

Data Mining

— Cluster Analysis —

What is Cluster Analysis?

- Cluster: a collection of data objects
 - Similar to one another within the same cluster
 - Dissimilar to the objects in other clusters
- Cluster analysis
 - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- **Unsupervised learning**: no predefined classes

Clustering Applications

- **Business Intelligence:** To organize a large no of customers into groups.
- **Image Pattern Recognition:** handwritten character recognition systems to improve overall recognition accuracy using multiple models based on multiple subclasses/clusters.
- **Web Search:** To organize the relevant web pages in a concise and easily accessible way.
- **Biology:** It can be used for classification among different species of plants and animals.
- **Information Retrieval:** cluster documents into topics
- **Marketing:** It can be used to characterize & discover customer segments for marketing purposes.

Examples of Clustering Applications

- **Libraries:** It is used in clustering different books on the basis of topics and information.
- **Insurance:** Identifying groups of motor insurance policy holders with a high average claim cost and identifying the frauds.
- As a DM function, clustering can be used
 - As a **stand-alone tool** to get insight into data distribution
 - As a **preprocessing step** for other algorithms
- Outlier detection

Requirements of Clustering in Data Mining

- Scalability:
 - Clustering on only a small sample of a large Dataset can lead to biased results.
- Ability to deal with different types of attributes:
 - Recently many applications need clustering on complex data types like graph, sequences, images and documents.
- Discovery of clusters with arbitrary shape
 - Clusters based on ED and MD are spherical in shape.
- Requirements to provide domain knowledge in the form of input parameters:
 - Hard to determine.(high dimensionality DS and no deep understanding)
 - makes quality of the clustering difficult to control.
 - Clustering results are sensitive to those parameters.

Requirements of Clustering in Data Mining

- Able to deal with noisy data
 - Clustering algorithms are sensitive to noise and may produce low quality clusters.
- Incremental clustering and insensitivity to input data order.
 - Incorporate incremental updates into existing clustering structures
- Capability of clustering high-dimensional data
 - High dimensional data can be very sparse and highly skewed.
- Incorporation of user-specified constraints
 - Clustering algorithms should be able to find clusters that satisfy the specified constraints.
- Interpretability and usability

Major Clustering Approaches

- Partitioning approach:
 - Given a set of n objects, a partitioning method constructs k partitions of the data, where each partition represents a cluster and $k \leq n$.
 - The basic partitioning methods typically adopt ***exclusive cluster separation***
 - Most partitioning methods are distance-based.
 - Given k , this approach creates initial partitioning and then uses Iterative relocation technique to improve it.
 - The general criterion of a good partitioning is that objects in the same cluster are “close” or related to each other, whereas objects in different clusters are “far apart” or very different .
 - Typical methods: k-means, k-medoids, CLARANS

The *K-Means* Clustering Method

- A centroid-based partitioning technique uses the **centroid** of a cluster, C_i , to represent that cluster.
- 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

$$E = \sum_{i=1}^k \sum_{p \in C_i} \text{dist}(\mathbf{p}, \mathbf{c}_i)^2,$$

where E is the sum of the squared error for all objects in the data set; \mathbf{p} is the point in space representing a given object; and \mathbf{c}_i is the centroid of cluster C_i (both \mathbf{p} and \mathbf{c}_i are multidimensional).

- This objective function tries to make the resulting k clusters as compact and as separate as possible.
- Optimizing the within-cluster variation is computationally challenging.
- To overcome the prohibitive computational cost for the exact solution, greedy
* approaches are often used in practice. e.g k-means algorithm

The *K-Means* Clustering Method

- Given k , the *k-means* algorithm is implemented as below :
 - First, it randomly selects k of the objects in D , each of which initially represents a cluster mean or cluster center.
 - For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the Euclidean distance between the object and the cluster mean.
 - The *k-means* algorithm then iteratively improves the within-cluster variation.
 - For each cluster, it computes the new mean using the objects assigned to the cluster in the previous iteration.
 - All the objects are then reassigned using the updated means as the new cluster centers.
 - The iterations continue until the assignment is stable, that is, the clusters formed in the current round are the same as those formed in the previous round.

The *K-Means* Clustering Method (5 steps)

Step 1: Choose the no.of clusters ,k.

Step2: Select at random k points, the centroids(not necessarily from your dataset)

Step3: Assign each data point to the closest centroid as that forms k clusters.

Step4: Compute and place new centroid of each cluster.

Step5: Reassign each data point to the new closest centroid . If any reassignment takes place go to step 4 . Otherwise (if the assignment is stable, i.e. the clusters formed in the current round are the same as those formed in the previous round) then STOP.

- The process of iteratively re assigning objects to clusters to improve the partitioning is referred to as ***iterative relocation technique***.

K- means Algorithm

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.

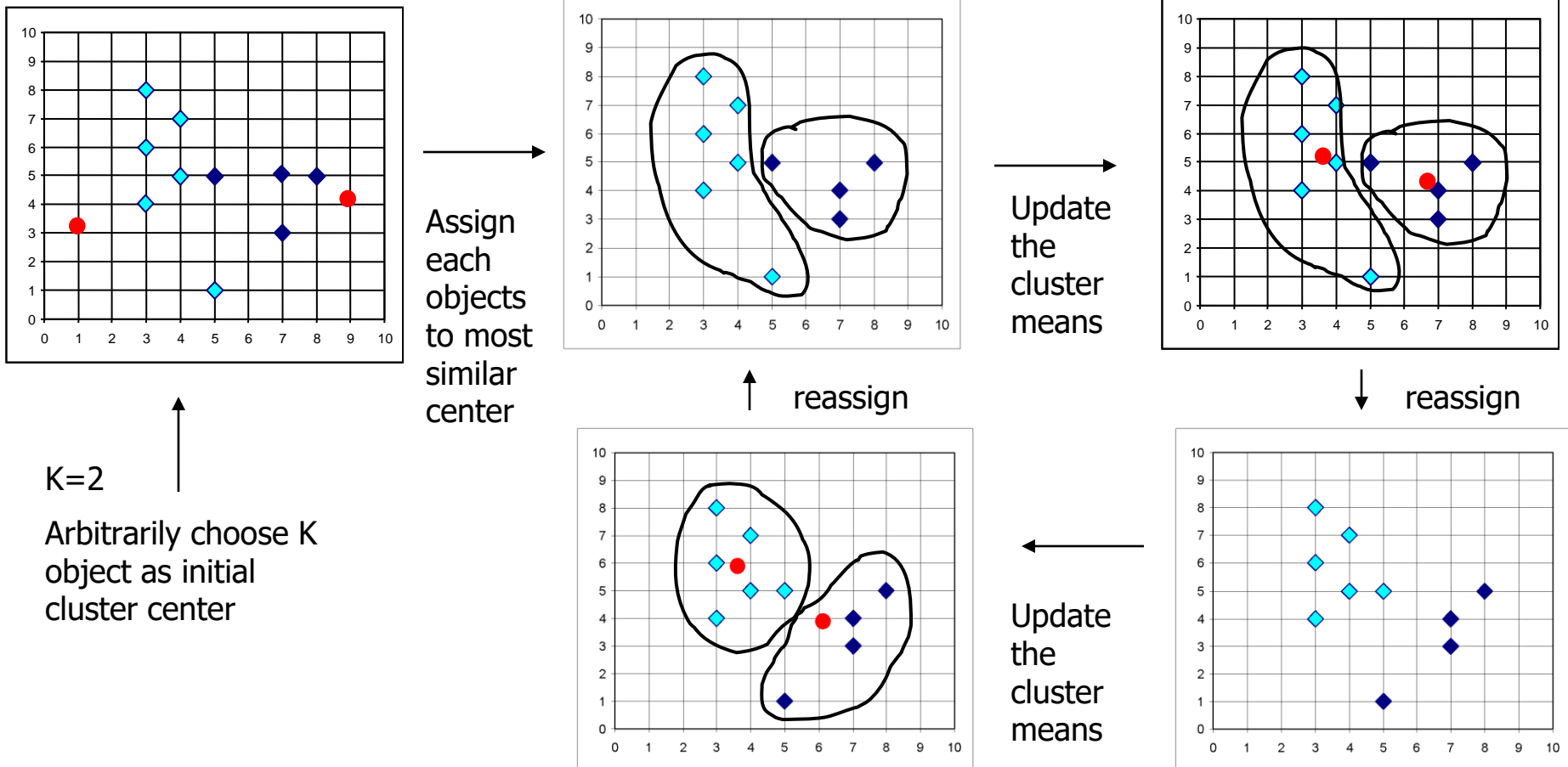
Output: A set of k clusters.

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* Clustering Method

■ Example



Example

- Suppose that the data mining task is to cluster points (with (x, y) representing location) into three clusters, where the points are $A1(2, 10), A2(2, 5), A3(8, 4), B1(5, 8), B2(7, 5), B3(6, 4), C1(1, 2), C2(4, 9)$. The distance function is Euclidean distance. Suppose initially we assign $A1$, $B1$, and $C1$ as the center of each cluster, respectively. Use the *k-means* algorithm to show *only*
 - (a) the three cluster centers after the first round of execution.
 - (b) the final three clusters.

Solution

- Suppose that the data mining task is to cluster points (with (x, y) representing location) into three clusters, where the points are $A1(2, 10), A2(2, 5), A3(8, 4), B1(5, 8), B2(7, 5), B3(6, 4), C1(1, 2), C2(4, 9)$. The distance function is Euclidean distance. Suppose initially we assign $A1, B1$, and $C1$ as the center of each cluster, respectively. Use the *k-means* algorithm to show *only*
 - (a) the three cluster centers after the first round of execution.

