Unit V

Cluster Analysis: Basic Concepts and Algorithms

- **What is Cluster Analysis?**
  - **Different Types of Clustering**
  - **Different Types of Clusters**
- **K – Means Algorithm**
  - **Issues**
  - **Bisecting K-Means**
  - **K-Means and different types of clusters**
  - **Strength and Weakness**
  - **K-Means as an Optimization Problem**
- **Agglomerative Hierarchical Algorithm**
  - **Basic Algorithm**
  - **Specific Techniques**
- **DBSCAN**
  - **Traditional Density : Center Based Approach**
  - **The DBSCAN Algorithm**
  - **Strength and Weakness**

Definition:

"**Cluster analysis** or **clustering** is the task of grouping a set of objects or data points in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters)."

Or

"**Clustering** is the process of grouping objects into different groups which are meaningful, useful or both".

Cluster analysis is related to other mining tasks that divide data objects into groups. For instance, *clustering can be regarded as a form of classification in that it creates a labeling of objects with class (cluster) labels*. Due to this reason ***Classification is known as "Supervised Learning" and Clustering is known as "Unsupervised Classification"***

Segmentation and **Partitioning** are sometimes used as ***synonyms for*** **Clustering**; these terms are frequently used for approaches outside the traditional bounds of cluster analysis. For example, the term partitioning is often used in connection with techniques that divide graphs into subgraphs and that are not strongly connected to clustering. Segmentation often refers to the division of data into groups using simple techniques; e.g., an image can be split into segments based only on pixel intensity and color.

Applications for Clustering:

- **Clustering for Understanding**

  Classes, or conceptually meaningful groups of objects that share common characteristics, play an important role to analyze and describe.

- ***Biology***
  Biologists have spent many years creating taxonomy (hierarchical classification) of all living things: *kingdom, phylum, class, order, family, genus, and species*. Biologists have applied clustering to analyze the large amounts of genetic information. For example, clustering has been used to find groups of genes that have similar functions.

- ***Information Retrieval***
  The World Wide Web consists of billions of Web pages, and the results of a query to a search engine can return thousands of pages. Clustering can be used to group these search results into a small number of clusters, each of which captures a particular aspect of the query.

- ***Climate***
  Understanding the Earth's climate requires finding patterns in the atmosphere and ocean. Cluster analysis has been applied to find patterns in the atmospheric pressure of Polar Regions and areas of the ocean that have a significant impact on land climate.

- ***Phycology and Medicine***
  An illness or condition frequently has a number of variations, and cluster analysis can be used to identify these different subcategories. Cluster analysis can also be used to detect patterns in the spatial or temporal distribution of a disease.

- ***Business***
  Businesses collect large amounts of information on current and potential customers. Clustering can be used to segment customers into a small number of groups for additional analysis and marketing activities.

- **Clustering for Utility**
  Cluster analysis provides an abstraction from individual data objects to the clusters in which those data objects reside. Some clustering techniques characterize each cluster in terms of a cluster prototype; i.e., a data object that is representative of the other objects in the cluster. These cluster prototypes can be used as the basis for a number of data analysis or data processing techniques. Therefore, in the context of utility, cluster analysis is the study of techniques for finding the most representative cluster prototypes.

  - ***Summarization***
    Many data analysis techniques, such as regression or PCA, have a time or space complexity of O(m2) or higher (where m is the number of objects), and thus, are not practical for large data sets. However, instead of applying the algorithm to the entire data set, it can be applied to a reduced data set consisting only of cluster prototypes.

  - ***Compression***
    Cluster prototypes can also be used for data compression. In particular, a table is created that consists of the prototypes for each cluster; i.e., each prototype is assigned an integer value that is its position (index) in the table. Each object is

represented by the index of the prototype associated with its cluster. This type of compression is known as vector quantization and is often applied to image, sound, and video data, where (1) many of the data objects are highly similar to one another, (2) some loss of information is acceptable, and (3) a substantial reduction in the data size is desired.

- ***Efficiently Finding Nearest Neighbors***
  Finding nearest neighbors can require computing the pairwise distance between all points. If objects are relatively close to the prototype of their cluster, then we can use the prototypes to reduce the number of distance computations that are necessary to find the nearest neighbors of an object. If two cluster prototypes are far apart, then the objects in the corresponding clusters cannot be nearest neighbors of each other. Consequently, to find an object's nearest neighbors it is only necessary to compute the distance to objects in nearby clusters, where the nearness of two clusters is measured by the distance between their prototypes.

