



Module 5

Clustering



Cluster Analysis Introduction

Imagine that you are the Director of Customer Relationships at *AllElectronics*, and you have five managers working for you. You would like to organize all the company's customers into five groups so that each group can be assigned to a different manager. Strategically, you would like that the customers in each group are as similar as possible. Moreover, two given customers having very different business patterns should not be placed in the same group. Your intention behind this business strategy is to develop customer relationship campaigns that specifically target each group, based on common features shared by the customers per group. What kind of data mining techniques can help you to accomplish this task?

Unlike in classification, the class label (or *group_ID*) of each customer is unknown. You need to *discover* these groupings. Given a large number of customers and many attributes describing customer profiles, it can be very costly or even infeasible to have a human study the data and manually come up with a way to partition the customers into strategic groups. You need a *clustering* tool to help.

Clustering is the process of grouping a set of data objects into multiple groups or *clusters* so that objects within a cluster have high similarity, but are very dissimilar to objects in other clusters. Dissimilarities and similarities are assessed based on the attribute values describing the objects and often involve distance measures.¹ Clustering as a data mining tool has its roots in many application areas such as biology, security, business intelligence, and Web search.



10.1.1 What Is Cluster Analysis?

Cluster analysis or simply **clustering** is the process of partitioning a set of data objects (or observations) into subsets. Each subset is a **cluster**, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. The set of clusters resulting from a cluster analysis can be referred to as a **clustering**. In this context, different clustering methods may generate different clusterings on the same data set. The partitioning is not performed by humans, but by the clustering algorithm. Hence, clustering is useful in that it can lead to the discovery of previously unknown groups within the data.


Types of Data in Cluster Analysis

In this section, we study the types of data that often occur in cluster analysis and how to preprocess them for such an analysis. Suppose that a data set to be clustered contains n objects, which may represent persons, houses, documents, countries, and so on. Main memory-based clustering algorithms typically operate on either of the following two data structures.

- **Data matrix** (or *object-by-variable structure*): This represents n objects, such as persons, with p variables (also called *measurements* or *attributes*), such as age, height, weight, gender, and so on. The structure is in the form of a relational table, or n -by- p matrix (n objects $\times p$ variables):

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix} \quad (7.1)$$

- **Dissimilarity matrix** (or *object-by-object structure*): This stores a collection of proximities that are available for all pairs of n objects. It is often represented by an n -by- n table:



$$\begin{bmatrix} 0 & & & & \\ d(2, 1) & 0 & & & \\ d(3, 1) & d(3, 2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n, 1) & d(n, 2) & \dots & \dots & 0 \end{bmatrix} \quad (7.2)$$

where $d(i, j)$ is the measured **difference** or **dissimilarity** between objects i and j . In general, $d(i, j)$ is a nonnegative number that is close to 0 when objects i and j are highly similar or “near” each other, and becomes larger the more they differ. Since $d(i, j) = d(j, i)$, and $d(i, i) = 0$, we have the matrix in (7.2). Measures of dissimilarity are discussed throughout this section.

The rows and columns of the data matrix represent different entities, while those of the dissimilarity matrix represent the same entity. Thus, the data matrix is often called a **two-mode** matrix, whereas the dissimilarity matrix is called a **one-mode** matrix. Many clustering algorithms operate on a dissimilarity matrix. If the data are presented in the form of a data matrix, it can first be transformed into a dissimilarity matrix before applying such clustering algorithms.




1. Interval-Scaled Variables

- “What are interval-scaled variables?” Interval-scaled variables are continuous measurements of a roughly linear scale.
- Typical examples include weight and height, latitude and longitude coordinates (e.g., when clustering houses), and weather temperature. The measurement unit used can affect the clustering analysis.
- For example, changing measurement units from meters to inches for height, or from kilograms to pounds for weight, may lead to a very different clustering structure.
- In general, expressing a variable in smaller units will lead to a larger range for that variable, and thus a larger effect on the resulting clustering structure. To help avoid dependence on the choice of measurement units, the data should be standardized.



2. Binary Variables

- A binary variable has only two states: 0 or 1, where 0 means that the variable is absent, and 1 means that it is present.
- Given the variable smoker describing a patient, for instance, 1 indicates that the patient smokes, while 0 indicates that the patient does not.
- Treating binary variables as if they are interval-scaled can lead to misleading clustering results. Therefore, methods specific to binary data are necessary for computing dissimilarities.




3. Categorical, Ordinal, and Ratio-Scaled Variables

Categorical Variables :

- ❑ A categorical variable is a generalization of the binary variable in that it can take on more than two states.
- ❑ For example, map color is a categorical variable that may have, say, five states: red, yellow, green, pink, and blue.
- ❑ Let the number of states of a categorical variable be M . The states can be denoted by letters, symbols, or a set of integers, such as $1, 2, \dots, M$.
- ❑ Notice that such integers are used just for data handling and do not represent any specific ordering.



Ordinal Variables:

- A discrete ordinal variable resembles a categorical variable, except that the M states of the ordinal value are ordered in a meaningful sequence.
 - Ordinal variables are very useful for registering subjective assessments of qualities that cannot be measured objectively.
 - For example, professional ranks are often enumerated in a sequential order, such as assistant, associate, and full for professors.
 - For example, the relative ranking in a particular sport (e.g., gold, silver, bronze) is often more essential than the actual values of a particular measure.
- 



Ratio-Scaled Variables

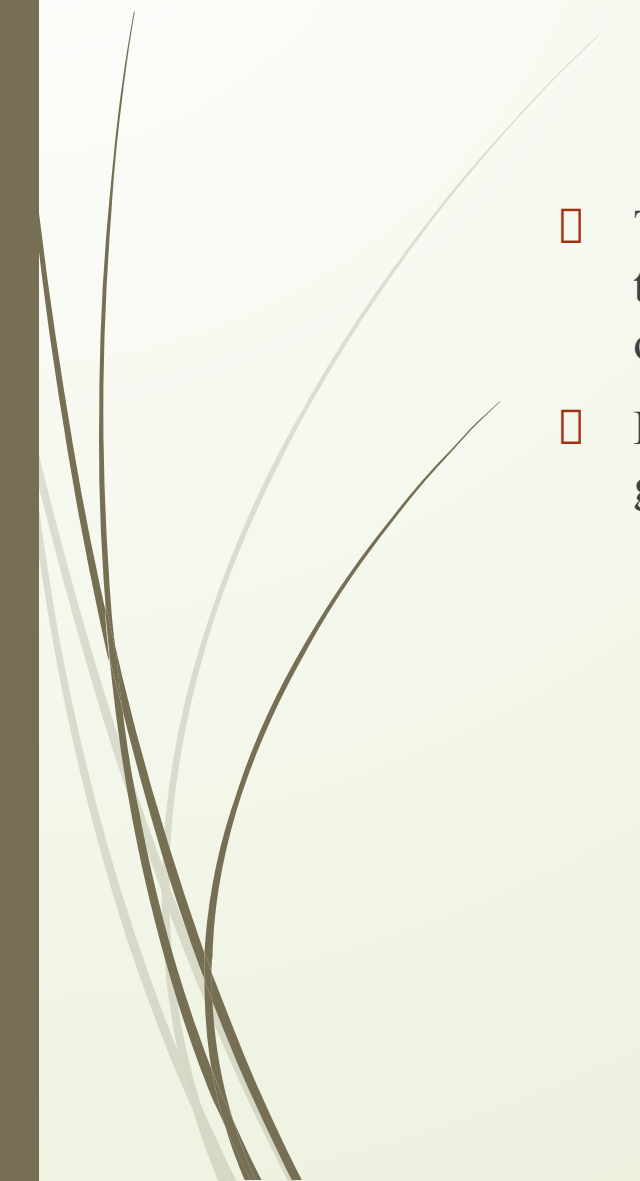
A ratio-scaled variable makes a positive measurement on a nonlinear scale, such as an exponential scale, approximately following the formula

