**UNIT V**

**CLUSTERING:** Similarity and distance measures, Hierarchical algorithms, Partitional algorithms, Clustering large databases, Clustering with categorical attributes.

**Objectives:**

To understand how the similarities among the data objects will be calculated.

To explore the applications of clustering

To implement basic clustering algorithms

**Cluster Analysis**

Cluster is a collection of data objects which are similar to one another within the same cluster.

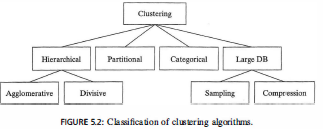
Cluster analysis: Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters

**5.1 Major issues in Clustering:**

* **Scalability:** The ability of the algorithm to perform well with large number of data objects (tuples).
* **Analyze mixture of attribute types**: The ability to analyze single as well as mixtures of attribute types.
* **Find arbitrary-shaped clusters**: The shape usually corresponds to the kinds of clusters an algorithm can find and we should consider this as a very important thing when choosing a method, since we want to be as general as possible. Different types of algorithms will be biased towards finding different types of cluster structures/shapes and it is not always an easy task to determine the shape or the corresponding bias.
* **Minimum requirements for input parameters:** Many clustering algorithms require some user-defined parameters, such as the number of clusters, in order to analyze the data.
* **Sensitivity to the order of input records**: The same data set, when presented to certain algorithms in different orders, may produce dramatically different results.
* **High dimensionality of data:** The number of attributes/dimensions in many data sets is large, and many clustering algorithms cannot handle more than a small number (eight to ten) of dimensions.
* **Interpretability and usability:** when it comes to comparing the results with preconceived ideas or constraints, some techniques fail to be satisfactory. Therefore, easy to understand results are highly desirable.

A classification o f the different types of clustering algorithms is shown i n Figure 5 .2.

* Clustering algorithms themselves may be viewed as hierarchical or partitional. With



hierarchical clustering, a nested set of clusters is created. Each level in the hierarchy has a separate set of clusters. At the lowest level, each item is in its own unique cluster. At the highest level, all items belong to the same cluster. With hierarchical clustering, the desired number of clusters i$ not input. With partitional clustering, the algorithm creates only one set of clusters. These approaches use the desired number of clusters to drive how the final set is created. Traditional clustering algorithms tend to be targeted to small numeric databases that fit into memory. There are, however, more recent clustering algorithms that look at categorical data and are targeted to larger, perhaps dynamic, databases. Algorithms targeted to larger databases may adapt to memory constraints by either sampling the database or using data structures, which can be compressed or pruned to fit mto memory regardless of the size of the database. Clustering algorithms may also differ basd on whether they produce overlapping or nonoverlapping clusters. Even though we consider only nonoverlapping clusters, it is possible to place an item in multiple clusters. In turn, nonoverlapping clusters can be viewed as extrinsic or intrinsic. Extrinsic techniques use labeling of the items to assist in the classification process. These algorithms are the traditional classification supervised learning algorithms in which a special input training set is used. Intrinsic algorithms do not use any a priori category labels, but depend only on the adjacency matrix containing the distance between objects

The types of clustering algorithms can be furthered classified based on the implementation technique used. Hierarchical algorithms can be categorized as agglomerative or divisive. "Agglomerative" implies that the clusters are created in a bottom-up fashion, while divisive algorithms work in a top-down fashion. Although both hierarchical and partitional algorithms could be described using the agglomerative vs. Divisive typically is more associated with hierarchical algorithms.

Another descnptlve tag medicates whether each individual element is handled one by one, serial (sometimes called incrementa[), or whether all items are examined together, simultaneous. If a specific tuple is viewed as having attribute values for all attributes in the schema, then clustering algorithms

could differ as to how the attribute values are examined. As is usually done with decision tree classification techniques, some algorithms examine attribute values one at a time, monothetic. Polythetic algorithms consider all attribute values at one time. Finally, clustering algorithms can be labeled based on the mathematical formulation given to the algorithm: graph theoretic or matrix algebra. In this chapter we generally use the graph approach and describe the input to the clustering algorithm as an adjacency matnx labeled with distance measures.