Difference between Classification and Clustering:

CLASSIFICATION	CLUSTERING
We have a Training set containing data that have been previously categorized	We do not know the characteristics of similarity of data in advance
Based on this training set, the algorithms finds the category that the new data points belong to	Using statistical concepts, we split the datasets into sub-datasets such that the Sub-datasets have "Similar" data
Since a Training set exists, we describe this technique as **Supervised learning**	Since Training set is not used, we describe this technique as **Unsupervised learning**
Example: We use training dataset which categorized customers that are loyal. Now based on this training set, we can classify whether a customer will be loyal to our shop or not.	**Example:** We use a dataset of customers and split them into sub-datasets of customers with "similar" characteristics. Now this information can be used to market a product to a specific segment of customers that has been identified by clustering algorithm

Types of Clustering:

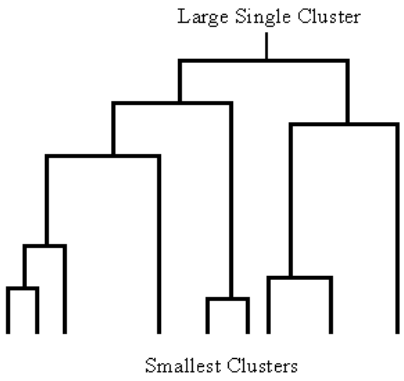
- Hierarchical clustering Vs Partitional Clustering
- Exclusive Clustering Vs Overlapping Clustering
- Fuzzy Clustering
- Complete Clustering Vs Partial Clustering

Hierarchical clustering Vs Partitional Clustering

Hierarchical Clustering:

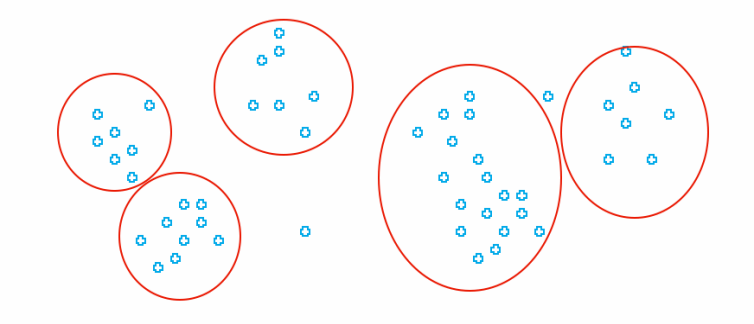
A division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset. Each node (cluster) in the tree is the union of its children, and the root of the tree is

the single large cluster contains nested clusters.



Partitional Clustering:

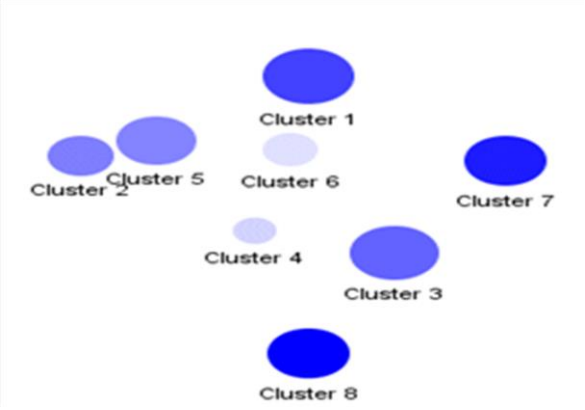
A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.



Exclusive Clustering Vs Overlapping Clustering

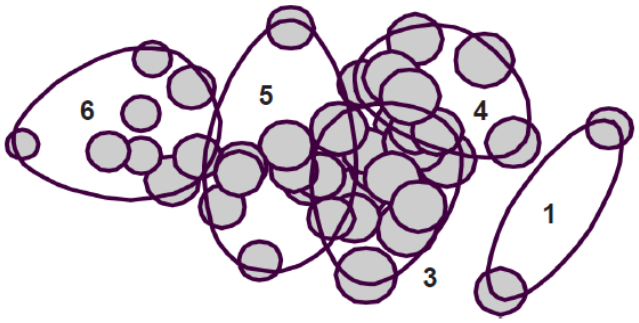
Exclusive Clustering:

In Exclusive clustering each data object is assigned to a single object.



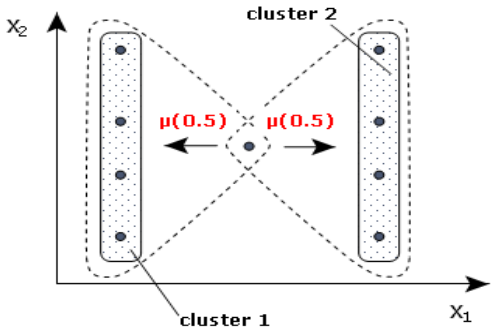
Overlapping Clustering:

In non-exclusive or overlapping clustering, points may belong to multiple clusters. This clustering is used to represent that an object can belong to more than a single cluster. For example, a person can act as student and faculty in the same organization.