$$Ae^{Bt} \quad \text{or} \quad Ae^{-Bt} \quad (7.14)$$

where A and B are positive constants, and t typically represents time. Common examples include the growth of a bacteria population or the decay of a radioactive element.

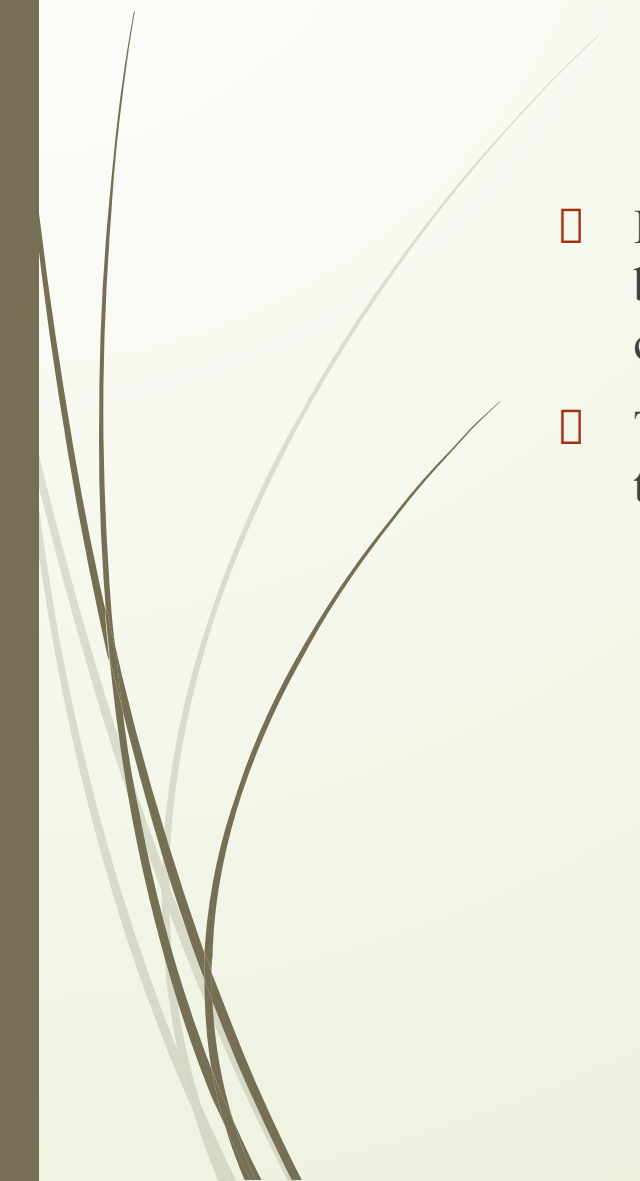


4. Variables of Mixed Types

- The dissimilarity between objects described by variables of the same type, where these types may be either interval-scaled, symmetric binary, asymmetric binary, categorical, ordinal, or ratio-scaled.
 - However, in many real databases, objects are described by a mixture of variable types. In general, a database can contain all of the six variable types listed above.
- 




5. Vector Objects

- In some applications, such as information retrieval, text document clustering, and biological taxonomy, we need to compare and cluster complex objects (such as documents) containing a large number of symbolic entities (such as keywords and phrases).
 - To measure the distance between complex objects, it is often desirable to abandon traditional metric distance computation and introduce a nonmetric similarity function.
- 



A Categorization of Major Clustering Methods



Many clustering algorithms exist in the literature. It is difficult to provide a crisp categorization of clustering methods because these categories may overlap, so that a method may have features from several categories. Nevertheless, it is useful to present a relatively organized picture of the different clustering methods. In general, the major clustering methods can be classified into the following categories



1. Partitioning Methods

7.4 Partitioning Methods

Given D , a data set of n objects, and k , the number of clusters to form, a **partitioning algorithm** organizes the objects into k partitions ($k \leq n$), where each partition represents a cluster. The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are “similar,” whereas the objects of different clusters are “dissimilar” in terms of the data set attributes.

7.4.1 Classical Partitioning Methods: k -Means and k -Medoids

The most well-known and commonly used partitioning methods are k -means, k -medoids, and their variations.

Centroid-Based Technique: The k -Means Method

The k -means algorithm takes the input parameter, k , and partitions a set of n objects into k clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. Cluster similarity is measured in regard to the *mean* value of the objects in a cluster, which can be viewed as the cluster's *centroid* or *center of gravity*.

“How does the k -means algorithm work?” The k -means algorithm proceeds as follows. First, it randomly selects k of the objects, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. It then computes the new mean for each cluster. This process iterates until the criterion function converges. Typically, the **square-error criterion** is used, defined as

$$E = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2, \quad (7.18)$$

where E is the sum of the square error for all objects in the data set; p is the point in space representing a given object; and m_i is the mean of cluster C_i (both p and m_i are multidimensional). In other words, for each object in each cluster, the distance from the object to its cluster center is squared, and the distances are summed. This criterion tries to make the resulting k clusters as compact and as separate as possible. The k -means procedure is summarized in Figure 7.2.

Algorithm: k -means. The k -means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- k : the number of clusters,
- D : a data set containing n objects.

Output: A set of k clusters.

Method:

- (1) arbitrarily choose k objects from D as the initial cluster centers;
- (2) **repeat**
- (3) (re)assign each object to the cluster to which the object is the most similar,
 based on the mean value of the objects in the cluster;
- (4) update the cluster means, i.e., calculate the mean value of the objects for
 each cluster;
- (5) **until** no change;

Figure 7.2 The k -means partitioning algorithm.

