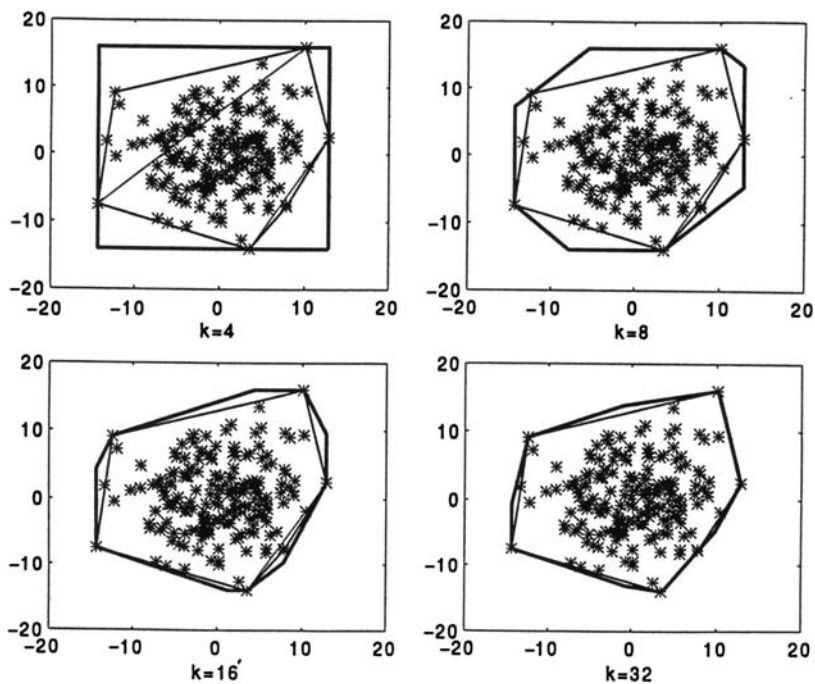


Fig. 2.35 The $C(S)$ and its inscribed and circumscribed approximations obtained by the CHCNN: case 1

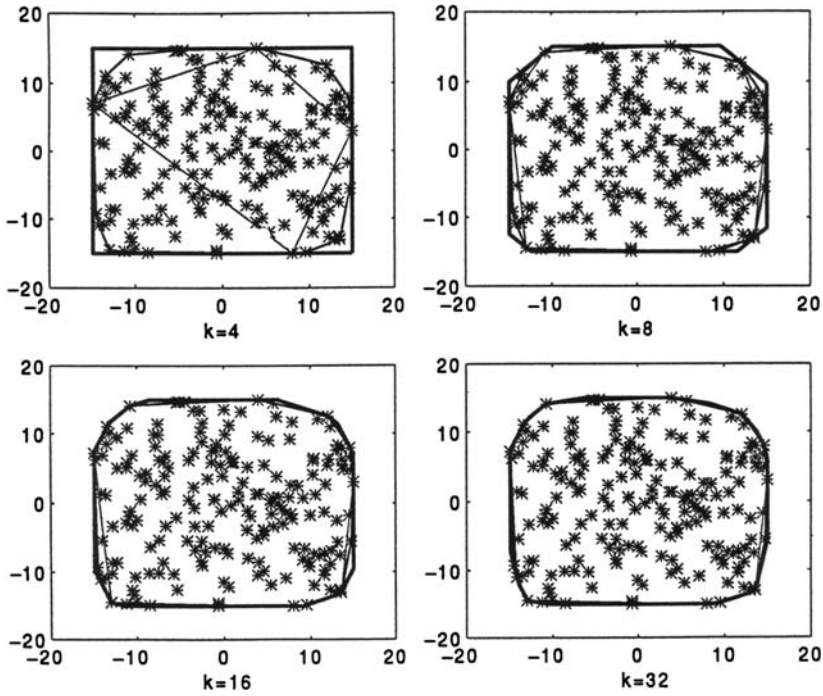


Fig. 2.36 The $C(S)$ and its inscribed and circumscribed approximations obtained by the CHCNN: case 2

The following Lemma Underlies the CHCNN:

Lemma 2.1. Let $\mathbf{U} = \{n^{(1)}, n^{(2)}, \dots, n^{(k)}\}$ be a given set of unit direction vectors in \mathbf{R}^N . If $i(j) \in \{1, 2, \dots, M\}$ is the index such that

$$\langle \mathbf{n}^{(j)}, p^{[i(j)]} \rangle = \max_{1 \leq i \leq M} \left\{ \langle \mathbf{n}^{(j)}, p^{(i)} \rangle \right\}, \quad (2.93)$$

we denote

$$\theta_j = \langle \mathbf{n}^{(j)}, p^{[i(j)]} \rangle \quad (2.94)$$

$$H^{(j)} = \left\{ \mathbf{x} \in \mathbf{R}^N : \langle \mathbf{n}^{(j)}, \mathbf{x} \rangle = \theta_j \right\} \quad (2.95)$$

$$\mathbf{V}^* = \left\{ p^{[i(1)]}, p^{[i(2)]}, \dots, p^{[i(k)]} \right\} \quad (2.96)$$

and,

$$\mathbf{H}^* = \cup \{H^{(j)} : j = 1, 2, \dots, k\}, \quad (2.97)$$

then

1. $C_H(\mathbf{H}^*) = \cap_{j=1}^k \bar{H}^{(j)}$ is a circumscribed approximation of $C(\mathbf{S})$.
2. $C(\mathbf{V}^*)$ is an inscribed approximation of $C(\mathbf{S})$.

(See Leung et al. (1997) for the proof)

Lemma 2.1 indicates that for a pre-specified set of directions \mathbf{U} , as long as the θ_j and $p^{[i(j)]}$ defined by (2.93) and (2.94) are known, the convex hulls $C(\mathbf{V}^*)$ and $C_H(\mathbf{H}^*)$ provide respectively two approximations of $C(\mathbf{S})$ in the inscribed and circumscribed manners. Furthermore, in this case, (2) in Lemma implies that the set \mathbf{V}^* defined by (2.96) offers a very good approximation to the vertex set $Ver(\mathbf{S})$, and hence to the convex hull under Problem 2. Also, (1) implies that every $H^{(j)}$ is a supporting hyperplane of the convex hull $C(\mathbf{S})$, and consequently yields an approximate solution to the convex hull under Problem 3 stated in Sect. 2.6.1.

2.6.3 The CHCNN Architecture

Given a set of k unit direction vectors $\mathbf{U} = \{n^{(1)}, n^{(2)}, \dots, n^{(k)}\}$. According to Lemma 2.1, the aim is then to build an appropriate neural network such that after adaptive training, the network can yield the vertex set \mathbf{V}^* and the hyperplanes \mathbf{H}^* . The network is the CHCNN shown in Fig. 2.37.

Topologically, CHCNN consists of one input layer of N neurons and one output layer of k neurons. Similar to the adaptive resonance theory (ART) developed by Carpenter and Grossberg (1987), the two layers of neurons communicate via a feedforward connection W and a feedback connection T . The input neurons are all McCulloch–Pitts type with zero threshold and linear input–output activation

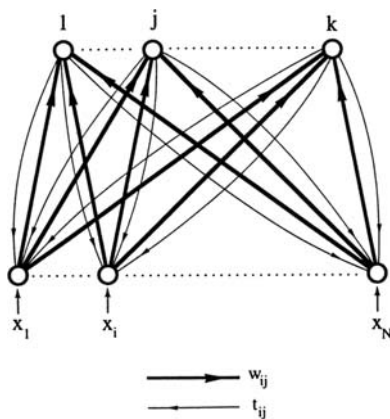


Fig. 2.37 The CHCNN architecture

function, but the output neurons all have nonzero thresholds and the hard-limiter input–output activation function defined by

$$f(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{if } x \leq 0. \end{cases} \quad (2.98)$$

Let w_{ij} and t_{ij} be, respectively, the feedforward and feedback connections (weights) between neuron i in the input layer and neuron j in the output layer. Let θ_j be the threshold value attached to the output neuron j . Denote

$$\begin{aligned} w^{(j)} &= (w_{1j}, w_{2j}, \dots, w_{Nj}), \\ t^{(j)} &= (t_{1j}, t_{2j}, \dots, t_{Nj}). \end{aligned}$$

In the CHCNN, the feedforward connection $w^{(j)}$ is fixed as the j th prespecified direction under $\mathbf{n}^{(j)}$. The feedback connection vector $t^{(j)}$ and the threshold θ_j are trained adaptively to yield the supporting point $p^{[i(j)]}$, defined in (2.93), and the maximum value defined in (2.94) [or equivalently, the hyperplane $H^{(j)}$ defined in (2.95)], respectively. Consequently, after training, the CHCNN is capable of yielding the vertex set \mathbf{V}^* and the hyperplanes \mathbf{H}^* specified in Lemma 2.1.

