UNIT-V

CLUSTER ANALYSIS: BASIC CONCEPTS AND METHODS

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 and similarities are assessed based on the attribute values describing the objects and often involve distance measures.

1. Cluster Analysis

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**. In this context, different clustering methods may generate different clusterings on the same data set.

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

In image recognition, clustering can be used to discover clusters or “subclasses” in handwritten character recognition systems. Suppose we have a data set of handwritten digits, where each digit is labeled as either 1, 2, 3, and so on.

Clustering has also found many applications in Web search. For example, a keyword search may often return a very large number of hits (i.e., pages relevant to the search) due to the extremely large number of web pages. Clustering can be used to organize the search results into groups and present the results in a concise and easily accessible way.

As a data mining function, cluster analysis can be used as a standalone tool to gain insight into the distribution of data, to observe the characteristics of each cluster, and to focus on a particular set of clusters for further analysis.

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.

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*.

Requirements for Cluster Analysis

the requirements for clustering as a data mining tool, as well as aspects that can be used for comparing clustering methods.

The following are typical requirements of clustering in data mining.

- **Scalability**: Many clustering algorithms work well on small data sets containing fewer than several hundred data objects; however, a large database may contain millions or even billions of objects, particularly in Web search scenarios.
- **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.
- **Discovery of clusters with arbitrary shape**: Many clustering algorithms determine clusters based on Euclidean or Manhattan distance measures.
- **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.
- **Ability to deal with noisy data**: Most real-world data sets contain outliers and/or missing, unknown, or erroneous data.
- **Incremental clustering and insensitivity to input order**: In many applications, incremental updates (representing newer data) may arrive at any time.
- **Capability of clustering high-dimensionality data**: A data set can contain numerous dimensions or attributes.

Overview of Basic Clustering Methods

There are many clustering algorithms in the literature. It is difficult to provide a crisp categorization of clustering methods because these categories may overlap so that a method may have features from several categories.

Partitioning methods: Given a set of *n* objects, a partitioning method constructs *k* partitions of the data, where each partition represents a cluster and *k* <_ *n*. That is, it divides the data into *k* groups such that each group must contain at least one object. In other words, partitioning methods conduct one-level partitioning on data sets. The basic partitioning methods typically adopt *exclusive cluster separation.*

Most partitioning methods are distance-based. Given *k*, the number of partitions to construct, a partitioning method creates an initial partitioning.

most applications adopt popular heuristic methods, such as greedy approaches like the *k-means* and the *k-medoids* algorithms, which progressively improve the clustering quality and approach a local optimum. These heuristic clustering methods work well for finding spherical-shaped clusters in small- to medium-size databases.

Hierarchical methods: A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either *agglomerative* or *divisive*, based on how the hierarchical decomposition is formed.

The *agglomerative approach*, also called the *bottom-up* approach, starts with each object forming a separate group. It successively merges the objects or groups 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 clustering methods can be distance-based or density- and continuity based.

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

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

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

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

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)

Overview of clustering methods discussed in this chapter. Note that some algorithms may combine various methods.

Similarity and Dissimilarity between Objects

After standardization, the dissimilarity (or similarity) between the objects described by interval-scaled variables is computed based on the distance between each pair of objects.

1) Euclidean distance

The most popular distance measure is Euclidean distance, which is defined as

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2},$$

where i=(xi1, xi2, : : : , xin) and j =(x j1, x j2, : : : , x jn) are two n-dimensional data objects.

2) Manhattan distance

Another metric is Manhattan distance, defined as

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}|.$$

Both the Euclidean distance and Manhattan distance satisfy the following mathematic requirements of a distance function

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

3. Minkowski distance:

Minkowski distance is a generalization of both Euclidean distance and Manhattan distance. It is defined as

$$d(i, j) = (|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \cdots + |x_{in} - x_{jn}|^p)^{1/p},$$

where p is a positive integer. Such a distance is also called Lp norm.

- If p=1, d is Manhattan distance (i.e., L1 norm)
- If p=2, d is Euclidean distance (i.e., L2 norm).

2. 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. To keep the problem specification concise, we can assume that the number of clusters is given as background knowledge. This parameter is the starting point for partitioning methods.

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, : : : ,*Ck*, that is, *Ci* ⊂ *D* and *Ci* ∩*Cj*=*Θ* ; 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.

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 **within cluster 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*.

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.

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:

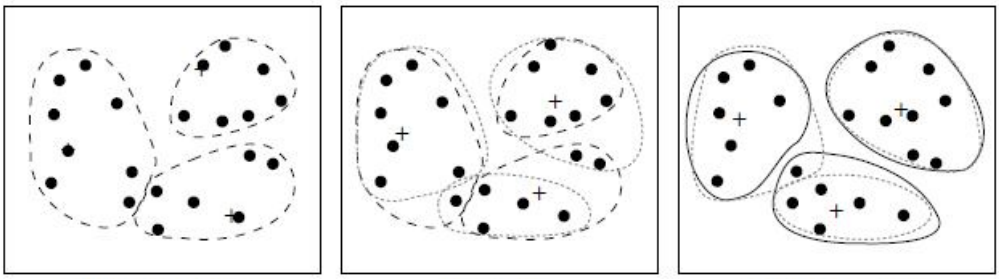
(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;

(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;



(a) Initial clustering (b) Iterate (c) Final clustering

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 +).