Fuzzy Clustering

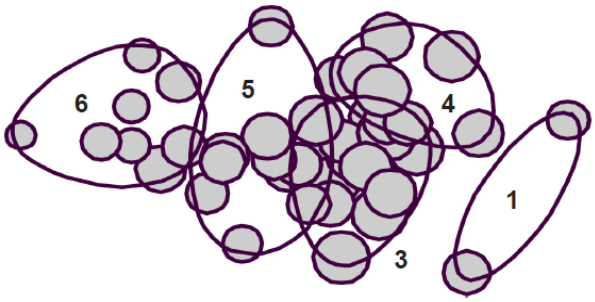
In fuzzy clustering, every object belongs to a cluster with a membership weight that is between 0 (absolutely doesn't belong) and 1 (absolutely belong). In fuzzy clustering we often impose that the sum of weights of an object must be 1.



Complete Clustering Vs Partial Clustering:

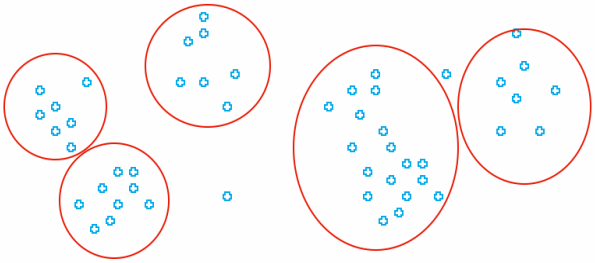
Complete Clustering:

A complete clustering assigns every object to a cluster.



Partial Clustering:

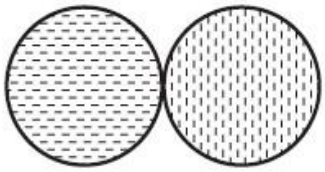
In Partial clustering, some of the data points are left alone as outliers or noises. These types of clusters are used for outlier analysis.



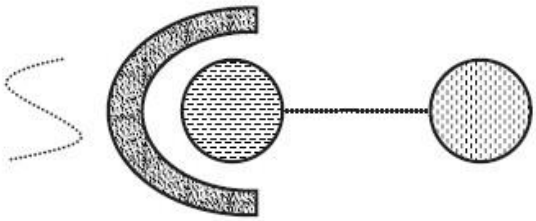
Types of Clusters:

- Well-separated clusters
- Center-based clusters
- Contiguous clusters
- Density-based clusters
- Property or Conceptual

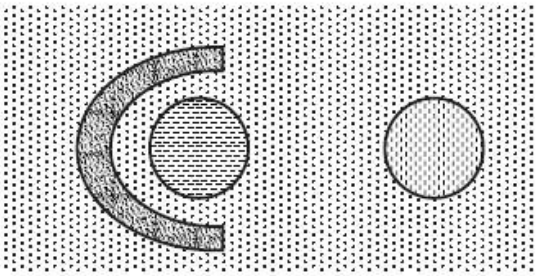
(a) Well-separated clusters. Each point is closer to all of the points in its cluster than to any point in another cluster.



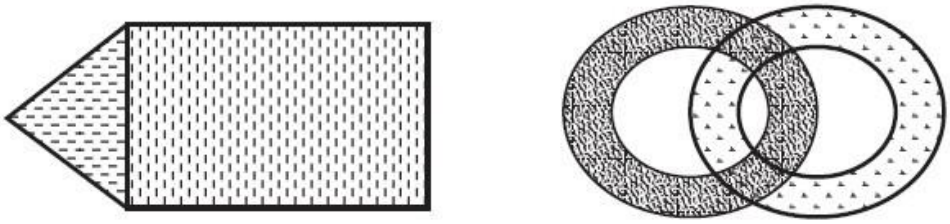
(b) Center-based clusters. Each point is closer to the center of its cluster than to the center of any other cluster.



(c) Contiguity-based clusters. Each point is closer to at least one point in its cluster than to any point in another cluster.



(d) Density-based clusters. Clusters are regions of high density separated by regions of low density.



(e) Conceptual clusters. Points in a cluster share some general property that derives from the entire set of points. (Points in the intersection of the circles belong to both.)

Well Separated Clusters:

A Well Separated cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.

