What Is Cluster Analysis?

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. A cluster of data objects can be treated collectively as one group and so may be considered as a form of data compression.

Although classification is an effective means for distinguishing groups or classes of objects, it requires the often costly collection and labeling of a large set of training tuples or patterns, which the classifier uses to model each group. It is often more desirable to proceed in the reverse direction: First partition the set of data into groups based on data similarity (e.g., using clustering), and then assign labels to the relatively small number of groups. Additional advantages of such a clustering-based process are that it is adaptable to changes and helps single out useful features that distinguish different groups.

Clustering is a challenging field of research in which its potential applications pose their own special requirements. The following are typical requirements of clustering in data mining:

Scalability: Many clustering algorithms work well on small data sets containing fewer than several hundred data objects; however, a large database may contain millions of objects. Clustering on a sample of a given large data set may lead to biased results. Highly scalable clustering algorithms are needed.

Ability to deal with different types of attributes: Many algorithms are designed to cluster interval-based (numerical) data. However, applications may require clustering other types of data, such as binary, categorical (nominal), and ordinal data, or mixtures of these data types.

Discovery of clusters with arbitrary shape: Many clustering algorithms determine clusters based on Euclidean or Manhattan distance measures. Algorithms based on such distance measures tend to find spherical clusters with similar size and density. However, a cluster could be of any shape. It is important to develop algorithms that can detect clusters of arbitrary shape.

Minimal requirements for domain knowledge to determine input parameters: Many clustering algorithms require users to input certain parameters in cluster analysis (such as the number of desired clusters). The clustering results can be quite sensitive to input parameters. Parameters are often difficult to determine, especially for data sets containing high-dimensional objects. This not only burdens users, but it also makes the quality of clustering difficult to control.

Ability to dealwith noisy data: Most real-world databases contain outliers or missing, unknown, or erroneous data. Some clustering algorithms are sensitive to such data and may lead to clusters

of poor quality. Incremental clustering and insensitivity to the order of input records: Some clustering algorithms cannot incorporate newly inserted data (i.e., database updates) into existing clustering structures and, instead, must determine a new clustering from scratch. Some clustering algorithms are sensitive to the order of input data.

That is, given a set of data objects, such an algorithm may return dramatically different clusterings depending on the order of presentation of the input objects. It is important to develop incremental clustering algorithms and algorithms that are insensitive to the order of input.

High dimensionality: A database or a data warehouse can contain several dimensions or attributes. Many clustering algorithms are good at handling low-dimensional data, involving only two to three dimensions. Human eyes are good at judging the quality of clustering for up to three dimensions. Finding clusters of data objects in high dimensional space is challenging, especially considering that such data can be sparse and highly skewed.

Constraint-based clustering: Real-world applications may need to perform clustering under various kinds of constraints. Suppose that your job is to choose the locations for a given number of new automatic banking machines (ATMs) in a city. To decide upon this, you may cluster households while considering constraints such as the city's rivers and highway networks, and the type and number of customers per cluster. A challenging task is to find groups of data with good clustering behavior that satisfy specified constraints.

Interpretability and usability: Users expect clustering results to be interpretable, comprehensible, and usable. That is, clustering may need to be tied to specific semantic interpretations and applications. It is important to study how an application goal may influence the selection of clustering features and methods.

Types of Data in Cluster Analysis

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 _p variables):

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: 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.

a) 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.

How can the data for a variable be standardized?" To standardize measurements, one choice is to convert the original measurements to unit less variables. Given measurements for a variable f , this can be performed as follows. 1. Calculate the mean absolute deviation, 2. Calculate the standardized measurement, or z-score:

After standardization, or without standardization in certain applications, the dissimilarity (or similarity) between the objects described by interval-scaled variables is typically computed based on the distance between each pair of objects. The most popular distance measure is Euclidean distance, Another well-known metric is Manhattan (or city block) distance, Both the Euclidean distance and Manhattan distance satisfy the following mathematic requirements of a distance function:

1. d(i, j) _ 0: Distance is a nonnegative number.
2. d(i, i) = 0: The distance of an object to itself is 0.
3. d(i, j) = d(j, i): Distance is a symmetric function.
4. d(i, j) _ d(i, h)+d(h, j): Going directly fromobject i to object j in space is no more than making a detour over any other object h (triangular inequality).

b) Binary Variables

Let us see how to compute the dissimilarity between objects described by either symmetric or asymmetric 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.

"What is the difference between symmetric and asymmetric binary variables?" A binary variable is

symmetric if both of its states are equally valuable and carry the same weight; that is, there is no preference on which outcome should be coded as 0 or 1. One such example could be the attribute gender having the states male and female. Dissimilarity that is based on symmetric binary variables is called symmetric binary dissimilarity. Its dissimilarity (or distance) measure, defined in Equation (7.9), can be used to assess the dissimilarity between objects i and j.

A binary variable is asymmetric if the outcomes of the states are not equally important, such as the positive and negative outcomes of a disease test. By convention, we shall code the most important outcome, which is usually the rarest one, by 1 (e.g., HIV positive) and the other by 0 (e.g., HIV negative). Given two asymmetric binary variables, the agreement of two 1s (a positive match) is then considered more significant than that of two 0s (a negative match). Therefore, such binary variables are often considered "monary" (as if having one state). The dissimilarity based on such variables is called asymmetric binary dissimilarity, where the number of negative matches, t, is considered unimportant and thus is ignored in the computation as shown below.

C) 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, : : : , M.Notice that such integers are used just for data handling and do not represent any specific ordering.

"How is dissimilarity computed between objects described by categorical variables?" The dissimilarity between two objects i and j can be computed based on the ratio of mismatches:
where m is the number of matches (i.e., the number of variables for which i and j are in the same state), and p is the total number of variables. Weights can be assigned to increase the effect of m or to assign greater weight to the matches in variables having a larger number of states.

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. A continuous ordinal variable looks like a set of continuous data of an unknown scale; that is, the relative ordering of the values is essential but their actual magnitude is not. 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. Ordinal variables may also be obtained from the discretization of interval-scaled quantities by splitting the value range into a finite number of classes. The values of
an ordinal variable can be mapped to ranks. For example, suppose that an ordinal variable f has Mf states. These ordered states define the ranking 1, : : : , Mf

"How are ordinal variables handled?" The treatment of ordinal variables is quite similar to that of intervalscaled variables when computing the dissimilarity between objects. Suppose that f is a variable from a set of ordinal variables describing n objects. The dissimilarity computation with respect to f involves the following steps:

1. The value of f for the ith object is xi f , and f has Mf ordered states, representing theranking 1, : : : , Mf . Replace each xi f by its corresponding rank, ri f 2 f1, : : : , Mf g.
2. Since each ordinal variable can have a different number of states, it is often necessary to map the range of each variable onto [0.0,1.0] so that each variable has equal weight. This can be achieved by replacing the rank ri f of the ith object in the f th variable by
3. Dissimilarity can then be computed using any of the distance measures for interval-scaled variables, using zi f to represent the f value for the ith object.

