UNIT-5

Cluster Analysis: Basic Concepts and Methods- Cluster Analysis, partitioning methods, Hierarchical Methods and evaluation of Clustering

Cluster Analysis: Basic Concepts and Methods:

- What Is Cluster Analysis?
- Requirements for Cluster Analysis
- Overview of Basic Clustering Methods

What is a Cluster?

The given data is divided into different groups by combining similar objects into a group. This group is nothing but a cluster.

A cluster is nothing but a collection of similar data which is grouped together.

Clustering is known as unsupervised learning because the class label information is not present. For this reason, clustering is a form of learning by observation, rather than learning by examples.

A good clustering algorithm aims to obtain clusters whose:

- The **intra-cluster similarities** are high, It implies that the data present inside the cluster is similar to one another.
- The **inter-cluster similarity** is low, and it means each cluster holds data that is not similar to other data.

What Is Cluster Analysis?

Cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets.

Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters.

The set of clusters resulting from a cluster analysis can be referred to as a **clustering**.

For example,

consider a dataset of vehicles given in which it contains information about different vehicles like cars, buses, bicycles, etc.

As it is unsupervised learning there are no class labels like Cars, Bikes, etc for all the vehicles, all the data is combined and is not in a structured manner.

Clustering is also called data segmentation in some applications because clustering partitions large data sets into groups according to their similarity. Clustering can also be used for outlier detection, where outliers (values that are “far away” from any cluster) may be more interesting than common cases.

Applications of outlier detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce. For example, exceptional cases in credit card transactions, such as very expensive and infrequent purchases, may be of interest as possible fraudulent activities.

Requirements for Cluster Analysis:

The following are 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.

Capability of clustering high-dimensionality data: A data set can contain numerous dimensions or attributes. When clustering documents, for example, each keyword can be regarded as a dimension, and there are often thousands of keywords. Most clustering algorithms are good at handling low-dimensional data such as data sets involving only two or three dimensions. Finding clusters of data objects in a highdimensional space is challenging, especially considering that such data can be very 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 teller 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 types and number of customers per cluster. A challenging task is to find data groups with good clustering behavior that satisfy specified constraints.

Interpretability and usability: Users want clustering results to be interpretable, comprehensible, and usable. It is important to study how an application goal may influence the selection of clustering features and clustering methods.

The following are orthogonal aspects with which clustering methods can be compared:

The partitioning criteria: In some methods, all the objects are partitioned so that no hierarchy exists among the clusters. That is, all the clusters are at the same level conceptually. Such a method is useful, for example, for partitioning customers into groups so that each group has its own manager.

Separation of clusters: Some methods partition data objects into mutually exclusive clusters. When clustering customers into groups so that each group is taken care of by one manager, each customer may belong to only one group.

Similarity measure: Some methods determine the similarity between two objects by the distance between them. Such a distance can be defined on Euclidean space. Similarity measures play a fundamental role in the design of clustering methods.

Clustering space: Many clustering methods search for clusters within the entire given data space. These methods are useful for low-dimensionality data sets. With highdimensional data, however, there can be many irrelevant attributes, which can make similarity measurements unreliable. Consequently, clusters found in the full space are often meaningless. It's often better to instead search for clusters within different subspaces of the same data set. Subspace clustering discovers clusters and subspaces (often of low dimensionality) that manifest object similarity.

Overview of Basic Clustering Methods:

The clustering methods can be classified into the following categories:

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

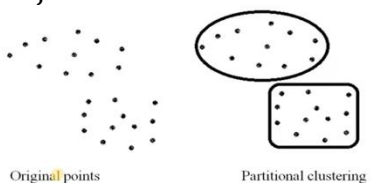
Partitioning Method

It is used to make partitions on the data in order to form clusters.

If "n" partitions are done on "p" objects of the database then each partition is represented by a cluster and n < p.

The two conditions which need to be satisfied with this Partitioning Clustering Method are:

- One objective should only belong to only one group.
- There should be no group without even a single purpose.
- In the partitioning method, there is one technique called iterative relocation, which means the object will be moved from one group to another to improve the partitioning



Most well-known and commonly used partitioning methods:

1) The k-means algorithm: where each cluster is represented by the mean value of the objects in the cluster.(Each cluster is represented by the center of the cluster)

2) The k-medoids algorithm: where each cluster is represented by one of the objects located near the center of the cluster.

Hierarchical clustering

Hierarchical clustering is a method of data mining that groups similar data points into clusters by creating a hierarchical structure of the clusters.

Identify the 2 clusters which can be closest together, and Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.

In Hierarchical Clustering, the aim is to produce a hierarchical series of nested clusters. A diagram called **Dendrogram** (A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits) graphically represents this hierarchy.

A hierarchical method can be classified as being either agglomerative or divisive

- **The agglomerative approach**, also called the bottom-up approach, starts with each object forming a separate group. It successively merges the objects or groups close to one another, until all the groups are merged into one (the topmost level of the hierarchy), or a termination condition holds.
- **The divisive approach**, also called the top-down approach, starts with all the objects in the same cluster. In each successive iteration, a cluster is split into smaller clusters, until eventually each object is in one cluster, or a termination condition holds.

Hierarchical methods suffer from the fact that once a step (merge or split) is done, it can never be undone.

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) or according to a density function (e.g., in DENCLUE). OPTICS is a density-based method that generates an augmented ordering of the data's clustering structure.

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.