Center-based Clusters:

A cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster. The center of a cluster is often a

centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster.

Contiguous Clusters (Nearest neighbor or Transitive):

A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.

Density-based Clusters:

A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density. Used when the clusters are irregular or intertwined, and when noise and outliers are present.

Shared Property or Conceptual Clusters:

Finds clusters that share some common property or represent a particular concept.

Road Map:

K-means:

This is a prototype-based, partitional clustering technique that attempts to find a user-specified number of clusters (K), which are represented by their centroids.

Agglomerative Hierarchical Clustering:

This clustering approach refers to a collection of closely related clustering techniques that produce a hierarchical clustering by starting with each point as a singleton cluster and then repeatedly merging the two closest clusters until a single cluster remains.

DBSCAN:

This is a density-based clustering algorithm that produces a partitional clustering, in which the number of clusters is automatically determined by the algorithm. Points in low-density regions are classified as noise and omitted; thus, DBSCAN does not produce a complete clustering.

K-Means:

A prototype based one level partitional clustering technique that attempts to find a user-specified number of clusters(k).

- It is the oldest and most widely used clustering algorithm
- K-means defines a prototype as a "centroid"
- Applied to objects in a continuous n-dimensional space.

The Basic K-Means Algorithm:

- Choose k initial centroids, k-user specified and k indicates number of clusters desired
- Each point is then assigned to the closest centroid, and each collection of points assigned to a centroid is a cluster.
- Centroid of each cluster is then updated based on the points assigned to the cluster
- Repeat the assignment and update steps until no point changes clusters.

Algorithm:

Algorithm Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

- K-means reaches a state in which no points are shifting from one cluster to another and hence, the centroids don't change.

Assigning Points to the Closest Centroid:

- ✓ To assign a point into a cluster, a measure must be calculated for finding the closest of point to a centroid.

Various measures are:

- Euclidean (L2) distance is often used for data points in Euclidean space.
- Cosine similarity is more appropriate for documents.
- Manhattan (L1) distance can be used for Euclidean data
- Jaccard measure is often employed for documents.

Table of notation.

Symbol	Description
$\mathbf{x}$	An object.
C_i	The i^{th} cluster.
$\mathbf{c}_i$	The centroid of cluster C_i.
$\mathbf{c}$	The centroid of all points.
m_i	The number of objects in the i^{th} cluster.
m	The number of objects in the data set.
K	The number of clusters.

Centroids and Objective Functions:

Minimize the squared distance of each point to its closest centroid.

Data in Euclidean Space:

- Objective function, measures the quality of a clustering, we use the sum of the squared error (SSE), which is also known as scatter.
- Calculate the error of each data point, i.e., its Euclidean distance to the closest centroid, and then compute the total sum of the squared errors (SSE).

- Given two different sets of clusters that are produced by two different runs of K-means, we prefer the one with the smallest squared error.
- SSE is formally defined as follows:

$$\text{SSE} = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} dist(\mathbf{c}_i, \mathbf{x})^2$$

where *dist* is the standard Euclidean (L2) distance between two objects in Euclidean space.

The centroid that minimizes the SSE of the cluster is the mean. The centroid (mean) of the i^{th} cluster is defined by:

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

To illustrate, the centroid of a cluster containing the three two-dimensional points, (1,1), (2,3), and (6,2), is ((1 + 2 + 6)/3,((1 + 3 + 2)/3) = (3,2).

Document data:

- k-means is also used for document data; for document data the cosine similarity measure.
- Document data is represented as a document – term matrix
- Maximize the similarity of the documents in a cluster to the cluster centroid
- This quality is called cohesion of the cluster

$$\text{Total Cohesion} = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} cosine(\mathbf{x}, \mathbf{c}_i)$$

Manhattan(L1) - median- is the objective of minimizing sum of the distances.

Choosing Initial centroids:

Randomly selected initial centroids may be poor. One technique that is commonly used to address the problem of choosing initial centroids is to perform multiple runs, each with a different set of randomly chosen initial centroids, and then select the set of clusters with the minimum SSE.