Ratio-Scaled Variables

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

"How can I compute the dissimilarity between objects described by ratio-scaled variables?" There are three methods to handle ratio-scaled variables for computing the dissimilarity between objects.

• Treat ratio-scaled variables like interval-scaled variables. This, however, is not usually a good choice since it is likely that the scale may be distorted.
• Apply logarithmic transformation to a ratio-scaled variable f having value xi f for object i by using the formula yi f = log(xi f). The yi f values can be treated as interval valued, Notice that for some ratioscaled variables, loglog or other transformations may be applied, depending on the variable's
definition and the application.
• Treat xi f as continuous ordinal data and treat their ranks as interval-valued.
The latter two methods are the most effective, although the choice of method used may depend on the given application.

d) Variables of Mixed Types

To compute 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.

"So, how can we compute the dissimilarity between objects of mixed variable types?" One approach is to group each kind of variable together, performing a separate cluster analysis for each variable type. This is feasible if these analyses derive compatible results. However, in real applications, it is unlikely that a separate cluster analysis per variable type will generate compatible results.

A more preferable approach is to process all variable types together, performing a single cluster analysis.

One such technique combines the different variables into a single dissimilarity matrix, bringing all of the meaningful variables onto a common scale of the interval [0.0,1.0].
Suppose that the data set contains p variables of mixed type. The dissimilarity d(i, j) between objects i and j is defined as

e) 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. There are several ways to define such a similarity function, s(x, y), to compare two vectors x and y. One popular way is to define the similarity function as a cosine measure as follows:
where xt is a transposition of vector x, jjxjj is the Euclidean normof vector x,1 jjyjj is the Euclidean norm of vector y, and s is essentially the cosine of the angle between vectors x and y. This value is invariant to rotation and dilation, but it is not invariant to translation and general linear transformation.

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.
Partitioning methods: Given a database of n objects or data tuples, a partitioning method constructs k partitions of the data, where each partition represents a cluster and k _ n. That is, it classifies the data into k groups, which together satisfy the following requirements: (1) each group must contain at least one object, and (2) each object must belong to exactly one group. Notice that the second requirement can be relaxed in some fuzzy partitioning techniques.

Given k, the number of partitions to construct, a partitioning method creates an initial partitioning. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. The general criterion of a good partitioning is that objects in the same cluster are “close” or related to each
other, whereas objects of different clusters are “far apart” or very different. There are various kinds of other criteria for judging the quality of partitions.

Hierarchical methods: A hierarchical method creates a hierarchical decomposition of the given set of data objects.

A hierarchical method can be classified as being either agglomerative or divisive, based on how the hierarchical decomposition is formed. The agglomerative approach, also called the bottom-up approach, starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one (the topmost

level of the hierarchy), or until a termination condition holds. The divisive approach, also called the top-down approach, starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

Density-based methods: Most partitioning methods cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes.

Other clustering methods have been developed based on the notion of density. Their general idea is to continue growing the given cluster as long as the density (number of objects or data points) in the "neighborhood" exceeds some threshold; that is, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise (outliers) and discover clusters of arbitrary shape.

DBSCAN and its extension, OPTICS, are typical density-based methods that grow clusters according to a density-based connectivity analysis. DENCLUE is a method that clusters objects based on the analysis of the value distributions of density functions.

Grid-based methods: Grid-based methods quantize the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on the grid structure (i.e., on the quantized space). The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.

STING is a typical example of a grid-based method. WaveCluster applies wavelet transformation for clustering analysis and is both grid-based and density-based. Gridbased clustering methods

Model-based methods: Model-based methods hypothesize a model for each of the clusters and find the best fit of the data to the given model. A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points. It also leads to a way of automatically determining the number of clusters based on standard statistics, taking "noise" or outliers into account and thus yielding robust clustering methods.

Clustering high-dimensional data: is a particularly important task in cluster analysis because many applications require the analysis of objects containing a large number of features or dimensions. For example, text documents may contain thousands of terms or keywords as features, and DNA microarray data may provide information on the expression levels of

thousands of genes under hundreds of conditions. Clustering high-dimensional data is challenging due to the curse of dimensionality.

Constraint-based clustering is a clustering approach that performs clustering by incorporation of user-specified or application-oriented constraints. A constraint expresses a user's expectation or describes "properties" of the desired clustering results, and provides an effective means for communicating with the clustering process. Various kinds of constraints can be specified, either by a user or as per application requirements. Our focus of discussion will be on spatial clustering with the existence of obstacles and clustering under user-specified constraints. In addition, semisupervised
clustering is described, which employs, for example, pairwise constraints (such as pairs of instances
labeled as belonging to the same or different clusters) in order to improve the quality of the resulting clustering

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 _ 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 thedata set attributes.

Classical Partitioning Methods: k-Means and k-Medoids
The mostwell-known and commonly used partitioningmethods are k-means, k-medoids, and their variations.

The k-Means Method – A Centroid-Based Technique

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 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 mi is the mean of cluster Ci (both p and mi 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.

Clustering by k-means partitioning:

Suppose that there is a set of objects located in space as depicted in the rectangle shown in Figure 7.3(a).

Let k = 3; that is, the user would like the objects to be partitioned into three clusters.
According to the algorithm in Figure 7.2, we arbitrarily choose three objects as the three initial cluster centers, where cluster centers are marked by a "+". Each object is distributed to a cluster based on the cluster center to which it is the nearest. Such a distribution forms silhouettes encircled by dotted curves, as shown in Figure 7.3(a).

Next, the cluster centers are updated. That is, the mean value of each cluster is recalculated based on the current objects in the cluster. Using the new cluster centers, the objects are redistributed to the clusters based on which cluster center is the nearest. Such a redistribution forms new silhouettes encircled by dashed curves, as shown in Figure 7.3(b).

This process iterates, leading to Figure 7.3(c). The process of iteratively reassigning objects to clusters to improve the partitioning is referred to as iterative relocation. Eventually, no redistribution of the objects in any cluster occurs, and so the process terminates. The resulting clusters are returned by the clustering process.

The algorithm attempts to determine k partitions that minimize the square-error function. It works well when the clusters are compact clouds that are rather well separated from one another. The method is relatively scalable and efficient in processing large data sets because the computational complexity of the algorithm is O(nkt), where n is the total number of objects, k is the number of clusters, and t is the number of iterations. Normally, $k \ll n$ and $t \ll n$. The method often terminates at a local optimum.