Method	General Characteristics
Partitioning methods	– Find mutually exclusive clusters of spherical shape – Distance-based – May use mean or medoid (etc.) to represent cluster center – Effective for small- to medium-size data sets
Hierarchical methods	– Clustering is a hierarchical decomposition (i.e., multiple levels) – Cannot correct erroneous merges or splits – May incorporate other techniques like microclustering or consider object "linkages"
Density-based methods	– Can find arbitrarily shaped clusters – Clusters are dense regions of objects in space that are separated by low-density regions – Cluster density: Each point must have a minimum number of points within its "neighborhood" – May filter out outliers
Grid-based methods	– Use a multiresolution grid data structure – Fast processing time (typically independent of the number of data objects, yet dependent on grid size)

Figure: Overview of clustering methods

Partitioning Methods:

- k-Means: A Centroid-Based Technique
- k-Medoids: A Representative Object-Based Technique

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, C1,...,Ck , that is, Ci ⊂ D and Ci ∩Cj = ∅ for (1 ≤ i,j ≤ 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.

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

To measure the distance between data points and centroid,
we can use any method such as

- Euclidean distance or
- Manhattan distance.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

A centroid-based partitioning technique uses the centroid of a cluster, Ci , 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 medoid of the objects (or points) assigned to the cluster. The difference between an object p ∈ Ci and ci , the representative of the cluster, is measured by dist(p,ci), where dist(x,y) is the Euclidean distance between two points x and y.

The quality of cluster Ci can be measured by the withincluster variation, which is the sum of squared error between all objects in Ci and the centroid ci , defined as

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(\boldsymbol{p}, \boldsymbol{c_i})^2,$$

where E is the sum of the squared error for all objects in the data set; p is the point in space representing a given object; and ci is the centroid of cluster Ci (both p and ci are multidimensional).

"How does the k-means algorithm work?"

The k-means algorithm defines the centroid of a cluster as the mean value of the points within the cluster. It proceeds as follows.

First, it randomly selects k of the objects in D, 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 Euclidean distance between the object and the cluster mean.

The k-means algorithm then iteratively improves the within-cluster variation. For each cluster, it computes the new mean using the objects assigned to the cluster in the previous iteration.

All the objects are then reassigned using the updated means as the new cluster centers. The iterations continue until the assignment is stable, that is, the clusters formed in the current round are the same as those formed in the previous round.

The process of iteratively reassigning objects to clusters to improve the partitioning is referred to as iterative relocation

The k-means procedure:

The time complexity of the k-means 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 <<n and t<< n. Therefore, the method is relatively scalable and efficient in processing large data sets.

There are several 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.

The k-means method can be applied only when the mean of a set of objects is defined. This may not be the case in some applications such as when data with nominal attributes are involved.

The k-modes method is a variant of k-means, which extends the k-means paradigm to cluster nominal data by replacing the means of clusters with modes. It uses new dissimilarity measures to deal with nominal 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 nominal values.

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

Input:

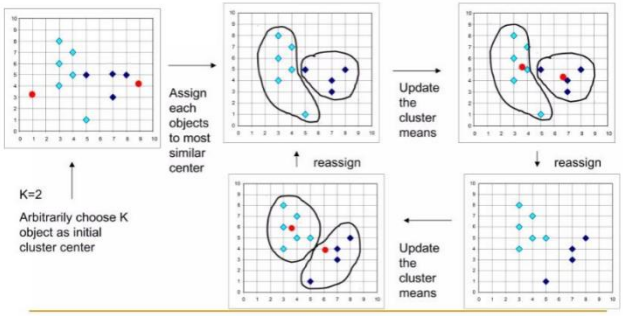
- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

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

Example: Clustering by k-means partitioning. Consider a set of objects located in 2-D space, as depicted in the following Figure Let k = 2, that is, the user would like the objects to be partitioned into two clusters.



Drawbacks of k-means:

- 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.

"How can we make the k-means algorithm more scalable?"

- One approach to making the k-means method more efficient on large data sets is to use a good-sized set of samples in clustering.
- Another is to employ a filtering approach that uses a spatial hierarchical data index to save costs when computing means.
- A third approach explores the microclustering idea, which first groups nearby objects into "microclusters" and then performs k-means clustering on the microclusters.

How to choose the value of "K number of clusters" in K-means Clustering?

Elbow Method

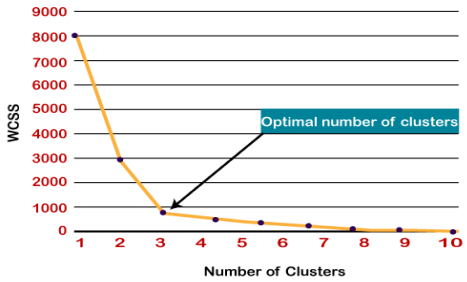
The Elbow method is one of the most popular ways to find the optimal number of clusters.

This method uses the concept of WCSS value. WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below**:**

$$WCSS= \sum_{P_i \text{ in Cluster1}} distance(P_i\ C_1)^2 + \sum_{P_i \text{ in Cluster2}} distance(P_i\ C_2)^2 + \sum_{P_i \text{ in CLuster3}} distance(P_i\ C_3)^2$$

To find the optimal value of clusters, the elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10). For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.
- Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



k-Medoids: A Representative Object-Based Technique:

"How can we modify the k-means algorithm to diminish such sensitivity to outliers?"

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 assigned to the cluster of which the representative object is the most similar. The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object p and its corresponding representative object. **That is, an absolute-error criterion is used, defined as**

$$E = \sum_{i=1}^{k} \sum_{\boldsymbol{p} \in C_i} dist(\boldsymbol{p}, \boldsymbol{o}_i),$$

where E is the sum of the absolute error for all objects p in the data set, and oi is the representative object of Ci . This is the basis for the k-medoids method, which groups n objects into k clusters by minimizing the absolute error.

The Partitioning Around Medoids (PAM) algorithm is a popular realization of k-medoids clustering. It tackles the problem in an iterative, greedy way. Like the k-means algorithm, the initial representative objects (called seeds) are chosen arbitrarily. We consider whether replacing a representative object by a non-representative object would improve the clustering quality. All the possible replacements are tried out.

The iterative process of replacing representative objects by other objects continues until the quality of the resulting clustering cannot be improved by any replacement.

This quality is measured by a cost function of the average dissimilarity between an object and the representative object of its cluster.