Answer:

After the first round, the three new clusters are: (1) $\{A1\}$, (2) $\{B1, A3, B2, B3, C2\}$, (3) $\{C1, A2\}$, and their centers are (1) $(2, 10)$, (2) $(6, 6)$, (3) $(1.5, 3.5)$.

(b) the final three clusters.

Answer:

The final three clusters are: (1) $\{A1, C2, B1\}$, (2) $\{A3, B2, B3\}$, (3) $\{C1, A2\}$

Comments on the *K-Means* Method

- Strength: *Relatively efficient*: $O(nkt)$, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t \ll n$. Therefore, the method is relatively scalable and efficient in processing large data sets.

- Weakness:
 - The k -means method is not guaranteed to converge to the global optimum and often terminates at a local optimum
 - The results may depend on the initial random selection of cluster centres.
 - Applicable only when *mean* is defined, then what about categorical data?
 - Need to specify k , the *number* of clusters, in advance.
 - Unable to handle noisy data and *outliers*
 - A small no. of such data can substantially influence the mean value.
 - Not suitable to discover clusters with *non-convex shapes* or clusters of very different size.

Variations of *K-Means* Method

- Handling categorical data: ***k-modes*** (Huang'98)
 - Pick K observations at random and use them as leaders/clusters
 - Calculate the dissimilarities and assign each observation to its closest cluster
 - Define new modes for the clusters
 - Repeat 2–3 steps until there is no re-assignment required
- Handling mixture of numeric and categorical data: ***k-prototype method***

K-modes clustering Example

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

k=3.

Step 1: Lets select P1,P7 and P8 as leaders/cluster

Step2 : Calculate the dissimilarities(no. of mismatches) and assign each observation to its closest cluster

Comparing leader/Cluster P1 to the observation P1 gives 0 dissimilarities.

Comparing leader/cluster P1 to the observation P2 gives 3(1+1+1) dissimilarities.

Likewise, calculate all the dissimilarities and put them in a matrix as shown below and assign the observations to their closest cluster(cluster that has the least dissimilarity)

	Cluster 1 (P1)	Cluster 2 (P7)	Cluster 3 (P8)	Cluster
P1	0 ✓	2	2	Cluster 1
P2	3 ✓	3	3	Cluster 1
P3	3	1 ✓	3	Cluster 2
P4	3	3	1 ✓	Cluster 3
P5	1 ✓	2	2	Cluster 1
P6	3	3	2 ✓	Cluster 3
P7	2	0 ✓	2	Cluster 2
P8	2	2	0 ✓	Cluster 3

K-modes clustering Example(Cntd..)

Step 3: Define new modes for the clusters (Mode is simply the most observed value.)

Mark the observations according to the cluster they belong to. K1: Yellow, Brick Red: K2, Purple: K3

person	hair color	eye color	skin color
P1	blonde	amber	fair
P2	brunette	gray	brown
P3	red	green	brown
P4	black	hazel	brown
P5	brunette	amber	fair
P6	black	gray	brown
P7	red	green	fair
P8	black	hazel	fair

Considering one cluster at a time, for each feature, look for the Mode and update the new leaders

Cluster 1 observations(P1, P2, P5) has brunette as the most observed hair color, amber as the most observed eye color, and fair as the most observed skin color.

Note: If you observe the same occurrence of values, take the mode randomly. In our case, the observations of Cluster 2(P3, P7) have one occurrence of brown, fair skin color. I randomly chose fair as the mode.

New Leaders			
	hair color	eye color	skin color
Cluster 1	brunette	amber	fair
Cluster 2	red	green	fair
Cluster 3	black	hazel	brown

Repeat steps 2–3

K-modes clustering Example(Cntd..)

	Cluster 1	Cluster 2	Cluster 3	Cluster
P1	1 ✓	2	3	Cluster 1
P2	2 ✓	3	2	Cluster 1
P3	3	1 ✓	2	Cluster 2
P4	3	3	0 ✓	Cluster 3
P5	0 ✓	2	3	Cluster 1
P6	3	3	1 ✓	Cluster 3
P7	2	0 ✓	3	Cluster 2
P8	2	2	1 ✓	Cluster 3

We stop here as we see there is no change in the assignment of observations.

K-Medoids Algorithm

- **K-Medoids:** Instead of taking the mean value of the objects in a cluster as a reference point, we can pick actual objects to represent the clusters, using one representative object per cluster (Medoid).
- Each remaining object is assigned to the cluster of which the representative object is the most similar.
- The partitioning method is then performed based on the principle of **minimizing the sum of the dissimilarities** between each object p and its corresponding representative object.
- An **absolute-error criterion** is defined as

$$E = \sum_{i=1}^k \sum_{p \in C_i} \text{dist}(p, o_i),$$

where E is the sum of the absolute error for all objects p in the data set, and 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

K-medoids, PAM

- A medoid of a finite dataset is a data point from this set, whose average dissimilarity to all the data points is minimal i.e. it is the most centrally located point in the set.

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;

Example: K-medoids, PAM

Given $D=\{1,2,6,7,8,10,15,17,20\}$ Form 3 clusters using k-medoid algorithm.
Assume initial medoids as 6,7 8 respectively for K1,K2,and K3.

What Is the Problem with PAM?

- Adv: PAM is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean
- Disadv: PAM works efficiently for small data sets but does not **scale well** for large data sets.
 - $O(k(n-k)^2)$ for each iteration : very costly computation for large n and k where n is # of data objects and k is # of clusters
- To deal with large data sets, a sampling based method called CLARA can be used.

CLARA

- **CLARA** (Clustering Large Applications), is an extension to **k-medoids (PAM)** methods to deal with data containing a large number of objects (more than several thousand observations) in order to reduce computing time and RAM storage problem. This is achieved using the sampling approach.
- Instead of finding medoids for the entire data set, CLARA considers a small sample of the data with fixed size (*sampsize*) and applies the PAM algorithm to generate an optimal set of medoids for the sample.
- The quality of resulting medoids is measured by the average dissimilarity between every object in the entire data set and the medoid of its cluster, defined as the cost function.
- CLARA repeats the sampling and clustering processes for a pre-specified number of times in order to minimize the sampling bias. The final clustering results correspond to the set of medoids with the minimal cost.

CLARA Algorithm

1. Create randomly, from the original dataset, multiple subsets with fixed size (sample size)
2. Compute PAM algorithm on each subset and choose the corresponding k representative objects (medoids). Assign each observation of the entire data set to the closest medoid.
3. Calculate the mean (or the sum) of the dissimilarities of the observations to their closest medoid. This is used as a measure of the goodness of the clustering.
4. Retain the sub-dataset for which the mean (or sum) is minimal. A further analysis is carried out on the final partition.

CLARA

- The complexity of computing the medoids on a random sample is $O(ks^2 + k.(n-k))$, where s is the size of the sample, k is the number of clusters, and n is the total number of objects.
- The effectiveness of CLARA depends on the sample size.
- If an object is one of the best k -medoids but is not selected during sampling, CLARA will never find the best clustering.(Example to demonstrate the same)

CLARANS

- A randomized algorithm called **CLARANS** (**C**lustering **L**arge **A**pplications based upon **RAN**domized **S**earch) presents a trade-off between the cost and the effectiveness of using samples to obtain clustering.

Steps of CLARANS algorithm

1. Select 'k' random data points and label them as medoids for the time being.
2. Select a random point say 'a' from the points picked in step (1), and another point say 'b' which is not included in those points.
3. We would already have the sum of distances of point 'a' from all other points since that computation is required for selecting the points in step (1). Perform similar computation for point 'b'.
4. If the sum of distances from all other points for point 'b' turns out to be less than that for point 'a', replace 'a' by 'b'.
5. The algorithm performs such a randomized search of medoids 'x' times where 'x' denotes the number of local minima computed, i.e. number of iterations to be performed, which we specify as a parameter. The set of medoids obtained after such 'x' number of steps is termed as 'local optimum'.
6. A counter is incremented every time a replacement of points is made. The process of examining the points for possible replacement is repeated till the counter does not exceed the maximum number of neighbors to be examined (specified as a parameter).
7. The set of medoids obtained when the algorithm stops is the best local optimum