The k-means method, however, can be applied only when the mean of a cluster is defined. This may not be the case in some applications, such as when data with categorical attributes are involved. The necessity for users to specify k, the number of clusters, in advance can be seen as a disadvantage. The k-means method is not suitable for discovering clusters with nonconvex shapes or clusters of very different size. Moreover, it is sensitive to noise and outlier data points because a small number of such data can substantially influence the mean value.

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 "+".)

There are quite a few variants of the k-means method. These can differ in the selection of the initial k means, the calculation of dissimilarity, and the strategies for calculating cluster means. An interesting strategy that often yields good results is to first apply a hierarchical agglomeration algorithm, which determines the number of clusters and finds an initial clustering, and then use iterative relocation to improve the clustering.

Another variant to k-means is the k-modes method, which extends the k-means paradigm to cluster

categorical data by replacing the means of clusters with modes, using new dissimilarity measures to deal with categorical objects and a frequency-based method to update modes of clusters. The k-means and the k-modes methods can be integrated to cluster data with mixed numeric and categorical values.

The k-Medoids Method - Representative Object-Based Technique:

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

where E is the sum of the absolute error for all objects in the data set; p is the point in space representing a given object in clusterCj; and oj is the representative object ofCj. 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 kmedoids 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, orandom, is a good replacement for a current representative object, oj, the following four cases are examined for each of the nonrepresentative objects, p, as illustrated in Figure 7.4.

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

PAM(Partitioning Around Medoids) was one of the first k-medoids algorithms introduced (Figure 7.5). It attempts to determine k partitions for n objects. After an initial random selection of k representative objects, the algorithm repeatedly tries to make a better choice of cluster representatives. All of the possible pairs of objects are analyzed, where one object in each pair is considered a representative object and the other is not. The quality of the resulting clustering is calculated for each such combination. An object, oj, is replaced with the object causing the greatest reduction in error. The set of best objects for each cluster in one iteration forms the representative objects for the next iteration. The final set of representative objects are the respective medoids of the clusters. The complexity of each iteration is O(k(n_k)2). For large values of n and k, such computation becomes very costly.

Figure 7.5 PAM, a k-medoids partitioning algorithm.

Partitioning Methods in Large Databases:
From k-Medoids to CLARAN

"How efficient is the k-medoids algorithm on large data sets?" A typical k-medoids partitioning algorithm like PAM works effectively for small data sets, but does not scale well for large data sets. To deal with larger data sets, a sampling-based method, called CLARA (Clustering LARge Applications), can be used.

The idea behind CLARA is as follows: Instead of taking the whole set of data into consideration, a small portion of the actual data is chosen as a representative of the data. Medoids are then chosen from this sample using PAM. If the sample is selected in a fairly random manner, it should closely represent the original data set. The representative objects (medoids) chosen will likely be similar to those that would have been chosen from the whole data set. CLARA draws multiple samples of the data set, applies PAM on each sample, and returns its best clustering as the output. As expected, CLARA can deal with larger data sets than PAM. The complexity of each iteration now becomes O(ks2+k(n_k)), where s is the size of the sample, k is the number of clusters, and n is the total number of objects.

The effectiveness of CLARA depends on the sample size. Notice that PAM searches for the best k medoids among a given data set, whereas CLARA searches for the best k medoids among the selected sample of the data set. CLARA cannot find the best clustering if any of the best sampled medoids is not among the best k medoids.

That is, if an object oi is one of the best k medoids but is not selected during sampling, CLARA will never find the best clustering. This is, therefore, a trade-off for efficiency. A good clustering based on sampling will not necessarily represent a good clustering of the whole data set if the sample is biased.

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. The quality of a pure hierarchical clustering method suffers fromits inability to performadjustment once amerge or split decision has been executed. That is, if a particular merge or split decision later turns out to have been a poor choice, the method cannot backtrack and correct it. Recent studies have emphasized the integration of hierarchical agglomeration with iterative relocation methods.

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, fa, b, c, d, eg. 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 C1 and C2 may be merged if an object in C1 and an object in C2 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 fa, b, c, d, eg.
In either agglomerative or divisive hierarchical clustering, the user can specify the desired number of clusters as a termination condition.

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 shows a dendrogram for the five objects presented in Figure 7.6, where l = 0 shows the five objects as singleton clusters at level 0. At l = 1, objects a and b are grouped together to form the first cluster, and they stay together at all subsequent levels. We can also use a vertical axis to show the similarity scale between clusters. For example, when the similarity of two groups of objects, fa, bg and fc, d, eg, is roughly 0.16, they are merged together to form a single cluster

Figure 7.7 Dendrogram representation for hierarchical clustering of data objects fa, b, c, d, eg.
2) BIRCH: Balanced Iterative Reducing and Clustering Using Hierarchies

BIRCH is designed for clustering a large amount of numerical data by integration of hierarchical clustering (at the initial microclustering stage) and other clustering methods such as iterative partitioning (at the later macroclustering stage). It overcomes the two difficulties of agglomerative clustering methods: (1) scalability and (2) the inability to undo what was done in the previous step.
BIRCH introduces two concepts, clustering feature and clustering feature tree (CF tree), which are used to summarize cluster representations. These structures help the clustering method

achieve good speed and scalability in large databases and also make it effective for incremental and dynamic clustering of incoming objects.

Let's look closer at the above-mentioned structures. Given n d-dimensional data objects or points in a cluster, we can define the centroid x0, radius R, and diameter D of the cluster as follows:
where R is the average distance from member objects to the centroid, and D is the average pairwise distance within a cluster. Both R and D reflect the tightness of the cluster around the centroid. A clustering feature (CF) is a threedimensional vector summarizing information about clusters of objects. Given n d-dimensional objects or points in a cluster, fxig, then the CF of the cluster is defined as where n is the number of points in the cluster, LS is the linear sum of the n points and SS is the square sum of the data points

Example 7.10 Clustering feature. Suppose that there are three points, (2, 5), (3, 2), and (4, 3), in a cluster, C1. The clustering feature of C1 is

CF1 = (3, (2+3+4;5+2+3), (22+32+42;52+22+32)) = (3, (9,10), (29,38)):

Suppose thatC1 is disjoint to a second cluster,C2, whereCF2 =(3, (35, 36), (417, 440) The clustering feature of a newcluster,C3, that is formed by mergingC1 andC2, is derived by adding CF1 and CF2. That is, CF3 = (3+3, (9+35,10+36), (29+417,38+440) = (6, (44,46), (446,478)):

A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering. An example is shown in Figure 7.8. By definition, a nonleaf node in a tree has descendants or "children." The nonleaf nodes store sums of the CFs of their children, and thus summarize clustering information about their children. A CF tree has two parameters: branching factor, B, and threshold, T. The branching factor specifies the maximum number of children per nonleaf node. The threshold parameter specifies the maximum diameter of subclusters stored at the leaf nodes of the tree. These two parameters influence the size of the resulting tree.

