



UNIT-IV

Clustering and Applications

Cluster analysis–Types of Data in Cluster Analysis–Categorization of Major Clustering Methods– Partitioning Methods, Hierarchical Methods– Density–Based Methods, Grid–Based Methods, Outlier Analysis.

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.

Cluster analysis has been widely used in many applications such as business intelligence, image pattern recognition, Web search, biology, and security. In business intelligence, clustering can be used to organize a large number of customers into groups, where customers within a group share strong similar characteristics. This facilitates the development of business strategies for enhanced customer relationship management. Moreover, consider a consultant company with a large number of projects.

To improve project management, clustering can be applied to partition projects into categories based on similarity so that project auditing and diagnosis (to improve project delivery and outcomes) can be conducted effectively

Requirements for Cluster Analysis:

Clustering is a challenging research field. In this section, you will learn about the requirements for clustering as a data mining tool, as well as aspects that can be used for comparing clustering methods. 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 or even billions of objects, particularly in Web search scenarios. Clustering on only a sample of a given large data set may lead to biased results. Therefore, highly scalable clustering algorithms are needed.

Ability to deal with different types of attributes: Many algorithms are designed to cluster numeric (interval-based) data. However, applications may require clustering other data types, such as binary, nominal (categorical), and ordinal data, or mixtures of these data types. Recently, more and more applications need clustering techniques for complex data types such as graphs, sequences, images, and documents.

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. Consider sensors, for example, which are often deployed for environment surveillance. Cluster analysis on sensor readings can detect interesting phenomena. We may want to use clustering to find the frontier of a running forest fire, which is often not spherical. It is important to develop algorithms that can detect clusters of arbitrary shape.

Requirements for domain knowledge to determine input parameters: Many clustering algorithms require users to provide domain knowledge in the form of input parameters such as the desired number of clusters.

Consequently, the clustering results may be sensitive to such parameters. Parameters are often hard to determine, especially for high-dimensionality data sets and where users have yet to grasp a deep understanding of their data. Requiring the specification of domain knowledge not only burdens users, but also makes the quality of clustering difficult to control.

Ability to deal with noisy data: Most real-world data sets contain outliers and/or missing,

unknown, or erroneous data. Sensor readings, for example, are often noisy—some readings may be inaccurate due to the sensing mechanisms, and some readings may be erroneous due to interferences from surrounding transient objects. Clustering algorithms can be sensitive to such noise and may produce poor-quality clusters. Therefore, we need clustering methods that are robust to noise.

Incremental clustering and insensitivity to input order: In many applications, incremental updates (representing newer data) may arrive at any time. Some clustering algorithms cannot incorporate incremental updates into existing clustering structures and, instead, have to recompute a new clustering from scratch. Clustering algorithms may also be sensitive to the input data order.

That is, given a set of data objects, clustering algorithms may return dramatically different clusterings depending on the order in which the objects are presented. Incremental clustering algorithms and algorithms that are insensitive to the input order are needed.

Types of Data in Cluster Analysis:

- Interval-Scaled variables
- Binary variables
- Nominal, Ordinal, and Ratio variables
- Variables of mixed types

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.

Binary variables:

A binary variable is a variable that can take only 2 values. For example, generally, gender variables can take 2 variables male and female.

Contingency Table For Binary Data:

Let us consider binary values 0 and 1
 Let $p = a + b + c + d$ \

Simple	matching	coefficient (invariant,	if	the	binary	variable	is	symmetric):
Jaccard	coefficient (noninvariant	if	the	binary	variable	is	asymmetric):	

Nominal variables:

A generalization of the binary variable in that it can take more than 2 states, e.g., red, yellow, blue, green.

Method 1: Simple matching

The dissimilarity between two objects i and j can be computed based on the simple matching.

m: Let m be no of matches (i.e., the number of variables for which i and j are in the same state).

p: Let p be total no of variables.

Method 2: use a large number of binary variables

Creating a new binary variable for each of the M nominal states.

Ordinal Variables:

An ordinal variable can be discrete or continuous. In this order is important, e.g., rank. It can be treated like interval-scaled By replacing x_{if} by their rank,

By mapping the range of each variable onto $[0, 1]$ by replacing the i -th object in the f -th variable by, Then compute the dissimilarity using methods for interval-scaled variables.

Ratio-Scaled Intervals

Ratio-scaled variable: It is a positive measurement on a nonlinear scale, approximately at an exponential scale, such as Ae^{Bt} or A^e-Bt .