Example

Clustering by *k*-means partitioning. Consider a set of objects located in 2-D space, as depicted in Figure (a). Let *k* D 3, that is, the user would like the objects to be partitioned into three clusters.

According to the algorithm , we arbitrarily choose three objects as the three initial cluster centers, where cluster centers are marked by a C. Each object is assigned to a cluster based on the cluster center to which it is the nearest. Such a distribution forms silhouettes encircled by dotted curves.

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

The process of iteratively reassigning objects to clusters to improve the partitioning is referred to as *iterative relocation*. Eventually, no reassignment of the objects in any cluster occurs and so the process terminates. The resulting clusters are returned by the clustering process.

Example

Use the k-mean algorithm and Euclidean distance to cluster

A1=(2,10) A2=(2,5),A3=(8,4) A4=(5,8)

A5=(7,5) A6=(6,4) A7=(1,2) A8=(4,9)

K=3

Initial cluster points are A1,A4,A7

	Distance from center (2,10)of cluster1(C1)	Distance from center (5,8)of cluster2(C2)	Distance from center (1,2)of cluster3(C3)	Cluster point	Note
(2,10)	0	√13	√65	C1	A1€C1
(2,5)	√25	√18	√10	C3	A2€C3
(8,4)	√36	√25	√53	C2	A3€C2
(5,8)	√13	√0	√52	C2	A4€C2
(7,5)	√50	√13	√45	C2	A5€C2
(6,4)	√52	√17	√29	C2	A6€C2
(1,2)	√65	√52	√0	C3	A7€C3
(4,9)	√5	√2	√58	C2	A8€C2

New Cluster

A)

Cluster1={A1}

Cluster2={A3,A4,A5,A6,A8}

Cluster3={A2,A7}

B)

C1=(2,10)

C2=

A3=(8,4) A4=(5,8)

A5=(7,5) A6=(6,4) A8=(4,9)

C2=(8+5+7+6+4/5,4+8+5+4+9/5)

=(30/5,30/5)=(6,6)

C3=

A2=(2,5) A7=(1,2)

C3=(2+1/2,5+2/2)

C3=(1.5,3.5)

	Distance from center (2,10)of cluster1(C1)	Distance from center (6,6)of cluster2(C2)	Distance from center (1.5,3.5)of cluster3(C3)	Cluster point	Note
(2,10)	0	√32	√42.5	C1	A1€C1
(2,5)	√25	√17	√2.5	C3	A2€C3
(8,4)	√36	√8	√44.5	C2	A3€C2
(5,8)	√13	√5	√32.5	C2	A4€C2
(7,5)	√50	√2	√32.5	C2	A5€C2
(6,4)	√52	√4	√20.5	C2	A6€C2
(1,2)	√65	√41	√2.5	C3	A7€C3
(4,9)	√5	√13	√36.5	C1	A8€C1

New Cluster

A)

Cluster1={A1,A8}

Cluster2={A3,A4,A5,A6 }

Cluster3={A2,A7}

B) Center for new Cluster

C1=(2,10) A8=(4,9)

C1=(2+4/2,10+9/2)

=(3,9.5)

C2=

A3=(8,4) A4=(5,8)

A5=(7,5) A6=(6,4)

C2=(8+5+7+6/4,4+8+5+4/4)

=(6.5,5.25)

C3=

A2=(2,5) A7=(1,2)

C3=(2+1/2,5+2/2)

C3=(1.5,3.5)

	Distance from center (3,9.5)of cluster1(C1)	Distance from center (6.5,5.25)of cluster2(C2)	Distance from center (1.5,3.5)of cluster3(C3)	Cluster point	Note
(2,10)	√1.25	√42.81	√42.5	C1	A1€C1
(2,5)	√21.25	√20.31	√2.5	C3	A2€C3
(8,4)	√55.25	√3.81	√44.5	C2	A3€C2
(5,8)	√6.25	√9.81	√32.5	C1	A4€C1
(7,5)	√36.25	√0.2065	√32.5	C2	A5€C2
(6,4)	√39.25	√1.81	√20.5	C2	A6€C2
(1,2)	√60.25	√40.18	√2.5	C3	A7€C3
(4,9)	√1.25	√20.31	√36.5	C1	A8€C1