BIRCH tries to produce the best clusters with the available resources. Given a limited amount of main memory, an important consideration is to minimize the time required for I/O. BIRCH applies a multiphase clustering technique: a single scan of the data set yields a basic good clustering, and one or more additional scans can (optionally) be used to further improve the quality. The primary phases are:

• Phase 1: BIRCH scans the database to build an initial in-memory CF tree, which can be viewed as
a multilevel compression of the data that tries to preserve the inherent clustering structure of the data.

• Phase 2: BIRCH applies a (selected) clustering algorithm to cluster the leaf nodes of the CF tree,
which removes sparse clusters as outliers and groups dense clusters into larger ones.

Figure 7.8 A CF tree structure.

3) ROCK: A Hierarchical Clustering Algorithm for Categorical Attributes

ROCK (RObust Clustering using linKs) is a hierarchical clustering algorithm that explores the concept of links (the number of common neighbors between two objects) for data with categorical attributes. Traditional clustering algorithms for clustering data with Boolean and categorical attributes use distance functions (such as those introduced for binary variables in Section 7.2.2). However, experiments show that such distance measures cannot lead to high-quality clusters when clustering categorical data. Furthermore, most clustering algorithms assess only the similarity between points when clustering; that is, at each step, points that are the most similar are
merged into a single cluster. This "localized" approach is prone to errors. For example, two distinct clusters may have a few points or outliers that are close; therefore, relying on the similarity between points to make clustering decisions could cause the two clusters to be merged. ROCK takes a more global approach to clustering by considering the neighborhoods of individual pairs of points. If two similar points also have similar neighborhoods, then the two points likely belong to the same cluster and so can be merged.

4) Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modeling

Chameleon is a hierarchical clustering algorithm that uses dynamic modeling to determine the similarity between pairs of clusters. It was derived based on the observed weaknesses of two hierarchical clustering algorithms: ROCK and CURE. ROCK and related schemes emphasize cluster interconnectivity while ignoring information regarding cluster proximity. CURE and related schemes consider cluster proximity yet ignore cluster interconnectivity. In Chameleon, cluster similarity is assessed based on how well-connected objects are within a cluster and on the proximity of clusters. That is, two clusters are merged if their interconnectivity is high and they are close together. Thus, Chameleon does not depend on a static, user-supplied model and can automatically adapt to the internal characteristics of the clusters being merged. The merge process facilitates the discovery of
natural and homogeneous clusters and applies to all types of data as long as a similarity function can be specified.

"How does Chameleon work?" The main approach of Chameleon is illustrated in Figure 7.9. Chameleon uses a k-nearest-neighbor graph approach to construct a sparse graph, where each

vertex of the graph represents a data object, and there exists an edge between two vertices (objects) if one object is among the k-most-similar objects of the other. The edges are weighted to reflect the similarity between objects. Chameleon uses a graph partitioning algorithm to partition the k-nearest-neighbor graph into a large number of relatively small subclusters. It then uses an agglomerative hierarchical clustering algorithm that repeatedly merges subclusters based on their

similarity. To determine the pairs of most similar subclusters, it takes into account both the interconnectivity as well as the closeness of the clusters. We will give a mathematical definition for these criteria shortly.

Figure 7.9 Chameleon: Hierarchical clustering based on k-nearest neighbors and dynamic modeling.

Remaining topics will send later….

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. OPTICS extends DBSCAN to produce a cluster ordering obtained from a wide range of parameter settings. DENCLUE clusters objects based on a set of density distribution functions.

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.

Example 7.12 Density-reachability and density connectivity. Consider Figure 7.10 for a given ε represented by the radius of the circles, and, say, let MinPts = 3. Based on the above definitions:

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.

2) OPTICS: Ordering Points to Identify the Clustering Structure

Although DBSCAN can cluster objects given input parameters such as ε and MinPts, it still leaves the user with the responsibility of selecting parameter values that will lead to the discovery of acceptable clusters. Actually, this is a problem associated with many other clustering algorithms. Such parameter settings are usually empirically set and difficult to determine, especially for real-world, high-dimensional data sets. Most algorithms are very sensitive to such parameter values: slightly different settings may lead to very different clusterings of the data.

Moreover, high-dimensional real data sets often have very skewed distributions, such that their intrinsic clustering structure may not be characterized by global density parameters.

To help overcome this difficulty, a cluster analysis method called OPTICS was proposed. Rather than

produce a data set clustering explicitly, OPTICS computes an augmented cluster ordering for automatic and interactive cluster analysis. This ordering represents the density-based clustering structure of the data. It contains information that is equivalent to density-based clustering obtained from a wide range of parameter settings. The cluster ordering can be used to extract basic clustering information (such as cluster centers or arbitrary-shaped clusters) as well as provide the intrinsic clustering structure.

By examining DBSCAN, we can easily see that for a constant MinPts value, densitybased clusters with respect to a higher density (i.e., a lower value for ε) are completely contained in density-connected sets obtained with respect to a lower density. Recall that the parameter ε is a distance—it is the neighborhood radius. Therefore, in order to produce a set or ordering of density-based clusters, we can extend the DBSCAN algorithm to process a set of distance parameter values at the same time. To construct the different clusterings simultaneously, the objects should be processed in a specific order. This order selects an object that is density-reachable with respect to the

lowest ε value so that clusters with higher density (lower ε) will be finished first. Based on this idea, two values need to be stored for each object—core-distance and reachability-distance:

• The core-distance of an object p is the smallest ε' value that makes {p} a core object. If p is not a core object, the core-distance of p is undefined.

• The reachability-distance of an object q with respect to another object p is the greater value of the core distance of p and the Euclidean distance between p and q. If p is not a core object, the reachability distance between p and q is undefined.

Figure 7.11 OPTICS terminology.

Example 7.13 Core-distance and reachability-distance. Figure 7.11 illustrates the concepts of coredistance and reachability-distance. Suppose that ε=6 mm and MinPts=5. The coredistance of p is the distance, ε0, between p and the fourth closest data object. The reachability-distance of q1 with respect to p is the core-distance of p (i.e., ε' =3 mm) because this is greater than the Euclidean distance from p to q1. The reachabilitydistance of q2 with respect to p is the Euclidean distance from p to q2 because this is greater than the core-distance of p.

Figure 7.12 Cluster ordering in OPTICS.

"How are these values used?" The OPTICS algorithm creates an ordering of the objects in a database,
additionally storing the core-distance and a suitable reachabilitydistance for each object. An algorithm was proposed to extract clusters based on the ordering information produced by OPTICS. Such information is sufficient for the extraction of all density-based clusterings with respect to any distance ε0 that is smaller than the distance ε used in generating the order.