PAM, a k-medoids partitioning algorithm:

Specifically, let o1,...,ok be the current set of representative objects (i.e., medoids).

To determine whether a non-representative object, denoted by o random, is a good replacement for a current medoid oj (1 ≤ j ≤ k),

we calculate the distance from every object p to the closest object in the set {o1,...,oj−1,orandom,oj+1,...,ok}, and use the distance to update the cost function.

The reassignments of objects to {o1,...,oj−1,orandom,oj+1,...,ok} are simple. Suppose object p is currently assigned to a cluster represented by medoid oj .

Do we need to reassign p to a different cluster if oj is being replaced by orandom?

Object p needs to be reassigned to either orandom or some other cluster represented by oi (i 6= j), whichever is the closest.

Algorithm: *k*-medoids. PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input:

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

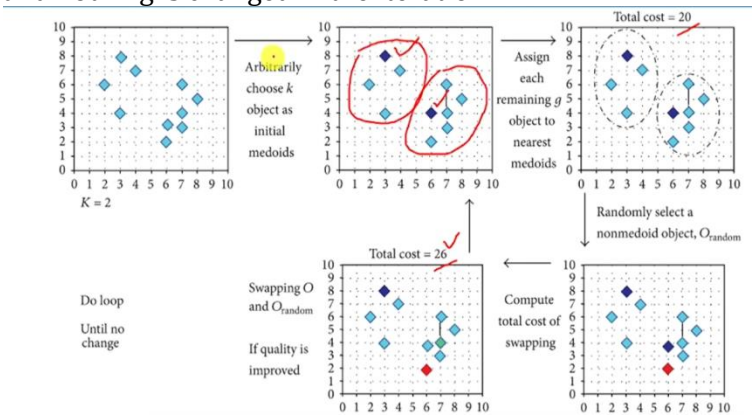
Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, *S*, of swapping representative object, o_j, with o_{random};
(6) **if** $S < 0$ **then** swap o_j with o_{random} to form the new set of *k* representative objects;
(7) **until** no change;

Each time a reassignment occurs, a difference in absolute error, E, is contributed to the cost function. Therefore, the cost function calculates the difference in absolute-error value if a current representative object is replaced by a no representative object. The total cost of swapping is the sum of costs incurred by all no representative

objects. If the total cost is negative, then oj is replaced or swapped with o random because the actual absolute-error E is reduced. If the total cost is positive, the current representative object, oj , is considered acceptable, and nothing is changed in the iteration.



"Which method is more robust—k-means or k-medoids?"

The k-medoids method is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean. However, the complexity of each iteration in the k-medoids algorithm is O(k(n − k) 2). For large values of n and k, such computation becomes very costly, and much more costly than the k-means method. Both methods require the user to specify k, the number of clusters.

"How can we scale up the k-medoids method?"

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. Instead of taking the whole data set into consideration, CLARA uses a random sample of the data set.

CLARA builds clusterings from multiple random samples and returns the best clustering as the output. A k-medoids partitioning algorithm like PAM works effectively for small data sets, but does not scale well for large data sets.

"How might we improve the quality and scalability of CLARA?"

A randomized algorithm called CLARANS (Clustering Large Applications based upon RANdomized Search) presents a trade-off between the cost and the effectiveness of using samples to obtain clustering.

Recall that when searching for better medoids, PAM examines every object in the data set against every current medoid,

whereas CLARA confines the candidate medoids to only a random sample of the data set.

First, it randomly selects k objects in the data set as the current medoids. It then randomly selects a current medoid x and an object y that is not one of the current medoids. Can replacing x by y improve the absolute-error criterion? If yes, the replacement is made. CLARANS conducts such a randomized search l times. The set of the current medoids after the l steps is considered a local optimum. CLARANS repeats this randomized process m times and returns the best local optimal as the final result.

Hierarchical Methods:

- Agglomerative versus Divisive Hierarchical Clustering
- Distance Measures in Algorithmic Methods
- BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees
- Chameleon: Multiphase Hierarchical Clustering Using Dynamic Modeling
- Probabilistic Hierarchical Clustering

Hierarchical Methods:

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.

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 microclustering (that is, grouping objects into "microclusters") and then operating on the microclusters with other clustering techniques such as iterative relocation (as in BIRCH).

For example, as the manager of human resources at AllElectronics, 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.

A hierarchical clustering method can be either agglomerative or divisive, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or topdown (splitting) fashion. Let's have a closer look at these strategies.

Agglomerative versus Divisive Hierarchical Clustering:

Agglomerative Hierarchical Clustering

- **An agglomerative hierarchical clustering method** uses a bottom-up strategy.
- It starts by letting each object form its own cluster and iteratively merges clusters into larger and larger clusters, until all the objects are in a single cluster or certain termination conditions are satisfied.
- The single cluster becomes the hierarchy's root.
- For the merging step, it finds the two clusters that are closest to each other (according to some similarity measure), and combines the two to form one cluster. Because two clusters are merged per iteration, where each cluster contains at least one object, an agglomerative method requires at most n iterations.

Divisive Hierarchical Clustering:

- **A divisive hierarchical clustering method** employs a top-down strategy.
- It starts by placing all objects in one cluster, which is the hierarchy's root. It then divides the root cluster into several smaller subclusters, and recursively partitions those clusters into smaller ones.
- The partitioning process continues until each cluster at the lowest level is coherent enough—either containing only one object, or the objects within a cluster are sufficiently similar to each other.

In either agglomerative or divisive hierarchical clustering, a user can specify the desired number of clusters as a termination condition.

Example :

Agglomerative versus divisive hierarchical clustering. Figure shows the application of AGNES (AGglomerative NESting), an agglomerative hierarchical clustering method, and DIANA (DIvisive ANAlysis), a divisive hierarchical clustering method, on a data set of five objects, {a,b,c,d, e}.