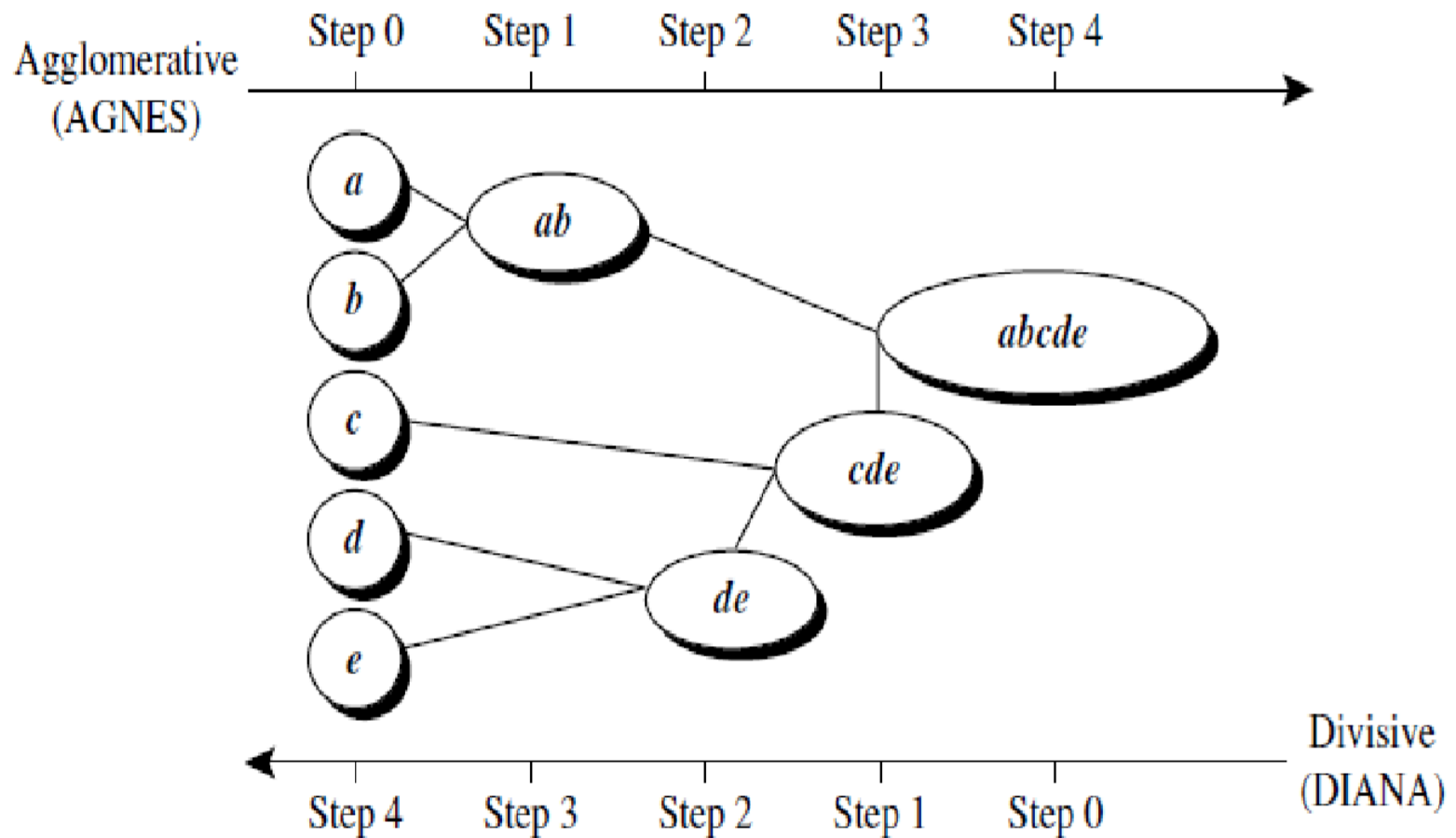
Comparison between CLARA & CLARANS

- Like CLARA, CLARANS does not check every neighbors of a node.
- Unlike CLARA, CLARANS does not restrict its search to a particular graph. It search the original graph $G_{n,k}$.
- CLARA draws a sample of *nodes* at the beginning of a search, CLARANS draws a sample of *neighbors* in each step of a search.
- CLARANS gives higher quality clustering than CLARA.
- CLARANS requires a very small number of searches than CLARA.

Hierarchical Clustering

- A **hierarchical clustering method** works by partitioning data objects into groups at different levels like a hierarchy or “tree” of clusters.
- Representing data objects in the form of a hierarchy is useful for data summarization and visualization.
- If necessary, hierarchical partitioning can be continued recursively until a desired granularity is reached.

Example : Application of AGNES(AGglomerative NESTing and DIANA(DIvisible ANAlysis)

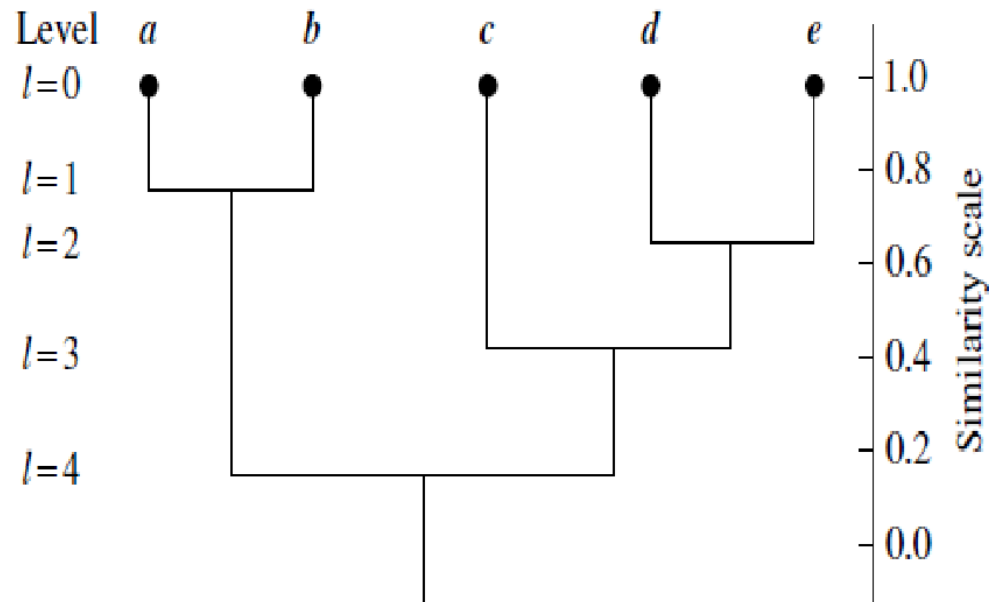


Hierarchical Clustering

(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.
- An **agglomerative hierarchical clustering method** :
 - The hierarchical decomposition is formed in a bottom-up (merging) way
- A **divisive hierarchical clustering method**
 - The hierarchical decomposition is formed in a top-down (splitting) way
- In either agglomerative or divisive hierarchical clustering, a user can specify the desired number of clusters as a termination condition.
- Hierarchical methods suffer from the fact that once a step (merge or split) is done, **it can never be undone**. So these techniques cannot correct erroneous decisions.

Hierarchical Clustering: Dendrogram



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

Divisive Method Challenges

- How to partition a large cluster into smaller ones is a challenge.
 - $(2^{n-1}-1)$ ways to partition n objects into 2 exclusive clusters.
 - For large n , it is computationally inhibitive to examine all possibilities .
- Also a Divisive method typically uses heuristics in partitioning which can lead to inaccurate results.
- For the sake of efficiency divisive methods typically do not backtrack on the partitioning decisions that have been made.
- Due to these challenges, Agglomerative clustering is mostly used than divisive clustering.

Distance Measures(*linkage measures*) in Agglomerative Method

- $|p - p'|$ is the distance between two objects or points, p and p' ;
- m_i is the mean for cluster, C_i ;
- n_i is the number of objects in C_i .

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(*linkage measures*) in Algorithmic 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**.
 - Here if the clustering process is terminated when the minimum distance between nearest clusters exceeds a user-defined threshold, it is called a **single-linkage algorithm**.
- 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**.
 - Here, 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**

Example: Agglomerative Clustering (Single Linkage)

	P_1	P_2	P_3	P_4	P_5
P_1	0				
P_2	9	0			
P_3	3	7	0		
P_4	6	5	9	0	
P_5	11	10	2	8	0



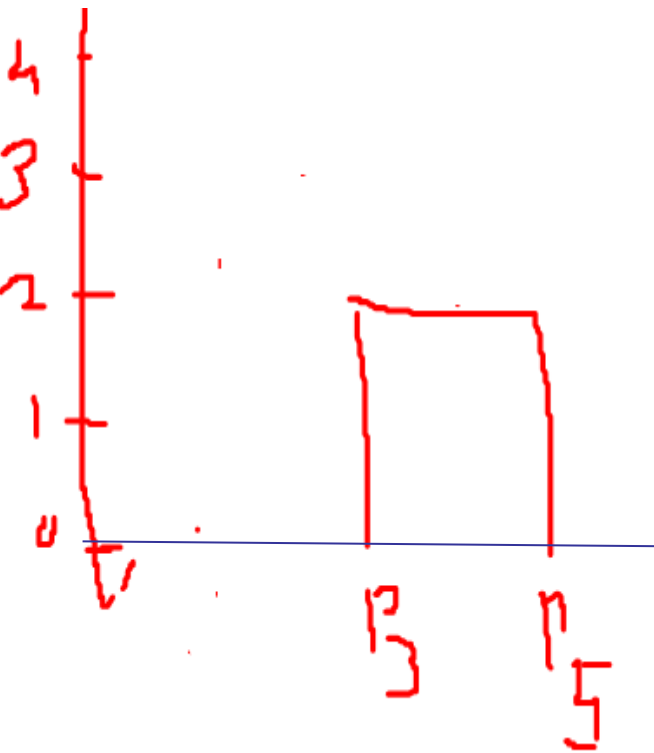
	P_1	P_2	$[P_3, P_5]$	P_4
P_1	0			
P_2	9	0		
$[P_3, P_5]$	3	7	0	
P_4	6	5	8	0

$$\begin{aligned} &\Rightarrow d(P_2, [P_3, P_5]) \\ &\Rightarrow \min(d(P_2, P_3), d(P_2, P_5)) \\ &\Rightarrow \min(7, 10) \Rightarrow 7 \\ &\Rightarrow d(P_4, [P_3, P_5]) \\ &\Rightarrow \min(d(P_4, P_3), d(P_4, P_5)) \\ &\Rightarrow \min(9, 8) \Rightarrow 8 \end{aligned}$$

$$\begin{aligned} &\Rightarrow d(P_1, [P_3, P_5]) \\ &\Rightarrow \min(d(P_1, P_3), d(P_1, P_5)) \\ &\Rightarrow \min(3, 11) \Rightarrow 3 \end{aligned}$$