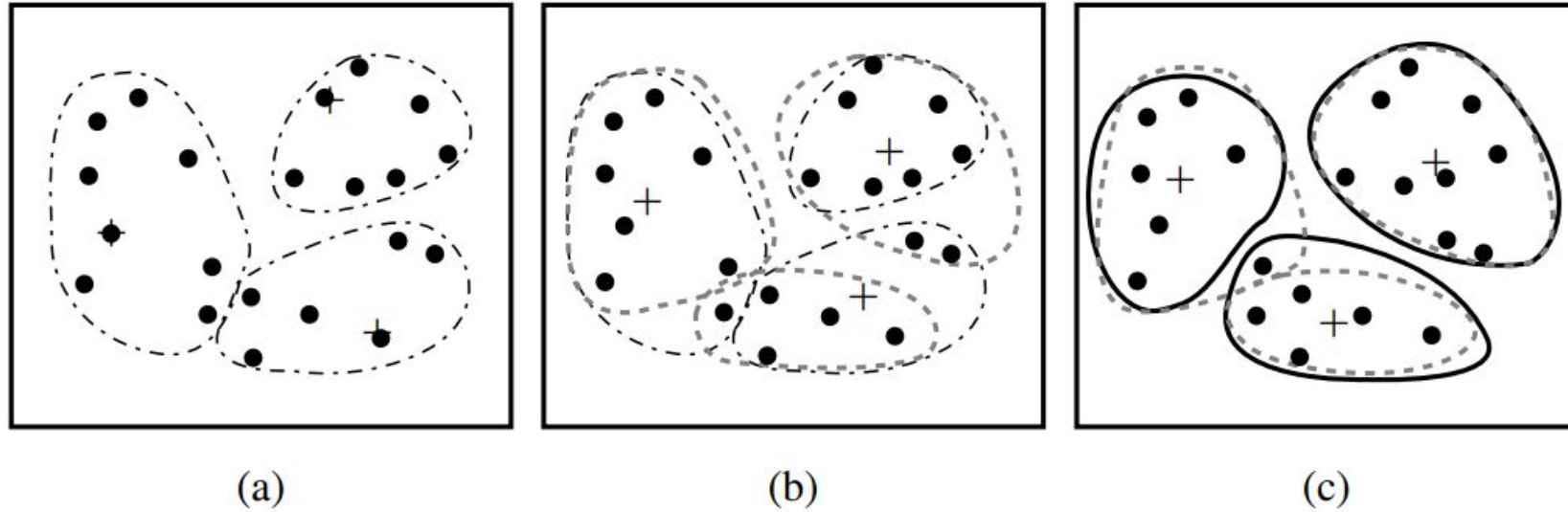



Figure 7.3 Clustering of a set of objects based on the k -means method. (The mean of each cluster is marked by a “+”.)

Representative Object-Based Technique: The k -Medoids Method

The k -means algorithm is sensitive to outliers because an object with an extremely large value may substantially distort the distribution of data. This effect is particularly exacerbated due to the use of the *square-error* function (Equation (7.18)).

“How might the algorithm be modified to diminish such sensitivity?” Instead of taking the mean value of the objects in a cluster as a reference point, we can pick actual objects to represent the clusters, using one representative object per cluster. Each remaining object is clustered with the representative object to which it is the most similar. The partitioning method is then performed based on the principle of minimizing the sum of



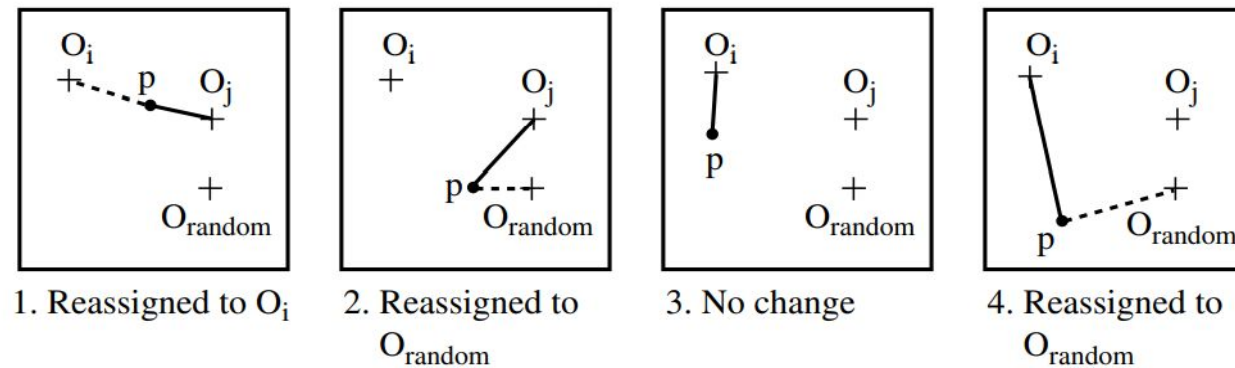
the dissimilarities between each object and its corresponding reference point. That is, an **absolute-error criterion** is used, defined as

$$E = \sum_{j=1}^k \sum_{p \in C_j} |\mathbf{p} - \mathbf{o}_j|, \quad (7.19)$$

where E is the sum of the absolute error for all objects in the data set; \mathbf{p} is the point in space representing a given object in cluster C_j ; and \mathbf{o}_j is the representative object of C_j . In general, the algorithm iterates until, eventually, each representative object is actually the **medoid**, or most centrally located object, of its cluster. This is the basis of the **k -medoids method** for grouping n objects into k clusters.

Let's look closer at k -medoids clustering. The initial representative objects (or seeds) are chosen arbitrarily. The iterative process of replacing representative objects by nonrepresentative objects continues as long as the quality of the resulting clustering is improved. This quality is estimated using a cost function that measures the average dissimilarity between an object and the representative object of its cluster. To determine whether a nonrepresentative object, \mathbf{o}_{random} , is a good replacement for a current representative object, \mathbf{o}_j , the following four cases are examined for each of the nonrepresentative objects, \mathbf{p} , as illustrated in Figure 7.4.

- **Case 1:** p currently belongs to representative object, o_j . If o_j is replaced by o_{random} as a representative object and p is closest to one of the other representative objects, o_i , $i \neq j$, then p is reassigned to o_i .
- **Case 2:** p currently belongs to representative object, o_j . If o_j is replaced by o_{random} as a representative object and p is closest to o_{random} , then p is reassigned to o_{random} .
- **Case 3:** p currently belongs to representative object, o_i , $i \neq j$. If o_j is replaced by o_{random} as a representative object and p is still closest to o_i , then the assignment does not change.
- **Case 4:** p currently belongs to representative object, o_i , $i \neq j$. If o_j is replaced by o_{random} as a representative object and p is closest to o_{random} , then p is reassigned to o_{random} .