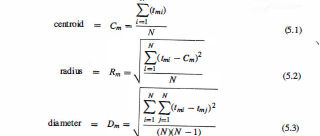
**5.2 Similarity measures**

There are many desirable properties for the clusters created by a solution to a specific clustering problem. The most important one is that a tuple within one cluster is more like tuples within that cluster than it is similar to tuples outside it. As with classification, then, we assume the definition of a similarity measure, sim(t; , t1 ), defined between any two tuples, t; , tt E D. This provides a more strict and alternative clustering definition, as found in Definition 5.2. Unless otherwise stated, we use the first definition rather than the second. Keep in mind that the similarity relationship stated within the second definition is a desirable, although not always obtainable, property .

A distance measure, dis (t; , tj ), as opposed to similarity, is often used in clustering.

Some clustering algorithms look only at numeric data, usually assuming metric

data points. Metric attributes satisfy the triangular inequality. The clusters can then be described by using several characteristic yalues. Given a cluster, Km of N points {tm l , tm 2 •... , tm N }, we make the following definitions [ZRL9 6] :

****

Many clustering algorithms require that the distance between clusters (rather than elements) be determined. This is not an easy task given that there are many interpretations for distance between clusters. Given clusters Ki and KJ , there are several standard alternatives to calculate the distance between clusters. A representative list is :

• Single link: Smallest distance between an element in one cluster and an eiement in the other. We thus have dis(Ki , KJ) = min(dis(tu , tJm )) Vtu E Ki fj KJ and Vtjm E KJ rj Ki .

• Complete link: Largest distance between an element in one cluster and an element in the other. We thus have dis(Ki , KJ) = max(dis(tu , tjm ))Vtu E Ki rj KJ and VtJm E KJ rj Ki .

• Average: Average distance between an element in one cluster and an element in the other. We thus have dis(Ki , Kj) = mean(dis(tu , tJm ))Vtu E Ki fj KJ and

Vtjm E KJ fj Ki .

• Centroid: If 􀅵lusters have a representative centroid, then the centroid distance is defined as the distance between the centroids. We thus have dis(Ki , K 1) = dis(Ci , CJ), where Ci is the centroid for Ki and similarly for Cj .

• Medoid: Using a medoid to represent each cluster, the distance between the clusters can be defined by the distance between the medoids: dis(Ki , Kj) = dis(Mi , MJ ) .

**5.3 Partitioning Algorithms:**

Given a database of n objects, a partition clustering algorithm constructs k partitions of the data, where each cluster optimizes a clustering criterion, such as the minimization of the sum of squared distance from the mean within each cluster. Partitioning algorithms try to locally improve a certain criterion. First, they compute the values of the similarity or distance, they order the results, and pick the one that optimizes the criterion. Hence, the majority of them could be considered as greedy-like algorithms.**K Means Algorithm:**

Input : ‘k’, the number of clusters to be partitioned; ‘n’, the number of objects.

Output: A set of ‘k’ clusters based on given similarity function.

Steps:

1. Arbitrarily choose ‘k’ objects as the initial cluster centers;

ii) Repeat,

a. (Re)assign each object to the cluster to which the object is the most similar; based on the given similarity function;

b. Update the centroid (cluster means), i.e., calculate the mean value of the objects for each cluster;

iii) Until no change

Strengths:

1. Relatively scalable and efficient in processing large data sets; complexity is O (i k n), where i is the total number of iterations, k is the total number of clusters, and n is the total number of objects. Normally, k<<n and i<<n.
2. Easy to understand and implement.

Weaknesses:

1. Applicable only when the mean of a cluster is defined; not applicable to categorical data.
2. Need to specify k, the total number of clusters in advance.
3. Not suitable to discover clusters with non-convex shape, or clusters of very different size.
4. Unable to handle noisy data and outliers.
5. May terminate at local optimum.
6. Result and total run time depends upon initial partition.

**K Medoids Algorithm:**

**Input**: ‘k’, the number of clusters to be partitioned; ‘n’, the number of objects

**Output**: A set of ‘k’ clusters that minimizes the sum of the dissimilarities of all the objects to their nearest medoid.

i) Arbitrarily choose ‘k’ objects as the initial medoids;

ii) Repeat,

1. Assign each remaining object to the cluster with the nearest medoid;
2. Randomly select a non-medoid object;
3. Compute the total cost of swapping old medoid object Compute the total cost of

swapping old medoid object.