A parameter-setting rule for \mathbf{U} and a training algorithm for T and θ are as follows:

2.6.3.1 Parameter-Setting and Training Rules

The CHCNN is inherently dependent on the setting of the direction vectors \mathbf{U} (which specify the direction of the supporting hyperplanes), and the training rule for adjusting the weights T (which record the supporting points) and the thresholds θ (which control the positions of the supporting hyperplanes).

1. *Setting of \mathbf{U} and \mathbf{W} .* Since every supporting hyperplanes $H^{(j)}$ bounds the convex hull in a given direction $\mathbf{n}^{(j)}$, so a very reasonable approximation should apply a set of uniformly distributed directions \mathbf{U} to direct the hyperplanes. Nevertheless, finding a uniformly distributed direction set \mathbf{U} is very difficult and complicated in high dimensions. Leung et al. (1997) suggest \mathbf{U} to be specified in such a way that they distribute regularly on a unit sphere as follows:

$$\begin{aligned} \mathbf{U} &= \mathbf{U}(\alpha) \\ &= \left\{ \mathbf{n}^{(j)} : j = 1, 2, \dots, k = 2d^{(N-1)} \right\} \end{aligned}$$

with

$$\mathbf{n}^{(j)} = \mathbf{n}^{(i_1, i_2, \dots, i_{N-1})}$$

$$\begin{aligned}
&= \left\{ \sin \left[\left(\frac{\pi i_1}{d} \right) + \alpha_1 \right] \sin \left[\left(\frac{\pi i_1}{d} \right) + \alpha_2 \right] \times \cdots \times \sin \left[\left(\frac{\pi i_{N-1}}{d} \right) + \alpha_{N-1} \right], \right. \\
&\quad \cos \left[\left(\frac{\pi i_1}{d} \right) + \alpha_1 \right] \sin \left[\left(\frac{\pi i_1}{d} \right) + \alpha_2 \right] \times \cdots \times \sin \left[\left(\frac{\pi i_{N-1}}{d} \right) + \alpha_{N-1} \right], \\
&\quad \cos \left[\left(\frac{\pi i_2}{d} \right) + \alpha_2 \right] \sin \left[\left(\frac{\pi i_3}{d} \right) + \alpha_3 \right] \times \cdots \times \sin \left[\left(\frac{\pi i_{N-1}}{d} \right) + \alpha_{N-1} \right], \\
&\quad \cos \left[\left(\frac{\pi i_3}{d} \right) + \alpha_3 \right] \sin \left[\left(\frac{\pi i_4}{d} \right) + \alpha_4 \right] \times \cdots \times \sin \left[\left(\frac{\pi i_{N-1}}{d} \right) + \alpha_{N-1} \right], \quad (2.99) \\
&\quad \dots \\
&\quad \cos \left[\left(\frac{\pi i_{(N-2)}}{d} \right) + \alpha_{N-2} \right] \times \sin \left[\left(\frac{\pi i_{N-1}}{d} \right) + \alpha_{N-1} \right], \\
&\quad \left. \cos \left[\left(\frac{\pi i_{N-1}}{d} \right) + \alpha_{N-1} \right] \right\}
\end{aligned}$$

where $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_{N-1}) \in \mathbf{R}^{N-1}$, $i_1 = 1, \dots, 2d$, $i_1 = 1, \dots, d$ for $l = 2, \dots, N-1$, and, $j = i_1 + 2d(i_2 - 1) + 2d^2(i_3 - 1) + \cdots + 2d^{N-2}(i_{N-1} - 1)$. The variable α is a rotation parameter whose components are randomly chosen. The function of α is explained in Theorem 2.4. It is shown that with such specified direction vectors \mathbf{U} , the CHCNN is always capable of yielding very accurate approximation of the convex hull $C(\mathbf{S})$. As mentioned previously, once the direction vectors \mathbf{U} is specified, the feedforward connections $\mathbf{W} = (w^{(1)}, w^{(2)}, \dots, w^{(k)})$ are fixed as the same as \mathbf{U} . That is,

$$w^{(j)} = [\mathbf{n}^{(j)}]^T, \quad j = 1, \dots, k.$$

2. A Learning Rule for T and $\boldsymbol{\theta}$. The j th neuron in the output layer of the CHCNN is said to be excited by an input x if $f(\langle w, x \rangle - \theta_j) = 1$, otherwise the neuron is said to be inhibited. The feedback connections $T = [t^{(1)}, t^{(2)}, \dots, t^{(k)}]$ and thresholds $\boldsymbol{\theta} = [\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(k)}]$ will then be adjusted according to the following learning rule.