- data object
- + cluster center
- before swapping
- after swapping

Figure 7.4 Four cases of the cost function for k -medoids clustering.




Comparison

“Which method is more robust— k -means or k -medoids?” The k -medoids method is more robust than k -means in the presence of noise and outliers, because a medoid is less influenced by outliers or other extreme values than a mean. However, its processing is more costly than the k -means method. Both methods require the user to specify k , the number of clusters.



2. Hierarchical Methods




A hierarchical clustering method works by grouping data objects into a tree of clusters. Hierarchical clustering methods can be further classified as either agglomerative or divisive, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or top-down (splitting) fashion.



7.5.1 Agglomerative and Divisive Hierarchical Clustering

In general, there are two types of hierarchical clustering methods:

- **Agglomerative hierarchical clustering:** This bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied. Most hierarchical clustering methods belong to this category. They differ only in their definition of intercluster similarity.
- **Divisive hierarchical clustering:** This top-down strategy does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster. It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions, such as a desired number of clusters is obtained or the diameter of each cluster is within a certain threshold.



Example 7.9 Agglomerative versus divisive hierarchical clustering. Figure 7.6 shows the application of AGNES (AGglomerative NESting), an agglomerative hierarchical clustering method, and DIANA (DIvisive ANALysis), a divisive hierarchical clustering method, to a data set of five objects, $\{a, b, c, d, e\}$. Initially, AGNES places each object into a cluster of its own. The clusters are then merged step-by-step according to some criterion. For example, clusters C_1 and C_2 may be merged if an object in C_1 and an object in C_2 form the minimum Euclidean distance between any two objects from different clusters. This is a **single-linkage** approach in that each cluster is represented by all of the objects in the cluster, and the similarity between two clusters is measured by the similarity of the *closest* pair of data points belonging to different clusters. The cluster merging process repeats until all of the objects are eventually merged to form one cluster.

In DIANA, all of the objects are used to form one initial cluster. The cluster is split according to some principle, such as the maximum Euclidean distance between the closest neighboring objects in the cluster. The cluster splitting process repeats until, eventually, each new cluster contains only a single object. ■

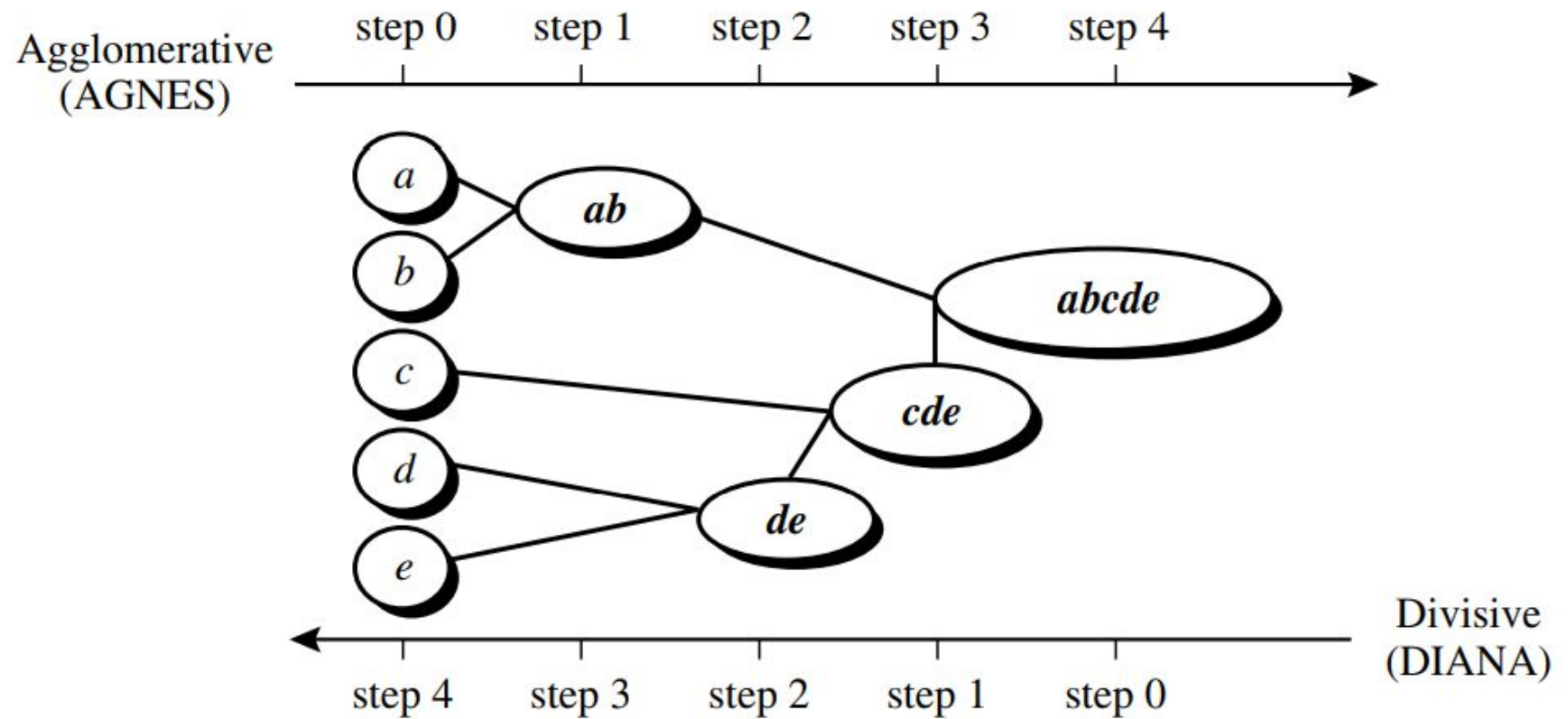


Figure 7.6 Agglomerative and divisive hierarchical clustering on data objects $\{a, b, c, d, e\}$.

- A tree structure called a dendrogram is commonly used to represent the process of hierarchical clustering. It shows how objects are grouped together step by step.