The cluster ordering of a data set can be represented graphically, which helps in its understanding. For example, Figure 7.12 is the reachability plot for a simple two-dimensional data set, which presents a general overview of how the data are structured and clustered. The data objects are plotted in cluster order (horizontal axis) together with their respective reachability-distance (vertical axis). The three Gaussian "bumps" in the plot reflect three clusters in the data set. Methods have also been developed for viewing clustering structures of high dimensional data at various levels of detail.

3) DENCLUE: Clustering Based on Density Distribution Functions
DENCLUE (DENsity-based CLUstEring) is a clustering method based on a set of density distribution
functions. The method is built on the following ideas: (1) the influence of each data point can be formally modeled using a mathematical function, called an influence function, which describes the impact of a data point within its neighborhood; (2) the overall density of the data space can be modeled analytically as the sum of the influence function applied to all data points; and (3) clusters can then be determined mathematically by identifying density attractors, where density attractors are local maxima of the overall density function.

Let x and y be objects or points in Fd, a d-dimensional input space. The influence function of data object y on x is a function, fy

B : Fd -> !R0+ , which is defined in terms of a basic influence function fB:

This reflects the impact of y on x. In principle, the influence function can be an arbitrary function that can be determined by the distance between two objects in a neighborhood. The distance function, d(x, y), should be reflexive and symmetric, such as the Euclidean distance function (Section7.2.1). It can be used to compute a square wave influence function, or a Gaussian influence function,

Figure 7.13 Possible density functions for a 2-D data set.

Figure 7.13 shows a 2-D data set together with the corresponding overall density functions for a square wave and a Gaussian influence function

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. Some typical examples of the grid-based approach include STING, which explores statistical information stored in the grid cells;

WaveCluster, which clusters objects using a wavelet transform method; and CLIQUE, which represents a grid-and density-based approach for clustering in high-dimensional data space

1) STING: STatistical INformation Grid

STING is a grid-based multiresolution clustering technique in which the spatial area is divided into

rectangular cells. There are usually several levels of such rectangular cells corresponding to different levels of resolution, and these cells form a hierarchical structure: each cell at a high level is partitioned to form a number of cells at the next lower level. Statistical information regarding the attributes in each grid cell (such as the mean, maximum, and minimum values) is precomputed and stored. These statistical parameters are useful for query processing, as described below.

Figure 7.15 A hierarchical structure for STING clustering.

Figure 7.15 shows a hierarchical structure for STING clustering. Statistical parameters of higher-level cells can easily be computed from the parameters of the lower-level cells. These parameters include the following: the attribute-independent parameter, count; the attribute-dependent parameters, mean, stdev (standard deviation), min (minimum), max (maximum); and the type of distribution that the attribute value in the cell follows, such as normal, uniform,

exponential, or none (if the distribution is unknown).When the data are loaded into the database, the parameters count, mean, stdev, min, and max of the bottom-level cells are calculated directly from the data. The value of distribution may either be assigned by the user if the distribution type is known beforehand or obtained by hypothesis tests such as the $\chi 2$ test. The type of distribution of a higher-level cell can be computed based on the majority of distribution types of its corresponding lower-level cells in conjunction with a threshold filtering process. If the distributions of the low erlevel cells disagree with each other and fail the threshold test, the distribution type of the high-level cell is set to none.

2) WaveCluster: Clustering Using Wavelet Transformation

WaveCluster is a multiresolution clustering algorithm that first summarizes the data by imposing a

multidimensional grid structure onto the data space. It then uses a wavelet transformation to transformthe original feature space, finding dense regions in the transformed space.

A wavelet transform is a signal processing technique that decomposes a signal into different frequency subbands. The wavelet model can be applied to d-dimensional signals by applying a one-dimensional wavelet transform d times. In applying a wavelet transform, data are transformed so as to preserve the relative distance between objects at different levels of resolution. This allows the natural clusters in the data to become more distinguishable. Clusters can then be identified by searching for dense regions in the new domain. Wavelet transforms are also discussed in

Chapter 2, where they are used for data reduction by compression.

"Why is wavelet transformation useful for clustering?" It offers the following advantages:

• It provides unsupervised clustering. It uses hat-shaped filters that emphasize regions where the points cluster, while suppressing weaker information outside of the cluster boundaries. Thus, dense regions in the original feature space act as attractors for nearby points and as inhibitors for points that are further away. This means that the clusters in the data automatically stand out and "clear" the regions around them. Thus, another advantage is that wavelet transformation can automatically result in the removal of outliers.

• The multiresolution property of wavelet transformations can help detect clusters at varying levels of accuracy. For example, Figure 7.16 shows a sample of two dimensional feature space, where each point in the image represents the attribute or feature values of one object in the spatial data set.

Figure 7.17 shows the resulting wavelet transformation at different resolutions, from a fine scale (scale 1) to a coarse scale (scale 3). At each level, the four subbands into which the original data are decomposed are shown. The subband shown in the upper-left quadrant emphasizes the average neighborhood around each data point.

The subband in the upper-right quadrant emphasizes the horizontal edges of the data. The subband in the lower-left quadrant emphasizes the vertical edges, while the subband in the lower-right quadrant emphasizes the corners.

Figure 7.16 A sample of two-dimensional feature space.

• Wavelet-based clustering is very fast, with a computational complexity of O(n), where n is the number of objects in the database. The algorithm implementation can be made parallel.

Figure 7.17 Multiresolution of the feature space in Figure 7.16 at (a) scale 1 (high resolution); (b) scale 2 (medium resolution); and (c) scale 3 (low resolution).

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

1) Statistical Distribution-Based Outlier Detection

The statistical distribution-based approach to outlier detection assumes a distribution or probability model for the given data set (e.g., a normal or Poisson distribution) and then identifies outliers with respect to the model using a discordancy test. Application of the test requires knowledge of the data set parameters (such as the assumed data distribution), knowledge of distribution parameters (such as the mean and variance), and the expected number of outliers

.

"How does the discordancy testing work?" A statistical discordancy test examines two hypotheses: a

working hypothesis and an alternative hypothesis. A working hypothesis, H, is a statement that the entire data set of n objects comes from an initial distribution model, F, that is,

The hypothesis is retained if there is no statistically significant evidence supporting its rejection. A discordancy test verifies whether an object, oi, is significantly large (or small) in relation to the distribution F.

2) Distance-Based Outlier Detection

The notion of distance-based outliers was introduced to counter the main limitations imposed by statistical methods. An object, o, in a data set, D, is a distance-based (DB) outlier with parameters pct and dmin,11 that is, a DB(pct;dmin)-outlier, if at least a fraction, pct, of the objects in D lie at a distance greater than dmin from o. In other words, rather than relying on statistical tests, we can think of distance-based outliers as those objects that do not have "enough" neighbors, where neighbors are defined based on distance from the given object. In comparison with statistical-based methods, distancebased outlier detection generalizes the ideas behind discordancy testing for

various standard distributions. Distance-based outlier detection avoids the excessive computation that can be associated with fitting the observed distribution into some standard distribution and in selecting discordancy tests.

