Data Mining

UNIT-VI

Cluster Analysis

B.Tech(CSE)-VI SEM

Course Outcomes

After Successful completion of the Course, the student will be able to:

CO1: Explain the concept of Data Mining and its functionalities (K2)

CO2: Discuss various Data Preprocessing Techniques (K2)

CO3: Demonstrate Association Analysis Techniques (K3)

CO4: Illustrate varions Classification Techniques (K3)

CO5: Demonstrate Alternative techniques for Classification (K3)

CO6: Use different Clustering techniques to cluster data (K3)

UNIT VI: Cluster Analysis: Contents

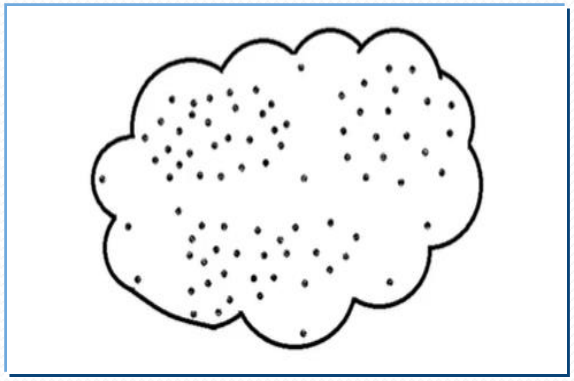
- **Cluster Analysis: Basic Concepts and Methods**
  - What is Cluster Analysis?
  - Requirements for Cluster Analysis
  - Overview of Basic Clustering Methods
- **Partitioning Methods**
  - k-Means: A Centroid-Based Technique
  - k-Medoids: A Representative Object-Based Technique
- **Hierarchical Methods**
  - 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
- **Density Based Methods**
  - DBSCAN: Density-Based Clustering Based on Connected Regions with High Density

Intended Learning Outcome:

- To illustrate basic concepts of Clustering

Cluster: (Cluster Analysis – Unsupervised learning)

- Clustering is the process of grouping a set of data objects into multiple groups or clusters so that objects within a cluster have high similarity , but are very dissimilar to objects in other clusters.



- Dissimilarities (or unrelated) and similarities (or related) are assessed based on the attribute values describing the objects and often involve distance measures.

- Clustering is known as **unsupervised learning** because the class label information is not present.

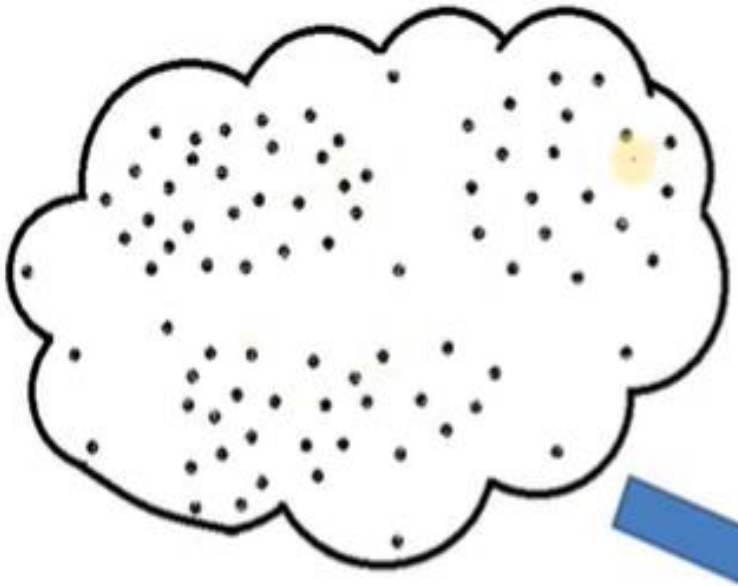
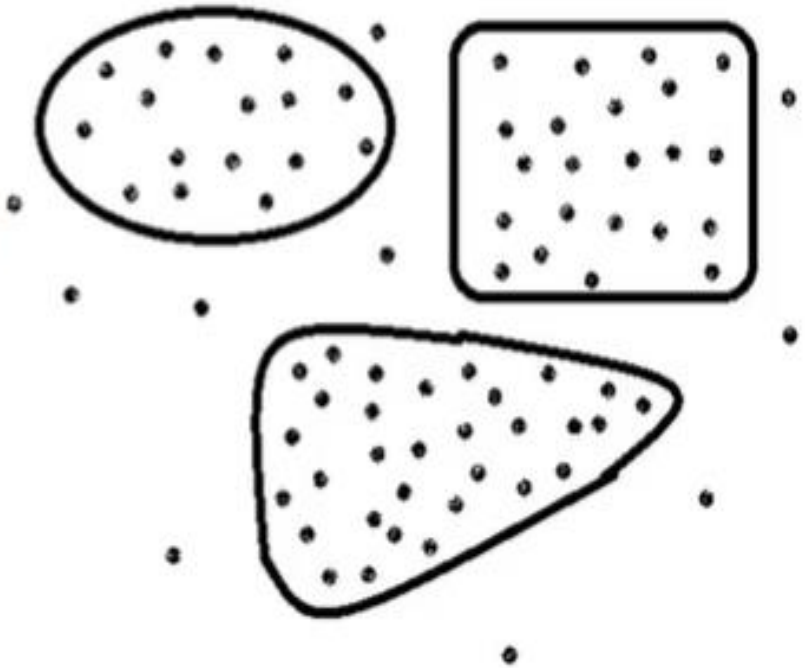
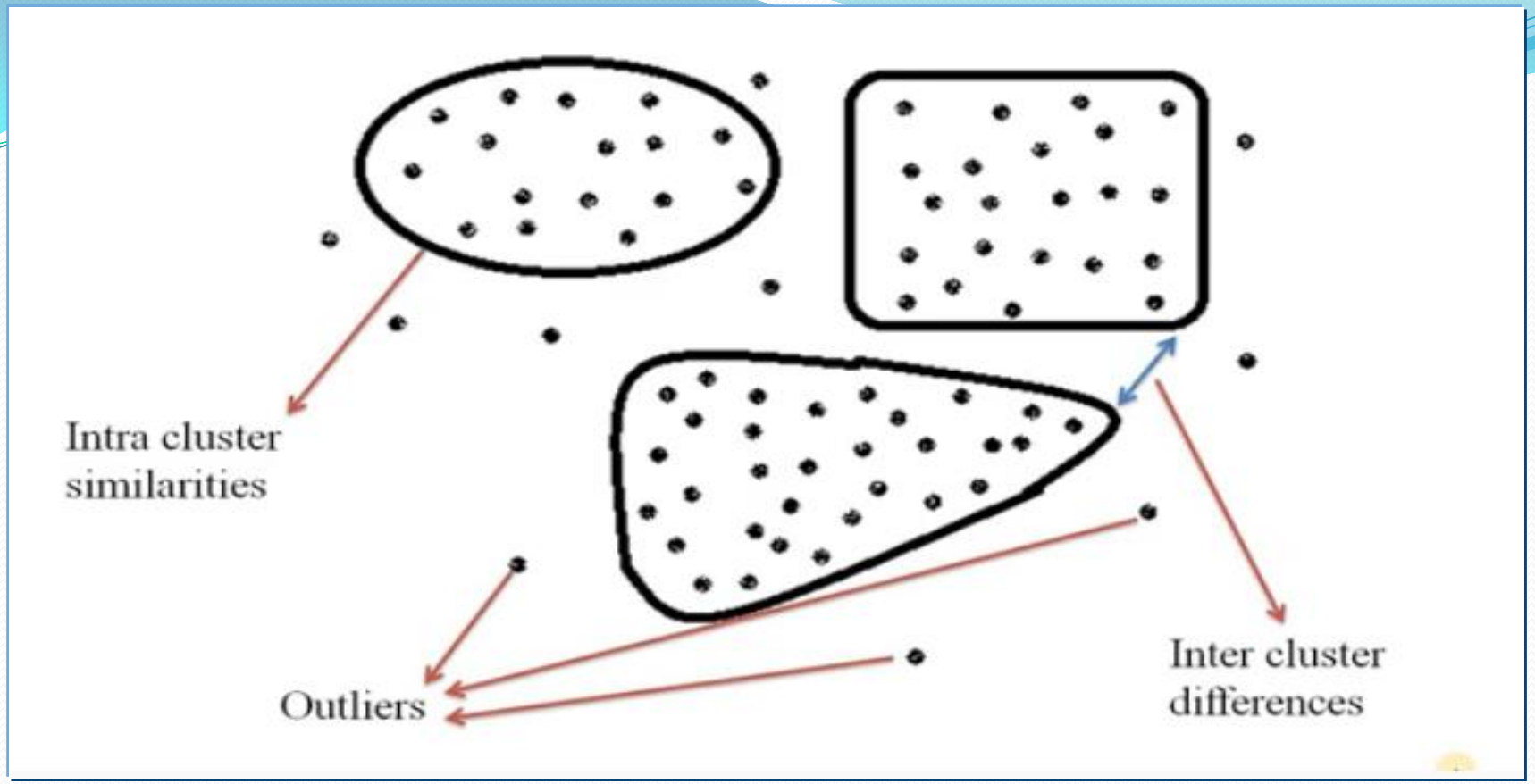


Fig: Group of Data points

Fig: Clusters based on similarities





-The **inter-class** cluster show the distance between data point with cluster center.

-The intra-class cluster show the distance between the data point of one cluster with the other data point in other cluster.

-Outliers are extreme values that fall a long way outside of the other observations.