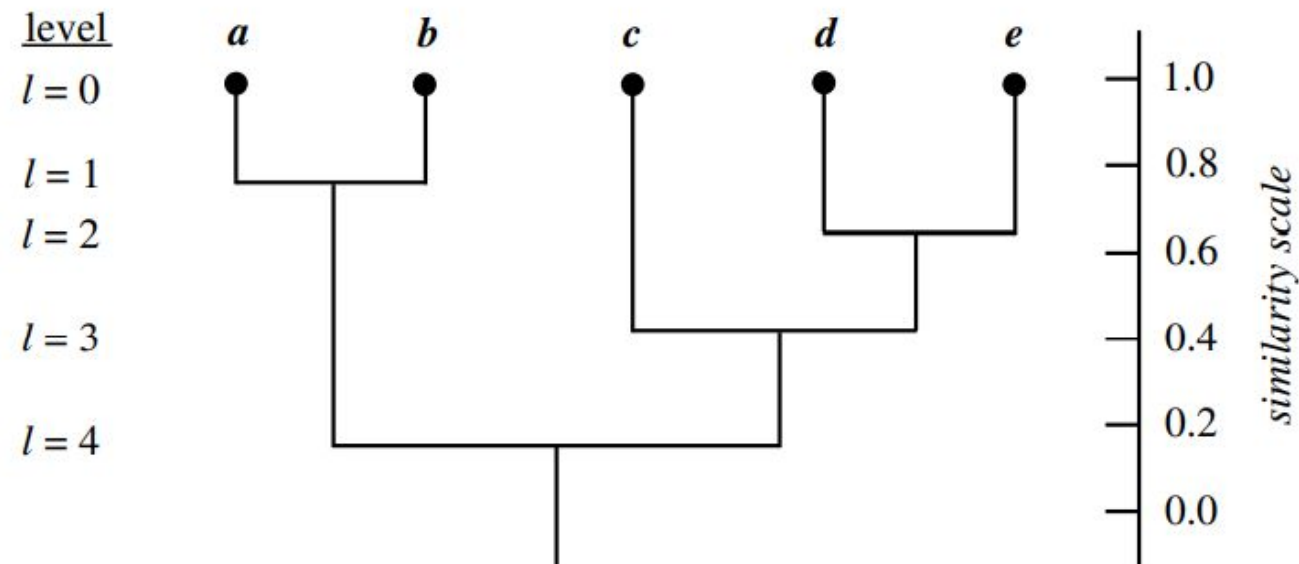


Figure 7.7 Dendrogram representation for hierarchical clustering of data objects $\{a, b, c, d, e\}$.



Density-Based Methods

- To discover clusters with arbitrary shape, density-based clustering methods have been developed. These typically regard clusters as dense regions of objects in the data space that are separated by regions of low density (representing noise). DBSCAN grows clusters according to a density-based connectivity analysis.

7.6.1 **DBSCAN: A Density-Based Clustering Method Based on Connected Regions with Sufficiently High Density**

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm. The algorithm grows regions with sufficiently high density into clusters and discovers clusters of arbitrary shape in spatial databases with noise. It defines a cluster as a maximal set of *density-connected* points.

The basic ideas of density-based clustering involve a number of new definitions. We intuitively present these definitions, and then follow up with an example.

A **density-based cluster** is a set of density-connected objects that is maximal with respect to density-reachability. Every object not contained in any cluster is considered to be *noise*.

“How does DBSCAN find clusters?” DBSCAN searches for clusters by checking the ϵ -neighborhood of each point in the database. If the ϵ -neighborhood of a point p contains more than $MinPts$, a new cluster with p as a core object is created. DBSCAN then iteratively collects directly density-reachable objects from these core objects, which may involve the merge of a few density-reachable clusters. The process terminates when no new point can be added to any cluster.

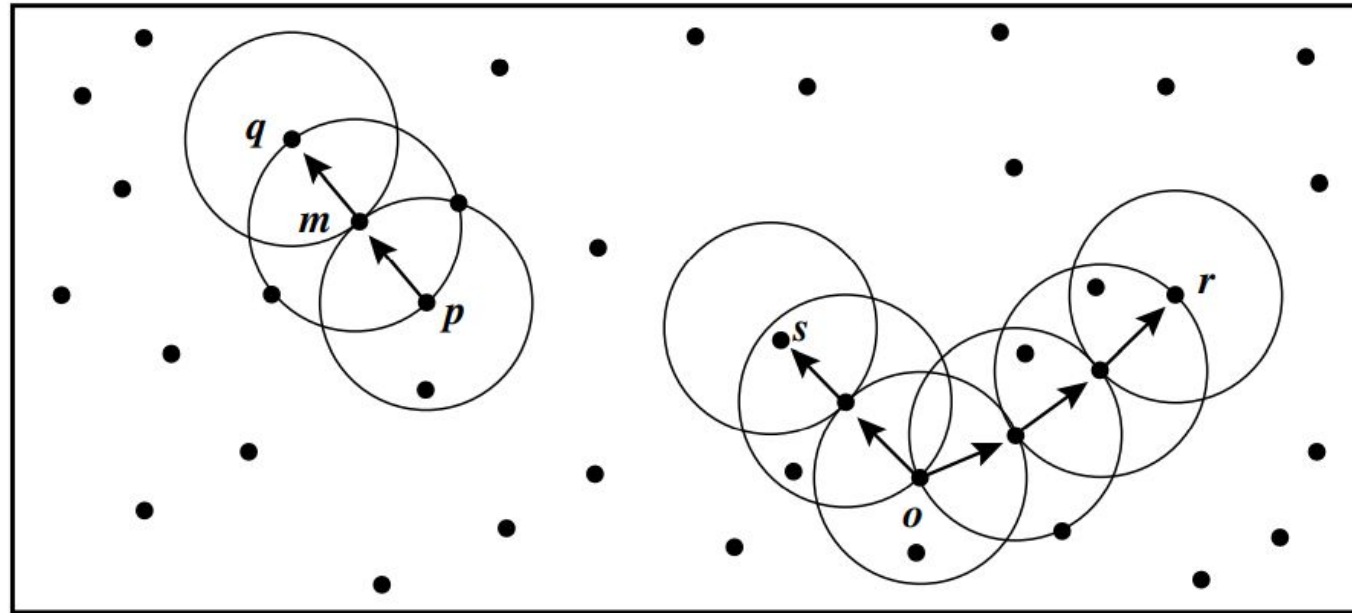


Figure 7.10 Density reachability and density connectivity in density-based clustering. Based on [EKSX96].



Grid-Based Methods

Grid-Based Methods

The grid-based clustering approach uses a multiresolution grid data structure. It quantizes the object space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed. The main advantage of the approach is its fast processing time, which is typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space.