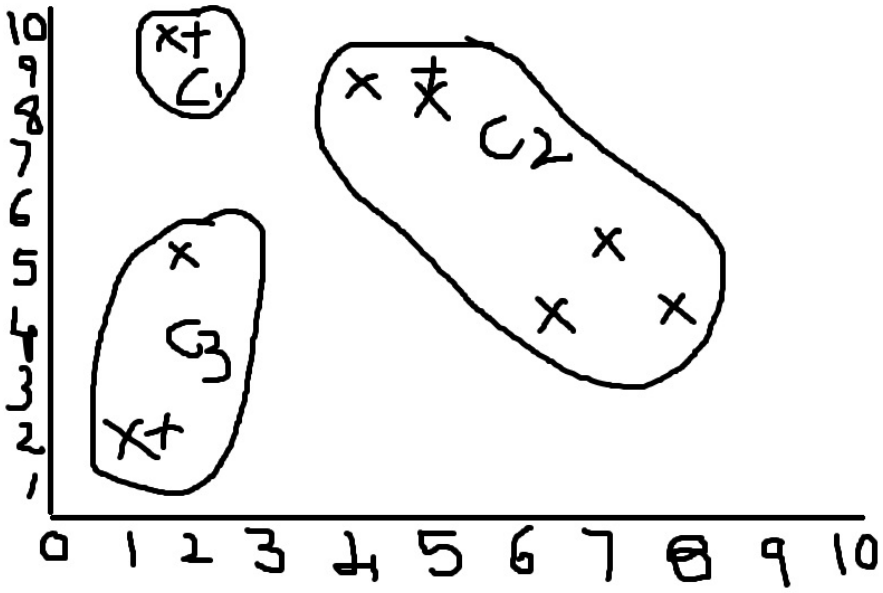
k-Means algorithm using Manhattan Distance for forming cluster

A1=(2,10) A2=(2,5) A3=(8,4) A4=(5,8)

A5=(7,5) A6=(6,4) A7=(1,2) A8=(4,9)

Suppose no of clusters are 3

Initial Cluster seeds are A1,A4,A7.



	(2,10) C1	(5,8) C2	(1,2) C3	Cluster Point	Note
(2,10)	0	5	9	C1	A1€C1
(2,5)	5	6	4	C3	A2€C3
(8,4)	12	7	9	C2	A3€C2
(5,8)	5	0	10	C2	A4€C2
(7,5)	10	5	9	C2	A5€C2
(6,4)	10	5	7	C2	A6€C2
(1,2)	9	10	0	C3	A7€C3
(4,9)	3	2	10	C2	A8€C2

(a) New Cluster

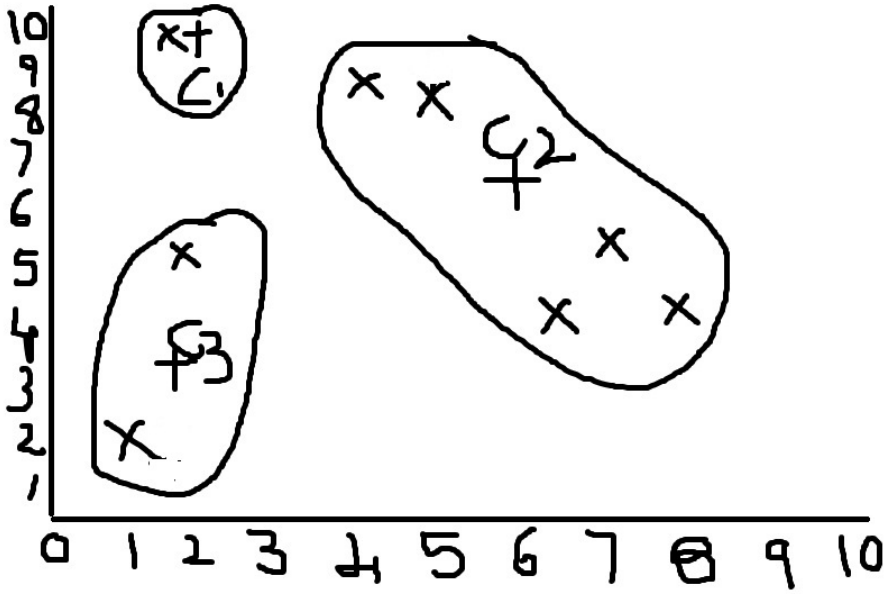
Cluster1={A1} Cluster2={A3,A4,A5,A6,A8} Cluster3={A2,A7}

(b) Center for new Cluster

C1=(2,10)

C2=((8+5+7+6+4)/5,(4+8+5+4+9)/5)=(6,6)

C3=((2+1)/2,(5+2)/2)=(1.5,3.5)



	(2,10) C1	(6,6) C2	(1.5,3.5) C3	Cluster Point	Note
(2,10)	0	8	7	C1	A1€C1
(2,5)	5	5	2	C3	A2€C3
(8,4)	12	4	7	C2	A3€C2
(5,8)	5	3	8	C2	A4€C2
(7,5)	10	2	7	C2	A5€C2
(6,4)	10	2	5	C2	A6€C2
(1,2)	9	9	2	C3	A7€C3
(4,9)	3	5	8	C1	A8€C1