1. If the total cost of swapping is less than zero, then perform that swap operation to form the new set of k- medoids.

iii) Until no change.

Strengths:

* More robust than k-means in the presence of noise and outliers; because a medoid is less influenced by outliers or other extreme values than a mean.

Weaknesses:

* Relatively more costly; complexity is O( i k (n-k)2), where i is the total number of iterations, is the total number of clusters, and n is the total number of objects.
* Relatively not so much efficient.
* Need to specify k, the total number of clusters in advance.
* Result and total run time depends upon initial partition.

**5.4 Hierarchical algorithms**

Hierarchical algorithms create a hierarchical decomposition of the objects. They are either agglomerative (bottom-up) or divisive (top-down):Agglomerative algorithms start with each object being a separate cluster itself, and successively merge groups according to a distance measure. The clustering may stop when all objects are in a single group or at any other point the user wants. These methods generally follow a greedy-like bottom-up merging.

(b) Divisive algorithms follow the opposite strategy. They start with one group of all objects and successively split groups into smaller ones, until each object falls in one cluster, or as desired. Divisive approaches divide the data objects in disjoint groups at every step, and follow the same pattern until all objects fall into a separate cluster. This is similar to the approach followed by divide-and-conquer algorithms.**Algorithmic steps for Agglomerative Nesting:**

Let  X = {x1, x2, x3, ..., xn} be the set of data points.

* Begin with the disjoint clustering having level L(0) = 0 and sequence number m = 0.
* Find the least distance pair of clusters in the current clustering, say pair (r), (s), according to d[(r),(s)] = min d[(i),(j)]   where the minimum is over all pairs of clusters in the current clustering.
* Increment the sequence number: m = m +1.Merge clusters (r) and (s) into a single cluster to form the next clustering   m. Set the level of this clustering to L(m) = d[(r),(s)].
* Update the distance matrix, D, by deleting the rows and columns corresponding to clusters (r) and (s) and adding a row and column corresponding to the newly formed cluster. The distance between the new cluster, denoted (r,s) and old cluster(k) is defined in this way: d[(k), (r,s)] = min (d[(k),(r)], d[(k),(s)]).
* If all the data points are in one cluster then stop, else repeat from step 2).

Divisive Analysis called DIANA is just the reverse of Agglomerative Hierarchical approach.

**Advantages**

1) No apriori information about the number of clusters required.

2) Easy to implement and gives best result in some cases.

**Disadvantages**

1) Algorithm can never undo what was done previously.

2) Time complexity of at least O(*n2log n*) is required, where *‘n’* is the number of data points.

3) Based on the type of distance matrix chosen for merging different algorithms can suffer with one or more of the following:

* + Sensitivity to noise and outliers
  + Breaking large clusters
  + Difficulty handling different sized clusters and convex shapes
  + No objective function is directly minimized

**Divisive Clustering**

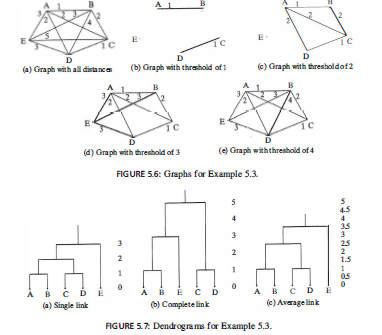
With divisive clustering, all items are initially placed in one cluster and clusters are repeatedly split in two until all items are in their own cluster. The idea is to split up clusters where some elements are not sufficiently close to other elements.

One simple example of a divisive algorithm is based on the MST version of the

single link algorithm. Here, however, we cut out edges from the MST from the largest to the smallest. Looking at Figure 5.8, we would start with a cluster containing all items: {A, B, C, D, E}. Looking at the MST, we see that the largest edge is between

D and E. Cutting this out of the MST, we then split the one cluster into two: { E } and

{ A , B, C, D}. Next we remove the edge between B and C. This splits the one large cluster into two: {A , B} and {C, D } . These will then be split at the next step. The order depends on how a specific implementation would treat identical values. Looking at the dendrogram in Figure 5.7(a), we see that we have created the same set of clusters as with the agglomerative approach, but in reverse order.