Initially, AGNES, the agglomerative method, 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 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 the objects are eventually merged to form one cluster.

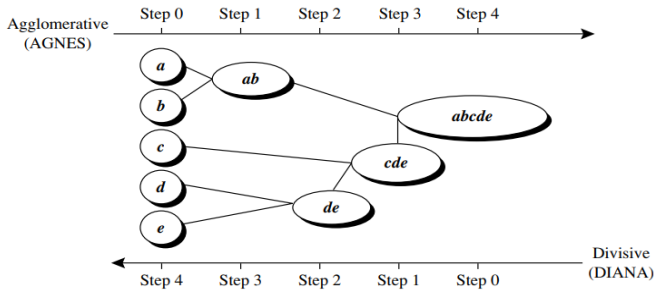


Figure-1: Agglomerative and divisive hierarchical clustering on data objects {a,b,c,d, e}

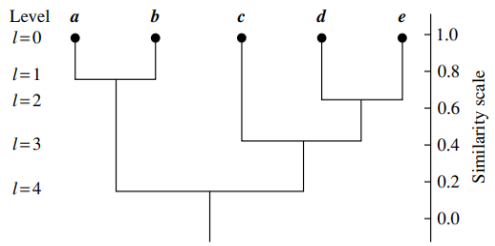


Figure-2 : Dendrogram representation for hierarchical clustering of data objects {a,b,c,d, e}.

DIANA, the divisive method, proceeds in the contrasting way. All 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.

Dendrogram:

A tree structure called a dendrogram is commonly used to represent the process of hierarchical clustering. It shows how objects are grouped together (in an agglomerative method) or partitioned (in a divisive method) step-by-step. Figure-2 shows a dendrogram for the five objects presented in Figure-1, 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, {a,b} and {c,d, e}, is roughly 0.16, they are merged together to form a single cluster.

A challenge with divisive methods is how to partition a large cluster into several smaller ones. For example, there are 2n−1 − 1 possible ways to partition a set of n objects into two exclusive subsets, where n is the number of objects.

Advantages of Hierarchical clustering:

- It is simple to implement and gives the best output in some cases.
- It is easy and results in a hierarchy, a structure that contains more information.
- It does not need us to pre-specify the number of clusters.

Disadvantages of hierarchical clustering

- It breaks the large clusters.
- It is Difficult to handle different sized clusters and convex shapes.
- It is sensitive to noise and outliers.
- The algorithm can never be changed or deleted once it was done previously.

Distance Measures in Algorithmic Methods:

Whether using an agglomerative method or a divisive method, a core need is to measure the distance between two clusters, where each cluster is generally a set of objects.

Four widely used measures for distance between clusters are as follows, where |p − p ' | is the distance between two objects or points, p and p ' ; mi is the mean for cluster, Ci ; and ni is the number of objects in Ci . They are also known as linkage measures.

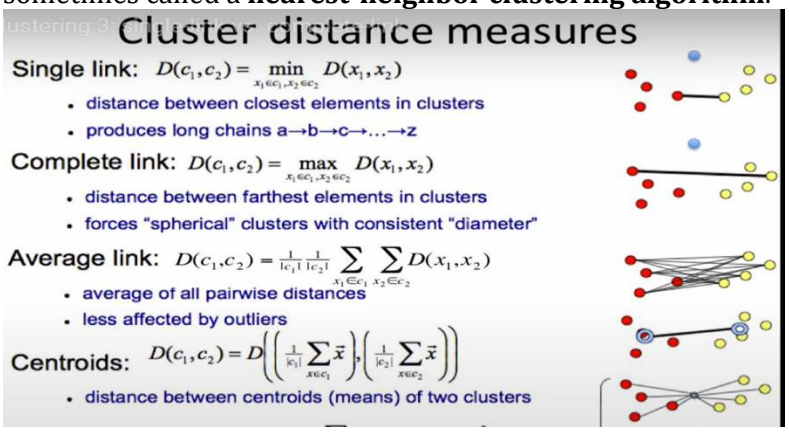
Minimum distance: $dist_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} \{|p - p'|\}$

Maximum distance: $dist_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} \{|p - p'|\}$

Mean distance: $dist_{mean}(C_i, C_j) = |m_i - m_j|$

Average distance: $dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, p' \in C_j} |p - p'|$

When an algorithm uses the minimum distance, dmin(Ci ,Cj), to measure the distance between clusters, it is sometimes called a **nearest-neighbor clustering algorithm**.



Example :

Single versus complete linkages. Let us apply hierarchical clustering to the data set of Figure 10.8(a). Figure 10.8(b) shows the dendrogram using single linkage. Figure 10.8(c) shows the case using complete linkage, where the edges between clusters {A,B,J,H} and {C,D,G,F,E} are omitted for ease of presentation. This example shows that by using single linkages we can find hierarchical clusters defined by local proximity, whereas complete linkage tends to find clusters opting for global closeness.

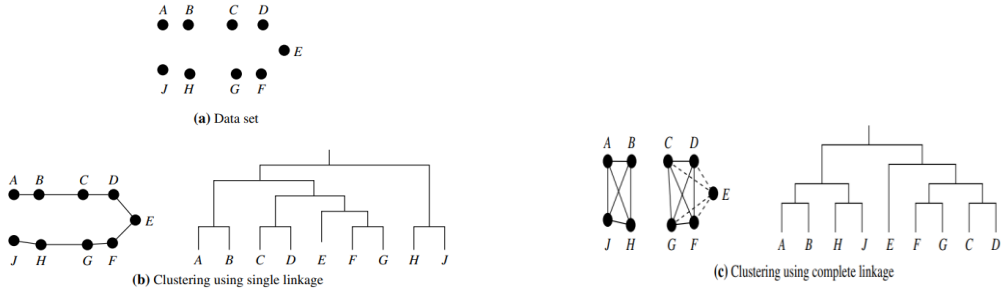


Figure 10.8 Hierarchical clustering using single and complete linkages.

BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees:

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is designed for clustering a large amount of numeric data by integrating hierarchical clustering (at the initial micro clustering stage) and other clustering methods such as iterative partitioning (at the later macro clustering stage).

It overcomes the two difficulties in agglomerative clustering methods:

(1) scalability and

(2) the inability to undo what was done in the previous step.

BIRCH uses the notions of

- clustering feature to summarize a cluster, and
- clustering feature tree (CF-tree) to represent a cluster hierarchy.

Consider a cluster of n d-dimensional data objects or points.

The **clustering feature (CF)** of the cluster is a 3-D vector summarizing information about clusters of objects. It is defined as

$$CF = \langle n, LS, SS \rangle,$$

where LS is the linear sum of the n points (i.e., $\sum_{i=1}^{n} \boldsymbol{x}_i$), and SS is the square sum of the data points (i.e., $\sum_{i=1}^{n} \boldsymbol{x}_i^2$).

A clustering feature is essentially a summary of the statistics for the given cluster. Using a clustering feature, we can easily derive many useful statistics of a cluster. For example, the cluster's centroid, x0, radius, R, and diameter, D, are

$$\boldsymbol{x_0} = \frac{\sum_{i=1}^{n} \boldsymbol{x_i}}{n} = \frac{LS}{n}, \qquad R = \sqrt{\frac{\sum_{i=1}^{n} (\boldsymbol{x_i} - \boldsymbol{x_0})^2}{n}} = \sqrt{\frac{nSS - 2LS^2 + nLS}{n^2}}, \qquad D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} (\boldsymbol{x_i} - \boldsymbol{x_j})^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}.$$

Here, 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.

Summarizing a cluster using the clustering feature can avoid storing the detailed information about individual objects or points. Instead, we only need a constant size of space to store the clustering feature. This is the key to BIRCH efficiency in space. Moreover, clustering features are additive. That is, for two disjoint clusters, C1 and C2, with the clustering features CF1 = <n1,LS1,SS1> and CF2 = <n2,LS2,SS2>, respectively, the clustering feature for the cluster that formed by merging C1 and C2 is simply

$$CF_1 + CF_2 = \langle n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2 \rangle.$$

Example: Clustering feature. Suppose there are three points, (2,5),(3,2), and (4,3), in a cluster, C1.

$$CF_1 = \langle 3, (2+3+4, 5+2+3), (2^2+3^2+4^2, 5^2+2^2+3^2) \rangle = \langle 3, (9,10), (29,38) \rangle.$$

Suppose that C1 is disjoint to a second cluster, C2, where CF2 = h3,(35,36),(417,440)i. The clustering feature of a new cluster, C3, that is formed by merging C1 and C2, is derived by adding CF1 and CF2. That is,

$$CF_3 = \langle 3+3, (9+35, 10+36), (29+417, 38+440) \rangle = \langle 6, (44,46), (446,478) \rangle.$$

A CF-tree is a height-balanced tree that stores the clustering features for a hierarchical clustering.

An example is shown in Figure. 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.

Parameters of BIRCH Algorithm :

- **threshold** : threshold is the maximum number of data points a sub-cluster in the leaf node of the CF tree can hold.
- **branching_factor** : This parameter specifies the maximum number of CF sub-clusters in each node (internal node).

- **n_clusters** : The number of clusters to be returned after the entire BIRCH algorithm is complete i.e., number of clusters after the final clustering step. If set to None, the final clustering step is not performed and intermediate clusters are returned.

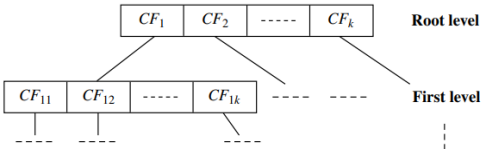
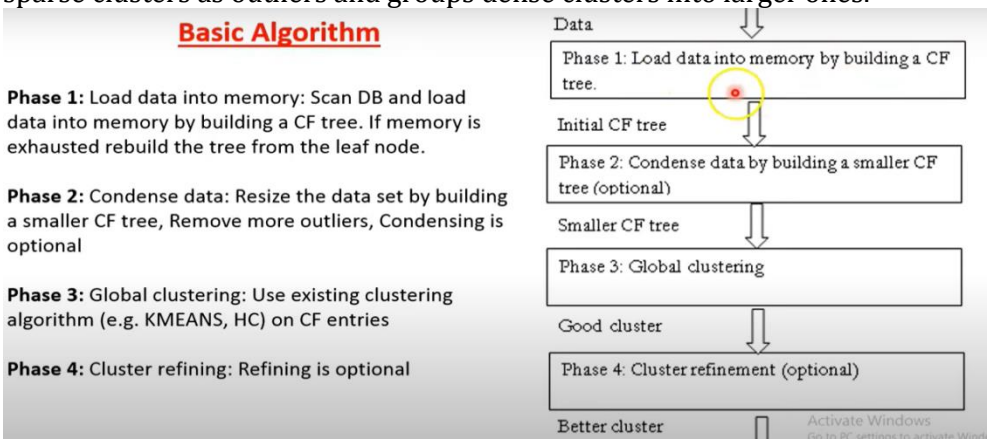


Fig: CF-tree structure.

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 data's inherent clustering structure.

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.



Chameleon: Multiphase Hierarchical Clustering Using Dynamic Modeling:

Chameleon is a hierarchical clustering algorithm that uses dynamic modeling to determine the similarity between pairs of clusters.