(a) New Cluster

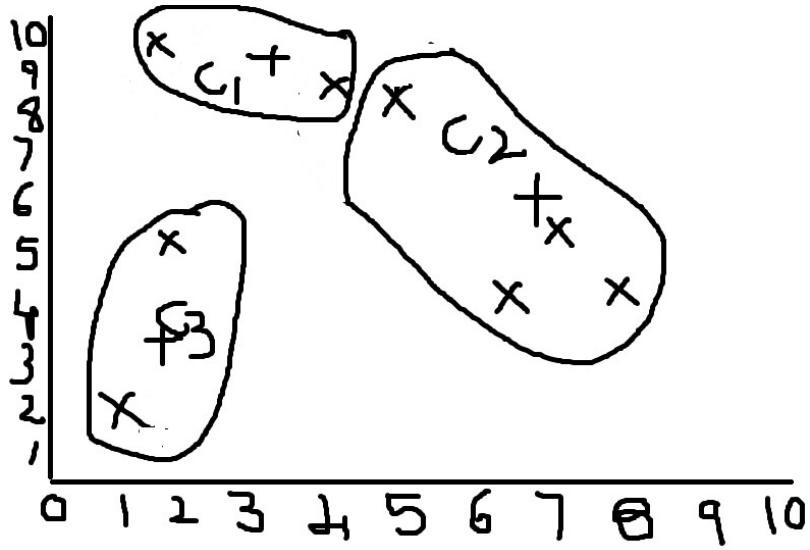
Cluster1={A1,A8} Cluster2={A3,A4,A5,A6} Cluster3={A2,A7}

(b) Center for new Cluster

C1=(3,9.5)

C2=(6.5,5.25)

C3=((2+1)/2,(5+2)/2)=(1.5,3.5)



	(3,9.5) C1	(6.5,5.25) C2	(1.5,3.5) C3	Cluster Point	Note
(2,10)	1.5	9.75	7	C1	A1€C1
(2,5)	5.5	4.75	2	C3	A2€C3
(8,4)	10.5	2.75	7	C2	A3€C2
(5,8)	3.5	4.25	8	C1	A4€C1
(7,5)	8.5	0.75	7	C2	A5€C2
(6,4)	8.5	1.75	5	C2	A6€C2
(1,2)	9.5	8.75	2	C3	A7€C3
(4,9)	1.5	6.25	8	C1	A8€C1

(a) New Cluster

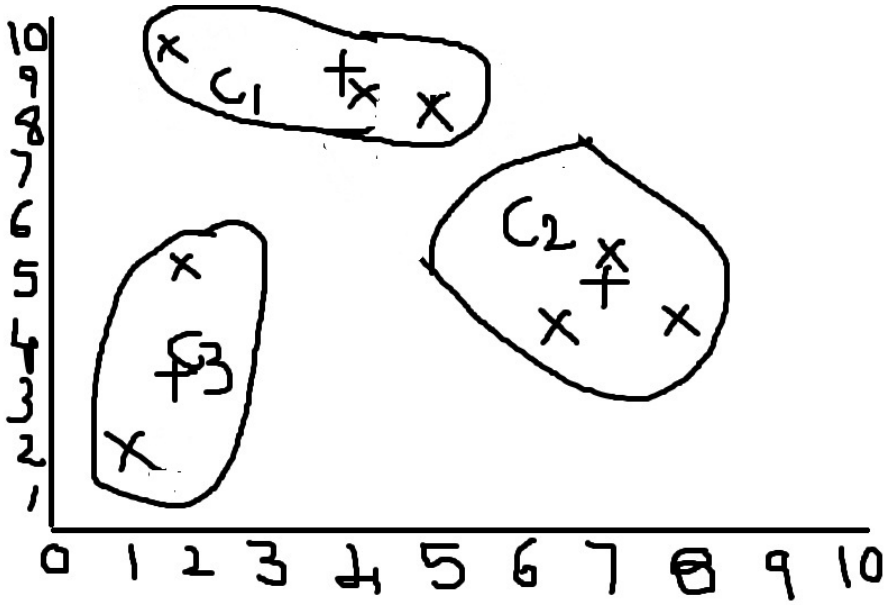
Cluster1={A1,A4,A8} Cluster2={A3,A5,A6} Cluster3={A2,A7}

(b) Center for new Cluster

C1=(3.6,9)

C2=(7,4.33)

C3=((2+1)/2,(5+2)/2)=(1.5,3.5)



	(3.6,9) C1	(7,4.33) C2	(1.5,3.5) C3	Cluster Point	Note
(2,10)	2.6	10.7	7	C1	A1€C1
(2,5)	5.6	5.7	2	C3	A2€C3
(8,4)	9.4	1.3	7	C2	A3€C2
(5,8)	2.4	5.7	8	C1	A4€C1
(7,5)	8.4	0.7	7	C2	A5€C2
(6,4)	7.4	1.3	5	C2	A6€C2
(1,2)	9.6	8.3	2	C3	A7€C3
(4,9)	0.4	7.7	8	C1	A8€C1

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

Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$

Comment: Often terminates at a *local optimal*.