PROBLEM

Use the k-means algorithm and Euclidean distance to cluster the following 8 examples into 3 clusters: A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9). The distance matrix based on the Euclidean distance is given below:

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
A2		0	$\sqrt{37}$	$\sqrt{18}$	$\sqrt{25}$	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{20}$
A3			0	$\sqrt{25}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{53}$	$\sqrt{41}$
A4				0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
A5					0	$\sqrt{2}$	$\sqrt{45}$	$\sqrt{25}$
A6						0	$\sqrt{29}$	$\sqrt{29}$
A7							0	$\sqrt{58}$
A8								0

Solution

d(a,b) denotes the Eucledian distance between a and b. It is obtained directly from the distance matrix or calculated as follows: $d(a,b) = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$

Select three centroids. A1 (C1 - Cluster 1), A4 (C2 - Cluster 2) and A7 (C3 - Cluster 3)

ITERATION 1

Find Eucledian Distance for A1 with C1, C2 & C3

d(A1, C1)=0

d(A1, C2)= $\sqrt{13}$

d(A1, C3)= $\sqrt{65}$

Add A1 to Cluster C1, because A1 is closer to the centroid of cluster C1.

Find Eucledian Distance for A2 with C1, C2 & C3

d(A2, C1)= $\sqrt{25}$

d(A2, C2)= $\sqrt{18}$

d(A2, C3)= $\sqrt{10}$

Add A2 to Cluster C3, because A2 is closer to the centroid of Cluster C3

Find Eucledian Distance for A3 with C1, C2 & C3
d(A3, C1)= $\sqrt{36}$
d(A3, C2)= $\sqrt{25}$
d(A3, C3)= $\sqrt{53}$

Add A3 to Cluster C2, because A3 is closer to the centroid of Cluster C2

Find Eucledian Distance for A4 with C1, C2 & C3
d(A4, C1)= $\sqrt{13}$
d(A4, C2)= 0
d(A4, C3)= $\sqrt{52}$

Add A4 to Cluster C2, because A4 is closer to the centroid of Cluster C2

Find Eucledian Distance for A5 with C1, C2 & C3
d(A5, C1)= $\sqrt{50}$
d(A5, C2)= $\sqrt{13}$
d(A5, C3)= $\sqrt{45}$

Add A5 to Cluster C2, because A5 is closer to the centroid of Cluster C2

Find Eucledian Distance for A6 with C1, C2 & C3
d(A6, C1)= $\sqrt{52}$
d(A6, C2)= $\sqrt{17}$
d(A6, C3)= $\sqrt{29}$

Add A6 to Cluster C2, because A6 is closer to the centroid of Cluster C2

Find Eucledian Distance for A7 with C1, C2 & C3
d(A7, C1)= $\sqrt{65}$
d(A7, C2)= $\sqrt{52}$
d(A7, C3)= 0

Add A7 to Cluster C3, because A7 is closer to the centroid of Cluster C3

Find Eucledian Distance for A8 with C1, C2 & C3
d(A8, C1)= $\sqrt{5}$
d(A8, C2)= $\sqrt{2}$
d(A8, C3)= $\sqrt{58}$

Add A8 to Cluster C2, because A8 is closer to the centroid of Cluster C2

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

New Formed Clusters:

Cluster 1 (C1) : {A1}

Cluster 2 (C2) : {A3, A4, A5, A6, A8}

Cluster 3 (C3) : {A2, A7}

Updated Centroid for Cluster 1 (C1)	**(2,10)**
Updated Centroid for Cluster 2 (C2)	**((8+5+7+6+4)/5 , (4+8+5+4+9)/5) = (6 , 6)**
Updated Centroid for Cluster 3 (C3)	**((2+1)/2 , (5+2)/2) = (1.5 , 3.5)**

New Centroids are C1 (2, 10), C2(6, 6) and C3(1.5, 3.5)

ITERATION 2

With this new centroids again calculate Eucledian Distance for points A1, A2, A3, A4, A5, A6, A7 AND A8

Newly formed clusters are:

Cluster 1 (C1) : {A1, A8}

Cluster 2 (C2) : {A3, A4, A5, A6}

Cluster 3 (C3) : {A2, A7}

Centroid for Cluster 1 (C1) : (3, 9.5)

Centroid for Cluster 2 (C2) : (6.5, 5.25)

Centroid for Cluster 3 (C3) : (1.5, 3.5)

ITERATION 3

With this new centroids again calculate Eucledian Distance for points A1, A2, A3, A4, A5, A6, A7 AND A8

Newly formed clusters are:

Cluster 1 (C1) : {A1, A4, A8}

Cluster 2 (C2) : {A3, A5, A6}

Cluster 3 (C3) : {A2, A7}

Centroid for Cluster 1 (C1) : (3.66, 9)

Centroid for Cluster 2 (C2) : (7, 4.33)

Centroid for Cluster 3 (C3) : (1.5, 3.5)

With these new centroids, the entire process is repeated until the centroids don’t change.