As 3 is minimum, combine $[P_3, P_5]$ and $[P_1]$

Dendrogram



Step 1



Step 2

	$[P_1 P_3 P_5]$	P_2	P_4
$[P_1 P_3 P_5]$	0		
P_2	7	0	
P_4	6	5	0

$$d(P_2, [P_1 P_3 P_5])$$

$$\Rightarrow \min(d(P_2, P_1), d(P_2, P_3), d(P_2, P_5))$$

$$\Rightarrow \min(9, 7, 10) \Rightarrow 7$$

Combine $[P_2]$ and $[P_4]$

	$[P_1 P_3 P_5]$	$[P_2 P_4]$
$[P_1 P_3 P_5]$	0	
$[P_2 P_4]$	6	0

$$d([P_1 P_3 P_5], [P_2 P_4])$$

combine $[P_1, P_3, P_5]$ and $[P_2, P_4]$ into one cluster

Dendrogram for the Example



Complete Linkage

	P_1	P_2	P_3	P_4	P_5
P_1	0				
P_2	9	0			
P_3	3	7	0		
P_4	6	5	9	0	
P_5	11	10	2	8	0

Combine $[P_3]$ and $[P_5]$



	P_1	P_2	$[P_3 P_5]$	P_4
P_1	0			
P_2	9	0		
$[P_3 P_5]$	11	10	0	
P_4	6	5	9	0



$$\begin{aligned}
 & d(P_2, [P_3 P_5]) \\
 & \Rightarrow \max(d(P_2, P_3), d(P_2, P_5)) \Rightarrow \max(7, 10) = 10 \\
 & d(P_1, [P_3 P_5]) \\
 & \Rightarrow \max(d(P_1, P_3), d(P_1, P_5)) \Rightarrow \max(3, 11) = 11 \\
 & d(P_4, [P_3 P_5]) \\
 & \Rightarrow \max(d(P_4, P_3), d(P_4, P_5)) \Rightarrow \max(9, 8) = 9
 \end{aligned}$$

Combine $[P_4]$ and $[P_2]$

Agglomerative Clustering(Complete Linkage)

	P_1	$[P_2 P_4]$	$[P_3 P_5]$
P_1			
$[P_2 P_4]$	(9)		
$[P_3 P_5]$	11	10	



	$[P_1 P_2 P_4]$	$[P_3 P_5]$
$[P_1 P_2 P_4]$		
$[P_3 P_5]$	(11)	

Combine $[P_2, P_4]$ and $[P_1]$

Combine $[P_1, P_2, P_4]$ and $[P_3, P_5]$ to form a single cluster



Dendrogram

BIRCH (1996)

- Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is designed for clustering a large amount of **numeric** data by integrating hierarchical clustering (at the initial *microclustering* stage) and other clustering methods such as iterative partitioning (at the later *macroclustering* stage).
- Small summary of large dataset.
- It overcomes the two difficulties in agglomerative clustering methods: (1) scalability and (2) the inability to undo what was done in the previous step.
- BIRCH uses the notions of *clustering feature* to summarize a cluster, and *clustering feature tree (CF-tree)* to represent a cluster hierarchy.
- These structures help the clustering method to achieve good speed and scalability in large DS and also make it effective for incremental and dynamic clustering of incoming data objects.

BIRCH (1996)

- A clustering feature is essentially a summary of the statistics for the given cluster.
- The **clustering feature (CF)** of the cluster is a 3-D vector summarizing information about clusters of objects.
 - $CF = (n, LS, SS)$ where
 - LS is the linear sum of the n points i.e., $\sum_{i=1}^n x_i$
 - SS is the square sum of the data points i.e., $\sum_{i=1}^n x_i^2$
- Using a clustering feature, we can easily derive many useful statistics of a cluster.
 - For example, the cluster's centroid, $\mathbf{x_0}$, radius, R , and diameter, D , are

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

Here, R is the average distance from member objects to the centroid, and D is the average pairwise distance within a cluster

Example

Clustering feature. Suppose there are three points, $(2,5)$, $(3,2)$, and $(4,3)$, in a cluster, C_1 . The clustering feature of C_1 is

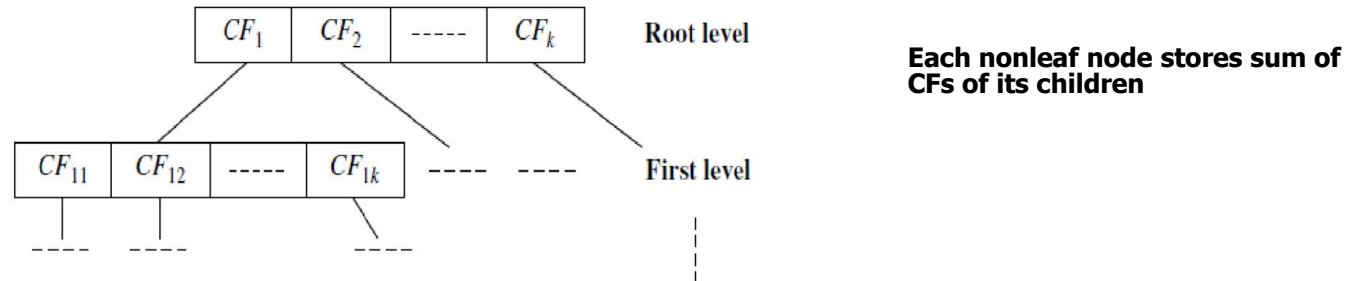
$$CF_1 = \langle 3, (2+3+4, 5+2+3), (2^2+3^2+4^2, 5^2+2^2+3^2) \rangle = \langle 3, (9, 10), (29, 38) \rangle.$$

Suppose that C_1 is disjoint to a second cluster, C_2 , where $CF_2 = \langle 3, (35, 36), (417, 440) \rangle$. 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,

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

Clustering Feature Tree (CF Tree)

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



CF-tree structure.

- A CF-tree has two parameters:
 1. *Branching factor, B* , specifies the maximum number of children per nonleaf node.
 2. *The threshold parameter T* , specifies the maximum diameter of subclusters stored at the leaf nodes of the tree.
- These two parameters implicitly control the resulting tree's size.

Clustering Feature Tree (CF Tree)

- BIRCH applies a *multiphase* clustering technique:
- 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.

Clustering Feature Tree (CF Tree)

- *"How effective is BIRCH?"*
 - 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.
- Drawbacks:
 - 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.
 - It can process only the metric attributes.
- **Implementation of BIRCH in Python: [Link](#)**

CF tree is built dynamically as objects are inserted.

- For each point in the input
 - Find closest leaf entry
 - Add point to leaf entry and update CF
 - If entry diameter $>$ max_diameter, then split leaf, and possibly parents

Example

Let Have Following Data

$X_1=(3,4)$, $x_2=(2,6)$, $x_3=(4,5)$, $x_4=(4,7)$, $x_5=(3,8)$, $x_6=(6,2)$, $x_7=(7,2)$, $x_8=(7,4)$, $x_9=(8,4)$, $x_{10}=(7,9)$

Cluster the Above Data Using BIRCH Algorithm, considering $T < 1.5$, and Max Branch = 2

For each Data Point we need to evaluate Radius and Cluster Feature:

->Consider Data Point (3,4):

As it is alone in the Feature map, Hence

1. Radius = 0

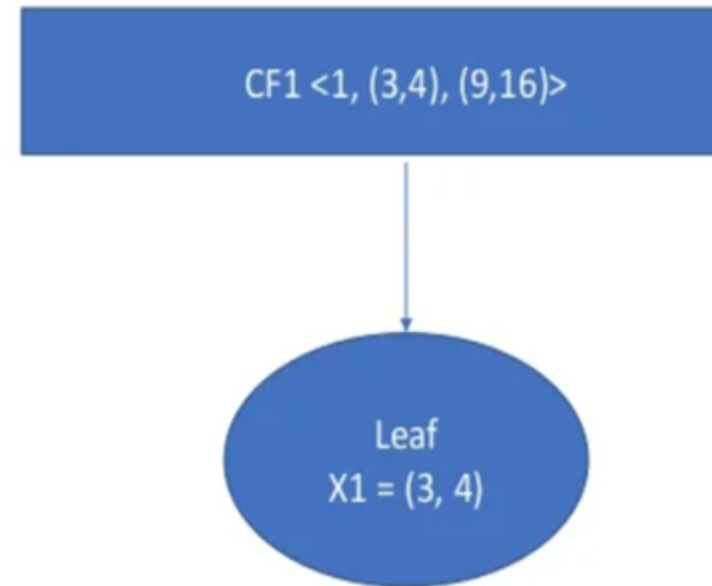
2. Cluster Feature $CF_1 < N, LS, SS >$

$N = 1$ as there is now one data point under consideration.

$LS = \text{Sum of Data Point under consideration} = (3,4)$

$SS = \text{Square Sum of Data Point Under Consideration}$
 $= (3^2, 4^2) = (9, 16)$

3. Now construct the Leaf with Data Point X_1 and Branch as CF_1 .



->Consider Data Point $x_2 = (2,6)$:

1. Linear Sum $LS = (3,4) + (2,6) = (5,10)$

2. Square Sum $SS = (3^2+2^2, 4^2+6^2) = (13, 52)$

Now Evaluate Radius considering $N=2$

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(13,52)-(5,10)^2/2}{2}} = \sqrt{\frac{(13,52)-(25,100)/2}{2}} =$$

$$\sqrt{\frac{(13,52)-(12.5,50)}{2}} = \sqrt{(6.5,26) - (6.25,25)} = \sqrt{(0.25,1)} = (0.5, 1) < T \text{ As}$$

$(0.25,1) < (T, T)$, hence x_2 will cluster with Leaf x_1 .

2. Cluster Feature $CF_1 <N, LS, SS> = <2,(5,10),(13,52)>$

$N = 2$ as there is now two data point under CF_1 .

$$LS = (3,4) + (2,6) = (5,10)$$

$$SS = (3^2+2^2, 4^2+6^2) = (13, 52)$$

$CF_1 <1, (5,10), (13,52)>$

Leaf

$x_1 = (3, 4),$

$x_2 = (2,6)$

->Consider Data Point $x_3 = (4,5)$ on CF1:

1. Linear Sum $LS = (4,5) + (5,10) = (9,15)$

2. Square Sum $SS = (4^2+13, 5^2+52) = (29, 77)$

Now Evaluate Radius considering $N=3$

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(29,77)-(9,15)^2/3}{3}} = (0.47, 0.4714) < T$$

As $(0.47, 0.471) < (T, T)$, hence X_3 will cluster with Leaf (X_1, x_2) .