2.6.3.2 Excited Learning Rule:

Initialize $t^{(j)}(0) = p^{(1)}$; $\theta_j(0) = \langle \mathbf{n}^{(j)}, p^{(1)} \rangle$.

Begin with $i = 1$.

- Step 1. Input $p^{(i)}$, and find all neurons excited by $p^{(i)}$ in the outer layer. Denote all the excited neurons by $J(i)$.
- Step 2. For every $j \in \{1, 2, \dots, k\}$, do the following (a and b).

(a) update $t^{(j)}$ according to

$$t^{(j)}(i) = t^{(j)}(i-1) + \Delta t^{(j)}(i)$$

with

$$\Delta t^{(j)}(i) = \begin{cases} p^{(i)} - t^{(j)}(i-1), & \text{if } j \in J(i), \\ 0, & \text{if } j \notin J(i). \end{cases}$$

(b) update θ_j according to

$$\theta_j(i) = \theta_j(i-1) + \Delta \theta_j(i)$$

with

$$\Delta \theta_j(i) = \begin{cases} \langle w^{(j)}, p^{(i)} \rangle - \theta_j(i-1), & \text{if } j \in J(i), \\ 0, & \text{if } j \notin J(i). \end{cases}$$

Step 3. If $i = M$, then terminate the learning process, otherwise, go to Step 1 with $i := i + 1$.

It is shown in Leung et al. (1997a) that the learning rule can guarantee convergence to the supporting points of $C(\mathbf{S})$ within M steps. That is, the CHCNN succeeds in M -step learning as summarized by the following theorem:

Theorem 2.2.

1. *The learning algorithm of the CHCNN converges in M steps.*
2. *The CHCNN algorithm is an on-line algorithm, processing every input in a single iteration.*
3. *The trained CHCNN yields \mathbf{V}^* and \mathbf{H}^* , such that $C(\mathbf{V}^*)$ is an inscribed approximation and $C_H(\mathbf{H}^*)$ is a circumscribed approximation of $C(\mathbf{S})$.*

From Theorem 2.2, we obtain the following conclusion:

Corollary 2.1. *The CHCNN algorithm has time complexity $O(M)$ for off-line problems and $O(1)$ for on-line problem.*

Theorems 2.3 and 2.4 below further show that $C(\mathbf{V}^*)$ and $C_H(\mathbf{H}^*)$ both actually provide very accurate approximations of $C(\mathbf{S})$.

Theorem 2.3. *Assume $d \geq 2$, \mathbf{V}^* is the supporting point set and \mathbf{H}^* is the supporting hyperplanes generated by the CHCNN, with the direction vectors \mathbf{U} defined as in (2.99). Then there is a constant K_N , which is only dependent of N , such that*

1. $\text{dist}[C_H(\mathbf{H}^*), C(\mathbf{V}^*)] \leq K_N \text{diam}(\mathbf{S}) k^{-1/(N-1)},$
2. $\text{dist}[C(\mathbf{V}^*), C(\mathbf{S})] \leq K_N \text{diam}(\mathbf{S}) k^{-1/(N-1)},$
3. $\text{dist}[C_H(\mathbf{H}^*), C(\mathbf{S})] \leq K_N \text{diam}(\mathbf{S}) k^{-1/(N-1)}.$

(See Leung et al. (1997) for the proof)

Remark 2.6. Let A and B be two subsets of \mathbf{R}^N . The diameter of the set A is defined by

$$\text{diam}(A) = \max\{\|x - y\| : x, y \in A\}.$$

The distance of a point p to the set A , denoted by $\text{dist}(p, A)$, is defined by

$$\text{dist}(p, A) = \min\{\|p - x\| : x \in A\},$$

and the distance between A and B is defined by

$$\text{dist}(A, B) = \max\left\{\max_{x \in A} \text{dist}(x, B), \max_{y \in B} \text{dist}(y, A)\right\}.$$

The distance between two sets can serve as a measure of the difference of the sets.