Time and space complexity:

Space for k-means are modest because only the data points and centroids are stored

i.e o((m+k)n) where m-no. of points, n-no. of attributes

time required is o(I*k*m*n) where I-no. of iterations, k – no. of clusters, m-no. of points and n-no. of attributes.

K-means: Additional Issues

Handling Empty Clusters:

One of the problems with the basic K-means algorithm given earlier is that empty clusters can be obtained if no points are allocated to a cluster during the assignment step. One approach is to choose the point that is farthest away from any current centroid. Another approach is to choose the replacement centroid from the cluster that has the highest SSE. This will typically split the cluster and reduce the overall SSE of the clustering. If there are several empty clusters, then this process can be repeated several times.

Outliers:

When outliers are present, the resulting cluster centroids (prototypes) thus have SSE at higher levels. Because of this, it is often useful to discover outliers and eliminate them before.

Reducing the SSE with Post-processing:

Two strategies that decrease the total SSE by increasing the number of clusters.

Split a cluster:

Choose the cluster with largest SSE, & split the cluster.

Introduce a new cluster centroid:

- Choose the farthest point from any cluster center (or) randomly chose point with the highest SSE

Strategies that decrease the number of clusters, while trying to minimize the increase in total SSE

Disperse a cluster:

Remove the centroid that corresponds to the cluster & reassigning the points to other clusters.

Merge two clusters:

Clusters with closest centroids are merged result in the smallest increase in total SSE.

Updating centroids incrementally:

Instead of updating cluster centroids after all points have been assigned to a cluster, the centroids can be updated incrementally, after each assignment of a point to a cluster. This requires either zero or two updates to cluster centroids at each step, since a point

either moves to a new cluster (two updates) or stays in its current cluster (zero updates).

Bisecting K-means:

Idea:

To obtain k clusters, split the set of all points into two clusters, select one of those clusters to split, and so on, until k clusters have been produced.

Algorithm:

Algorithm Bisecting K-means algorithm.

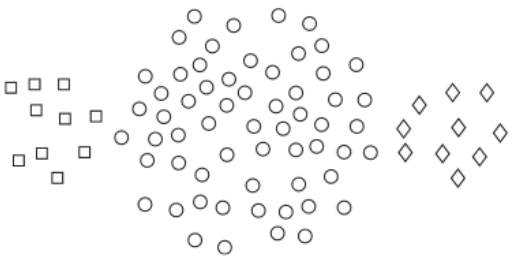
1: Initialize the list of clusters to contain the cluster consisting of all points.
2: **repeat**
3: Remove a cluster from the list of clusters.
4: {Perform several "trial" bisections of the chosen cluster.}
5: **for** $i = 1$ to *number of trials* **do**
6: Bisect the selected cluster using basic K-means.
7: **end for**
8: Select the two clusters from the bisection with the lowest total SSE.
9: Add these two clusters to the list of clusters.
10: **until** Until the list of clusters contains K clusters.

Bisecting K-means has less trouble with initialization because it performs several trial bisections and takes the one with the lowest SSE, and because there are only two centroids at each step.

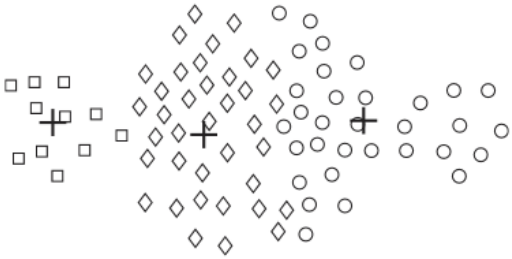
K-Means and Different Types of Clusters:

K-means and its variations have a number of limitations with respect to finding different types of clusters. K-means has difficulty:

- In detecting the "natural" clusters
- In detecting clusters when clusters have non-spherical shapes or widely different in sizes
- In detecting clusters when clusters have non-spherical shapes or widely different in densities

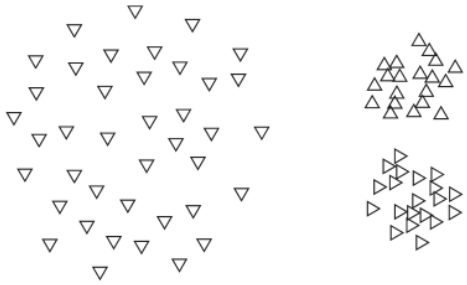


(a) Original points.

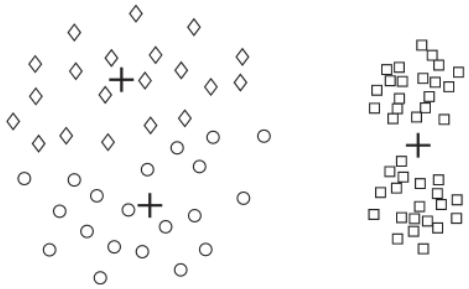


(b) Three K-means clusters.