WaveCluster is a grid-based and density-based algorithm. It conforms with many of the requirements of a good clustering algorithm: It handles large data sets efficiently, discovers clusters with arbitrary shape, successfully handles outliers, is insensitive to the order of input, and does not require the specification of input parameters such as the

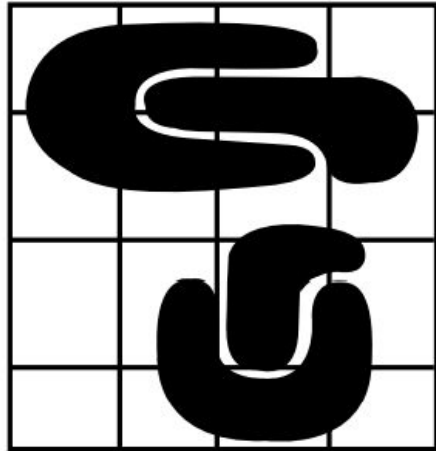

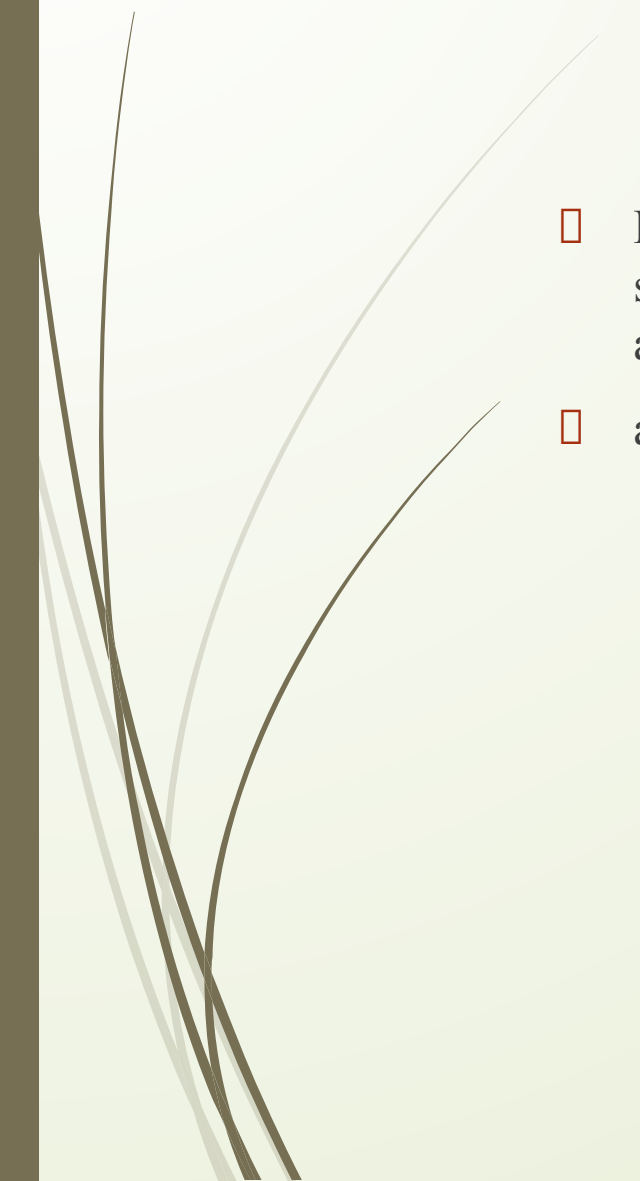


Figure 7.16 A sample of two-dimensional feature space. From [SCZ98].



Model-Based Clustering Methods

- Model-based clustering methods attempt to optimize the fit between the given data and some mathematical model. Such methods are often based on the assumption that the data are generated by a mixture of underlying probability distributions.
 - an extension of the k-means partitioning algorithm, called Expectation-Maximization.
- 

7.8.1 Expectation-Maximization

In practice, each cluster can be represented mathematically by a parametric probability distribution. The entire data is a *mixture* of these distributions, where each individual distribution is typically referred to as a *component distribution*. We can therefore cluster the data using a finite **mixture density model** of k probability distributions, where each distribution represents a cluster. The problem is to estimate the parameters of the probability distributions so as to best fit the data. Figure 7.18 is an example of a simple finite mixture density model. There are two clusters. Each follows a normal or Gaussian distribution with its own mean and standard deviation.

The EM (**Expectation-Maximization**) algorithm is a popular iterative refinement algorithm that can be used for finding the parameter estimates. It can be viewed as an extension of the k -means paradigm, which assigns an object to the cluster with which it is most similar, based on the cluster mean (Section 7.4.1). Instead of assigning each object to a dedicated cluster, EM assigns each object to a cluster according to a weight representing the probability of membership. In other words, there are no strict boundaries between clusters. Therefore, new means are computed based on weighted measures.

EM starts with an initial estimate or “guess” of the parameters of the mixture model (collectively referred to as the *parameter vector*). It iteratively rescores the objects against the mixture density produced by the parameter vector. The rescored objects are then used to update the parameter estimates. Each object is assigned a probability that it would possess a certain set of attribute values given that it was a member of a given cluster. The algorithm is described as follows:

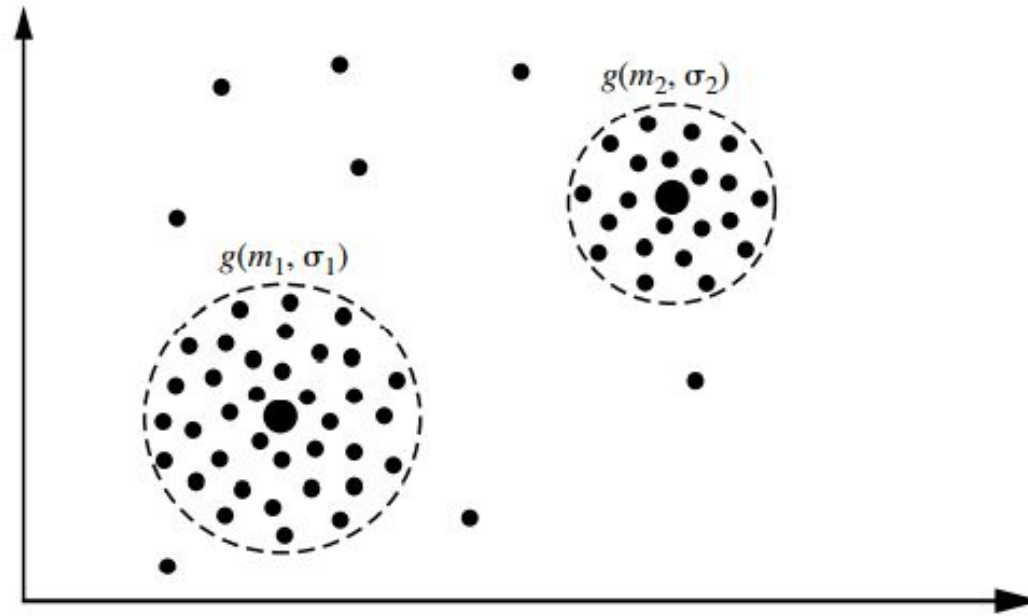
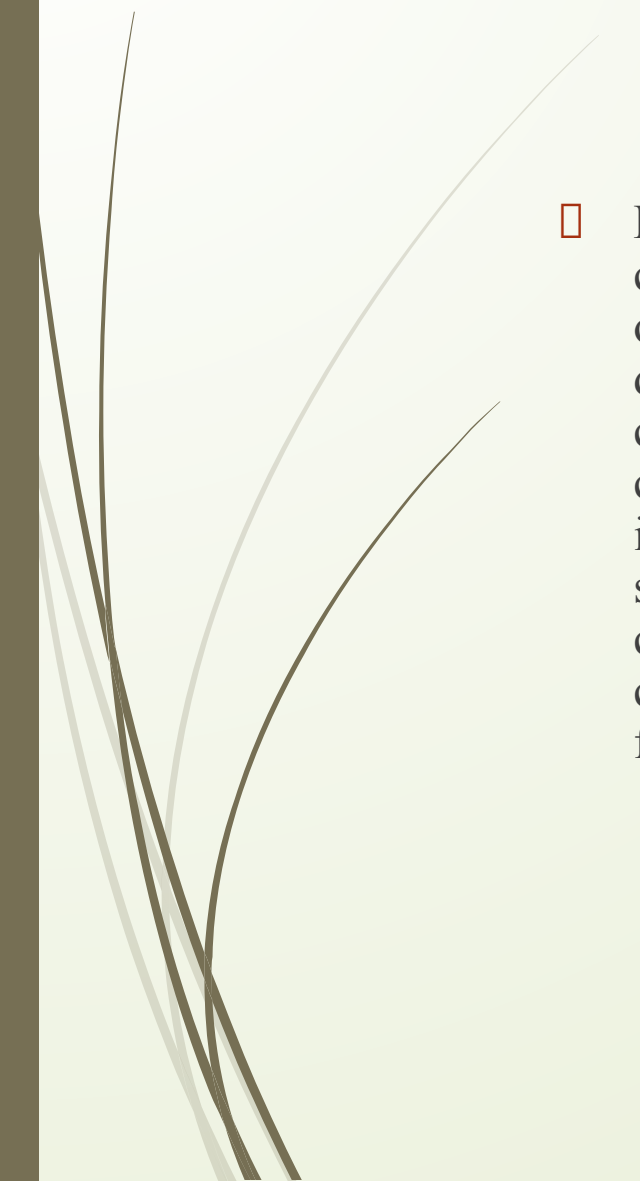


Figure 7.18 Each cluster can be represented by a probability distribution, centered at a mean, and with a standard deviation. Here, we have two clusters, corresponding to the Gaussian distributions $g(m_1, \sigma_1)$ and $g(m_2, \sigma_2)$, respectively, where the dashed circles represent the first standard deviation of the distributions.



Clustering High-Dimensional Data

- 
- Most clustering methods are designed for clustering low-dimensional data and encounter challenges when the dimensionality of the data grows really high (say, over 10 dimensions, or even over thousands of dimensions for some tasks). This is because when the dimensionality increases, usually only a small number of dimensions are relevant to certain clusters, but data in the irrelevant dimensions may produce much noise and mask the real clusters to be discovered. Moreover, when dimensionality increases, data usually become increasingly sparse because the data points are likely located in different dimensional subspaces. When the data become really sparse, data points located at different dimensions can be considered as all equally distanced, and the distance measure, which is essential for cluster analysis, becomes meaningless. To overcome this difficulty, we may consider using feature (or attribute) transformation and feature (or attribute) selection techniques.

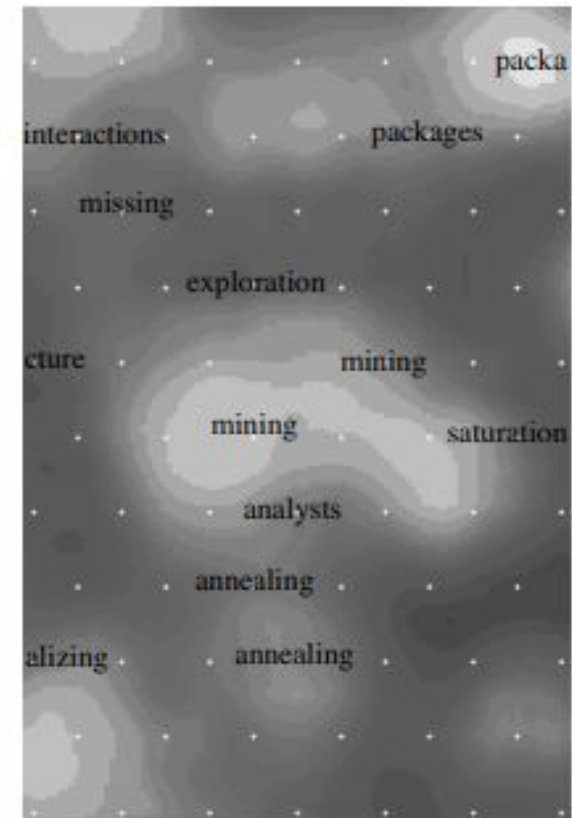
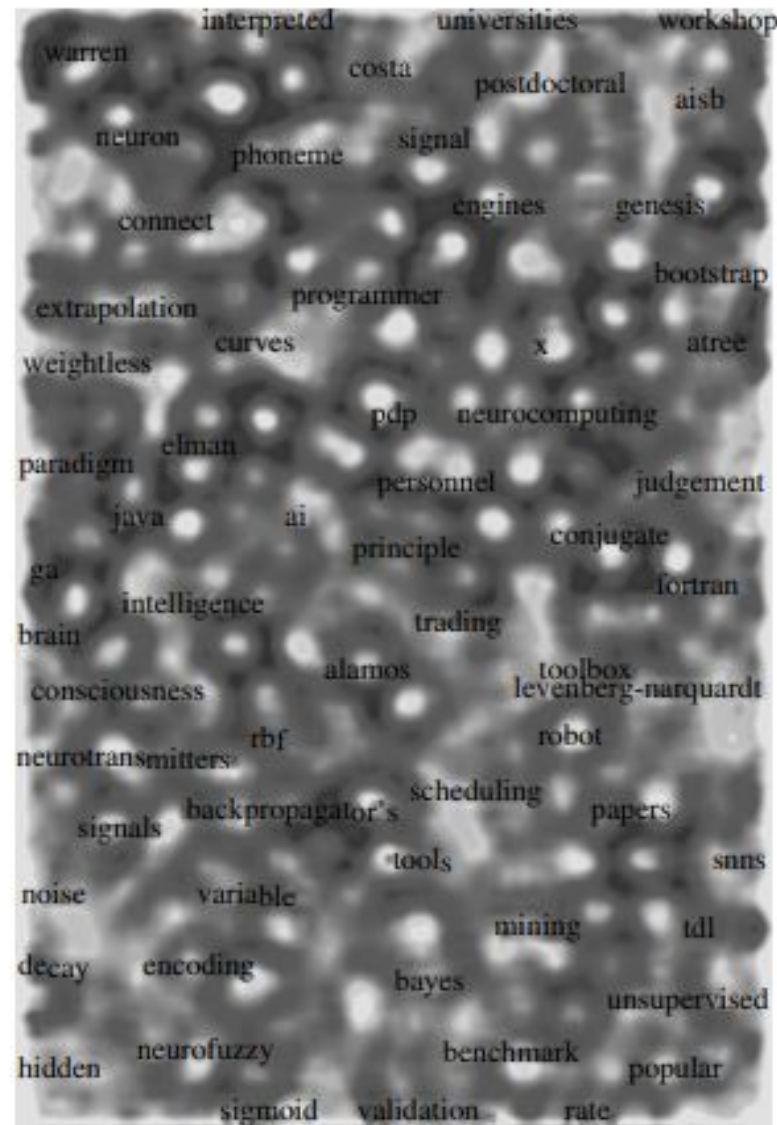