In Chameleon, cluster similarity is assessed based on

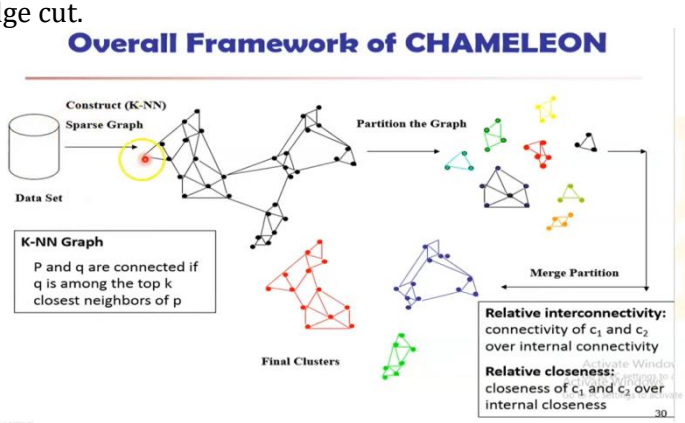
- how well connected objects are within a cluster and
- the proximity of clusters.

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 data types as long as a similarity function can be specified.

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 to 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 such that it minimizes the edge cut.

- Measures the similarity based on a dynamic model
  - Two clusters are merged only if the *interconnectivity* and *closeness (proximity)* between two clusters are high *relative to* the internal interconnectivity of the clusters and closeness of items within the clusters
- Graph-based, and a two-phase algorithm
  1. Use a graph-partitioning algorithm: cluster objects into a large number of relatively small sub-clusters
  2. Use an agglomerative hierarchical clustering algorithm: find the genuine clusters by repeatedly combining these sub-clusters



Chameleon: hierarchical clustering based on k-nearest neighbors and dynamic modeling.

Chameleon then uses an agglomerative hierarchical clustering algorithm that iteratively merges subclusters based on their similarity.

To determine the pairs of most similar subclusters, it takes into account both the interconnectivity and the

closeness of the clusters.
Specifically, Chameleon determines the similarity between each pair of clusters Ci and Cj according to their relative interconnectivity, RI(Ci ,Cj), and their relative closeness, RC(Ci ,Cj).
The relative interconnectivity, RI(Ci ,Cj), between two clusters, Ci and Cj , is defined as the absolute interconnectivity between Ci and Cj , normalized with respect to the internal interconnectivity of the two clusters, Ci and Cj . That is,

$$RI(C_i, C_j) = \frac{|EC_{\{C_i, C_j\}}|}{\frac{1}{2}(|EC_{C_i}| + |EC_{C_j}|)},$$

where EC{Ci ,Cj} is the edge cut as previously defined for a cluster containing both Ci and Cj . Similarly, ECCi (or ECCj) is the minimum sum of the cut edges that partition Ci (or Cj) into two roughly equal parts.
The relative closeness, RC(Ci ,Cj), between a pair of clusters, Ci and Cj , is the absolute closeness between Ci and Cj , normalized with respect to the internal closeness of the two clusters, Ci and Cj . It is defined as

$$RC(C_i, C_j) = \frac{\bar{S}_{EC_{\{C_i, C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{EC_{C_j}}},$$

where SEC{Ci ,Cj } is the average weight of the edges that connect vertices in Ci to vertices in Cj , and SECCi (or SECCj) is the average weight of the edges that belong to the mincut bisector of cluster Ci (or Cj).
Probabilistic Hierarchical Clustering:
Probabilistic hierarchical clustering aims to overcome some of these disadvantages by using probabilistic models to measure distances between clusters.
A probabilistic hierarchical clustering method can adopt the agglomerative clustering framework, but use probabilistic models to measure the distance between clusters
A probabilistic model is an unsupervised technique that helps us solve density estimation or "soft" clustering problems.
In probabilistic clustering, data points are clustered based on the likelihood that they belong to a particular distribution.
The Gaussian Mixture Model (GMM) is the one of the most commonly used probabilistic clustering methods.
Generative models are statistical models capable of generating new datasets based on the probability distribution, which is then estimated, thereby generating a distribution similar to the original one.
For example, when we conduct clustering analysis on a set of marketing surveys, we assume that the surveys collected are a sample of the opinions of all possible customers. Here, the data generation mechanism is a probability distribution of opinions with respect to different customers, which cannot be obtained directly and completely. The task of clustering is to estimate the generative model as accurately as possible using the observed data objects to be clustered.
In practice, we can assume that the data generative models adopt common distribution functions, such as Gaussian distribution or Bernoulli distribution, which are governed by parameters. The task of learning a generative model is then reduced to finding the parameter values for which the model best fits the observed data set.

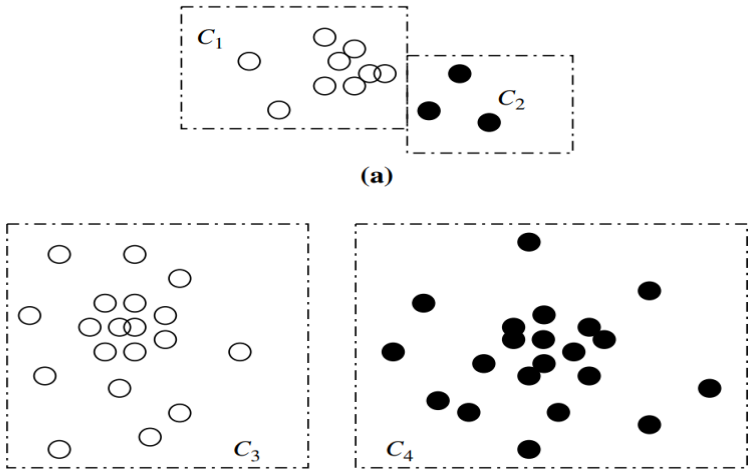


Fig: Merging clusters in probabilistic hierarchical clustering: (a) Merging clusters C1 and C2 leads to an increase in overall cluster quality, but merging clusters (b) C3 and (c) C4 does not.

Algorithm: A probabilistic hierarchical clustering algorithm.

Input:

- $D = \{o_1, \ldots, o_n\}$: a data set containing n objects;

Output: A hierarchy of clusters.