Cluster Analysis: Applications

- **Biology:** taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- **Information retrieval:** document clustering
- **Land use:** Identification of areas of similar land use in an earth observation database
- **Marketing:** Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **City-planning:** Identifying groups of houses according to their house type, value, and geographical location
- **Earth-quake studies:** Observed earth quake epicenters should be clustered along continent faults
- **Climate:** understanding earth climate, find patterns of atmospheric and ocean
- **Economic Science:** market research

Clustering as a Preprocessing Tool (Utility)

- **Summarization:**
  - Preprocessing for regression, PCA, classification, and association analysis
- **Compression:**
  - Image processing: vector quantization
- **Finding K-nearest Neighbors:**
  - Localizing search to one or a small number of clusters.
- **Outlier detection:**
  - Outliers are often viewed as those “far away” from any cluster

Quality: What Is Good Clustering?

- A **good clustering** method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters

- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

Requirements for Cluster Analysis

The following are typical requirements of clustering in data mining:

- **Scalability**
  - Clustering all the data instead of only on samples
  - Therefore, We need highly scalable clustering algorithms to deal with large databases.

- **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.
  - More applications need clustering techniques for complex data types such as graphs, sequences, images, and documents.

Requirements for Cluster Analysis (Contd…)

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

- **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.
  - Requiring the specification of domain knowledge not only burdens users, but also makes the quality of clustering difficult to control.

Requirements for Cluster Analysis (Contd…)

- **Ability to deal with noisy data**
  - Most real-world data sets contain outliers and/or missing, unknown, or erroneous data.
  - 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
  - Clustering algorithms may also be sensitive to the input data order
  - Incremental clustering algorithms and algorithms that are insensitive to the input order are needed

Requirements for Cluster Analysis (Contd…)

- **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.
  - Finding clusters of data objects in a high-dimensional 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.
  - 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.

Considerations for Cluster Analysis

Clustering methods can be compared using the following aspects:

- **Partitioning criteria**
  - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable)
- **Separation of clusters**
  - Exclusive (e.g., one customer belongs to only one region) vs. non-exclusive (e.g., one document may belong to more than one class)
- **Similarity measure**
  - Distance-based (e.g., Euclidian, road network, vector) vs. connectivity-based (e.g., density or contiguity)
- **Clustering space**
  - Full space (often when low dimensional) vs. subspaces (often in high-dimensional clustering)

Overview of Major Clustering Approaches

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)

Intended Learning Outcome:

- To illustrate Partitioning Methods in Clustering

Partitioning Methods

- The simplest and most fundamental version of cluster analysis is partitioning, which organizes the objects of a set into several exclusive groups or clusters.

- 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 ≤ 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.

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, $C_1,\ldots,C_k$, that is, $C_i \subset D$ and $C_i \cap C_j = \phi$ 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.
- The objective function aims for high intra-cluster similarity and low inter-cluster 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 medoid of the objects (or points) assigned to the cluster.
- The difference between an object $\mathbf{p} \in C_i$ and $\mathbf{c_i}$, 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.**

k-Means: A Centroid-Based Technique

- The quality of cluster C_i can be measured by the **within-cluster variation**, which is the sum of squared error between all objects in C_i and the centroid $\mathbf{c_i}$, defined as

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

- 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 c_i is the centroid of cluster C_i (both p and c_i are multidimensional).

- This objective function tries to make the resulting k clusters as compact and as separate as possible

k-Means: A Centroid-Based Technique

- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when the assignment does not change

k-Means: A Centroid-Based Technique

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.

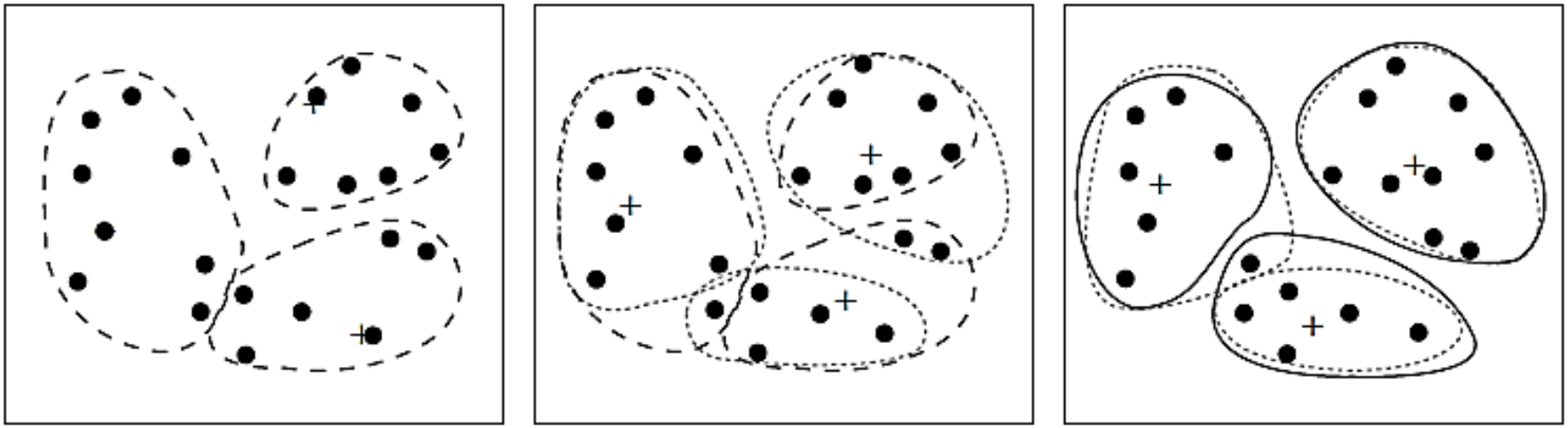
Output: A set of *k* clusters.

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;

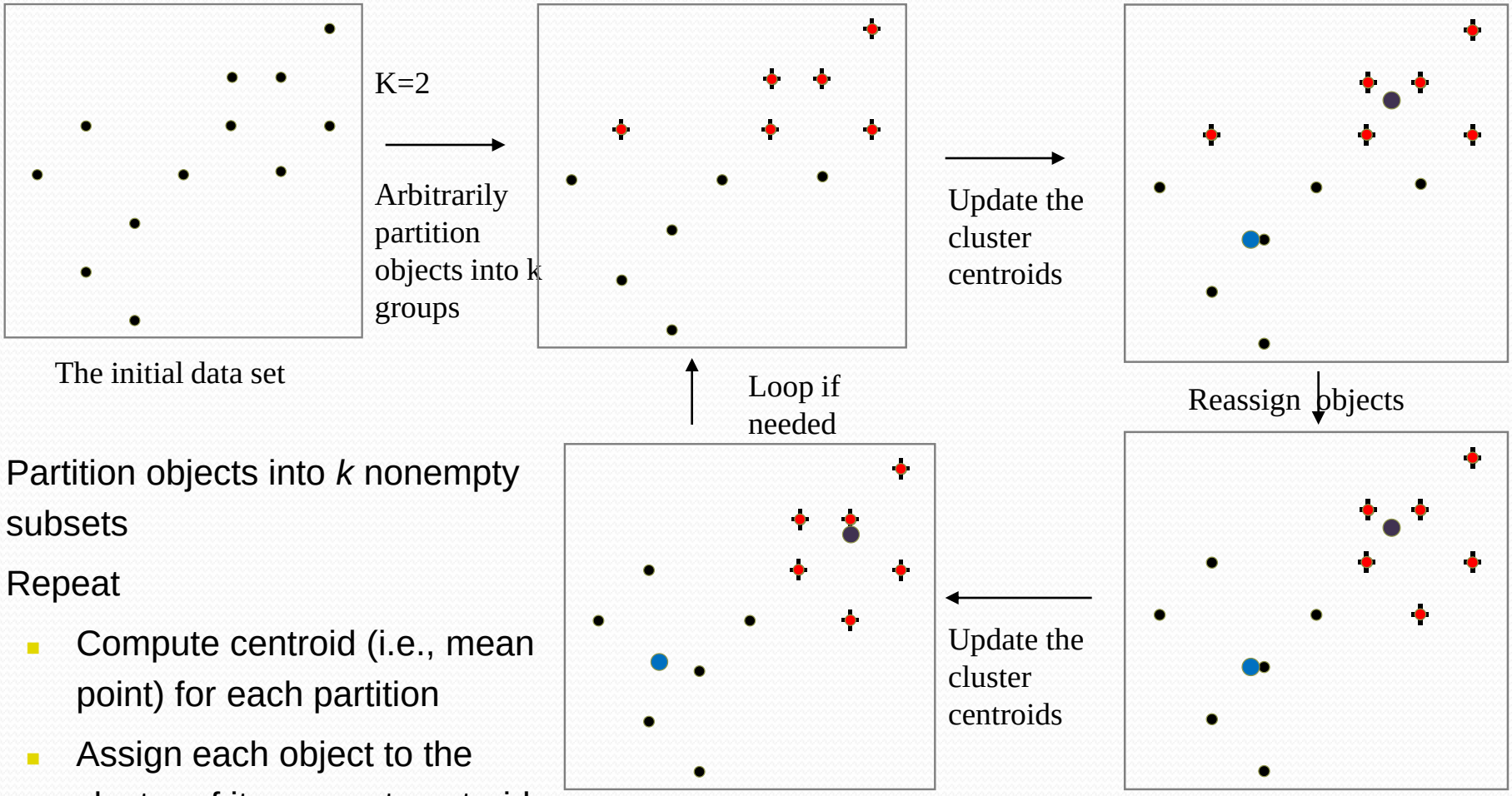
The *k*-means partitioning algorithm.

k-Means: A Centroid-Based Technique



Clustering of a set of objects using the *k*-means method; for (b) update cluster centers and reassign objects accordingly (the mean of each cluster is marked by a +)

An Example of K-Means Clustering



- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

K-Means Clustering Algorithm (contd…)

Step1: Take the mean value

Step2: Find the nearest no. of Mean and put in cluster.