Figure 7.20 The result of SOM clustering of 12,088 Web articles on comp.ai.neural-nets (left), and of drilling down on the keyword: “mining” (right). Based on <http://websom.hut.fi/web-som/comp.ai.neural-nets-new/html/root.html>.



Constraint-Based Cluster Analysis




- In many applications, it is desirable to have the clustering process take user preferences and constraints into consideration. Examples of such information include the expected number of clusters, the minimal or maximal cluster size, weights for different objects or dimensions, and other desirable characteristics of the resulting clusters. Moreover, when a clustering task involves a rather high-dimensional space, it is very difficult to generate meaningful clusters by relying solely on the clustering parameters. User input regarding important dimensions or the desired results will serve as crucial hints or meaningful constraints for effective clustering. In general, we contend that knowledge discovery would be most effective if one could develop an environment for human-centered, exploratory mining of data, that is, where the human user is allowed to play a key role in the process. Foremost, a user should be allowed to specify a focus—directing the mining algorithm toward the kind of “knowledge” that the user is interested in finding.



Constraint-based clustering finds clusters that satisfy user-specified preferences or constraints. Depending on the nature of the constraints, constraint-based clustering may adopt rather different approaches. Here are a few categories of constraints.

- 1. Constraints on individual objects:** We can specify constraints on the objects to be clustered. In a real estate application, for example, one may like to spatially cluster only those luxury mansions worth over a million dollars. This constraint confines the set of objects to be clustered. It can easily be handled by preprocessing (e.g., performing selection using an SQL query), after which the problem reduces to an instance of unconstrained clustering.
- 2. Constraints on the selection of clustering parameters:** A user may like to set a desired range for each clustering parameter. Clustering parameters are usually quite specific to the given clustering algorithm. Examples of parameters include k , the desired number of clusters in a k -means algorithm; or ϵ (the radius) and *MinPts* (the minimum number of points) in the DBSCAN algorithm. Although such user-specified parameters may strongly influence the clustering results, they are usually confined to the algorithm itself. Thus, their fine tuning and processing are usually not considered a form of constraint-based clustering.



3. Constraints on distance or similarity functions: We can specify different distance or similarity functions for specific attributes of the objects to be clustered, or different distance measures for specific pairs of objects. When clustering sportsmen, for example, we may use different weighting schemes for height, body weight, age, and skill level. Although this will likely change the mining results, it may not alter the clustering process per se. However, in some cases, such changes may make the evaluation of the distance function nontrivial, especially when it is tightly intertwined with the clustering process. This can be seen in the following example.




Outlier Analysis



“What is an outlier?” Very often, there exist data objects that do not comply with the general behavior or model of the data. Such data objects, which are grossly different from or inconsistent with the remaining set of data, are called **outliers**.


Outliers can be caused by measurement or execution error. For example, the display of a person’s age as –999 could be caused by a program default setting of an unrecorded age. Alternatively, outliers may be the result of inherent data variability. The salary of the chief executive officer of a company, for instance, could naturally stand out as an outlier among the salaries of the other employees in the firm.

Many data mining algorithms try to minimize the influence of outliers or eliminate them all together. This, however, could result in the loss of important hidden information because *one person’s noise could be another person’s signal*. In other words, the outliers may be of particular interest, such as in the case of fraud detection, where outliers may indicate fraudulent activity. Thus, outlier detection and analysis is an interesting data mining task, referred to as **outlier mining**.



Outlier mining has wide applications. As mentioned previously, it can be used in fraud detection, for example, by detecting unusual usage of credit cards or telecommunication services. In addition, it is useful in customized marketing for identifying the spending behavior of customers with extremely low or extremely high incomes, or in medical analysis for finding unusual responses to various medical treatments.

Outlier mining can be described as follows: Given a set of n data points or objects and k , the expected number of outliers, find the top k objects that are considerably dissimilar, exceptional, or inconsistent with respect to the remaining data. The outlier mining problem can be viewed as two subproblems: (1) define what data can be considered as inconsistent in a given data set, and (2) find an efficient method to mine the outliers so defined.



Data Mining Applications

Refer Book

Applications and Trends in Data Mining 649

11.1 Data Mining Applications 649

11.1.1 Data Mining for Financial Data Analysis 649


11.1.2 Data Mining for the Retail Industry 651

11.1.3 Data Mining for the Telecommunication Industry 652

11.1.4 Data Mining for Biological Data Analysis 654

11.1.5 Data Mining in Other Scientific Applications 657

11.1.6 Data Mining for Intrusion Detection 658



Data Mining System Products and Research Prototypes,
Additional Themes on Data Mining and Social Impacts of Data Mining.

Refer Book

Data Mining System Products and Research Prototypes 660

11.2.1 How to Choose a Data Mining System 660

11.2.2 Examples of Commercial Data Mining Systems 663

Additional Themes on Data Mining 665

11.3.1 Theoretical Foundations of Data Mining 665

11.3.2 Statistical Data Mining 666

11.3.3 Visual and Audio Data Mining 667

11.3.4 Data Mining and Collaborative Filtering 670

Social Impacts of Data Mining 675

11.4.1 Ubiquitous and Invisible Data Mining 675

11.4.2 Data Mining, Privacy, and Data Security 678