Method:

(1) **create** a cluster for each object $C_i = \{o_i\}$, $1 \le i \le n$;
(2) **for** $i = 1$ to n
(3) **find** pair of clusters C_i and C_j such that $C_i, C_j = \arg\max_{i \neq j} \log \frac{P(C_i \cup C_j)}{P(C_i)P(C_j)}$;
(4) **if** $\log \frac{P(C_i \cup C_j)}{P(C_i)P(C_j)} > 0$ then merge C_i and C_j;
(5) **else** stop;

A probabilistic hierarchical clustering algorithm.

Generative model. Suppose we are given a set of 1-D points $X = \{x_1, \ldots, x_n\}$ for clustering analysis. Let us assume that the data points are generated by a Gaussian distribution,

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \qquad (10.14)$$

where the parameters are μ (the mean) and σ^2 (the variance).

The probability that a point $x_i \in X$ is then generated by the model is

The task of learning the generative model is to find the parameters μ and σ^2 such that the likelihood $L(\mathcal{N}(\mu, \sigma^2) : X)$ is maximized, that is, finding

$$\mathcal{N}(\mu_0, \sigma_0^2) = \arg\max\{L(\mathcal{N}(\mu, \sigma^2) : X)\}, \qquad (10.17$$

where $\max\{L(\mathcal{N}(\mu, \sigma^2) : X)\}$ is called the *maximum likelihood.*

A **drawback** of using probabilistic hierarchical clustering is that it outputs only one hierarchy with respect to a chosen probabilistic model. It cannot handle the uncertainty of cluster hierarchies.

Evaluation of Clustering:

Clustering evaluation assesses the feasibility of clustering analysis on a data set and the quality of the results generated by a clustering method. The tasks include assessing clustering tendency, determining the number of clusters, and measuring clustering quality.

The major tasks of clustering evaluation include the following:

Assessing clustering tendency: Before applying any clustering method on your data, it's important to evaluate whether the data sets contains meaningful clusters (i.e.: non-random structures) or not. If yes, then how many clusters are there. This process is defined as the assessing of **clustering tendency** or the feasibility of the clustering analysis.

Determining the number of clusters in a data set: Determining the **optimal number of clusters** in a data set is a fundamental issue in partitioning clustering, such as k-means clustering, which requires the user to specify the number of clusters k to be generated.

Measuring clustering quality: After applying a clustering method on a data set, we want to assess how good the resulting clusters are. A number of measures can be used. Some methods measure how well the clusters fit the data set, while others measure how well the clusters match the ground truth, if such truth is available. There are also measures that score clusterings and thus can compare two sets of clustering results on the same data set.

Assessing Clustering Tendency:

"How can we assess the clustering tendency of a data set?"

we can try to measure the probability that the data set is generated by a uniform data distribution. This can be achieved using statistical tests for spatial randomness. To illustrate this idea, let's look at a simple yet effective

statistic called the Hopkins Statistic.

Hopkins Statistic:

The Hopkins Statistic is a spatial statistic that tests the spatial randomness of a variable as distributed in a space. Given a data set, D, which is regarded as a sample a random variable, o, we want to determine how far away o is from being uniformly distributed in the data space.

We calculate the Hopkins Statistic as follows:

1. Sample n points, p1, . . . , pn, uniformly from D. That is, each point in D has the same probability of being included in this sample. For each point, pi , we find the nearest neighbor of pi (1 ≤ i ≤ n) in D, and let xi be the distance between pi and its nearest neighbor in D. That is,

$$x_i = \min_{\boldsymbol{v} \in D} \{dist(\boldsymbol{p_i}, \boldsymbol{v})\}.$$

2. Sample n points, q1, . . . , qn, uniformly from D. For each qi (1 ≤ i ≤ n), we find the nearest neighbor of qi in D – {qi}, and let yi be the distance between qi and its nearest neighbor in D– {qi}. That is,

$$y_i = \min_{\boldsymbol{v} \in D, \boldsymbol{v} \neq \boldsymbol{q_i}} \{dist(\boldsymbol{q_i}, \boldsymbol{v})\}.$$

3. Calculate the Hopkins Statistic, H, as

$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}.$$

"What does the Hopkins Statistic tell us about how likely data set D *follows a uniform distribution in the data space?"* If D were uniformly distributed, then $\sum_{i=1}^{n} y_i$ and $\sum_{i=1}^{n} x_i$ would be close to each other, and thus H would be about 0.5. However, if D were highly skewed, then $\sum_{i=1}^{n} y_i$ would be substantially smaller than $\sum_{i=1}^{n} x_i$ in expectation, and thus H would be close to 0.

Determining the Number of Clusters:

In Clustering algorithms like K-Means clustering, we have to determine the right number of clusters for our dataset. This ensures that the data is properly and efficiently divided. An appropriate value of 'k' i.e. the number of clusters helps in ensuring proper granularity of clusters and helps in maintaining a good balance between compressibility and accuracy of clusters.

A simple method is to set the number of clusters to about $\sqrt{\frac{n}{2}}$ for a data set of n points. In expectation, each cluster has $\sqrt{2n}$ points.

Elbow Method

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$WCSS = \sum_{P_i \text{ in Cluster1}} distance(P_i\ C_1)^2 + \sum_{P_i \text{ in Cluster2}} distance(P_i\ C_2)^2 + \sum_{P_i \text{ in CLuster3}} distance(P_i\ C_3)^2$$

In the above formula of WCSS,

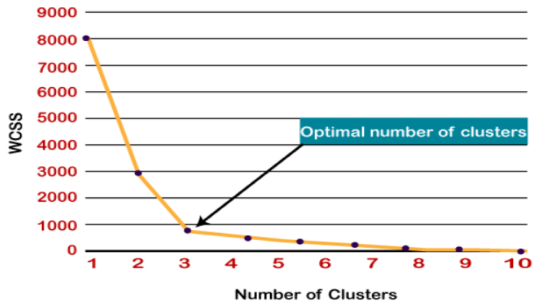
$\sum_{Pi \text{ in Cluster1}} distance(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



Measuring Clustering Quality:

There are two types of evaluation metrics for clustering,

- **Extrinsic Methods**: These measures require ground truth labels, which may not be available in practice
- **Intrinsic Methods**: These measures do not require ground truth labels (applicable to all unsupervised learning results)

Extrinsic Methods :

In general, a measure Q on clustering quality is effective if it satisfies the following four essential criteria:

- Cluster homogeneity
- Cluster completeness.
- Rag bag.
- Small cluster preservation.