For many discordancy tests, it can be shown that if an object, o, is an outlier according to the given test, then o is also a DB(pct, dmin)-outlier for some suitably defined pct and dmin. For example, if objects that lie three or more standard deviations from the mean are considered to be outliers, assuming a normal distribution, then this definition can be generalized by a DB(0:9988, 0:13σ) outlier.

Several efficient algorithms for mining distance-based outliers have been developed. These are outlined as follows.

Index-based algorithm: Given a data set, the index-based algorithm uses multidimensional indexing

structures, such as R-trees or k-d trees, to search for neighbors of each object o within radius dmin around that object. Let M be the maximum number of objects within the dmin-neighborhood of an outlier. Therefore, onceM+1 neighbors of object o are found, it is clear that o is not an outlier. This algorithm has a worst-case complexity of O(n2k), where n is the number of objects in the data set and k is the dimensionality. The index-based algorithm scales well as k increases. However, this complexity evaluation takes only the search time into account, even though the task of building an index in itself can be computationally intensive.

Nested-loop algorithm: The nested-loop algorithm has the same computational complexity as the indexbased algorithm but avoids index structure construction and tries to minimize the number of I/Os. It divides the memory buffer space into two halves and the data set into several logical blocks. By carefully choosing the order in which blocks are loaded into each half, I/O efficiency can be achieved.

Cell-based algorithm: To avoidO(n2) computational complexity, a cell-based algorithm was developed for memory-resident data sets. Its complexity is O(ck +n), where c is a constant depending on the number of cells and k is the dimensionality. In this method, the data space is partitioned into cells with a side length equal to dmin 2pk . Each cell has two layers surrounding it. The first layer is one cell thick, while the second is d2pk _ 1e cells thick, rounded up to the closest integer. The algorithm counts outliers on a cell-by-cell rather than an object-by-object basis. For a
given cell, it accumulates three counts—the number of objects in the cell, in the cell and the first layer together, and in the cell and both layers together. Let's refer to these counts as cell count, cell + 1 layer count, and cell + 2 layers count, respectively.

"How are outliers determined in this method?" Let M be the maximum number of outliers that can exist in the dmin-neighborhood of an outlier.

• An object, o, in the current cell is considered an outlier only if cell + 1 layer count is less than or equal to M. If this condition does not hold, then all of the objects in the cell can be removed from further investigation as they cannot be outliers.

• If cell + 2 layers count is less than or equal to M, then all of the objects in the cell are considered outliers. Otherwise, if this number is more than M, then it is possible that some of the objects in the cell may be outliers. To detect these outliers, object-by-object processing is used where, for each object, o, in the cell, objects in the second layer of o are examined. For objects in the cell, only those objects having no more than M points in their dmin-neighborhoods are outliers. The dmin-neighborhood of an object consists of the object's cell, all of its first layer, and some of its second layer.

3) Density-Based Local Outlier Detection

Statistical and distance-based outlier detection both depend on the overall or "global" distribution of the given set of data points, D. However, data are usually not uniformly distributed. These methods encounter difficulties when analyzing data with rather different density distributions, as illustrated in the following example.

Necessity for density-based local outlier detection. Figure 7.27 shows a simple 2-D data set containing 502 objects, with two obvious clusters. Cluster C1 contains 400 objects. Cluster C2 contains 100 objects. Two additional objects, o1 and o2 are clearly outliers. However, by distance-based outlier detection (which generalizes many notions from statistical-based outlier detection), only o1 is a reasonableDB(pct, dmin)-outlier, because if dmin is set to be less than the minimum distance between o2 andC2, then all 501 objects are further away from o2 than dmin.

Thus, o2 would be considered a DB(pct, dmin)-outlier, but so would all of the objects inC1! On the other hand, if dmin is set to be greater than the minimum distance between o2 andC2, then even when o2 is not regarded as an outlier, some points in C1 may still be considered outliers.

Figure 7.27 The necessity of density-based local outlier analysis.

4) Deviation-Based Outlier Detection

Deviation-based outlier detection does not use statistical tests or distance-based measures to identify exceptional objects. Instead, it identifies outliers by examining the main characteristics of objects in a group.

Objects that "deviate" from this description are considered outliers. Hence, in this approach the term deviations is typically used to refer to outliers. In this section, we study two techniques for deviation-based outlier detection. The first sequentially compares objects in a set, while the second employs an OLAP data cube approach.

Sequential Exception Technique

The sequential exception technique simulates the way in which humans can distinguish unusual objects from among a series of supposedly like objects. It uses implicit redundancy of the data. Given a data set, D, of n objects, it builds a sequence of subsets, {D1, D2, : : : , Dm}, of these objects with 2<= m <=n such that Dissimilarities are assessed between subsets in the sequence. The technique introduces the following key terms.

§ Exception set: This is the set of deviations or outliers. It is defined as the smallest subset of objects

whose removal results in the greatest reduction of dissimilarity in the residual set.

§ Dissimilarity function: This function does not require a metric distance between the objects. It is any function that, if given a set of objects, returns a low value if the objects are similar to one another. The greater the dissimilarity among the objects, the higher the value returned by the function. The dissimilarity of a subset is incrementally computed based on the subset prior to it in the sequence.

Given a subset of n numbers, {x1, …..,xn}, a possible dissimilarity function is the variance of the

numbers in the set, that is, where x is the mean of the n numbers in the set. For character strings, the dissimilarity function may be in the form of a pattern string (e.g., containing wildcard characters) that is used to cover all of the patterns seen so far.

The dissimilarity increases when the pattern covering all of the strings in Dj_1 does not cover any string in Dj that is not in Dj_1.

• Cardinality function:This is typically the count of the number of objects in a given set.

• Smoothing factor: This function is computed for each subset in the sequence. It assesses how much the dissimilarity can be reduced by removing the subset from the original set of objects. This value is scaled by the cardinality of the set. The subset whose smoothing factor value is the largest is the exception set.

OLAP Data Cube Technique

An OLAP approach to deviation detection uses data cubes to identify regions of anomalies in large

multidimensional data. This technique was described in detail in Chapter 4. For added efficiency, the deviation detection process is overlapped with cube computation. The approach is a form of discovery-driven exploration, in which precomputed measures indicating data exceptions are used to guide the user in data analysis, at all levels of aggregation. A cell value in the cube is considered an exception if it is significantly different from the expected value, based on a statistical model. The method uses visual cues such as background color to reflect the degree of exception of each

cell. The user can choose to drill down on cells that are flagged as exceptions. The measure value of a cell may reflect exceptions occurring at more detailed or lower levels of the cube, where these exceptions are not visible from the current level.