Step3: Repeat 1 & 2 steps until we get the same mean.

Example:

Data pts, k={2, 4, 6, 9, 12, 16, 20, 24, 26}

No. of Clusters=2 {4, 12}

k1={2,4,6}

=2+4+6/3

=4

k2={9, 12, 16, 20, 24, 26}

=9+12+16+20+24+26/6

=18

k1={2,4,6,9}

=21/4=5.25=5

k2={12, 16, 20, 24, 26}

=19.6=20

k1={2,4,6,9,12}

=6.6=7

k2={ 16, 20, 24, 26}

=21.5=22

k1={2,4,6,9,12}

=6.6=7

k2={ 16, 20, 24, 26}

=21.5=22

Comments on the *K-Means* Method

- **Strength:** *Efficient*: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, $t << n$.

- **Comment:** Often terminates at a *local optimal*.

- **Weakness**
  - Need to specify k, the *number* of clusters, in advance (there are ways to automatically determine the best k)
  - Sensitive to noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

k-Medoids: A Representative Object-Based Technique

A drawback of k-means: Consider six points in 1-D space having the values 1, 2, 3, 8, 9, 10, and 25, respectively. Intuitively, by visual inspection we may imagine the points partitioned into the clusters {1, 2,3} and {8, 9,10}, where point 25 is excluded because it appears to be an outlier.

- How would k-means partition the values? If we apply k-means using k =2 and Eq. (1), the partitioning {1,2,3}, {8,9,10,25} has the within-cluster variation

- Gi $(1-2)^2+(2-2)^2+(3-2)^2+(8-13)^2+(9-13)^2+(10-13)^2+(25-13)^2=196,$ 13.
- Comparing this to the partitioning {1, 2, 3,8}, {9, 10,25}, for which k-means computes the within cluster variation as

$$(1-3.5)^2+(2-3.5)^2+(3-3.5)^2+(8-3.5)^2+(9-14.67)^2 + (10-14.67)^2+(25-14.67)^2=189.67,$$

k-Medoids: A Representative Object-Based Technique

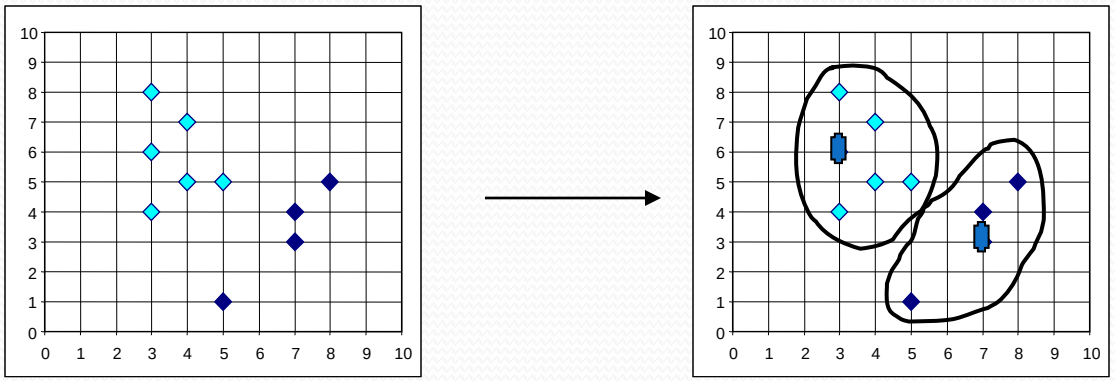
- Given that 3.5 is the mean of cluster {1, 2, 3,8} and 14.67 is the mean of cluster {9, 10,25}.
- The latter partitioning has the lowest within-cluster variation; therefore, the k-means method assigns the value 8 to a cluster different from that containing 9 and 10 due to the outlier point 25.
- Moreover, the center of the second cluster, 14.67, is substantially far from all the members in the cluster
- How can we modify the k-means algorithm to diminish such sensitivity to outliers?

Intended Learning Outcome:

- To illustrate Partitioning Methods in Clustering – K mediods

k-Medoids: A Representative Object-Based Technique

- One of the data object acts as cluster center (one representative object per cluster) instead of taking the mean value of the objects in a cluster (as in k-means algorithm).
- We call this cluster representative as a cluster mediod or simply mediod.



k-Medoids: A Representative Object-Based Technique

- **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:

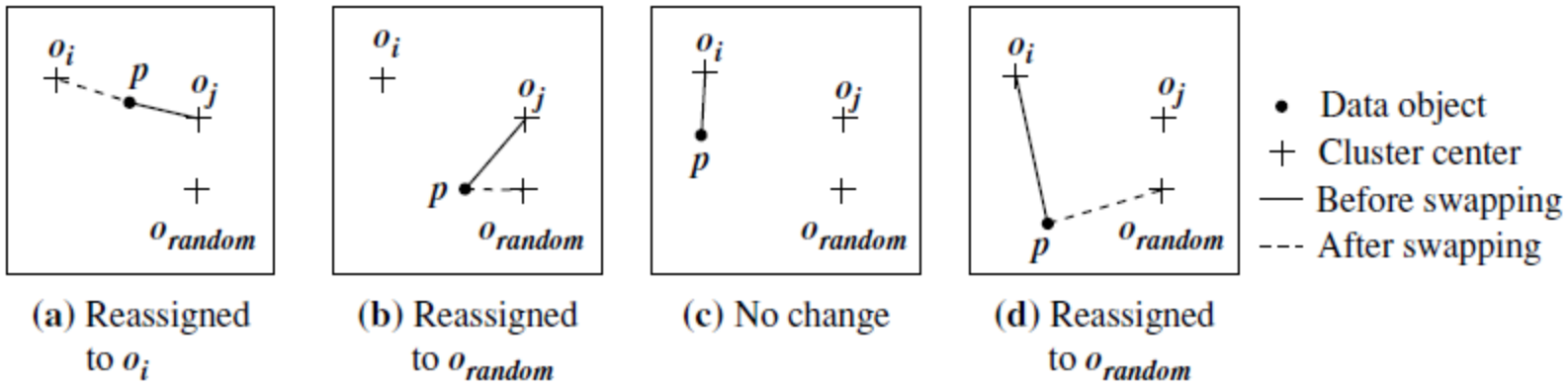
$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(\boldsymbol{p}, \boldsymbol{o}_i),$$

- where E is the sum of the absolute error for all objects **p** in the data set, and $\mathbf{o_i}$ is the representative object of C_i . This is the basis for the **k-medoids method**, which groups n objects into k clusters by minimizing the absolute error
- When k = 1, we can find the exact median in $O(n^2)$ time. However, when k is a general positive number, the k-medoid problem is NP-hard.

k-Medoids: A Representative Object-Based Technique

- **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.
- 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.

k-Medoids: A Representative Object-Based Technique



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

k-Medoids: A Representative Object-Based Technique

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

Input:

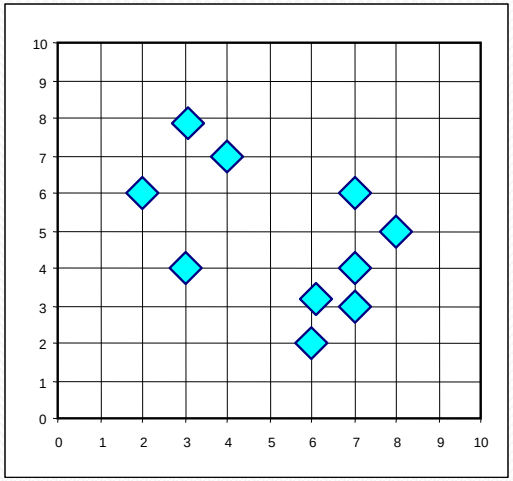
- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

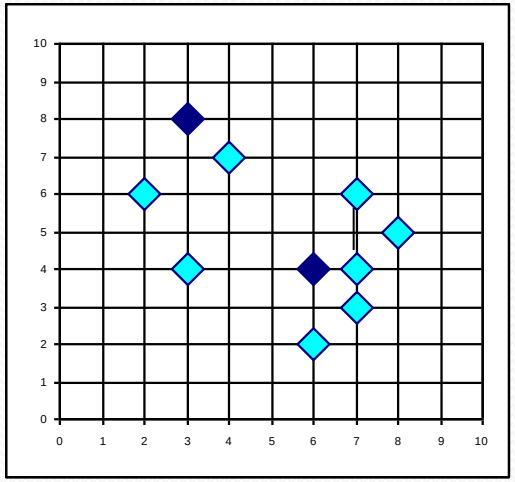
(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, *S*, of swapping representative 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;