Cluster Homogeneity: This requires that the more pure the clusters in a clustering are, the better the clustering. Suppose that ground truth says that the objects in a data set, D, can belong to categories L1,...,Ln. Consider clustering, C1, wherein a cluster C ∈ C1 contains objects from two categories Li ,Lj (1 ≤ i < j ≤ n). Also consider clustering C2, which is identical to C1 except that C2 is split into two clusters containing the objects in Li and Lj , respectively. A clustering quality measure, Q, respecting cluster homogeneity should give a higher score to C2 than C1, that is, Q(C2,Cg) > Q(C1,Cg).

Cluster completeness: Cluster completeness is the essential parameter for good clustering, if any two data objects are having similar characteristics then they are assigned to the same category of the cluster according to ground truth. Cluster completeness is high if the objects are of the same category.

Rag bag: In some situations, there can be a few categories in which the objects of those categories cannot be merged with other objects. Then the quality of those cluster categories is measured by the Rag Bag method. According to the rag bag method, we should put the heterogeneous object into a rag bag category.

Small cluster preservation: If a small category of clustering is further split into small pieces, then those small pieces of cluster become noise to the entire clustering and thus it becomes difficult to identify that small category from the clustering. The small cluster preservation criterion states that are splitting a small category into pieces is not advisable and it further decreases the quality of clusters as the pieces of clusters are distinctive.

Many clustering quality measures satisfy some of these four criteria. Here, we introduce the BCubed precision and recall metrics, which satisfy all four criteria.

BCubed evaluates the precision and recall for every object in a clustering on a given data set according to ground truth.

The **precision** of an object indicates how many other objects in the same cluster belong to the same category as the object.

The **recall** of an object reflects how many objects of the same category are assigned to the same cluster.

Then, for two objects, oi and oj , (1 ≤ i,j,≤ n,i 6= j), the correctness of the relation between oi and oj in clustering C is given by

$$\text{Correctness}(o_i, o_j) = \begin{cases} 1 & \text{if } L(o_i) = L(o_j) \Leftrightarrow C(o_i) = C(o_j) \\ 0 & \text{otherwise.} \end{cases}$$

BCubed precision and recall is defined as

$$\text{Precision BCubed} = \frac{\sum_{i=1}^{n} \frac{\sum_{o_j: i \neq j, C(o_i)=C(o_j)} \text{Correctness}(o_i, o_j)}{\|\{o_j | i \neq j, C(o_i) = C(o_j)\}\|}}{n}.$$

$$\text{Recall BCubed} = \frac{\sum_{i=1}^{n} \frac{\sum_{o_j: i \neq j, L(o_i)=L(o_j)} \text{Correctness}(o_i, o_j)}{\|\{o_j | i \neq j, L(o_i) = L(o_j)\}\|}}{n}.$$

Intrinsic Methods:

When the ground truth of a data set is not available, we have to use an intrinsic method to assess the clustering quality

Silhouette Coefficient:

Silhouette refers to a method of interpretation and validation of consistency within clusters of data.

For a data set, D, of n objects, suppose D is partitioned into k clusters, C1,...,Ck . For each object o ∈ D, we calculate a(o) as the average distance between o and all other objects in the cluster to which o belongs. Similarly, b(o) is the minimum average distance from o to all clusters to which o does not belong. Formally, suppose o ∈ Ci (1 ≤ i ≤ k); then

$$a(o) = \frac{\sum_{o' \in C_i, o \neq o'} dist(o, o')}{|C_i| - 1} \qquad b(o) = \min_{C_j : 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{o' \in C_j} dist(o, o')}{|C_j|} \right\}$$

The silhouette coefficient of o is then defined as

$$s(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}}.$$

The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

UNIT WISE IMPORTANT QUESTIONS:

1. Briefly describe and give examples of each of the following approaches to clustering: partitioning methods and hierarchical methods.
2. Suppose that the data mining task is to cluster points (with (x, y) representing location) into three clusters, where the points are A1(2,10),A2(2,5),A3(8,4),B1(5,8),B2(7,5),B3(6,4),C1(1,2),C2(4,9). The distance function is Euclidean distance. Suppose initially we assign A1, B1, and C1 as the center of each cluster, respectively. Use the k-means algorithm to show only
   (a) The three cluster centers after the first round of execution.
   (b) The final three clusters
3. Provide the pseudocode of the object reassignment step of the PAM algorithm.
4. Show that BCubed metrics satisfy the four essential requirements for extrinsic clustering evaluation methods.
5. Compare k-means with k- medoids algorithms for clustering.
6. Discuss various evaluation measures used to evaluate clustering algorithms
7. Discuss the similarity measures and distance measures frequently used in clustering the data.
8. Discuss about key issues in Hierarchical clustering.
9. What is the main objective of clustering? Give the categorization of clustering approaches. Briefly discuss them.
10. What are the requirements of clustering in data mining? Explain
11. Explain about types of Partitioning clustering algorithm
12. With the help of example explain K-Means Partitioning clustering algorithm
13. With the help of example explain K-Medoid Partitioning clustering algorithm
14. Explain about the basic Agglomerative Hierarchical clustering algorithm.
15. Explain about the Divisive Hierarchical clustering algorithm.
16. Discuss about BIRCH clustering algorithm.
17. Discuss about Chameleon clustering algorithm