The model considers variations and patterns in the measure value across all of the dimensions to which a cell belongs. For example, suppose that you have a data cube for sales data and are viewing the sales summarized per month. With the help of the visual cues, you notice an increase in sales in December in comparison to all other months. This may seem like an exception in the

time dimension. However, by drilling down on the month of December to reveal the sales per item in that month, you note that there is a similar increase in sales for other items during December. Therefore, an increase in total sales in December is not an exception if the item dimension is considered. The model considers exceptions hidden at all aggregated group-by's of a data cube. Manual detection
of such exceptions is difficult because the search space is typically very large, particularly when there are many dimensions involving concept hierarchies with several levels.

APPLICATIONS AND TRENDS IN DATA MINING

Data Mining Applications

1) Data Mining for Financial Data Analysis

Most banks and financial institutions offer a wide variety of banking services (such as checking and savings accounts for business or individual customers), credit (such as business, mortgage, and automobile loans), and investment services (such as mutual funds). Some also offer insurance services and stock investment services.

Financial data collected in the banking and financial industry are often relatively complete, reliable, and of high quality, which facilitates systematic data analysis and data mining. Here we present a few typical cases:

• Design and construction of data warehouses for multidimensional data analysis and data mining:

Like many other applications, data warehouses need to be constructed for banking and financial data. Multidimensional data analysis methods should be used to analyze the general properties of such data.

• Loan payment prediction and customer credit policy analysis: Loan payment prediction and customer credit analysis are critical to the business of a bank. Many factors can strongly or weakly influence loan payment performance and customer credit rating. Data mining methods, such as attribute selection and attribute relevance ranking, may help identify important factors and eliminate irrelevant ones. For example, factors related to the risk of loan payments include loan-to-value ratio, term of the loan, debt ratio (total amount of monthly debt versus the total monthly income), payment to-income ratio, customer income level, education level, residence region, and credit history.

• Classification and clustering of customers for targeted marketing: Classification and clustering methods can be used for customer group identification and targeted marketing. For example, we can use classification to identify the most crucial factors that may influence a customer's decision regarding banking.

• Detection of money laundering and other financial crimes: To detect money laundering and other

financial crimes, it is important to integrate information from multiple databases (like bank transaction databases, and federal or state crime history databases), as long as they are potentially related to the study. Multiple data analysis tools can then be used to detect unusual patterns, such as large amounts of cash flow at certain periods, by certain groups of customers.

2) Data Mining for the Retail Industry

Retail data mining can help identify customer buying behaviors, discover customer shopping patterns and trends, improve the quality of customer service, achieve better customer retention and satisfaction, enhance goods consumption ratios, design more effective goods transportation and distribution policies, and reduce the cost of business.

A few examples of data mining in the retail industry are outlined as follows.

• Design and construction of data warehouses based on the benefits of data mining: Because retail data cover a wide spectrum (including sales, customers, employees, goods transportation, consumption, and services), there can be many ways to design a data warehouse for this industry. The levels of detail to include may also vary substantially. The outcome of preliminary data mining exercises can be used to help guide the design and development of data warehouse structures. This involves deciding which dimensions and levels to include and what preprocessing to perform in order to facilitate effective data mining.

• Multidimensional analysis of sales, customers, products, time, and region: The retail industry requires timely information regarding customer needs, product sales, trends, and fashions, as well
as the quality, cost, profit, and service of commodities. It is therefore important to provide powerful
multidimensional analysis and visualization tools, including the construction of sophisticated data cubes according to the needs of data analysis. The multifeature data cube, introduced in Chapter 4,
is a useful data structure in retail data analysis because it facilitates analysis on aggregates with complex conditions.

• Analysis of the effectiveness of sales campaigns: The retail industry conducts sales campaigns using advertisements, coupons, and various kinds of discounts and bonuses to promote products and attract customers. Careful analysis of the effectiveness of sales campaigns can help improve company profits.

• Customer retention—analysis of customer loyalty: With customer loyalty card information, one can register sequences of purchases of particular customers. Customer loyalty and purchase trends can be analyzed systematically. Goods purchased at different periods by the same

customers can be grouped into sequences.

• Product recommendation and cross-referencing of items: By mining associations from sales records, one may discover that a customer who buys a digital camera is likely to buy another set of
items. Such information can be used to form product recommendations. Collaborative recommender systems use data mining techniques to make personalized product recommendations during live customer transactions, based on the opinions of other customers

3) Data Mining for the Telecommunication Industry

The telecommunication industry has quickly evolved from offering local and long distance telephone services to providing many other comprehensive communication services, including fax, pager, cellular phone, Internet messenger, images, e-mail, computer and Web data transmission, and other data traffic. The integration of telecommunication, computer network, Internet, and numerous other means of communication and computing is also underway.

The following are a few scenarios for which data mining may improve telecommunication services:

• Multidimensional analysis of telecommunication data: Telecommunication data are intrinsically multidimensional, with dimensions such as calling-time, duration, location of caller, location of callee, and type of call. The multidimensional analysis of such data can be used to identify and compare the data traffic, system workload, resource usage, user group behavior, and profit. For example, analysts in the industry may wish to regularly view charts and graphs regarding calling source, destination, volume, and time-of-day usage patterns. Therefore, it is often useful to consolidate telecommunication data into large data warehouses and routinely perform multidimensional analysis using OLAP and visualization tools.

• Fraudulent pattern analysis and the identification of unusual patterns: Fraudulent activity costs the telecommunication industry millions of dollars per year. It is important to (1) identify potentially fraudulent users and their atypical usage patterns; (2) detect attempts to gain fraudulent entry to customer accounts; and (3) discover unusual patterns that may need special attention, such as busy-hour frustrated call attempts, switch and route congestion patterns, and periodic calls from automatic dial-out equipment (like fax machines) that have been improperly programmed

• Multidimensional association and sequential pattern analysis: The discovery of association and sequential patterns in multidimensional analysis can be used to promote telecommunication services. For example, suppose you would like to find usage patterns for a set of communication

services by customer group, by month, and by time of day. The calling records may be grouped by customer in the following form: (Customer-ID, residence; office, time, date, service 1, service 2, …)

• Mobile telecommunication services: Mobile telecommunication, Web and information services, and mobile computing are becoming increasingly integrated and common in our work and life. One important feature of mobile telecommunication data is its association with spatiotemporal information.

• Use of visualization tools in telecommunication data analysis: Tools for OLAP visualization, linkage
visualization, association visualization, clustering, and outlier visualization have been shown to be very useful for telecommunication data analysis.