PAM: A Typical K-Medoids Algorithm



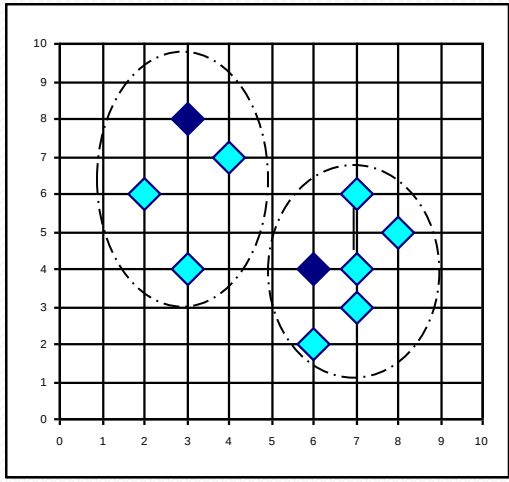
K=2

Arbitrary choose k object as initial medoids

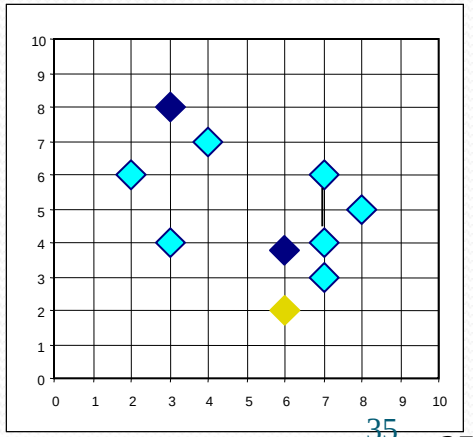


Assign each remaining object to nearest medoids

Total Cost = 20

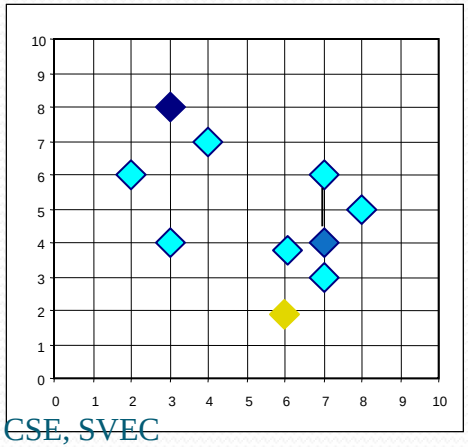


Randomly select a nonmedoid object, O_{random}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{random}

If quality is improved.

Do loop

Until no change

K-means vs k-mediods

1. ***Comparing k-means with k-mediods:***
   - Both algorithms need predefined value for k, the number of clusters , prior to the training algorithm.
   - Also both algorithms arbitrarily choose the initial cluster centriods.
   - The k-mediod method is more robust than k-means in the presence of outliers, because a mediod is less influenced by outliers than a mean.

2. Time Complexity of k-mean is O(tkn)

where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, $k, t << n$.

3. Time Complexity of PAM is $O(k(n-k)^2)$

Variations of k-Mediods

- Applicability of PAM:
  - PAM does not scale well to large database because of its computational complexity.
  - There are some variants of PAM that are designed mainly for large datasets are:
    - CLARA (Clustering LARge Applications) and
    - CLARANS (Clustering Large Applications based upon RANdomized Search.
  - CLARANS is an improvement of CLARA

k-Medoids: A Representative Object-Based Technique

- The effectiveness of CLARA depends on the sample size.
- PAM searches for the best k-medoids among a given data set, whereas CLARA searches for the best k-medoids among the selected sample of the data set.
- CLARA cannot find a good clustering if any of the best sampled medoids is far from the best k-medoids.
- If an object is one of the best k-medoids but is not selected during sampling, CLARA will never find the best clustering.

How might we improve the quality and scalability of CLARA?

- 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.
- 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.

k-Medoids: A Representative Object-Based Technique

CLARANS (Clustering Large Applications based upon RANdomized Search):

- 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.

Intended Learning Outcome:

- To illustrate Hierarchical Methods in Clustering

Hierarchical Methods

- A hierarchical clustering method works by **grouping data objects into a hierarchy or "tree"** of clusters . It can be visualized as a **dendogram.**
- **Types:**
  - Agglomerative versus Divisive Hierarchical Clustering
  - **BIRCH:** Multiphase Hierarchical Clustering Using Clustering Feature Trees
  - **Chameleon:** Multiphase Hierarchical Clustering Using Dynamic Modeling
  - Probabilistic Hierarchical Clustering

- 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 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.
- 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

Hierarchical Methods: Agglomerative versus Divisive Hierarchical Clustering

- An **agglomerative hierarchical clustering method** uses a bottom-up strategy.
- It typically 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.

Hierarchical Methods: Agglomerative versus Divisive Hierarchical Clustering

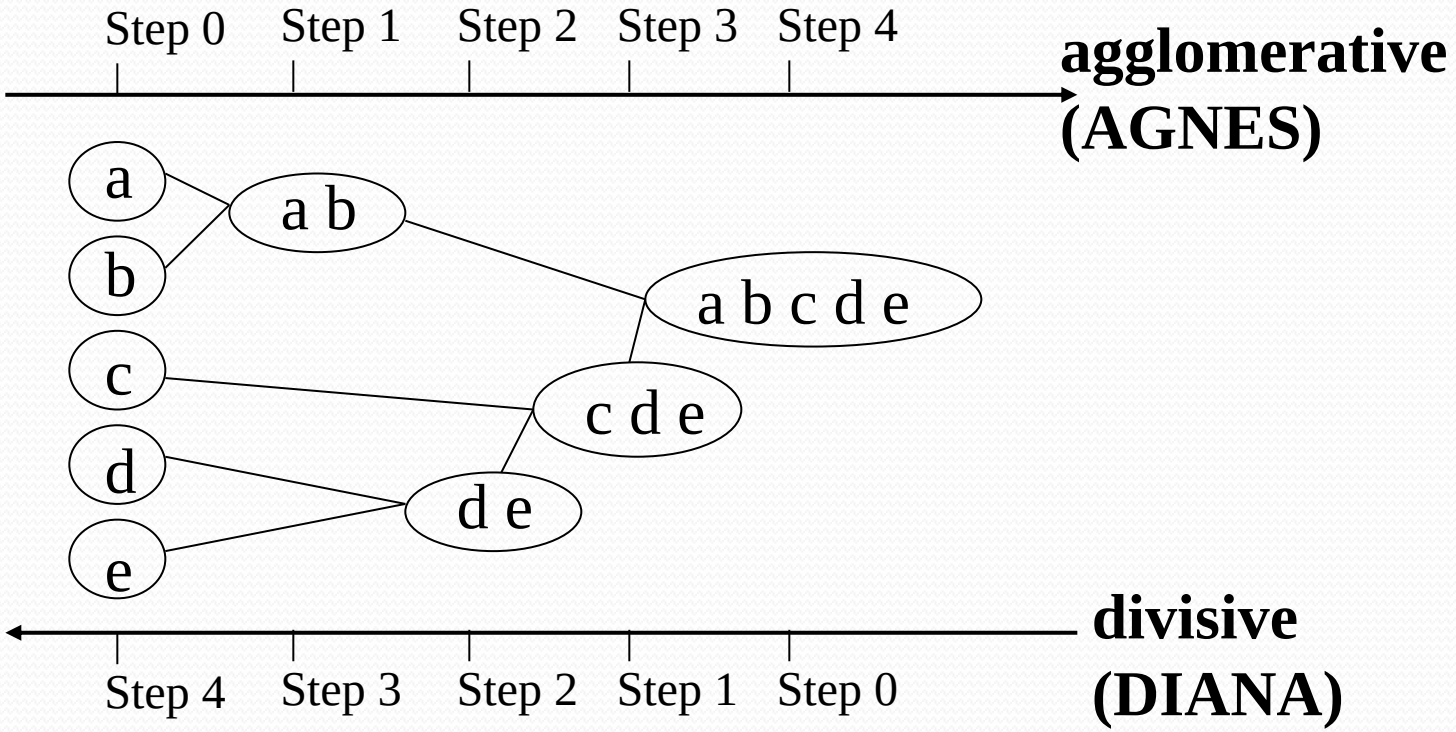
- A **divisive hierarchical clustering method** employs a top-down strategy.

- 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.

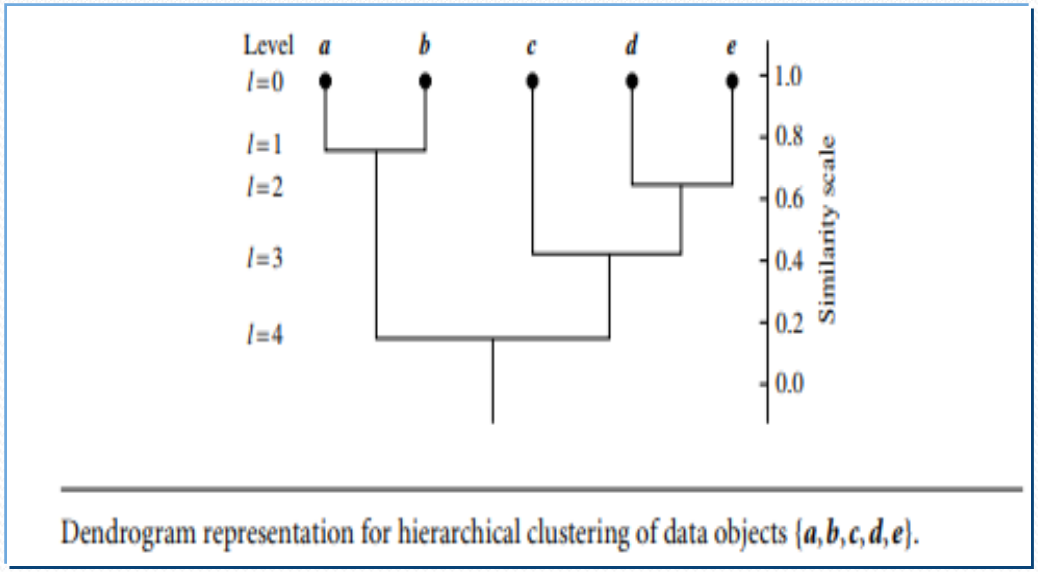
In either agglomerative or divisive hierarchical clustering, a user can specify the desired number of clusters as a termination condition.