Theorem 2.3 says that the $C(\mathbf{V}^*)$ and $C_H(\mathbf{H}^*)$ generated by the CHCNN approximate $C(\mathbf{S})$ with the same accuracy $O(k^{-1/(n-1)})$, which is proportional to the number of neurons adopted in the CHCNN and is independent of the specified \mathbf{S} . The significance of this is twofold. First, one can determine the size of the neural network based directly on this accuracy of estimation in a given convex-hull computation application with any prespecified level of approximation. Second, it follows that the approximation accuracy of $C(\mathbf{V}^*)$ and $C_H(\mathbf{H}^*)$ can assuredly increase as k increases. Thus, any highly accurate approximation of $C(\mathbf{S})$ can be ascertained via the CHCNN. This shows further that CHCNN, as an approximate algorithm, can converge to the exact convex hull with sufficient large number of neurons.

The following theorem further explains that it is not necessary to have an infinite number of neurons in order to get an exact approximation of $C(\mathbf{S})$ via the CHCNN. Clarification of this is tightly related to another important issue: whether or not the supporting point set \mathbf{V} generated by the CHCNN is a portion of $\text{Ver}(\mathbf{S})$. An affirmative answer to this question is offered in the theorem:

Theorem 2.4. *Let $\mathbf{V}^*(\alpha)$ be the set of supporting points generated by the CHCNN with the direction vectors $\mathbf{U}(\alpha)$ defined by (2.99). Then we have the following:*

1. *For almost every α in \mathbf{R}^{N-1} (namely, every α except a zero measure set), $\mathbf{V}^*(\alpha)$ is a portion of $\text{Ver}(\mathbf{S})$.*
2. *There is a constant $K(\mathbf{S})$ such that, for almost every α in \mathbf{R}^{N-1} , $\mathbf{V}^*(\alpha) = \text{Ver}(\mathbf{S})$ whenever $k \geq K(\mathbf{S})$.*

(See Leung et al. (1997a) for the proof)

Remark 2.7. Property (2) in Theorem 2.4 shows that for any given point set \mathbf{S} , the CHCNN with a finite number of neurons is capable of almost always yielding the exact vertices of $C(\mathbf{S})$. Therefore, it provides an accurate solution to convex-hull problem 2. In this case, $C(\mathbf{V}^*)$ then provides an accurate solution to convex-hull Problem 1 stated in Sect. 2.6.1.

2.6.4 Applications in Cluster Characterization

2.6.4.1 Determining Whether a Point p is Inside $C(\mathbf{S})$, a Cluster.

Given a point p , check whether or not p belongs to $C(\mathbf{S})$ is a basic point-location problem in computational geometry. This naturally arises in applications such as collision avoidance problem for robot motion planning, and infection area detection problem in epidemics. The idea in this application is that instead of checking if $p \in C(\mathbf{S})$, we can check if p belongs to $C_H(\mathbf{H}^*)$, which is known to be a circumscribed approximation of $C(\mathbf{S})$. Obviously, the latter can easily be accomplished by the CHCNN. The main step are as follows:

Step 1. Input p into the neural network trained by \mathbf{S} .

Step 2. If there is no neuron being excited, i.e.,

$$\langle p, \mathbf{n}^{(i)} \rangle \leq \theta_i, \quad i = 1, \dots, k,$$

then, $p \in \bar{H}^{(i)}$ holds for any i . Therefore, $p \in C_H(\mathbf{H}^*)$. Otherwise, there is a neuron, denoted by j , being excited, i.e.,

$$\langle p, \mathbf{n}^{(j)} \rangle > \theta_j, \quad j = 1, \dots, k.$$

Thus, $p \notin C_H(\mathbf{H}^*)$. This also means $p \notin C(\mathbf{S})$ since $C(\mathbf{S}) \subseteq C_H(\mathbf{H}^*)$.

It should be observed that for this application, only one iteration is required by the CHCNN to determine whether p belongs to $C_H(\mathbf{H}^*)$. This is clearly an optimal property one can expect of an on-line algorithm for dynamically changing problems such as spatial spread of epidemics.