4) Data Mining for Biological Data Analysis

The past decade has seen an explosive growth in genomics, proteomics, functional genomics, and
biomedical research. Examples range from the identification and comparative analysis of the genomes of human and other species (by discovering sequencing patterns, gene functions, and evolution paths) to the investigation of genetic networks and protein pathways, and the development of new pharmaceuticals and advances in cancer therapies. Biological data mining has become an essential part of a new research field called bioinformatics.

DNA sequences form the foundation of the genetic codes of all living organisms. All DNA sequences are comprised of four basic building blocks, called nucleotides: adenine (A), cytosine (C), guanine (G), and thymine (T).

These four nucleotides (or bases) are combined to form long sequences or chains that resemble a twisted ladder.

The DNA carry the information and biochemical machinery that can be copied from generation to generation. During the processes of "copying," insertions, deletions, or mutations (also called substitutions) of nucleotides are introduced into the DNA sequence, forming different evolution paths. A gene usually comprises hundreds of individual nucleotides arranged in a particular order. The nucleotides can be ordered and sequenced in an almost unlimited number of ways to form distinct genes. A genome is the complete set of genes of an organism. The human genome is estimated to contain around 20,000 to 25,000 genes. Genomics is the analysis of genome
sequences.

A linear string or sequence of DNA is translated into a sequence of amino acids, forming a protein (Figure 11.1). A proteome is the complete set of protein molecules present in a cell, tissue, or organism. Proteomics is the study of proteome sequences. Proteomes are dynamic, changing from minute to minute in response to tens of thousands of intra- and extra cellular environmental signals.

Figure 11.1 A DNA sequence and corresponding amino acid sequence.

5) Data Mining in Other Scientific Applications

• Data warehouses and data preprocessing: Data warehouses are critical for information exchange and data mining. In the area of geospatial data, however, no true geospatial data warehouse exists today. Creating such a warehouse requires finding means for resolving geographic and temporal data incompatibilities, such as reconciling semantics, referencing systems, geometry, accuracy, and precision.

• Mining complex data types: Scientific data sets are heterogeneous in nature, typically involving semistructured and unstructured data, such as multimedia data and georeferenced stream data. Robust methods are needed for handling spatiotemporal data, related concept hierarchies, and complex geographic relationships

• Graph-based mining: It is often difficult or impossible to model several physical phenomena and processes due to limitations of existing modeling approaches. Alternatively, labeled graphs may be used to capture many of the spatial, topological, geometric, and other relational characteristics present in scientific data sets. In graph modeling, each object to be mined is represented by a vertex in a graph, and edges between vertices represent relationships between objects.

• Visualization tools and domain-specific knowledge: High-level graphical user interfaces and visualization tools are required for scientific data mining systems. These should be integrated with existing domain-specific information systems and database systems to guide researchers and general users in searching for patterns, interpreting and visualizing discovered patterns, and using discovered knowledge in their decision making.

Trends in Data Mining

Some of the trends in data mining that reflect the pursuit of these challenges:

Ø Application exploration
Ø Scalable and interactive data mining methods
Ø Integration of data mining with database systems, data warehouse systems
Ø Web database systems

Ø Standardization of data mining language
Ø Visual data mining
Ø New methods for mining complex types of data
Ø Biological data mining
Ø Data mining and software engineering
Ø Web mining
Ø Distributed data mining
Ø Real-time or time-critical data mining
Ø Graph mining, link analysis, and social network analysis
Ø Multi-relational and multi-database data mining
Ø Privacy protection and information security in data mining

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. In this section, we describe three examples of model-based clustering. It presents an extension of the k-means partitioning algorithm, called Expectation-Maximization.

1) Expectation-Maximization

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. 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.

2) Conceptual Clustering

Conceptual clustering is a form of clustering in machine learning that, given a set of unlabeled objects, produces a classification scheme over the objects. Unlike conventional clustering, which primarily identifies groups of like objects, conceptual clustering goes one step further by also finding characteristic descriptions for each group, where each group represents a concept or class. Hence, conceptual clustering is a two-step process: clustering is performed first, followed by characterization. Here, clustering quality is not solely a function of the individual objects. Rather, it incorporates factors such as the generality and simplicity of the derived concept descriptions.

3) Neural Network Approach

The neural network approach is motivated by biological neural networks. Roughly speaking, a neural

network is a set of connected input/output units, where each connection has a weight associated with it. Neural networks have several properties that make them popular for clustering. First, neural networks are inherently

parallel and distributed processing architectures. Second, neural networks learn by adjusting their interconnection weights so as to best fit the data. This allows them to "normalize" or "prototype" the patterns and act as feature (or attribute) extractors for the various clusters. Third, neural networks process numerical vectors and require object patterns to be represented by quantitative features only. Many clustering tasks handle only numerical data or can transform their data into quantitative features if needed.

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.

1) CLIQUE: A Dimension-Growth Subspace Clustering Method

CLIQUE (CLustering InQUEst) was the first algorithm proposed for dimension-growth subspace clustering in high-dimensional space. In dimension-growth subspace clustering, the clustering process starts at singledimensional subspaces and grows upward to higher-dimensional ones. Because CLIQUE partitions each dimension like a grid structure and determines whether a cell is dense based on the number of points it contains, it can also be viewed as an integration of density-based and grid-based clustering methods. However, its overall approach is typical of subspace clustering for high-dimensional space, and so it is introduced

The ideas of the CLIQUE clustering algorithm are outlined as follows.

Ø Given a large set of multidimensional data points, the data space is usually not uniformly occupied by the data points. CLIQUE's clustering identifies the sparse and the "crowded" areas in space (or units), thereby discovering the overall distribution patterns of the data set.

Ø A unit is dense if the fraction of total data points contained in it exceeds an input model parameter. In CLIQUE, a cluster is defined as a maximal set of connected dense units.

2) PROCLUS: A Dimension-Reduction Subspace Clustering Method

PROCLUS (PROjected CLUStering) is a typical dimension-reduction subspace clustering method. That is, instead of starting from single-dimensional spaces, it starts by finding an initial approximation of the clusters in the high-dimensional attribute space. Each dimension is then assigned a weight for each cluster, and the updated weights are used in the next iteration to regenerate the clusters. This leads to the exploration of dense regions in all subspaces of some desired dimensionality and avoids the generation of a large number of overlapped clusters in projected dimensions of lower dimensionality.

Constraint-Based Cluster Analysis

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.

Important 16 mark Questions in Unit-V

Ø Explain Categorization of Major Clustering Methods
Ø Explain Partitioning Methods

Ø Explain Hierarchical Methods
Ø Explain Density-Based Methods and Grid Based Methods
Ø Explain Outlier Analysis
Ø Explain Data Mining Applications and trends in data mining