Hierarchical Methods: Agglomerative versus Divisive Hierarchical Clustering



The above 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}.

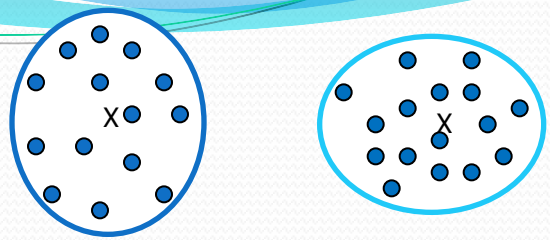
- A tree structure called a **dendrogram** is commonly used to represent the process of hierarchical clustering. It shows how objects are grouped together (in an agglomerative method) or partitioned (in a divisive method) step-by-step.
- Figure shows a dendrogram for the five objects , where l = o shows the five objects as singleton clusters at level o. 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



Dendrogram representation for hierarchical clustering of data objects {*a*,*b*,*c*,*d*,*e*}.

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

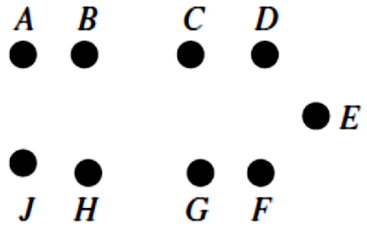
Distance Measures in Hierarchical Methods

- When an algorithm uses the minimum distance, $d_{min}(C_i, C_j)$, to measure the distance between clusters, it is sometimes called a **nearest-neighbor clustering algorithm**.
- If the clustering process is terminated when the distance between nearest clusters exceeds a user-defined threshold, it is called a **single-linkage algorithm**
- If we view the data points as nodes of a graph, with edges forming a path between the nodes in a cluster, then the merging of two clusters, Ci and Cj , corresponds to adding an edge between the nearest pair of nodes in Ci and Cj .
- Because edges linking clusters always go between distinct clusters, the resulting graph will generate a tree
- Thus, an agglomerative hierarchical clustering algorithm that uses the minimum distance measure is also called a **minimal spanning tree algorithm**

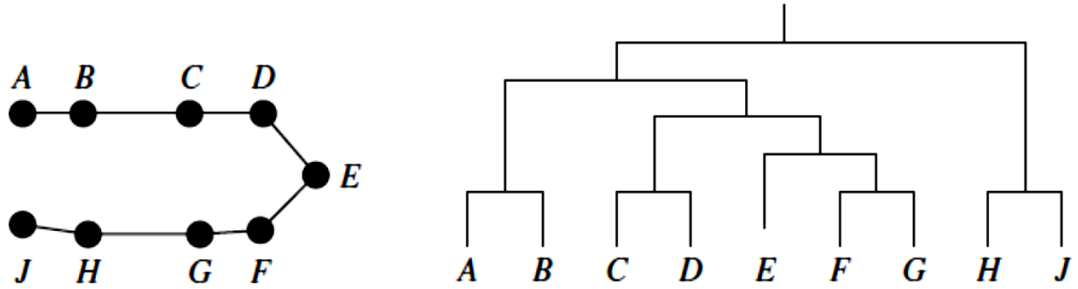
Distance Measures in Hierarchical Methods

- When an algorithm uses the maximum distance, $d_{max}(C_i, C_j)$, to measure the distance between clusters, it is sometimes called a **farthest-neighbor clustering algorithm**.
- If the clustering process is terminated when the maximum distance between nearest clusters exceeds a user-defined threshold, it is called a **complete-linkage algorithm**.
- The minimum and maximum measures represent two extremes in measuring the distance between clusters.
- They tend to be overly sensitive to outliers or noisy data.
- The use of mean or average distance is a compromise between the minimum and maximum distances and overcomes the outlier sensitivity problem.
- Whereas the mean distance is the simplest to compute, the average distance is advantageous in that it can handle categoric as well as numeric data.
- The computation of the mean vector for categoric data can be difficult or impossible to define.

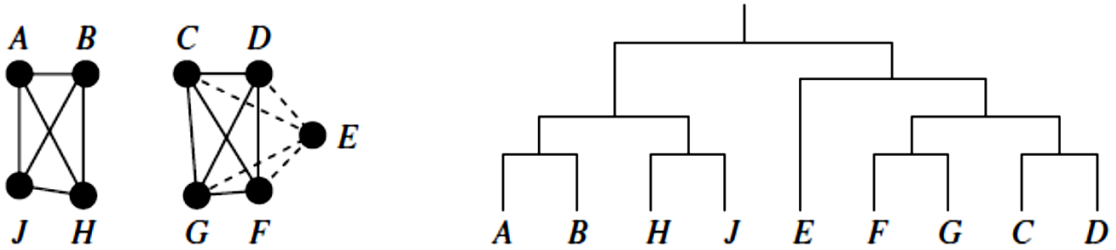
Hierarchical Methods



(a) Data set



(b) Clustering using single linkage



(c) Clustering using complete linkage

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.

- Done by integrating hierarchical clustering (at the initial microclustering stage) and other clustering methods such as iterative partitioning (at the later macroclustering stage).

- **It overcomes the two difficulties in agglomerative clustering methods**:
  (1) scalability and
  (2) the inability to undo what was done in the previous step.

- **BIRCH introduces to concepts:**
  (i) Clustering feature (CF)
  (ii) Clustering Feature tree (CF-tree) , which is used to summarize the cluster representation.

- CF tree, a height balanced tree that stores the clustering feature for hierarchical clustering.

BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

- Consider a cluster of n d-dimensional data objects or points.
- The clustering feature(CF) is represented by;

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

Where

n -> number of points in the cluster

LS -> linear sum of the n points (i.e., $\sum_{i=1}^{n} x_i$),

SS -> square sum of the data points (i.e., $\sum_{i=1}^{n} x_i^2$).

- 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.

BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

- **Measures of Clustering :** Given n d-dimensional data objects or points in a cluster then: **centroid, x_o,** radius, R, and diameter, D, are

Centroid, $$x_0 = \frac{\sum_{i=1}^{n}(x_i)}{n} = \frac{LS}{n}$$

Centroid: Middle of a cluster.
Radius: Average distance from member object to the centroid.
Diameter: Average pairwise distance within a cluster.

radius, $$R = \sqrt{\frac{\sum_{i=1}^{n}(x_i - x_0)^2}{n}} = \sqrt{\frac{nSS - 2LS^2 + nLS}{n^2}},$$

Diameter, $$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}.$$

- Here, R and D reflect the tightness of the cluster around the centroid**.**
- Moreover, clustering features are additive. That is, for two disjoint clusters, C_1 and C_2, with the clustering features $CF_1 = <n_1, LS_1, SS_1>$ and $CF_2 = <n_2, LS_2, SS_2>$, respectively, the clustering feature for the cluster that formed by merging C_1 and C_2 is simply

$$\mathbf{CF_1 + CF_2 = <n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2>}$$

BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

- **Example: Clustering feature.** Suppose there are three points, (2,5),(3,2), and (4,3), in a cluster, C_1. **CF = <n,LS,SS>**
- The clustering feature of C_1 is

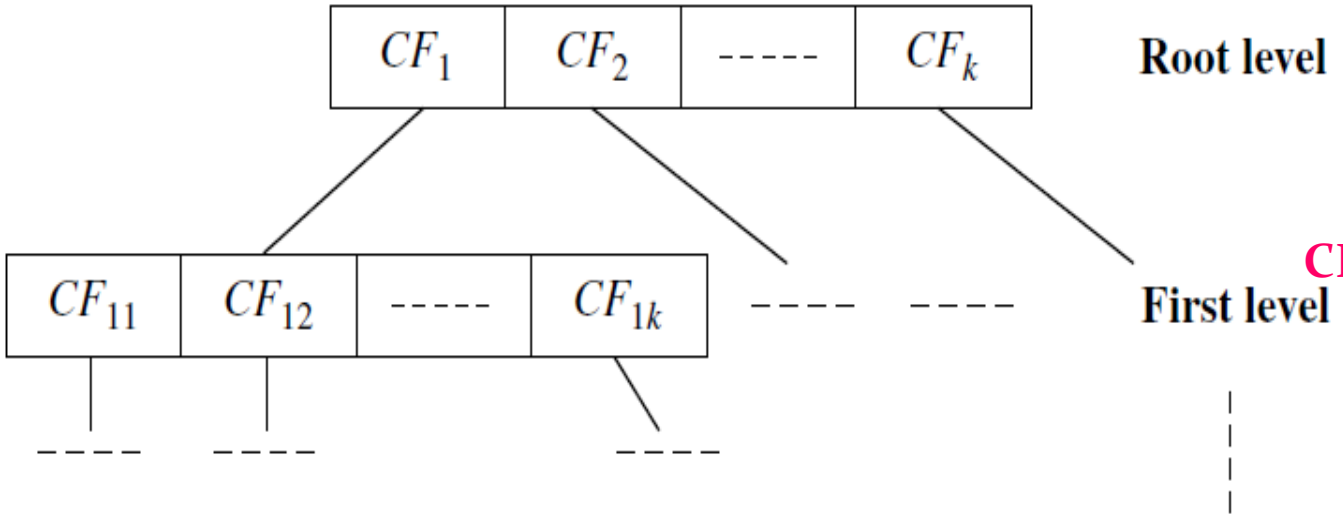
  $CF_1 = <3, (2+3+4,5+2+3),(2^2+3^2+4^2,5^2+2^2+3^2)> = <3,(9,10),(29,38)>$
- Suppose that C_1 is disjoint to a second cluster, C_2 , where

  $CF_2 = <3, (35,36),(417,440)>$
- The clustering feature of a new cluster, C_3, that is formed by merging C_1 and C_2, is derived by adding CF_1 and CF_2. That is,

  $CF_3 = <3+3, (9+35,10+36), (29+417,38+440)> =<6, (44,46),(446,478)>$
- A CF-tree is a height-balanced tree that stores the clustering features for a hierarchical clustering.

BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

- The nonleaf nodes store sums of the CFs of their children, and thus summarize clustering information about their children.
- **A CF-tree has two parameters:** branching factor, B, and threshold, T.
- The **branching factor** specifies the maximum number of children per nonleaf node.
- The **threshold** parameter specifies the maximum diameter of subclusters stored at the leaf nodes of the tree.
- These two parameters implicitly control the resulting tree's size.



CF-tree structure

BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

- Given a limited amount of main memory, an important consideration in BIRCH is to minimize the time required for input/output (I/O).
- BIRCH applies a multiphase clustering technique: A single scan of the data set yields a basic, good clustering, and one or more additional scans can optionally be used to further improve the quality. The primary phases are
- **Phase 1**: BIRCH scans the database to build an initial in-memory CF-tree, which can be viewed as a multilevel compression of the data that tries to preserve the 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.

BIRCH: Multiphase Hierarchical Clustering Using Clustering Feature Trees

- The time complexity of the algorithm is O(n), where n is the number of objects to be clustered.
- Experiments have shown the linear scalability of the algorithm with respect to the number of objects, and good quality of clustering of the data.
- However, since each node in a CF-tree can hold only a limited number of entries due to its size, a CF-tree node does not always correspond to what a user may consider a natural cluster.
- Moreover, if the clusters are not spherical in shape, BIRCH does not perform well because it uses the notion of radius or diameter to control the boundary of a cluster.

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:

  (1) how well connected objects are within a cluster and

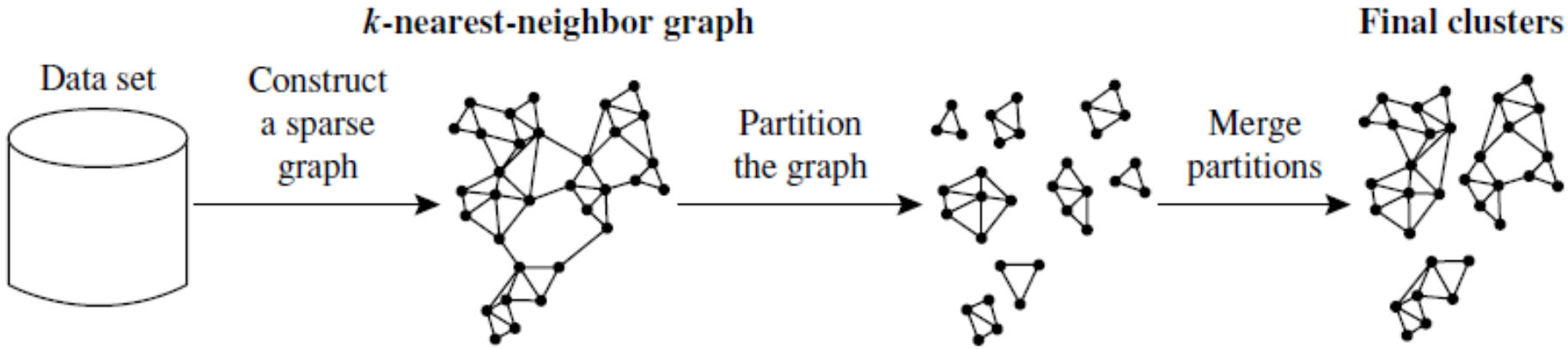
  (2) the proximity of clusters.
- That is, two clusters are merged if their interconnectivity is high and they are close together.

- 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: Multiphase Hierarchical Clustering Using Dynamic Modeling

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



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.

K-NN Graph: Points p and q are connected if q is among the top-k closest neighbors of p

Chameleon: Multiphase Hierarchical Clustering Using 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, $RI(C_i, C_j)$, and their relative closeness, $RC(C_i, C_j)$
- **The relative interconnectivity**, $RI(C_i, C_j)$ between two clusters, C_i and C_j , is defined as the absolute interconnectivity between C_i and C_j , normalized with respect to the internal interconnectivity of the two clusters, C_i and C_j . That is,

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

- where $EC_{\{C_i, C_j\}}$ is the edge cut as previously defined for a cluster containing both Ci and Cj . Similarly, EC_{C_i} (or EC_{C_j}) is the minimum sum of the cut edges that partition Ci (or Cj) into two roughly equal parts.

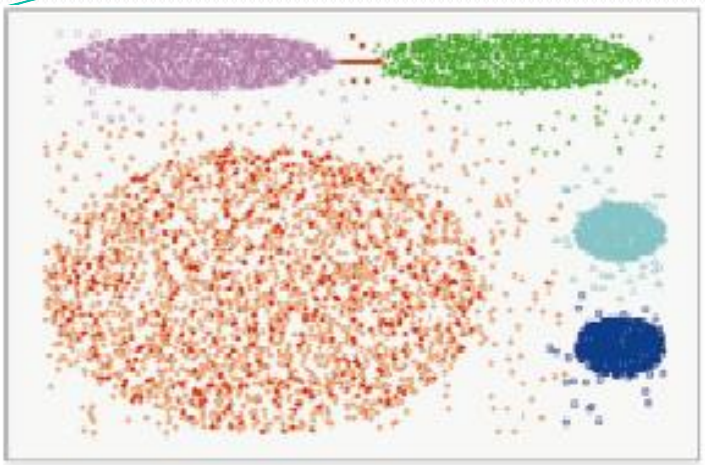
Chameleon: Multiphase Hierarchical Clustering Using Dynamic Modeling

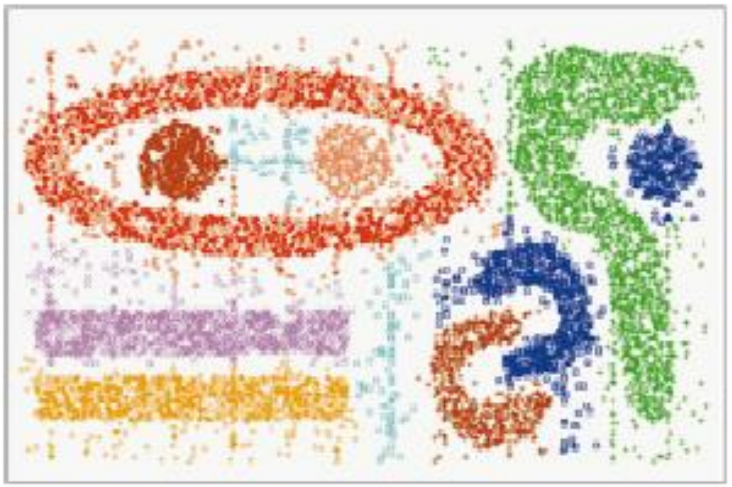
- The **relative closeness**, $RC(C_i, C_j)$, between a pair of clusters, C_i and C_j , is the absolute closeness between C_i and C_j, normalized with respect to the internal closeness of the two clusters, C_i and C_j. It is defined as

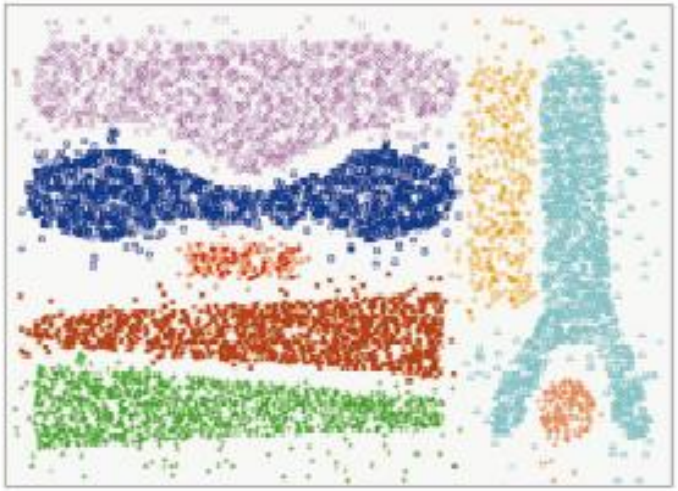
$$RC(C_i, C_j) = \frac{\overline{S}_{EC_{\{C_i,C_j\}}}}{\frac{|C_i|}{|C_i|+|C_j|}\overline{S}_{EC_{C_i}} + \frac{|C_j|}{|C_i|+|C_j|}\overline{S}_{EC_{C_j}}},$$

- where $\overline{S}_{EC_{\{C_i, C_j\}}}$ is the average weight of the edges that connect vertices in C_i to vertices in C_j , and $\overline{S}_{EC_{C_i}}$ (or $\overline{S}_{EC_{C_j}}$) is the average weight of the edges that belong to the min-cut bisector of cluster Ci (or Cj).
- Chameleon has been shown to have greater power at discovering arbitrarily shaped clusters of high quality than several well-known algorithms such as BIRCH and density based DBSCAN.
- However, the processing cost for high-dimensional data may require $O(n^2)$ time for n objects in the worst case.