2. Cluster Feature $CF1 <N, LS, SS> = <3,(9,15),(29,77)>$

$N = 3$ as there is now Three data point under $CF1$.

$LS = (4,5) + (5,10) = (9,15)$

$SS = (4^2+13, 5^2+52) = (29, 77)$

$CF1 <1, (9,15), (29,77)>$

Leaf

$X_1 = (3, 4),$

$X_2 = (2,6),$

$X_3 = (4,5)$

->Consider Data Point $x_4 = (4,7)$ on CF1:

1. Linear Sum $LS = (4,7) + (9,15) = (13,22)$
2. Square Sum $SS = (4^2+29, 7^2 + 77) = (45, 126)$

Now Evaluate Radius considering $N=4$

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(45,126)-(13,22)^2/4}{4}} = (0.41, 0.55)$$

As $(0.41, 0.55) < (T, T)$, hence X_4 will cluster with Leaf (X_1, x_2, x_3) .

2. Cluster Feature $CF1 <N, LS, SS> = <4,(13,22),(45,126)>$

$N = 4$ as there is now four data point under $CF1$.

$$LS = (4,7) + (9,15) = (13,22)$$

$$SS = (4^2+29, 7^2 + 77) = (45, 126)$$

$CF1 <1, (13,22), (45,126)>$

Leaf

$x_1 = (3, 4),$

$x_2 = (2,6),$

$x_3 = (4,5),$

$x_4 = (4,7)$

->Consider Data Point $x_5 = (3,8)$ on CF1:

1. Linear Sum $LS = (3,8) + (13,22) = (16,30)$

2. Square Sum $SS = (3^2+45, 8^2+126) = (54, 190)$

Now Evaluate Radius considering $N=5$

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(54,190)-(16,30)^2/5}{5}} = (0.33, 0.63)$$

As $(0.33, 0.63) < (T, T)$, hence X_5 will cluster with Leaf (X_1, x_2, x_3, x_4) .

2. Cluster Feature $CF1 \langle N, LS, SS \rangle = \langle 5, (16,30), (54,190) \rangle$

$N = 5$ as there is now four data point under $CF1$.

$CF1 \langle 5, (16,30), (54,190) \rangle$

Leaf

$X_1 = (3, 4),$

$X_2 = (2,6),$

$X_3 = (4,5),$

$X_4 = (4,7)$

$X_5 = (3,8)$

->Consider Data Point $x_6 = (6,2)$ on CF1:

1. Linear Sum $LS = (6,2) + (16,30) = (22,32)$
2. Square Sum $SS = (6^2+54, 2^2 + 190) = (90, 194)$

Now Evaluate Radius considering $N=6$

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(90,194)-(22,32)^2/6}{6}} = (1.24, 1.97)$$

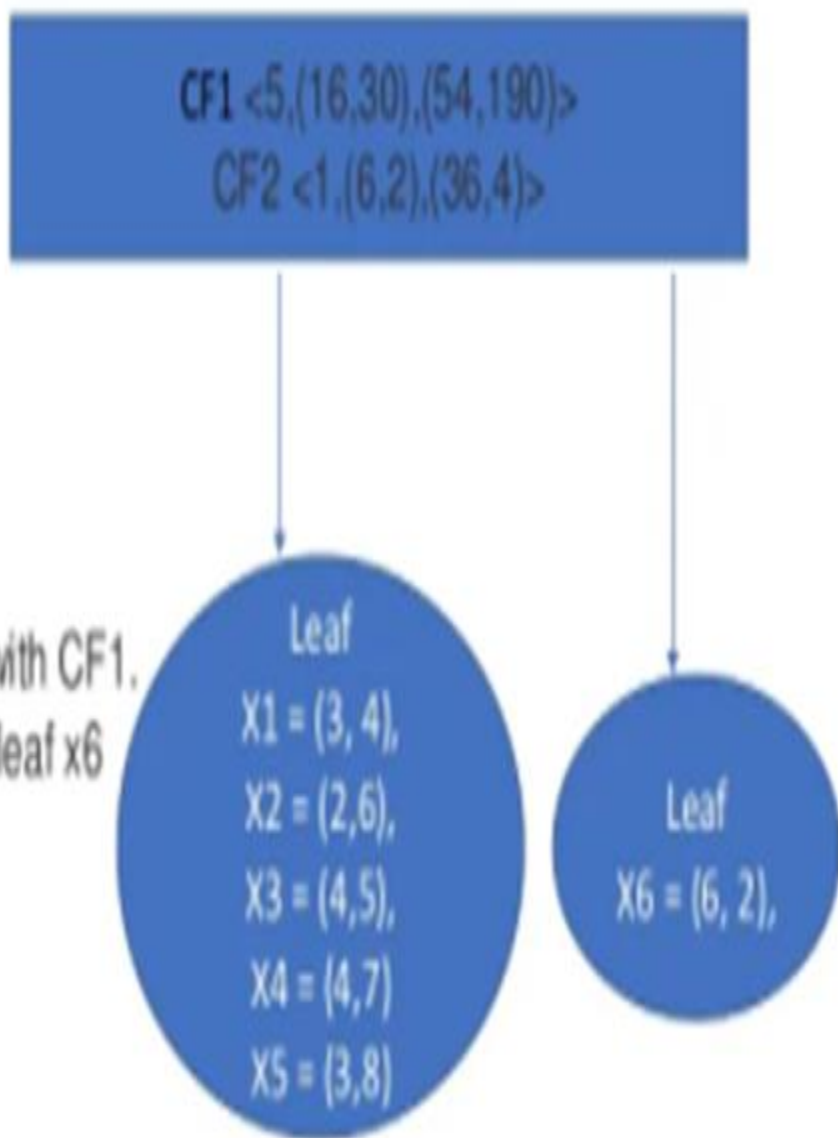
As $(1.24, 1.97) < (T, T)$, False. hence x_6 will Not form cluster with CF1.
CF1 will remain as it was in previous step. And New CF2 with leaf x_6 will be created.

2. Cluster Feature CF2 $\langle N, LS, SS \rangle = \langle 1, (6,2), (36,4) \rangle$

$N = 1$ as there is now one data point under CF2.

$$LS = (6,2)$$

$$SS = (6^2, 2^2) = (36,4)$$



->Consider Data Point $x_7 = (7,2)$. As There are Two Branch CF1 and CF2 hence we need to find with which branch X_7 is nearer, then with that leaf radius will be evaluated.

With CF1 = $LS/N = (16,30)/5 = (8,6)$ As there are $N=5$ Data Point

With CF2 = $LS/N = (6,2)/1 = (6,2)$ As there is $N=1$ Data Point

Now x_7 is closer to $(6,2)$ than $(8,6)$. Hence X_7 will calculate radius with CF2.

1. Linear Sum $LS = (7,2) + (6,2) = (13,4)$

2. Square Sum $SS = (7^2+36, 2^2+4) = (85, 8)$

Now Evaluate Radius considering $N=2$

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(85,8)-(13,4)^2/2}{2}} = (0.5, 0)$$

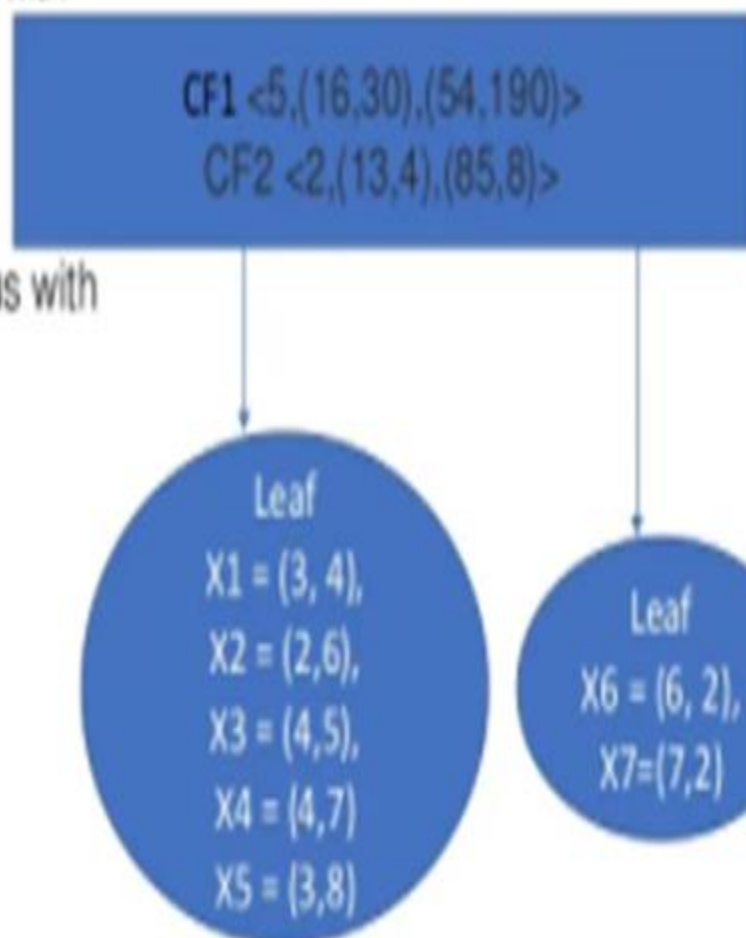
As $(0.5, 0) < (T, T)$, True. hence X_7 will form cluster with CF2

2. Cluster Feature CF2 $\langle N, LS, SS \rangle = \langle 2, (13,4), (85,8) \rangle$

$N = 2$ as there is now two data point under CF2.