K-means with clusters of different size.

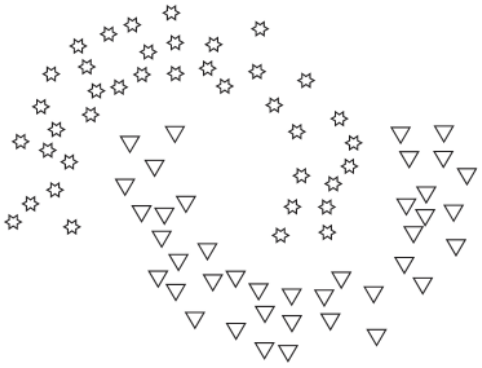


(a) Original points.

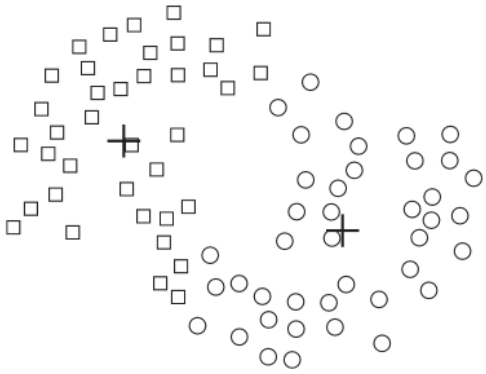


(b) Three K-means clusters.

K-means with clusters of different density.



(a) Original points.



(b) Two K-means clusters.

K-means with non-globular clusters.

K-Means Strengths & Weakness:

- K-means is simple and can be used for a wide variety of data types.
- It is quite efficient, even though multiple runs are often performed.
- Bisecting K-means, are even more efficient, and are less susceptible to initialization problems.

- It cannot handle no-globular clusters of different sizes and densities
- Faces problems if data set contains outliers
- It is restricted to the data which has centroid

Note: K-Means is not suitable for some type of data

K-Means as an Optimization Problem:

Minimizing SSE is the objective of K-Means. Gradient Descent Approach is used for minimizing SSE. It is a two-step process. Computes the change to the solution that best optimizes the objective function and then update the solution.

Derivation of K-means as an Algorithm to Minimize the SSE

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2$$

$$\begin{aligned} \frac{\partial}{\partial c_k}\mathrm{SSE} &= \frac{\partial}{\partial c_k}\sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2 \\ &= \sum_{i=1}^{K} \sum_{x \in C_i} \frac{\partial}{\partial c_k}(c_i - x)^2 \\ &= \sum_{x \in C_k} 2 * (c_k - x_k) = 0 \end{aligned}$$

$$\sum_{x \in C_k} 2 * (c_k - x_k) = 0 \Rightarrow m_k c_k = \sum_{x \in C_k} x_k \Rightarrow c_k = \frac{1}{m_k} \sum_{x \in C_k} x_k$$

The best centroid for minimizing the SSE of a cluster is the mean of the points in the cluster.

Derivation of K-means for SAE (Sum of Absolute Error):

$$\mathrm{SAE} = \sum_{i=1}^{K} \sum_{x \in C_i} dist_{\mathrm{L}_1}(c_i, x)$$

$$\begin{aligned}\frac{\partial}{\partial c_k}\text{SAE} &= \frac{\partial}{\partial c_k}\sum_{i=1}^{K}\sum_{x\in C_i}|c_i - x| \\ &= \sum_{i=1}^{K}\sum_{x\in C_i}\frac{\partial}{\partial c_k}|c_i - x| \\ &= \sum_{x\in C_k}\frac{\partial}{\partial c_k}|c_k - x| = 0\end{aligned}$$

$$\sum_{x\in C_k}\frac{\partial}{\partial c_k}|c_k - x| = 0 \Rightarrow \sum_{x\in C_k} sign(x - c_k) = 0$$

Agglomerative Hierarchical Clustering:

There are two basic approaches for generating a hierarchical clustering:

Agglomerative:

Start with the points as individual clusters and, at each step, merge the closest pair of clusters. This requires defining a notion of cluster proximity.