Methods:

- First, treat them like interval-scaled variables — not a good choice! (why?)
- Then apply logarithmic transformation i.e. $y = \log(x)$

Categorization of Major Clustering Methods:

Clustering methods can be classified into the following categories –

- Partitioning Method
- Hierarchical Method
- Density-based Method
- Grid-Based Method
- Outlier Analysis

Partitioning Method:

The simplest and most fundamental version of cluster analysis is partitioning, which organizes the objects of a set into several exclusive groups or clusters. To keep the problem specification concise, we can assume that the number of clusters is given as background knowledge. This parameter is the starting point for partitioning methods. Formally, given a data set, D , 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” to one another and “dissimilar” to objects in other clusters in terms of the data set attributes. In this section you will learn the most well-known and commonly used partitioning methods— k -means (Section 10.2.1) and k -medoids (Section 10.2.2). You will also learn several variations of these classic partitioning methods and how they can be scaled up to handle large data sets.

k-Means: A Centroid-Based Technique:

Suppose a data set, D , contains n objects in Euclidean space. Partitioning methods distribute the objects in D into k clusters, C_1, \dots, C_k , that is, $C_i \subset D$ and $C_i \cap C_j = \emptyset$ for $(1 \leq i, j \leq k)$. An objective function is used to assess the partitioning quality so that objects within a cluster are similar to one another but dissimilar to objects in other clusters. This is, the objective function aims for high intracluster similarity and low intercluster similarity. A centroid-based partitioning technique uses the centroid of a cluster, C_i , to represent that cluster. Conceptually, the centroid of a cluster is its center point. The centroid can be defined in various ways such as by the mean or method of the objects (or points) assigned to the cluster. The difference between an object $p \in C_i$ and c_i , the representative of the cluster, is measured by $\text{dist}(p, c_i)$, where $\text{dist}(x, y)$ is the Euclidean distance between two points x and y .

Hierarchical Method:

While partitioning methods meet the basic clustering requirement of organizing a set of objects into a number of exclusive groups, in some situations we may want to partition our data into groups at different levels such as in a hierarchy. A hierarchical clustering method works by grouping data objects into a hierarchy or “tree” of clusters. Representing data objects in the form of a hierarchy is useful for data summarization and visualization. For example, as the manager of human resources at All Electronics, you may organize your employees into major groups such as executives, managers, and staff. You can further partition these groups into smaller subgroups. For instance, the general group of staff can be further divided into subgroups of senior officers, officers, and trainees. All these groups form a hierarchy. We can easily summarize or characterize the data that are organized into a hierarchy, which can be used to find, say, the average salary of managers and of officers. Consider handwritten character recognition as another example. A set of handwriting samples may be first partitioned into general groups where each group corresponds to a unique character. Some groups can be further partitioned into subgroups since a character may be written in multiple substantially different ways. If necessary, the hierarchical partitioning can be continued recursively until a desired granularity is reached. In the previous examples, although we partitioned the data hierarchically, we did not assume that the data have a hierarchical structure (e.g., managers are at the same level in our All Electronics hierarchy as staff). Our use of a hierarchy here is just to summarize and represent the underlying data in a compressed way. Such a hierarchy is particularly useful for data visualization. Alternatively, in some applications we may believe that the data bear an underlying hierarchical structure that we want to discover. For example, hierarchical clustering may uncover a hierarchy for All Electronics employees structured on, say, salary. In the study of evolution, hierarchical clustering may group animals according to their biological features to uncover evolutionary paths, which are a hierarchy of species. As another example, grouping configurations of a strategic game (e.g., chess or checkers) in a hierarchical way may help to develop game strategies that can be used to train players.

Density-based Method:

Partitioning and hierarchical methods are designed to find spherical-shaped clusters. They have difficulty finding clusters of arbitrary shape such as the “S” shape and oval clusters in Figure 10.13. Given such data, they would likely inaccurately identify convex regions, where noise or outliers are included in the clusters. To find clusters of arbitrary shape, alternatively, we can model clusters as dense regions in the data space, separated by sparse regions. This is the main strategy behind density-based clustering methods, which can discover clusters of non spherical shape. In this section, you will learn the basic techniques of density-based clustering by studying three representative methods, namely, DBSCAN (Section 10.4.1), OPTICS (Section 10.4.2), and DENCLUE (Section 10.4.3).