Weakness

Applicable only to objects in a continuous n-dimensional space

Using the k-modes method for categorical data

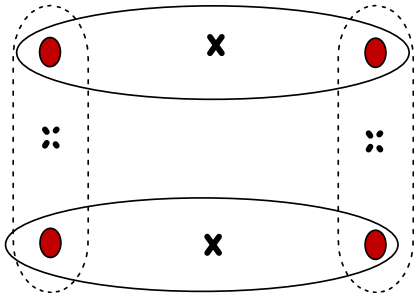
In comparison, k-medoids can be applied to a wide range of data

Need to specify *k*, the *number* of clusters, in advance (there are ways to automatically determine the best k (see Hastie et al., 2009)

Sensitive to noisy data and *outliers*

Not suitable to discover clusters with *non-convex shapes*

- Most of the variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means



- Handling categorical data: *k-modes*
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

k-Medoids: A Representative Object-Based Technique

The *k*-means algorithm is sensitive to outliers because such objects are far away from the majority of the data, and thus, when assigned to a cluster, they can dramatically distort the mean value of the cluster. This inadvertently affects the assignment of other objects to clusters

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 $k = 2$, the partitioning {1, 2,3}, {8, 9, 10,25} has the within-cluster variation.

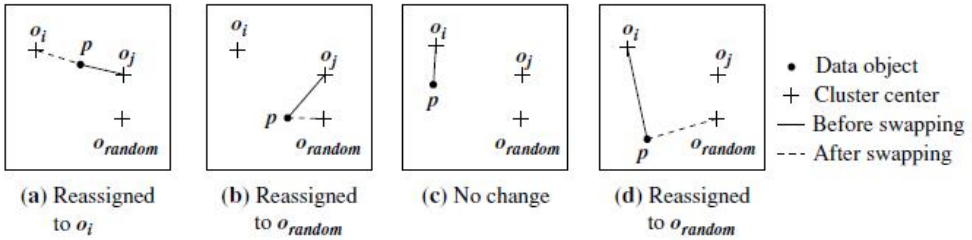
$$(1-2)^2+(2-2)^2+(3-2)^2+(8-13)^2+(9-13)^2+(10-13)^2+(25-13)^2=196,$$

given that the mean of cluster $\{1,2,3\}$ is 2 and the mean of $\{8,9,10,25\}$ is 13. Compare this to the partitioning $\{\{1,2,3,8\},\{9,10,25\}\}$, for which *k*-means computes the within-cluster variation as

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

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.

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.



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

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.

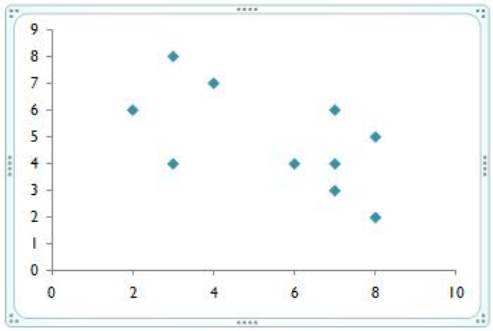
Output: A set of *k* clusters.

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 non representative object, ***Orandom***;
(5) compute the total cost, *S*, of swapping representative object, ***oj***, with ***Orandom***;
(6) **if** $S < 0$ **then** swap ***oj*** with ***Orandom*** to form the new set of *k* representative objects;
(7) **until** no change;

EXAMPLE: DataSet and Plots

	X	Y
0	7	6
1	2	6
2	3	8
3	8	5
4	7	4
5	4	7
6	8	2
7	7	3
8	6	4
9	3	4



Step 1:Let us randomly consider k=2

selected 2 medoids be C1=(3,4) C2=(7,4)

Step 2:Calculating Cost by $d=|x_2-x_1|+|y_2-y_1|$

The dissimilarity of each non-medoid point with medoids

calculated and tabulated

	X	Y	Dissimilarity From C1	Dissimilarity From C2	cluster
0	7	6	6	2	C2
1	2	6	3	7	C1
2	3	8	4	8	C1
3	8	5	6	2	C2
4	7	4	4	0	C2
5	4	7	4	6	C1
6	8	2	5	3	C2
7	7	3	5	1	C2
8	6	4	3	1	C2
9	3	4	0	4	C1

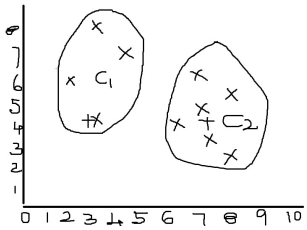
Step 3: Each point is assigned to the cluster of that medoids whose dissimilarity less. Points in cluster are:

C1=(1,2,5) and C2=(0,3,6,7,8)

Cost C=(3+4+4)+(2+2+3+1+1) =20

Step 4: Now randomly select one non medoid point and recalculate the cost.

Let the randomly selected point be (7,3) the dissimilarity of each non medoids point with the medoids C1=(3,4) and C2=(7,3) is calculated tabulated.



	X	Y	Dissimilarity From C I	Dissimilarity From C2
0	7	6	6	3
I	2	6	3	8
2	3	8	4	9
3	8	5	6	3
4	7	4	4	I
5	4	7	4	7
6	8	2	5	4
7	7	3	5	0
8	6	4	3	2
9	3	4	0	5

Each point is assigned to that cluster whose dissimilarity is less. So, the points 1,2,5 go to cluster C1, and 0,3,6,7,8 go to cluster C2.