2.6.4.2 Computing the Diameter of a Cluster \mathbf{S}

The diameter of a set \mathbf{S} is defined by

$$\text{Diam}(\mathbf{S}) = \max\{\|x - y\| : x, y \in \mathbf{S}\}. \quad (2.100)$$

The problem of determining the diameter of a set \mathbf{S} occurs in various applications. For instance, in clustering techniques, the “minimum diameter K -clustering” problem can be stated in the following way:

Given a set of m points in \mathbf{R}^N , partition them into K clusters C_1, C_2, \dots, C_k such that the maximum diameter of C_i , $i = 1, \dots, K$, is as small as possible (Preparata and Shamos 1985).

The success of applying the CHCNN to this problem is in part due to the following well-known result:

Lemma 2.2. *The diameter of a set equals that of its convex hull, which in turn is the greatest distance between parallel supporting hyperplanes (Preparata and Shamos 1985).*

It should be noted that in the CHCNN developed in Subsection 2.6.2, if $\mathbf{n} \in \mathbf{U}$ is the pre-specified direction vector defined in (2.99), then the direction $-\mathbf{n}$ must also belong to \mathbf{U} . Therefore, if $\theta(\mathbf{n}), \theta^-(\mathbf{n}), t(\mathbf{n}), t^-(\mathbf{n}), H(\mathbf{n}), H^-(\mathbf{n})$ respectively denote the corresponding threshold values, supporting points and supporting hyperplanes for \mathbf{n} and $-\mathbf{n}$ in the CHCNN trained by \mathbf{S} , then $H(\mathbf{n})$ and $H^-(\mathbf{n})$ would be parallel to each other, and the distance between them is equal to $|\theta(\mathbf{n}) - \theta^-(\mathbf{n})|$. According to Lemma 2.2, we thus can use

$$\max_{\mathbf{n} \in \mathbf{U}} \{|\theta^-(\mathbf{n}) - \theta(\mathbf{n})|\} \quad (2.101)$$

as an approximation of the diameter of \mathbf{S} . However, $t(\mathbf{n})$ and $t^-(\mathbf{n})$ are both the supporting points (therefore belong to \mathbf{S}), which shows $\text{Diam}(\mathbf{S}) \geq |t(\mathbf{n}) - t^-(\mathbf{n})|$ by the definition in (2.100). From the inequality $|t(\mathbf{n}) - t^-(\mathbf{n})| \geq |\theta(\mathbf{n}) - \theta^-(\mathbf{n})|$, it then follows that a more accurate approximation of $\text{Diam}(\mathbf{S})$ should be given by

$$\max_{\mathbf{n} \in \mathbf{U}} \{|t(\mathbf{n}) - t^-(\mathbf{n})|\}. \quad (2.102)$$

The advantage of this computational method is that it is not only very easy to implement but also very efficient for solving high dimensional problems. Table 2.7 shows the simulation results for diameters of a set of ten four-dimensional point sets, with all sets containing 200 points randomly chosen and the CHCNN was run with $k = 20$. In Table 2.7, $D(\mathbf{S})$ is the exact diameter of the set \mathbf{S} , and

$$D_1(\mathbf{S}) = \max_{\mathbf{n} \in \mathbf{U}} \{|\theta^-(\mathbf{n}) - \theta(\mathbf{n})|\} \quad (2.103)$$

and

$$D_2(\mathbf{S}) = \max_{\mathbf{n} \in \mathbf{U}} \{|t(\mathbf{n}) - t^-(\mathbf{n})|\} \quad (2.104)$$

are the approximations defined respectively by (2.101) and (2.102).

Table 2.7 Diamater of a set S

S_i	$D(S_i)$	$D_1(S_i)$	$D_2(S_i)$
S_1	131.1992	130.0681	131.1992
S_2	143.9056	143.8920	143.9056
S_3	137.0960	135.9434	137.0960
S_4	138.4248	136.4933	138.4248
S_5	144.4680	142.7688	144.4680
S_6	135.6954	134.5078	135.6954
S_7	136.9296	134.8861	134.8861
S_8	146.5135	144.9885	146.5135
S_9	149.6796	149.0020	149.6796
S_{10}	146.6331	145.0052	146.6331

A pleasant and surprising result found in Table 2.7 is that $D_2(S)$ almost always yield the exact diameter of a set S . It implies that CHCNN is highly effective and efficient in computing the diameter of a cluster.



<http://www.springer.com/978-3-642-02663-8>

Knowledge Discovery in Spatial Data

Leung, Y.

2009, XXIX, 360 p. 113 illus., Hardcover

ISBN: 978-3-642-02663-8