DBSCAN: Density-Based Clustering Based on Connected Regions with High Density “How can we find dense regions in density-based clustering?” The density of an object o can be measured by the number of objects close to o . DBSCAN (Density-Based Spatial Clustering of Applications with Noise) finds core objects, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters. “How does DBSCAN quantify the neighborhood of an object?” A user-

specified parameter $\epsilon > 0$ is used to specify the radius of a neighborhood we consider for every object. The ϵ -neighborhood of an object o is the space within a radius centered at o . Due to the fixed neighborhood size parameterized by ϵ , the density of a neighborhood can be measured simply by the number of objects in the neighborhood. To determine whether a neighborhood is dense or not, DBSCAN uses another user-specified parameter, Min Pts, which specifies the density threshold of dense regions. An object is a core object if the ϵ -neighborhood of the object contains at least MinPts objects. Core objects are the pillars of dense regions. Given a set, D , of objects, we can identify all core objects with respect to the given parameters, ϵ and MinPts. The clustering task is therein reduced to using core objects and their neighborhoods to form dense regions, where the dense regions are clusters. For a core object q and an object p , we say that p is directly density-reachable from q (with respect to ϵ and Min Pts) if p is within the ϵ -neighborhood of q . Clearly, an object p is directly density-reachable from another object q if and only if q is a core object and p is in the ϵ -neighborhood of q . Using the directly density-reachable relation, a core object can “bring” all objects from its ϵ -neighborhood into a dense region.

Grid-Based Method:

The clustering methods discussed so far are data-driven—they partition the set of objects and adapt to the distribution of the objects in the embedding space. Alternatively, a grid-based clustering method takes a space-driven approach by partitioning the embedding space into cells independent of the distribution of the input objects. 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. In this section, we illustrate grid-based clustering using two typical examples. STING (Section 10.5.1) explores statistical information stored in the grid cells. CLIQUE (Section 10.5.2) represents a grid- and density-based approach for subspace clustering in a high-dimensional data space.

STING: Statistical Information Grid:

STING is a grid-based multiresolution clustering technique in which the embedding spatial area of the input objects is divided into rectangular cells. The space can be divided in a hierarchical and recursive way. Several levels of such rectangular cells correspond to different levels of resolution and 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 pre computed and stored as statistical parameters.

These statistical parameters are useful for query processing and for other data analysis tasks. Figure 10.19 shows a hierarchical structure for STING clustering. The 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; and 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). Here, the attribute is a selected measure for analysis such as price for house objects. 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 χ^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 lower-level cells disagree with each other and fail the threshold test, the distribution type of the high-level cell is set to none. “How is this statistical information useful for query answering?” The statistical parameters can be used in a top-down, grid-based manner as follows. First, a layer within the hierarchical structure is determined from which the query-answering process

is to start. This layer typically contains a small number of cells. For each cell in the current layer, we compute the confidence interval (or estimated probability range) reflecting the cell's relevancy to the given query. The irrelevant cells are removed from further consideration. Processing of the next lower level examines only the remaining relevant cells. This process is repeated until the bottom layer is reached. At this time, if the query specification is met, the regions of relevant cells that satisfy the query are returned. Otherwise, the data that fall into the relevant cells are retrieved and further processed until they meet the query's requirements. An interesting property of STING is that it approaches the clustering result of DBSCAN if the granularity approaches 0 (i.e., toward very low-level data).

In other words, using the count and cell size information, dense clusters can be identified approximately using STING. Therefore, STING can also be regarded as a density-based clustering method. "What advantages does STING offer over other clustering methods?" STING offers several advantages: (1) the grid-based computation is query-independent because the statistical information stored in each cell represents the summary information of the data in the grid cell, independent of the query; (2) the grid structure facilitates parallel processing and incremental updating; and (3) the method's efficiency is a major advantage: STING goes through the database once to compute the statistical parameters of the cells, and hence the time complexity of generating clusters is $O(n)$, where n is the total number of objects. After generating the hierarchical structure, the query processing time is $O(g)$, where g is the total number of grid cells at the lowest level, which is usually much smaller than n . Because STING uses a multiresolution approach to cluster analysis, the quality of STING clustering depends on the granularity of the lowest level of the grid structure. If the granularity is very fine, the cost of processing will increase substantially; however, if the bottom level of the grid structure is too coarse, it may reduce the quality of cluster analysis. Moreover, STING does not consider the spatial relationship between the children and their neighboring cells for construction of a parent cell. As a result, the shapes of the resulting clusters are isothetic, that is, all the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected. This may lower the quality and accuracy of the clusters despite the fast processing time of the technique.

Outlier Analysis:

Outlier is a data object that deviates significantly from the rest of the data objects and behaves in a different manner. An outlier is an object that deviates significantly from the rest of the objects. They can be caused by measurement or execution errors. The analysis of outlier data is referred to as outlier analysis or outlier mining.

An outlier cannot be termed as a noise or error. Instead, they are suspected of not being generated by the same method as the rest of the data objects.