The cost C=(3+4+4)+(2+2+1+3+3)

C=22

Swap cost=present cost- previous cost 22-20 =2>0

As the swap cost is not less than zero we undo the swap.

3. Hierarchical Methods

A **hierarchical clustering method** works by grouping data objects into a hierarchy or "tree" of clusters.

Hierarchical clustering methods can encounter difficulties regarding the selection of merge or split points. Such a decision is critical, because once a group of objects is merged or split, the process at the next step will operate on the newly generated clusters. It will neither undo what was done previously, nor perform object swapping between clusters. Thus, merge or split decisions,

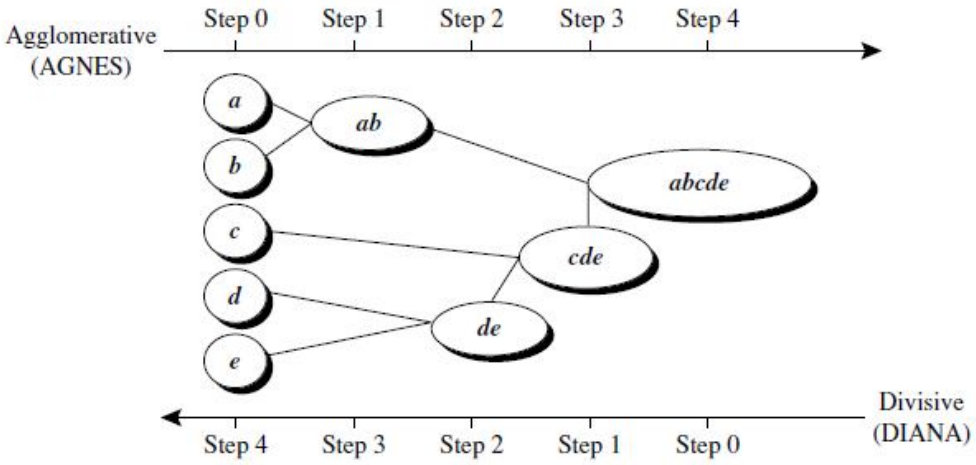
Agglomerative versus Divisive Hierarchical Clustering

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

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.

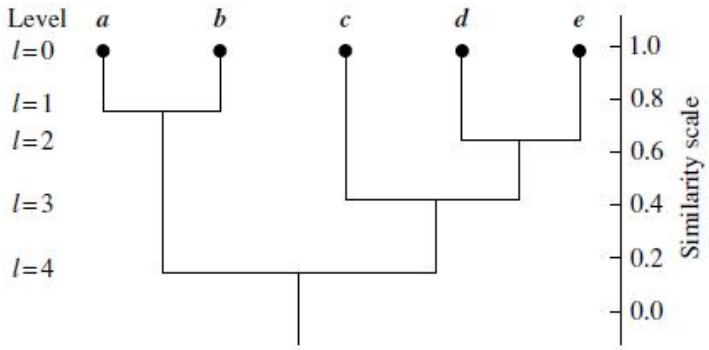
A **divisive hierarchical clustering method** employs a top-down strategy. It starts by placing all objects in one cluster, which is the hierarchy's root. It then divides the root cluster into several smaller subclusters, and recursively partitions those clusters into smaller ones.

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.



Agglomerative and divisive hierarchical clustering on data 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. a dendrogram for the five objects presented .



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

4.Density-Based Methods

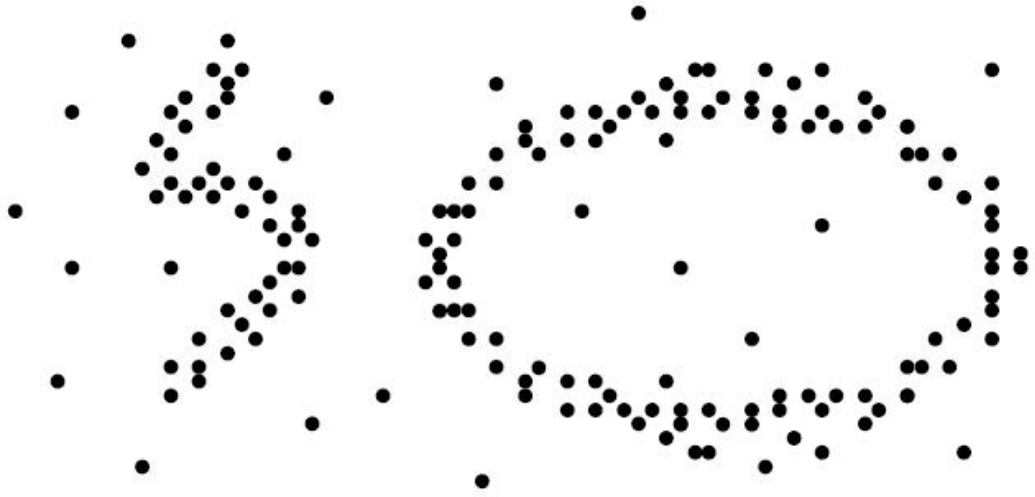
To find clusters of arbitrary shape, alternatively, we can model clusters as dense regions in the data space, separated by sparse regions. This is the main strategy behind *density-based clustering methods*, which can discover clusters of nonspherical shape.