$LS = (7,2) + (6,2) = (13,4)$

$SS = (7^2+36, 2^2+4) = (85, 8)$



-> Consider Data Point $x_8 = (7,4)$. As There are Two Branch CF1 and CF2 hence we need to find with which branch x_8 is nearer, then with that leaf, radius will be evaluated. → 3.2

With CF1 = $LS/N = (16,30)/5 = (8,6)$ As there are $N=5$ Data Point

With CF2 = $LS/N = (13,4)/2 = (6.5,2)$ As there is $N=2$ Data Point

Now x_8 is closer to $(6.5,2)$ than $(8,6)$. Hence x_8 will calculate radius with CF2.

1. Linear Sum $LS = (7,4) + (13,4) = (20,8)$

2. Square Sum $SS = (7^2+8^2, 4^2+8^2) = (134, 24)$

Now Evaluate Radius considering $N=3$

$$R = \sqrt{\frac{SS-LS^2/N}{N}} = \sqrt{\frac{(134,24)-(20,8)^2/3}{3}} = (0.47, 0.94)$$

As $(0.47, 0.94) < (T, T)$, True. hence x_8 will form cluster with CF2

2. Cluster Feature CF2 $\langle N, LS, SS \rangle = \langle 3, (20,8), (134,24) \rangle$

$N = 3$ as there is now two data point under CF2.

$LS (7,4) + (13,4) = (20,8)$

$SS = (134,24)$

CF1 $\langle 5, (16,30), (54,190) \rangle$

CF2 $\langle 3, (20,8), (134,24) \rangle$

Leaf

$x_1 = (3, 4),$

$x_2 = (2,6),$

$x_3 = (4,5),$

$x_4 = (4,7)$

$x_5 = (3,8)$

Leaf

$x_6 = (6, 2),$

$x_7 = (7,2),$

$x_8 = (7,4)$

-> Consider Data Point $x_9 = (8, 4)$. As There are Two Branch CF1 and CF2 hence we need to find with which branch x_9 is nearer, then with that leaf, radius will be evaluated.

With CF1 = $LS/N = (16, 30)/5 = (8, 6)$ As there are $N=5$ Data Point

With CF2 = $LS/N = (20, 8)/3 = (6.6, 2.6)$ As there is $N=3$ Data Point

Now x_9 is closer to $(6.6, 2.6)$ then $(8, 6)$. Hence x_9 will calculate radius with CF2.

1. Linear Sum $LS = (8, 4) + (20, 8) = (28, 12)$

2. Square Sum $SS = (8^2 + 134, 4^2 + 24) = (198, 40)$

Now Evaluate Radius considering $N=4$

$$R = \sqrt{\frac{SS - LS^2/N}{N}} = \sqrt{\frac{(198, 40) - (28, 12)^2/4}{4}} = (0.70, 1)$$

As $(0.7, 1) < (T, T)$, True. hence x_9 will form cluster with CF2

2. Cluster Feature CF2 $\langle N, LS, SS \rangle = \langle 4, (28, 12), (198, 40) \rangle$

$N = 4$ as there is now four data point under CF2.

$LS = (28, 12)$

$SS = (198, 40)$

CF1 $\langle 5, (16, 30), (54, 190) \rangle$

CF2 $\langle 4, (28, 12), (198, 40) \rangle$

Leaf

$x_1 = (3, 4)$,

$x_2 = (2, 6)$,

$x_3 = (4, 5)$,

$x_4 = (4, 7)$

$x_5 = (3, 8)$

Leaf

$x_6 = (6, 2)$,

$x_7 = (7, 2)$,

$x_8 = (7, 4)$,

$x_9 = (8, 4)$

-> Consider Data Point $x_{10} = (7,9)$. As There are Two Branch CF1 and CF2 hence we need to find with which branch x_{10} is nearer, then with that leaf, radius will be evaluated.

With $CF1 = LS/N = (16,30)/5 = (8,6)$ As there are $N=5$ Data Point

With $CF2 = LS/N = (28,12)/4 = (7,3)$ As there is $N=4$ Data Point

Now x_{10} is closer to $(8,6)$ than $(7,3)$. Hence x_{10} will calculate radius with CF1.

1. Linear Sum $LS = (7,9) + (16,30) = (23,39)$

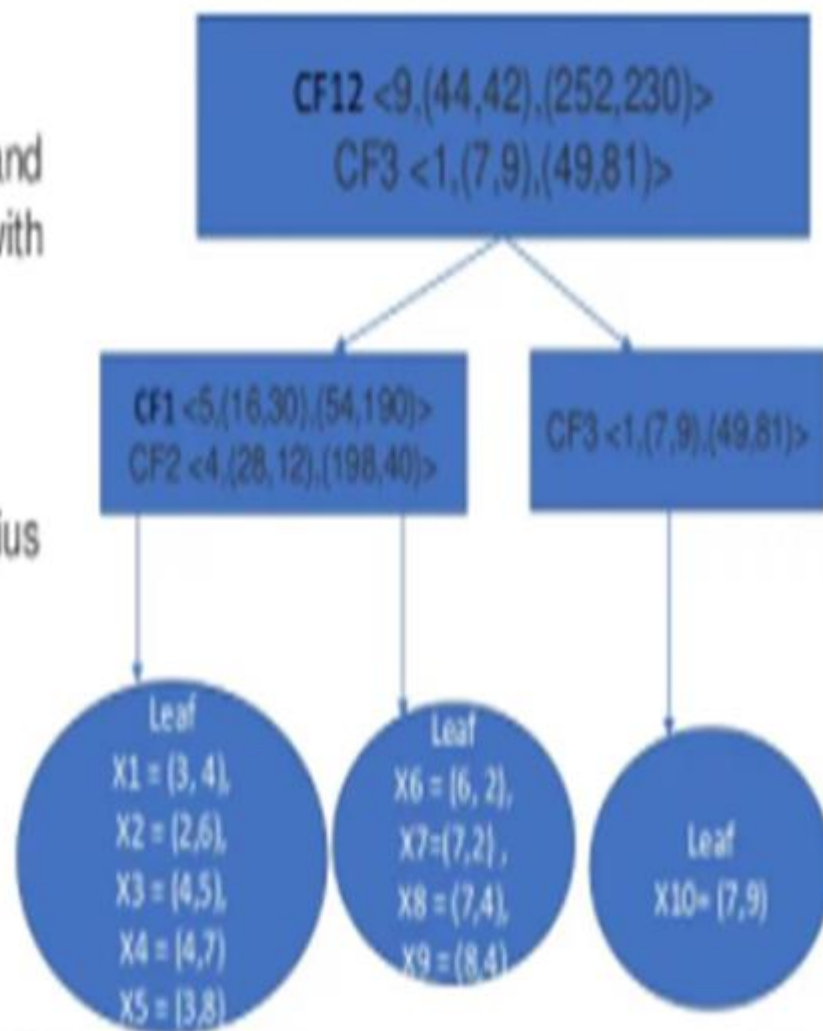
2. Square Sum $SS = (7^2+54, 9^2+190) = (103, 271)$

Now Evaluate Radius considering $N=6$

$$R = \sqrt{\frac{SS - LS^2/N}{N}} = \sqrt{\frac{(103,271) - (23,39)^2/6}{6}} = (1.57, 1.70)$$

As $(1.57, 1.70) < (T, T)$, False. hence x_{10} will become new leaf and Create new cluster feature CF3. But in a Branch only two CF is allowed hence Branch will Split.

2. Cluster Feature $CF3 <N, LS, SS> = <1, (7,9), (49,81)>$



Density Based Clustering

- Density-based approach:
 - Based on connectivity and density functions.
 - The strategy used is to model the dense clusters separated by sparse clusters.
 - The 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.
 - Such a method can be used to filter out noise or outliers and discover clusters of arbitrary shape.
 - Major features:
 - Discover clusters of arbitrary shape
 - Handle noise
 - One scan
 - Need density parameters as termination condition

□ Typical methods: DBSCAN, OPTICS, DenClue

DBSCAN(Density Based Spatial Clustering of Application with Noise)

- Density-Based Clustering Algorithm Based on Connected Regions with High Density.
- The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.
- **Why DBSCAN?**
 - Real life data may contain irregularities, like:
 1. Clusters can be of arbitrary shape.
 2. Data may contain noise.