CHAMELEON: Clustering Complex Objects







CHAMELEON is capable of generate quality clusters at clustering complex objects.

Probabilistic Hierarchical Clustering

- **Algorithmic hierarchical clustering methods can suffer from several drawbacks.**
  - Nontrivial to choose a good distance measure
  - Hard to handle missing attribute values.
  - Optimization goal not clear: heuristic, local search
  - Consequently, the optimization goal of the resulting cluster hierarchy can be unclear.

- **Probabilistic hierarchical clustering aims to overcome some of these disadvantages**
  - using probabilistic models to measure distances between clusters.
  - Generate model: regard the set of data objects to be clustered as a sample of the underlying data generation mechanism to be analyzed.
  - Easy to understand, same efficiency as algorithmetic agglomerative clustering method, can handle partially observed data.

- In practice, assume the generative models adopt common distribution functions, such as Gaussian distribution or Bernoulli distribution, which are governed by parameters.

62

Probabilistic Hierarchical Clustering

- ***Given a set of 1-D points $X = \{x_1, \ldots, x_n\}$ for clustering analysis and assuming they are generated by a Gaussian distribution,***

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

where the parameters are μ (the mean) and σ² (the variance).

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

$$P(x_i|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}.$$

Probabilistic Hierarchical Clustering

- ***The likelihood that X is generated by the model is***

$$L(\mathcal{N}(\mu,\sigma^2) : X) = P(X|\mu,\sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

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

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

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

Probabilistic Hierarchical Clustering

- For a set of objects partitioned into m clusters $C_1, \ldots, C_m$, the quality can be measured by

$$Q(\{C_1, \ldots, C_m\}) = \prod_{i=1}^{m} P(C_i),$$

- where P() is the maximum likelihood. If we merge two clusters, C_{j_1} and C_{j_2}, into a cluster, $C_{j_1} \cup C_{j_2}$, then, the change in quality of the overall clustering is

$$Q((\{C_1, \ldots, C_m\} - \{C_{j_1}, C_{j_2}\}) \cup \{C_{j_1} \cup C_{j_2}\}) - Q(\{C_1, \ldots, C_m\})$$

$$= \frac{\prod_{i=1}^{m} P(C_i) \cdot P(C_{j_1} \cup C_{j_2})}{P(C_{j_1})P(C_{j_2})} - \prod_{i=1}^{m} P(C_i)$$

$$= \prod_{i=1}^{m} P(C_i) \left(\frac{P(C_{j_1} \cup C_{j_2})}{P(C_{j_1})P(C_{j_2})} - 1 \right).$$

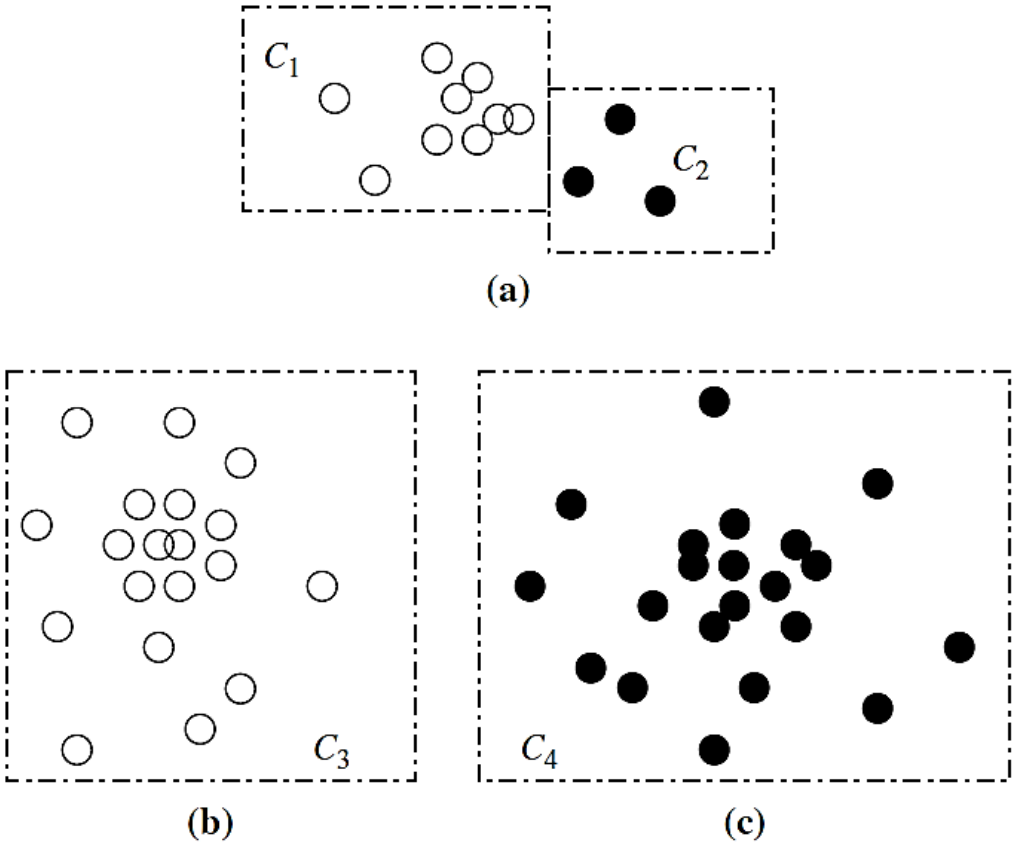
Probabilistic Hierarchical Clustering

- When choosing to merge two clusters in hierarchical clustering, $\prod_{i=1}^{m} P(Ci)$ is constant for any pair of clusters. Therefore, given clusters C1 and C2, the distance between them can be measured by

$$dist(C_i, C_j) = -\log \frac{P(C_1 \cup C_2)}{P(C_1)P(C_2)}$$

- Merging two clusters may not always lead to an improvement in clustering quality, that is $\frac{P(C_{j1} \cup C_{j2})}{P(C_{j1})P(C_{j2})}$ may be less than 1

Probabilistic Hierarchical Clustering



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 as no Gaussian functions can fit the merged cluster well

Probabilistic Hierarchical Clustering

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 $C_i = \{o_i\}$, $1 \leq i \leq n$;
(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;

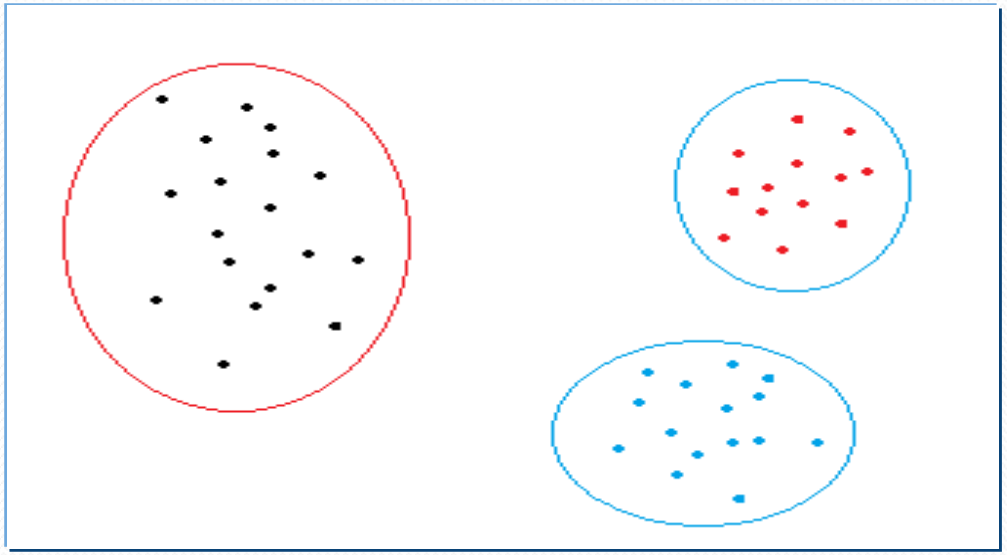
Probabilistic Hierarchical Clustering

- Probabilistic hierarchical clustering methods are easy to understand, and generally have the same efficiency as algorithmic agglomerative hierarchical clustering methods; in fact, they share the same framework.
- Probabilistic models are more interpretable, but sometimes less flexible than distance metrics.
- Probabilistic models can handle partially observed data
- 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.