DBSCAN: Density-Based Clustering Based on Connected Regions with High Density

"How can we find dense regions in density-based clustering?" The *density* of an object ***o*** can be measured by the number of objects close to ***o***. **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) finds *core objects*, that is, objects that have dense neighborhoods. It connects core objects and their neighborhoods to form dense regions as clusters.

"How does **DBSCAN** *quantify the neighborhood of an object?"* A user-specified parameter >0 is used to specify the radius of a neighborhood we consider for every object. The €-**neighborhood** of an object ***o*** is the space within a radius € centered at ***o***.

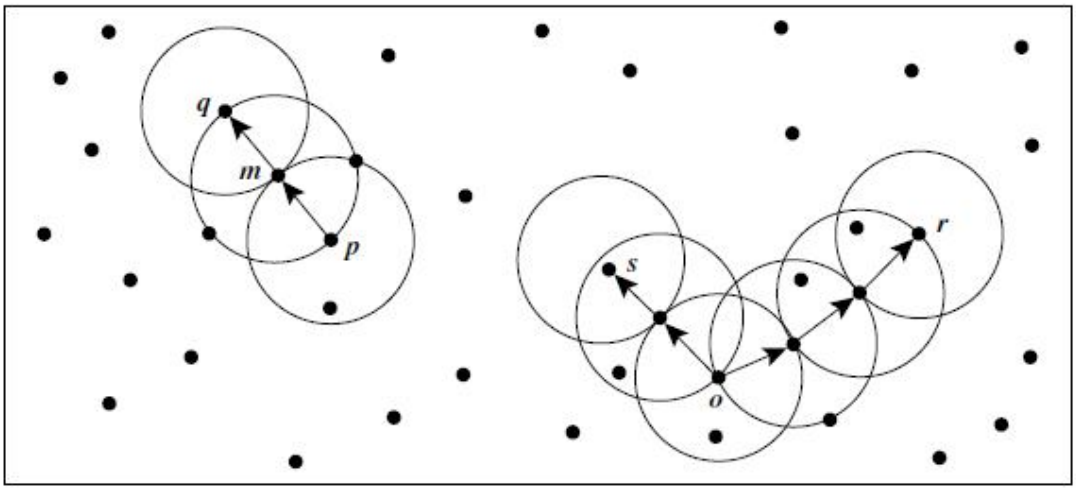
MinPts, which specifies the density threshold of dense regions. An object is a **core object** if the €_-neighborhood of the object contains at least *MinPts* objects. Core objects are the pillars of dense regions.



Clusters of arbitrary shape.

Given a set, *D*, of objects, we can identify all core objects with respect to the given parameters, € and *MinPts*. The clustering task is therein reduced to using core objects and their neighborhoods to form dense regions, where the dense regions are clusters. For a core object ***q*** and an object ***p***, we say that ***p*** is **directly density-reachable** from ***q*** (with respect to € and *MinPts*) if ***p*** is within the €-neighborhood of ***q***. Clearly, an object ***p*** is directly density-reachable from another object ***q*** if and only if ***q*** is a core object and ***p*** is in the €-

neighborhood of ***q***. Using the directly density-reachable relation, a core object can "bring" all objects from its €-neighborhood into a dense region.



Example :Density-reachability and density-connectivity. Consider Figure for a given _ represented by the radius of the circles, and, say, let *MinPts* D 3. Of the labeled points, ***m***,***p***,***o***, ***r*** are core objects because each is in an €-neighborhood containing at least three points. Object ***q*** is directly density-reachable from ***m***. Object ***m*** is directly density-reachable from ***p*** and vice versa.

Object ***q*** is (indirectly) density-reachable from ***p*** because ***q*** is directly density reachable from ***m*** and ***m*** is directly density-reachable from ***p***. However, ***p*** is not density reachable from ***q*** because ***q*** is not a core object. Similarly, ***r*** and ***s*** are density-reachable from ***o*** and ***o*** is density-reachable from ***r***. Thus, ***o***, ***r***, and ***s*** are all density-connected

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- *D*: a data set containing *n* objects,
- €: the radius parameter, and
- *MinPts*: the neighborhood density threshold.

Output: A set of density-based clusters.