****

**5.5 Clustering large data bases**

**CURE**

It first partitions the random sample and partially clusters the data points in each partition. After eliminating outliers, the pre clustered data in each partition is then clustered in a final pass to generate the final clusters.

(1) The clustering algorithm can recognize arbitrarily shaped clusters (e.g., ellipsoidal),

(2) The algorithm is robust to the presence of outliers,

(3) The algorithm uses space that is linear in the input size n and has a worst-case time Complexity of O(n2 logn). For lower dimensions (e.g., two), the complexity can be shown to further reduce to O(n2).

(4) It appropriate for handling large data sets.

**BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)**

BIRCH is an agglomerative hierarchical clustering algorithm proposed by Charikar et al. in 1997. It is especially suitable for very large databases. This method has been designed so as to minimize the number of I/O operations. BIRCH incrementally and dynamically clusters incoming multi-dimensional metric data points to try to produce the best quality clustering with the available resources (i. e., available memory and time constraints). BIRCH can typically find a good clustering with a single scan of the data, and improve the quality further with a few additional scans. BIRCH is also the first clustering algorithm proposed in the database area to handle ''noise'' (data points that are not part of the underlying pattern) effectively. The data pre-processing algorithm BIRCH groups the data set into compact sub clusters that have summary statistics (called Clustering Features (CF)) associated to each of them. These CF's are computed and updated as the sub clusters are being constructed. The end result is an ``in-memory'' summary of the data, where ``local'' compact sub clusters are represented by appropriate summary statistics.

Clustering Feature is a triplet defined as CF = (N, LS, SS)

N: Number of data points

LS: ∑Ni=1=Xi

SS: ∑Ni=1=Xi2

Advantages:

* Scales linearly: finds a good clustering with a single scan and improves the quality with a few additional scans.
* Computation complexity of the algorithm is O(n), where n is number-objects.

Disadvantages:

* Handles only numeric data, and sensitive to the order of the data record.
* Favors only clusters with spherical shape and similar sizes, because it uses the notion of

Diameter to control the boundary of a cluster.

**Density-Based Clustering Methods**

Clustering based on density (local cluster criterion), such as density-connected points

Major features:

Discover clusters of arbitrary shape

Handle noise

One scan

Need density parameters as termination condition

**DBSCAN: Density Based Spatial Clustering of Applications with Noise**

* *Eps*: Maximum radius of the neighborhood
* *MinPts*: Minimum number of points in an Eps-neighbourhood of that point
* *NEps(p)*:  *{q belongs to D* | *dist(p,q) <= Eps}*
* Directly density-reachable: A point *p* is directly density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if *p* belongs to *NEps(q)* core point condition:|*NEps (q)*| >= *MinPts*
* Density-reachable: A point *p* is density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if there is a chain of points *p1*, …, *pn*, *p1* = *q*, *pn* = *p* such that *pi+1* is directly density-reachable from *pi*
* Density-connected: A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point *o* such that both, *p* and *q* are density-reachable from *o* w.r.t. *Eps* and *MinPts*

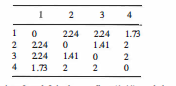
Algorithm:

* Arbitrary select a point *p*
* Retrieve all points density-reachable from *p* w.r.t. *Eps* and *MinPts*.
* If *p* is a core point, a cluster is formed.
* If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database.
* Continue the process until all of the points have been processed.

**5.6 CLUSTE RING WITH CATEGORICAL ATTRI BUTES**

Traditional algorithms do not always work with categorical data. Example 5.8 illustrates some problems that exist when clustering categorical data. This example uses a hierarchical-based centroid algorithm to illustrate the problems The problem illustrated here is that the centroid tends to weaken the relationship between the associated cluster and other clusters. The problems worsens as more and more clusters are merged. The number of attributes appearing in the mean increases, while the individual values actually decreases. This makes the centroid representations very similar and makes distinguishing between clusters difficult.