DBSCAN

■ DBSCAN algorithm requires two parameters:

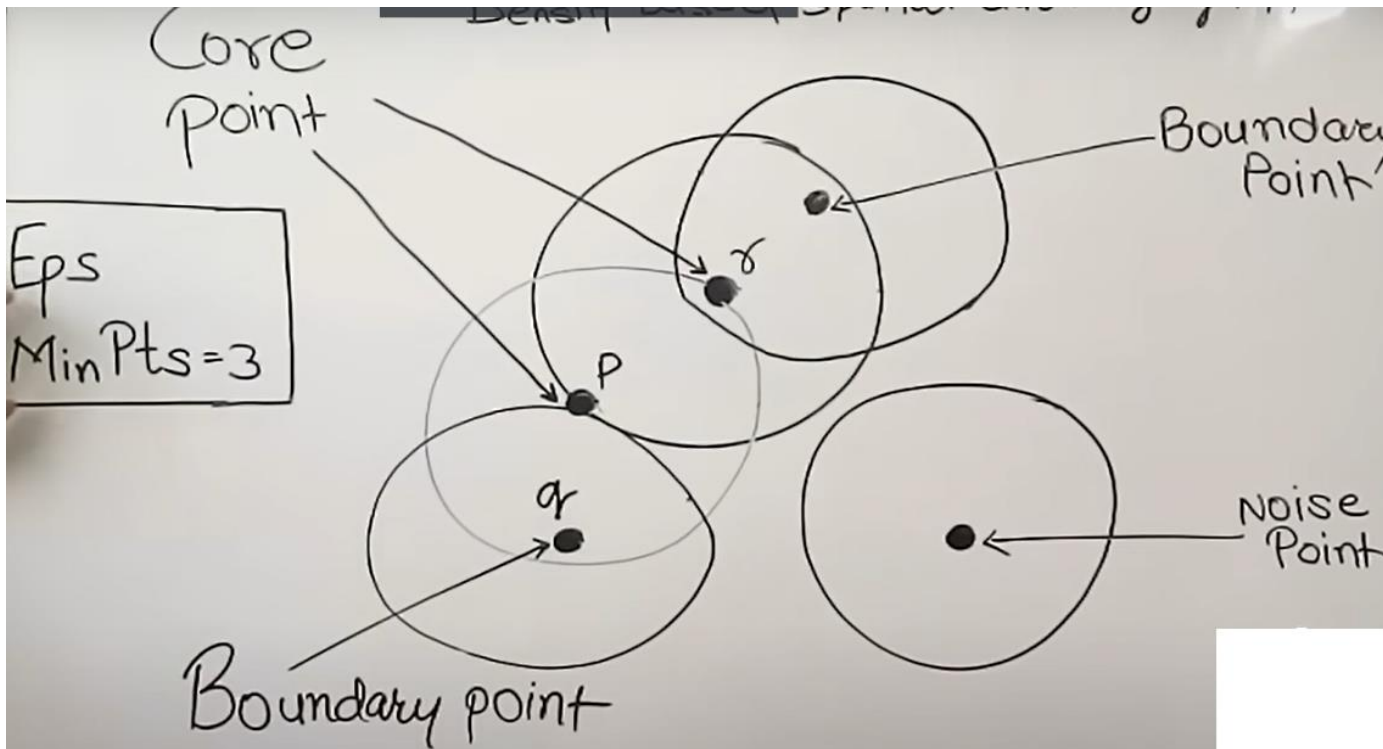
1. **eps** : It defines the neighborhood around a data point
 - if the distance between two points is \leq 'eps' then they are considered as neighbors.
 - If the eps value is chosen too small then large part of the data will be considered as outliers.
 - If it is chosen very large then the clusters will merge and majority of the data points will be in the same clusters.
 - One way to find the eps value is based on the ***k-distance graph***.
2. **MinPts**: Minimum number of neighbors (data points) within eps radius.
 - Larger the dataset, the larger value of MinPts must be chosen.
 - As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, $\text{MinPts} \geq D+1$.
 - The minimum value of MinPts must be chosen as at least 3.

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

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

DBSCAN

- In this algorithm, we have 3 types of data points.
 - **Core Point:** A point is a core point if it has more than MinPts points within eps.
 - **Border Point:** A point which has fewer than MinPts within eps(i.e it is not a core point) but it is in the neighborhood of a core point.
 - **Noise or outlier:** A point which is not a core point or border point.

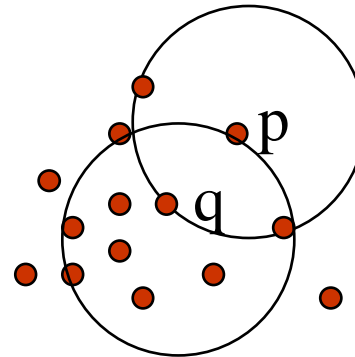


MinPts=3

Density-Based Clustering: Basic Concepts

- Two parameters:
 - *Eps* : Maximum radius of the neighborhood
 - *MinPts* : Minimum number of points in an Eps-neighbourhood of that point
- $N_{Eps}(p)$: $\{q \text{ belongs to } D \mid \text{dist}(p,q) \leq Eps\}$
- **Directly density-reachable**: A point p is directly density-reachable from a point q w.r.t. Eps , $MinPts$ if
 - p belongs to $N_{Eps}(q)$
 - core point condition:

$$|N_{Eps}(q)| \geq MinPts$$



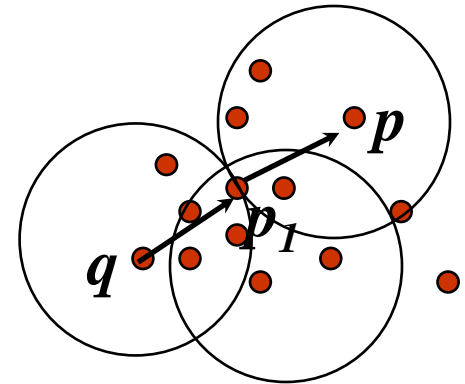
MinPts = 5

Eps = 1 cm

Density-Reachable and Density-Connected

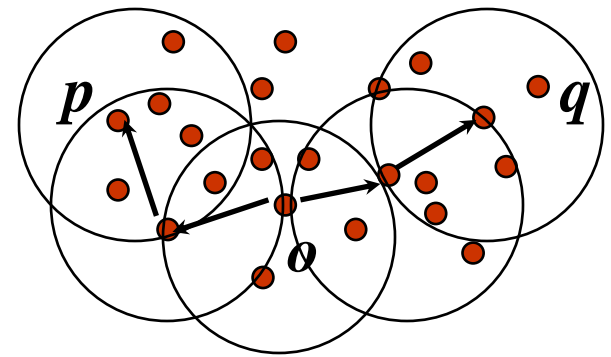
- Density-reachable:

- A point p is **density-reachable** from a point q w.r.t. Eps , $MinPts$ if there is a chain of points p_1, \dots, p_n , $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i



- Density-connected

- A point p is **density-connected** to a point q w.r.t. Eps , $MinPts$ if there is a point o such that both, p and q are density-reachable from o w.r.t. Eps and $MinPts$



DBSCAN: The Algorithm

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

DBSCAN Algorithm

Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D : a data set containing n objects,
- ϵ : 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 ϵ -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 ϵ -neighborhood of p ;
- (8) for each point p' in N
- (9) if p' is unvisited
- (10) mark p' as visited;
- (11) if the ϵ -neighborhood of p' has at least $MinPts$ points,
 add those points to N ;
- (12) if p' is not yet a member of any cluster, add p' to C ;
- (13) **end for**
- (14) output C ;
- (15) **else** mark p as noise;
- (16) **until** no object is unvisited;

DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

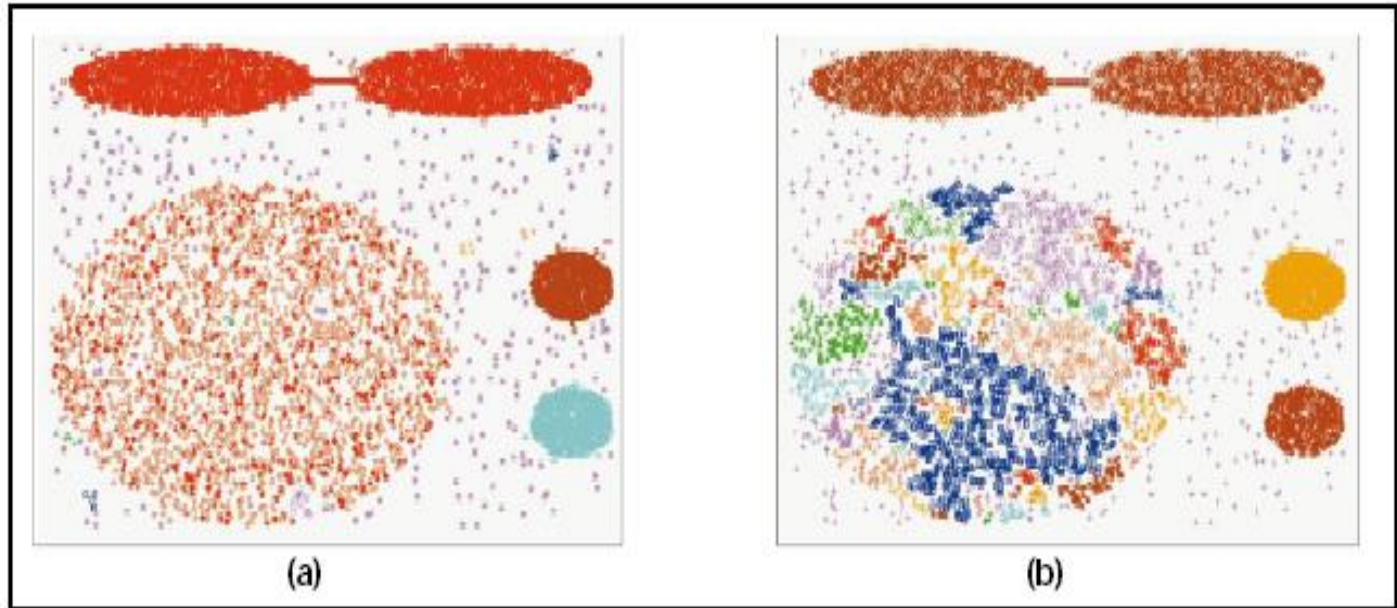
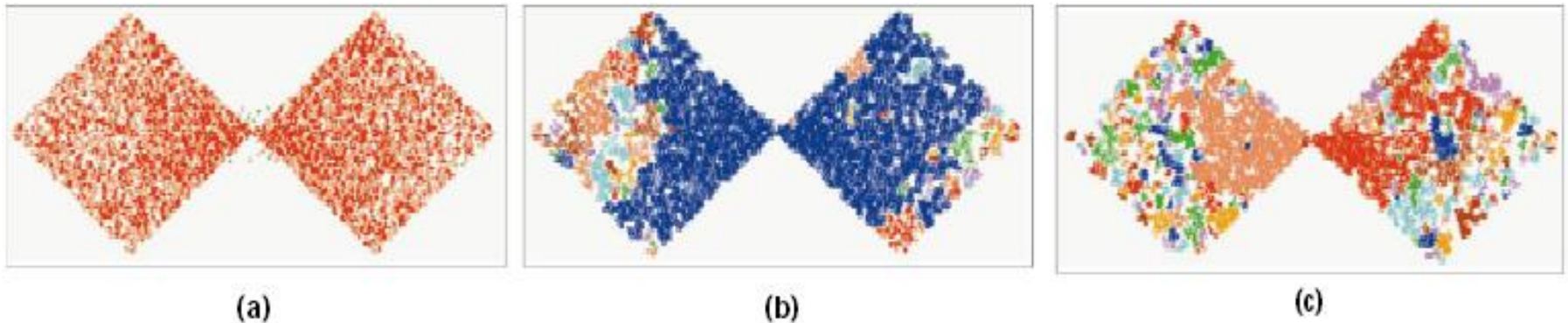