Method:

(1) mark all objects as unvisited;

(2) **do**

(3) randomly select an unvisited object ***p***;

(4) mark ***p*** as visited;
(5) **if** the €-neighborhood of ***p*** has at least *MinPts* objects
(6) create a new cluster *C*, and add ***p*** to *C*;
(7) let *N* be the set of objects in the €-neighborhood of ***p***;
(8) **for** each point ***p***0 in *N*
(9) if $\boldsymbol{p}^1$ is unvisited
(10) mark ***p***0 as visited;
(11) if the €-neighborhood of ***p***0 has at least *MinPts* points, add those points to *N*;
(12) if $\boldsymbol{p}^1$ is not yet a member of any cluster, add ***p***0 to *C*;
(13) **end for**
(14) output *C*;
(15) **else** mark ***p*** as noise;
(16) **until** no object is unvisited;

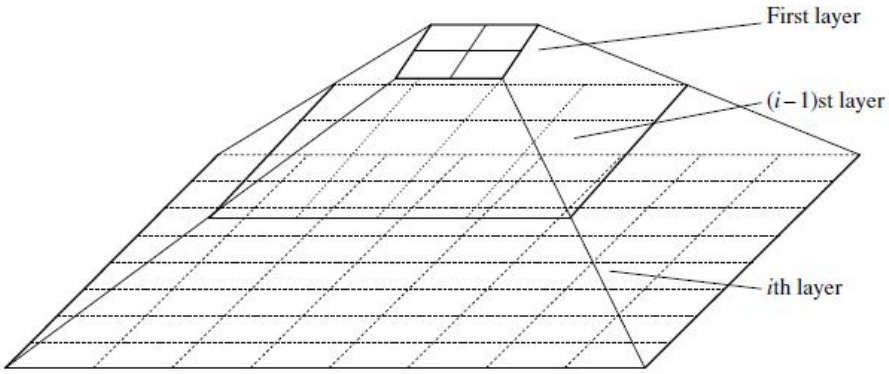
Grid-Based Methods

The clustering methods discussed so far are data-driven—they partition the set of objects and adapt to the distribution of the objects in the embedding space. Alternatively, a **grid-based clustering** method takes a space-driven approach by partitioning the embedding space into *cells* independent of the distribution of the input objects.

The *grid-based clustering* approach uses a multi resolution grid data structure. It quantizes the object space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed

STING: STatistical INformation Grid

STING is a grid-based multiresolution clustering technique in which the embedding spatial area of the input objects is divided into rectangular cells. The space can be divided in a hierarchical and recursive way. Several levels of such rectangular cells correspond to different levels of resolution and form a hierarchical structure: Each cell at a high level is partitioned to form a number of cells at the next lower level. Statistical information regarding the attributes in each grid cell, such as the mean, maximum, and minimum values, is pre computed and stored as *statistical parameters*. These statistical parameters are useful for query processing and for other data analysis tasks.



The statistical parameters of higher-level cells can easily be computed from the parameters of the lower-level cells. These parameters include the following: the attribute-independent parameter, *count*; and the attribute-dependent parameters, *mean*, *stdev* (standard deviation), *min* (minimum), *max* (maximum), and the type of *distribution* that the attribute value in the cell follows such as *normal*, *uniform*, *exponential*, or *none* (if the distribution is unknown).

the attribute is a selected measure for analysis such as *price* for house objects. When the data are loaded into the database, the parameters *count*, *mean*, *stdev*, *min*, and *max* of the bottom-level cells are calculated directly from the data. The value of *distribution* may either be assigned by the user if the distribution type is known beforehand or obtained by hypothesis tests such as the _2 test. The type of distribution of a higher-level cell can be computed based on the majority of distribution types of its corresponding lower-level cells in conjunction with a threshold filtering process.

"How is this statistical information useful for query answering?" The statistical parameters can be used in a top-down, grid-based manner as follows. First, a layer within the hierarchical structure is determined from which the query-answering process is to start. This layer typically contains a small number of cells.

Advantages of STING

STING offers several advantages:

(1) the grid-based computation is *query-independent* because the statistical information stored in each cell represents the summary information of the data in the grid cell, independent of the query;

(2) the grid structure facilitates parallel processing and incremental updating; and

(3) the method's efficiency is a major advantage: STING goes through the database once to compute the statistical parameters of the cells,

SUBJECTIVE QUESTIONS

1. Given the two objects represented by the tuples (22, 1, 42, 10) and (20, 0, 36, 8);

   (i) Compute the Euclidean distance between the two objects.
   (ii) Compute the Manhattan distance between the two objects.
   (iii) Compute the Murkowski distance between the two objects, using q=3.

2. Describe clustering with an example.

3. List the classical partitioning methods and explain them briefly

4. Illustrate with suitable example k-means clustering algorithm. What are its advantages and disadvantages?

5. You are to cluster eight points: x1=(2,10), x2=(2,5), x3=(8,4), x4=(5,8), x5=(7,5), x6=(6,4), x7=(1,2) and x8=(4,9). Suppose, you assigned x1, x4 and x7 as initial cluster centres for Kmeans clustering(k= 3). Using K-means with the Manhattan distance, compute the three clusters for each round of the algorithm until convergence.

6. What is the difference between k-means and k-medoids algorithms? Explain your answer with an example.

7. State the strengths and weaknesses of k-means clustering algorithm.

8. What is cluster analysis? Briefly explain different types of clustering methods.

9. Explain briefly agglomerative hierarchical clustering with an example

10. Distinguish between agglomerative and divisive hierarchical clustering.

11. What is density based clustering? Explain DBSCAN algorithm.

12. What is grid based clustering? Explain STING algorithm.