Consider an information retrieval system where documents may contain keywords {book, water, sun, sand, swim, read}. Suppose there are four documents, where the first contains the word {book}, the second contains {water, sun, sand, swim}, the third contains {water, sun, swim, read}, and the fourth contains {read, sand}. We can represent the four books using the following boolean points: ( 1 , 0, 0, 0, 0, 0), (0, 1, 1 , 1 , 1, 0), (0, 1 , 1, 0, 1 , 1), (0, 0, 0, 1 , 0, 1). We can use the Euclidean distance to develop the following adjacency matrix of distances:



The ROCK (RObust Clustering using linKs) clustering algoritlun is targeted to both boolean data and categorical data. A novel approach to identifying similarity is based on the number of links between items. A pair of items are said to be neighbors if their similarity exceeds some threshold. This need not be defined based on a precise metric, but rather a more intuitive approach using domain experts could be used. The number of links between two items is defined as the number of common neighbors they have. The objective of the clustering algoritlun is to group together points that have more links. The algorithm is a hierarchical agglomerative algorithm using the number of links as the similarity measure rather than a measure based on distance.

Instead of using a Euclidean distance, a different distance, such as the Jaccard

coefficient, has been proposed. One proposed similarity measure based on the Jaccard coefficient is defined as 1 I t· n t · sim(t; , tJ) = ' 1 I I t; u t1 I (5. 1 6)

If the tuples are viewed to be sets of items purchased (i.e., market basket data), then we look at the number of items they have in common divided by the total number in both. The denominator is used to normalize the value to be between 0 and 1 . The number of links between a pair of points can be viewed as the number of unique paths of length 2 between them. The authors argue that the use of links rather than similarity (distance) measures provides a more global approach because the similarity between points is impacted by other points as well. Example 5.9 illustrates the use of links by the ROCK algorithm using the data from Example 5.8 using the Jaccard coefficient.

Note that different threshold values for neighbors could be used to get different results. Also note that a hierarchical approach could be used with different threshold values for each level in the dendrogram.

**SUMMARY**

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. The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering.

Cluster analysis has extensive applications, including business intelligence, image pattern recognition, Web search, biology, and security. Cluster analysis can be used as a standalone data mining tool to gain insight into the data distribution, or as a preprocessing step for other data mining algorithms operating on the detected clusters.

Clustering is a dynamic ﬁeld of research in data mining. It is related to unsupervised learning in machine learning.

Clustering is a challenging ﬁeld. Typical requirements of it include scalability, the ability to deal with different types of data and attributes, the discovery of clusters in arbitrary shape, minimal requirements for domain knowledge to determine input parameters, the ability to deal with noisy data, incremental clustering and insensitivity to input order, the capability of clustering high-dimensionality data, constraint-based clustering, as well as interpretability and usability.

Many clustering algorithms have been developed. These can be categorized from several orthogonal aspects such as those regarding partitioning criteria, separation of clusters, similarity measures used, and clustering space. This chapter discusses major fundamental clustering methods of the following categories: partitioning methods, hierarchical methods, density-based methods, and grid-based methods. Some algorithms may belong to more than one category.

A partitioning method ﬁrst creates an initial set of k partitions, where parameter k is the number of partitions to construct. It then uses an iterative relocation technique that attempts to improve the partitioning by moving objects from one group to another. Typical partitioning methods include k-means, k-medoids, and

CLARANS. A hierarchical method creates a hierarchical decomposition of the given set of data objects. The method can be classiﬁed as being either agglomerative (bottom-up) or divisive (top-down), based on how the hierarchical decomposition is formed. To compensate for the rigidity of merge or split, the quality of hierarchical agglomeration can be improved by analyzing object linkages at each hierarchical partitioning (e.g., in Chameleon), or by ﬁrst performing micro clustering (that is, grouping objects into “microclusters”) and then operating on the microclusters with other clustering techniques such as iterative relocation (as in BIRCH).

A density-based method clusters objects based on the notion of density. It grows clusters either according to the density of neighborhood objects (e.g., in DBSCAN)

A grid-based method ﬁrst quantizes the object space into a ﬁnite number of cells that form a grid structure, and then performs clustering on the grid structure. STING is a typical example of a grid-based method based on statistical information stored in grid cells. CLIQUE is a grid-based and subspace clustering algorithm.