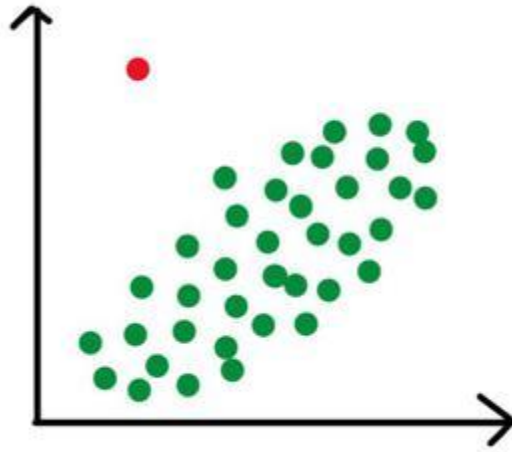
Outliers are of three types, namely –

- Global (or Point) Outliers
- Collective Outliers
- Contextual (or Conditional) Outliers

1. Global Outliers

They are also known as *Point Outliers*. These are the simplest form of outliers. If, in a given dataset, a data point strongly deviates from all the rest of the data points, it is known as a global outlier. Mostly, all of the outlier detection methods are aimed at finding global outliers.

For example, In Intrusion Detection System, if a large number of packages are broadcast in a very short span of time, then this may be considered as a global outlier and we can say that that particular system has been potentially hacked.

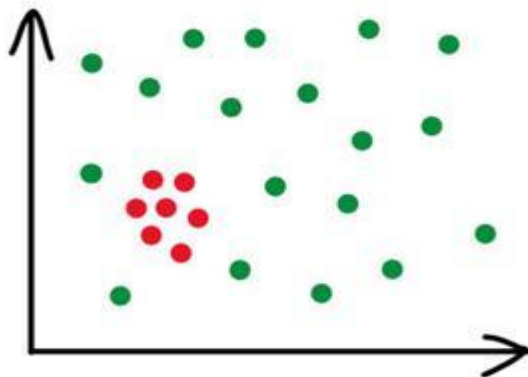


The red data point is a global outlier.

2. Collective Outliers

As the name suggests, if in a given dataset, some of the data points, as a whole, deviate significantly from the rest of the dataset, they may be termed as collective outliers. Here, the individual data objects may not be outliers, but when seen as a whole, they may behave as outliers. To detect these types of outliers, we might need background information about the relationship between those data objects showing the behavior of outliers.

For example: In an Intrusion Detection System, a DOS (denial-of-service) package from one computer to another may be considered as normal behavior. However, if this happens with several computers at the same time, then this may be considered as abnormal behavior and as a whole they can be termed as collective outliers.

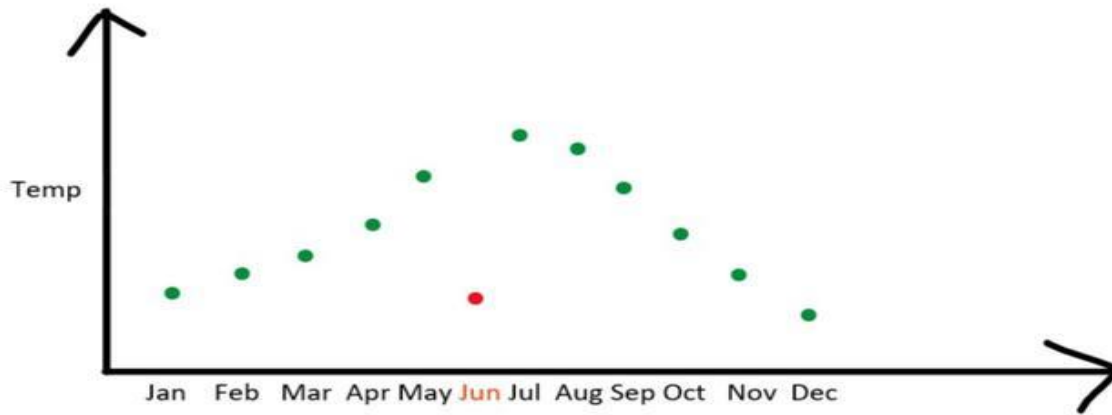


The red data points as a whole are collective outliers.

3. Contextual Outliers

They are also known as *Conditional Outliers*. Here, if in a given dataset, a data object deviates significantly from the other data points based on a specific context or condition only. A data point may be an outlier due to a certain condition and may show normal behavior under another condition. Therefore, a context has to be specified as part of the problem statement in order to identify contextual outliers. Contextual outlier analysis provides flexibility for users where one can examine outliers in different contexts, which can be highly desirable in many applications. The attributes of the data point are decided on the basis of both contextual and behavioral attributes.

For example: A temperature reading of 40°C may behave as an outlier in the context of a “winter season” but will behave like a normal data point in the context of a “summer season”.



A low temperature value in June is a contextual outlier because the same value in December is not an outlier.