Density-Based Methods

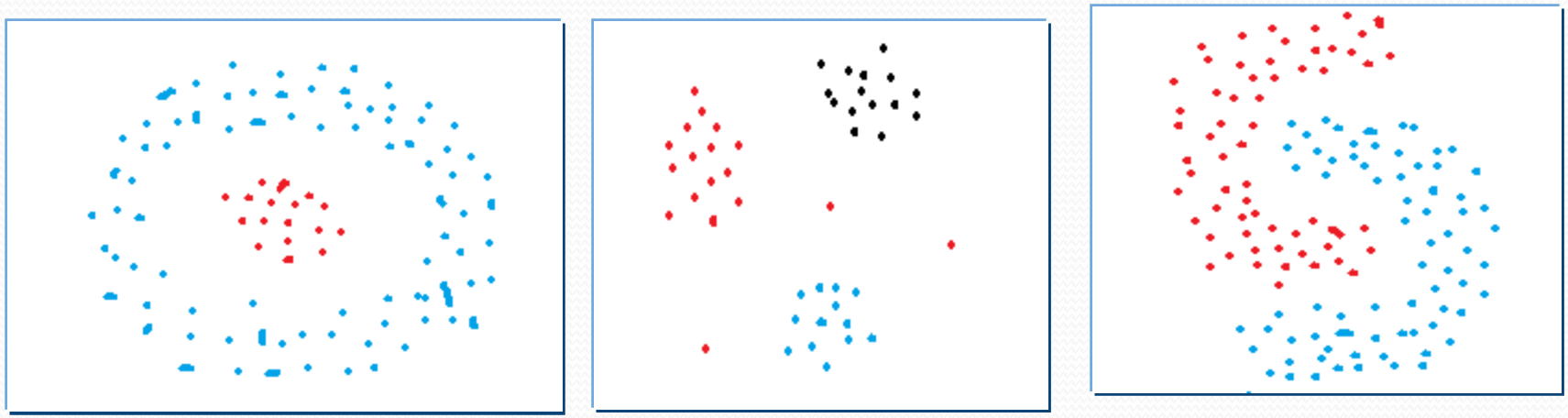
- Partitioning and hierarchical methods are designed to find spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data, density-based techniques are more efficient.

- For example, the dataset in the figure below can easily be divided into three clusters using k-means algorithm.



Density-Based Methods

Consider the following figures:



The data points in these figures are grouped in arbitrary shapes or include outliers. Density-based clustering algorithms are very efficient at finding high-density regions and outliers. It is very important to detect outliers for some task, e.g. anomaly detection.

DBSCAN

- DBSCAN stands for **d**ensity-**b**ased **s**patial **c**lustering of **a**pplications with **n**oise.

  - 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.

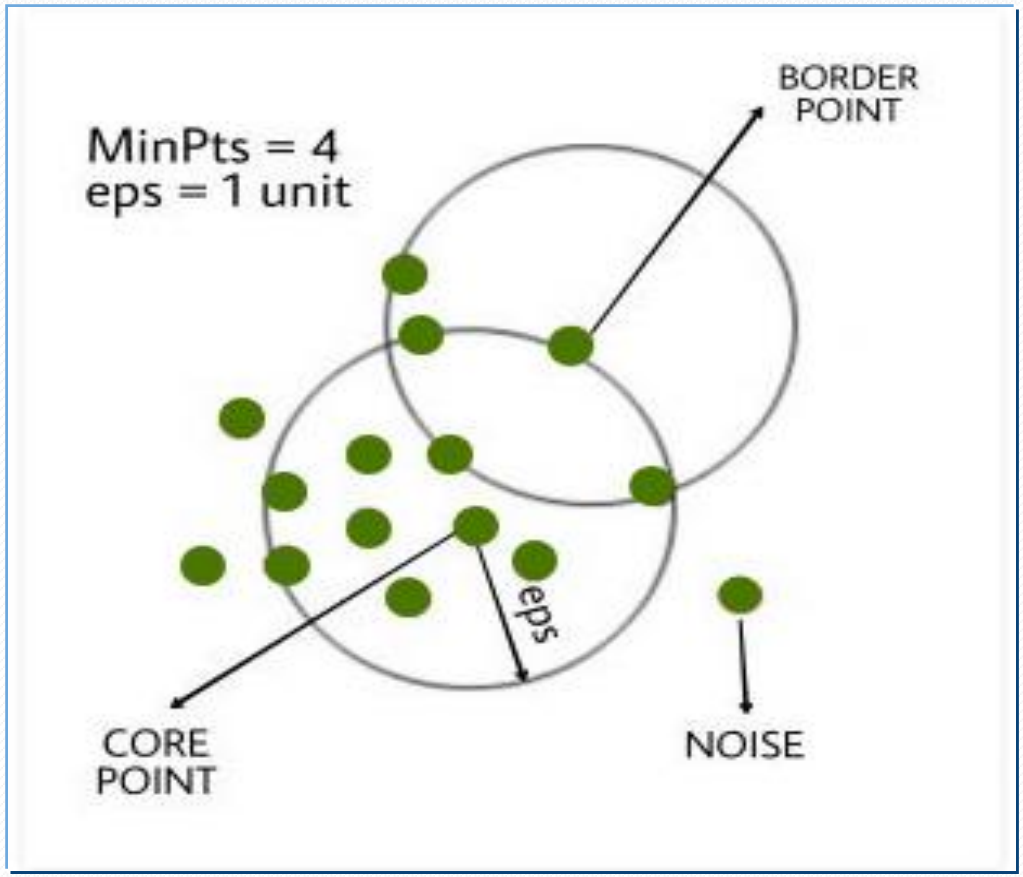
Clusters of arbitrary shape

DBSCAN

- ***There are two key parameters of DBSCAN:***
  - **eps**: The distance that specifies the neighborhoods. Two points are considered to be neighbors if the distance between them are less than or equal to eps.
  - **minPts:** Minimum number of data points to define a cluster.

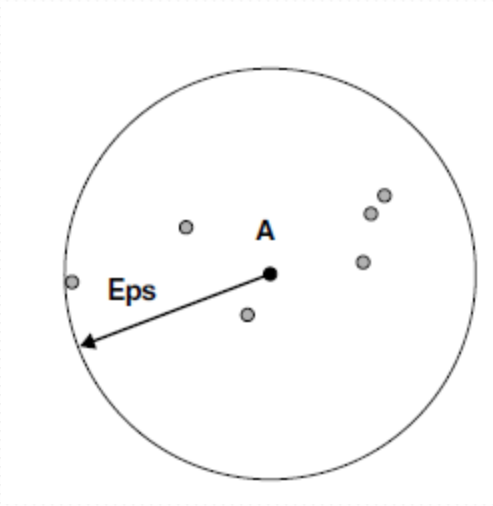
- ***Based on these two parameters, points are classified as core point, border point, or outlier:***
  - **Core points:** A point is a core point if it has more than minPts number of points (including the point itself) in its surrounding area with radius eps.
  - **Border points:** A border point is not a core point, but falls within the neighborhood of a core point. A border point can fall within the neighborhoods of several core points.
  - **Noise points:** A noise point is any point that is neither a core point nor a border point.

DBSCAN

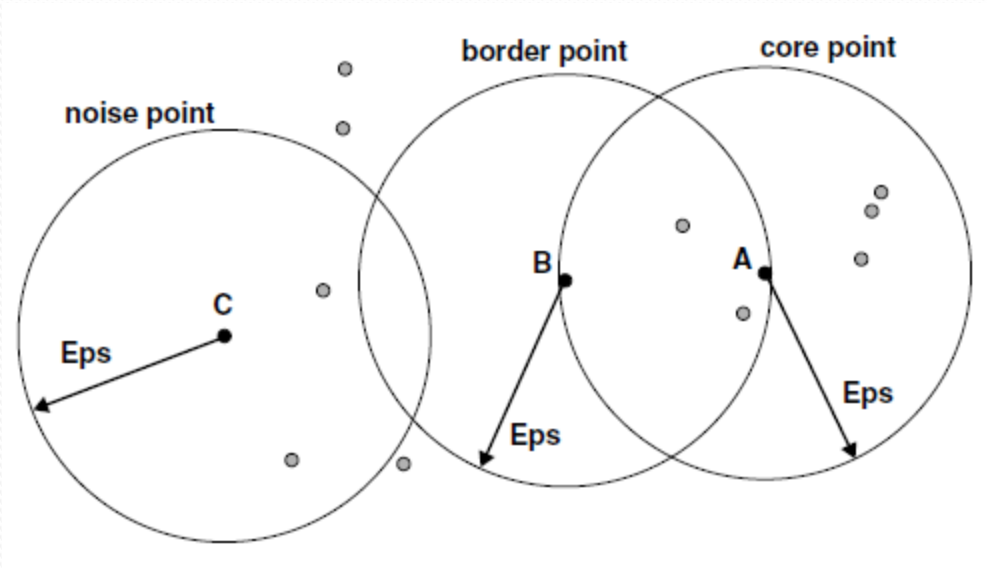


DBSCAN

- In the center-based approach, density is estimated for a particular point in the data set by counting the number of points within a specified radius, Eps, of that point. This includes the point itself



Center-based density



Core, border, and noise points.

DBSCAN

DBSCAN Algorithm

1. Label all points as core, border, or noise points.
2. Eliminate noise points.
3. Put an edge between all core points that are within Eps of each other.
4. Make each group of connected core points into a separate cluster.
5. Assign each border point to one of the clusters of its associated core points

- If a spatial index is used, the computational complexity of DBSCAN is O(nlogn), where n is the number of database objects.
- Otherwise, the complexity is $O(n^2)$.
- With appropriate settings of the user-defined parameters, $\in$ and MinPts, the algorithm is effective in finding arbitrary-shaped clusters.

DBSCAN

- *Advantages of DBSCAN:*
  - It will separate clusters of high density versus clusters of low density within a given dataset.
  - Handle outliers within the dataset.

- *Disadvantages of DBSCAN:*
  - Does not work well when dealing with clusters of varying densities.
  - It struggles with high dimensionality data.

Reference:

- Data Mining Concepts and Techniques, Jiawei Han, Micheline Kamber, Jian Pei,3rd Edition, Morgan Kaufmann Publishers