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



Example: DBSCAN

A1	2	10
A2	2	5
A3	8	4
A4	5	8
A5	7	5
A6	6	4
A7	1	2
A8	4	9

Find clusters using DBSCAN $\epsilon = 2$, MinPts=2

Step1 : Find distance matrix

Euclidean Distance	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	5	8.49	3.61	7.07	7.21	8.06	2.24
A2	5	0	6.08	4.24	5	4.12	3.16	4.47
A3	8.49	6.08	0	5	1.41	2	7.28	6.4
A4	3.61	4.24	5	0	3.61	4.12	7.21	1.41
A5	7.07	5	1.41	3.61	0	1.41	6.71	5
A6	7.21	4.12	2	4.12	1.41	0	5.39	5.39
A7	8.06	3.16	7.28	7.21	6.71	5.39	0	7.62
A8	2.24	4.47	6.4	1.41	5	5.39	7.62	0

Example: DBSCAN

Step2 : Find which data points are core points

$\epsilon = 2$, MinPts=2

Cluster1={A3,A5,A6}
Cluster2={A4,A8}

A1,A2,A7 are outliers

Euclidean Distance	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	5	8.49	3.61	7.07	7.21	8.06	2.24
A2	5	0	6.08	4.24	5	4.12	3.16	4.47
A3	8.49	6.08	0	5	1.41	2	7.28	6.4
A4	3.61	4.24	5	0	3.61	4.12	7.21	1.41
A5	7.07	5	1.41	3.61	0	1.41	6.71	5
A6	7.21	4.12	2	4.12	1.41	0	5.39	5.39
A7	8.06	3.16	7.28	7.21	6.71	5.39	0	7.62
A8	2.24	4.47	6.4	1.41	5	5.39	7.62	0

ϵ -neighborhood

A1

A2

A3,
A5,
A6

A4,
A8

A3,
A5,
A6

A3,
A5,
A6

A7

A4,
A8

The no. of data points in the ϵ -neighbourhood of A1= 1(A1 itself) . < MinPts . Hence A1 is not a core point

Example: DBSCAN

Find clusters using DBSCAN $\epsilon = 2$, MinPts=3

Euclidean Distance	A	B	C	D	E	F
A	0	0.7	5.7	3.6	4.2	3.2
B	0.7	0	4.9	2.9	3.5	2.5
C	5.7	4.9	0	2.9	1.4	2.5
D	3.6	2.9	2.9	0	1	0.5
E	4.2	3.5	1.4	1	0	1.1
F	3.2	2.5	2.5	0.5	1.1	0

Find which data points are core points, Boundary points and Outliers.

Example: DBSCAN

Find clusters using DBSCAN $\epsilon = 2$, MinPts=3

Euclidean Distance	A	B	C	D	E	F
A	0	0.7	5.7	3.6	4.2	3.2
B	0.7	0	4.9	2.9	3.5	2.5
C	5.7	4.9	0	2.9	1.4	2.5
D	3.6	2.9	2.9	0	1	0.5
E	4.2	3.5	1.4	1	0	1.1
F	3.2	2.5	2.5	0.5	1.1	0

ϵ -neighborhood A,B A,B C,E D,E,F C,D,E,F D,E,F

Find which data points are core points, Boundary Points and Outliers

D,E and F are core points.

A and B are outliers

C is Boundary point as it is in ϵ -neighborhood of core point E

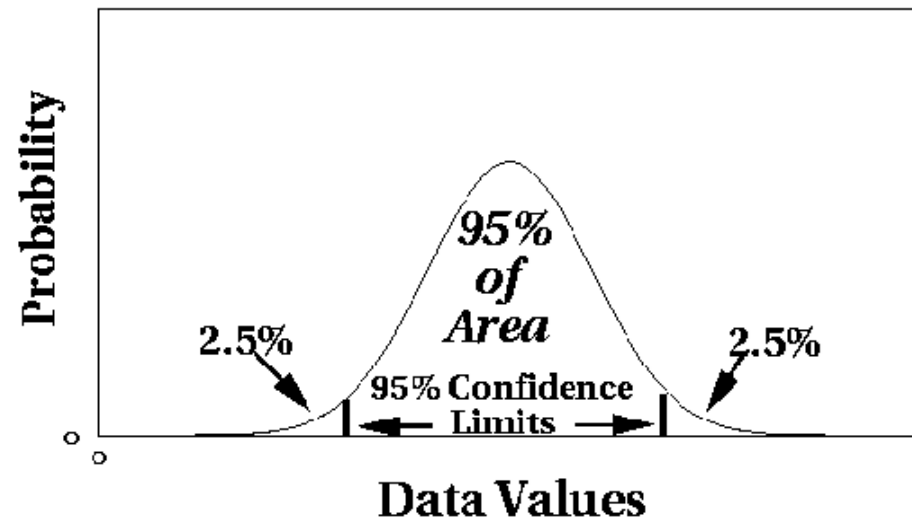
Self Learning

- Hierarchical methods :
 - Chameleon, Density based methods: OPTICS,
- Grid based methods: STING, CLIQUE

What Is Outlier Discovery?

- What are outliers?
 - The set of objects are considerably dissimilar from the remainder of the data
 - Example: Sports: Michael Jordon, Wayne Gretzky, ...
- Problem: Define and find outliers in large data sets
- Applications:
 - Credit card fraud detection
 - Telecom fraud detection
 - Customer segmentation
 - Medical analysis

Outlier Discovery: Statistical Approaches



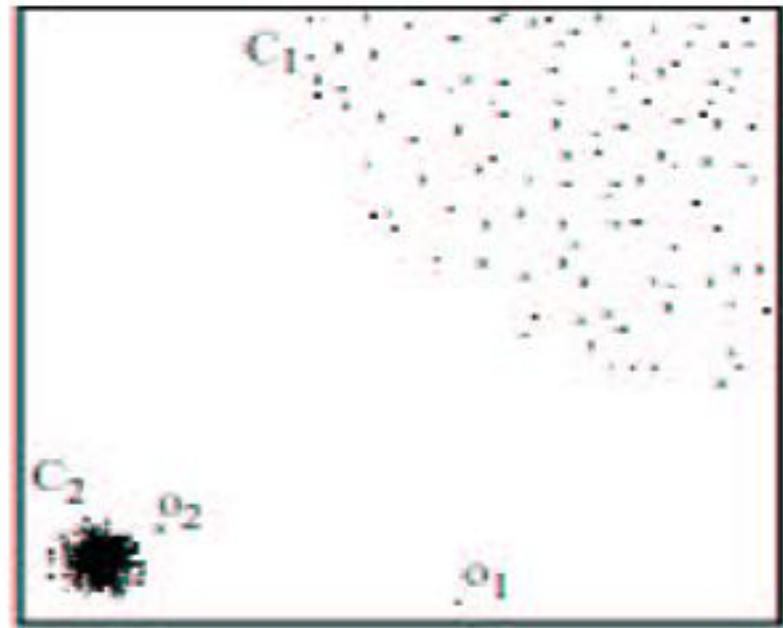
- Assume a model underlying distribution that generates data set (e.g. normal distribution)
- Use discordancy tests depending on
 - data distribution
 - distribution parameter (e.g., mean, variance)
 - number of expected outliers
- Drawbacks
 - most tests are for single attribute
 - In many cases, data distribution may not be known

Outlier Discovery: Distance-Based Approach

- Introduced to counter the main limitations imposed by statistical methods
 - We need multi-dimensional analysis without knowing data distribution
- Distance-based outlier: A $DB(p, D)$ -outlier is an object O in a dataset T such that at least a fraction p of the objects in T lies at a distance greater than D from O
- Algorithms for mining distance-based outliers
 - Index-based algorithm
 - Nested-loop algorithm
 - Cell-based algorithm

Density-Based Local Outlier Detection

- Distance-based outlier detection is based on global distance distribution
- It encounters difficulties to identify outliers if data is not uniformly distributed
- Ex. C_1 contains 400 loosely distributed points, C_2 has 100 tightly condensed points, 2 outlier points o_1 , o_2
- Distance-based method cannot identify o_2 as an outlier
- Need the concept of local outlier



- Local outlier factor (LOF)
 - Assume outlier is not crisp
 - Each point has a LOF

Outlier Discovery: Deviation-Based Approach

- Identifies outliers by examining the main characteristics of objects in a group
- Objects that “deviate” from this description are considered outliers
- Sequential exception technique
 - simulates the way in which humans can distinguish unusual objects from among a series of supposedly like objects
- OLAP data cube technique
 - uses data cubes to identify regions of anomalies in large multidimensional data

Problems and Challenges

- Considerable progress has been made in scalable clustering methods
 - Partitioning: k-means, k-medoids
 - Hierarchical: AGNES, DIANA, BIRCH,
 - Density-based: DBSCAN,
- Current clustering techniques do not address all the requirements adequately, still an active area of research