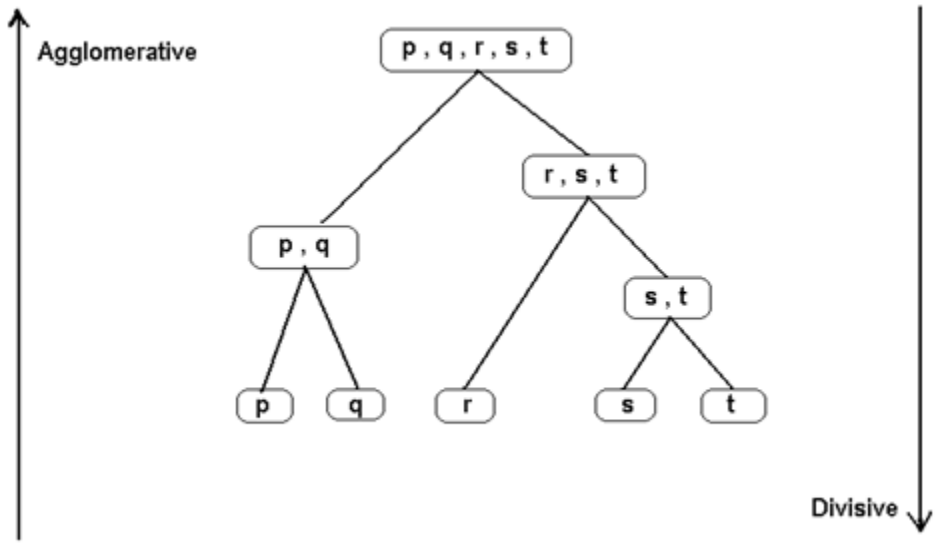
- In HAC Data objects are grouped in a bottom-up fashion.
- Initially each data object is in its own cluster.
- Then HAC merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied.
- For HAC termination condition can be specified by the user, as the desired number of clusters

Divisive:

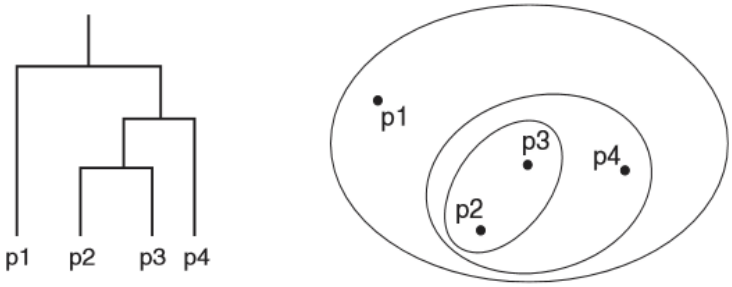
Start with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain. In this case, we need to decide which cluster to split at each step and how to do the splitting.

- In this data objects are grouped in a top down manner
- Initially all objects are in one cluster
- Then the cluster is subdivided into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions as the desired number of clusters is obtained.

Agglomerative vs Divisive



A hierarchical clustering is often displayed graphically using a tree-like diagram called a **dendrogram**, which displays both the cluster-subcluster relationships and the order in which the clusters were merged (agglomerative view) or split (divisive view). A hierarchical clustering can also be graphically represented using a **nested cluster diagram**.



(a) Dendrogram. (b) Nested cluster diagram.

A hierarchical clustering of four points shown as a dendrogram and as nested clusters.

Basic Agglomerative Hierarchical Clustering Algorithm

Algorithm Basic agglomerative hierarchical clustering algorithm.

1: Compute the proximity matrix, if necessary.
2: **repeat**
3: Merge the closest two clusters.
4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5: **until** Only one cluster remains.

Time and Space Complexity

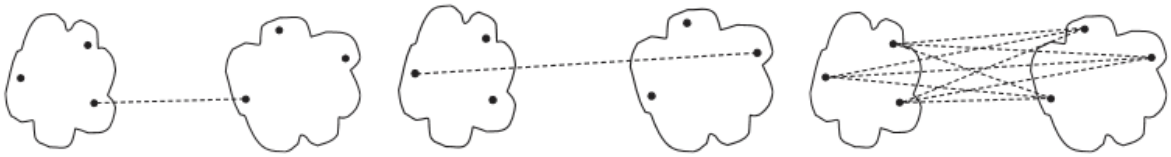
Storage of $\frac{1}{2}m^2$ proximity , m – no. of data points

Total space complexity = $O(m^2)$

Time Complexity =$O(m^2 \log(m))$

Defining Proximity between Clusters/ Agglomerative hierarchical clustering-Specific Techniques

- Key operation is , the computation of proximity between two clusters
- Cluster proximity differentiates various agglomerative hierarchical techniques



(a) MIN (single link.) (b) MAX (complete link.) (c) Group average.

MIN (single link):

Single link: smallest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \min\{d(x_{ip}, x_{jq})\}$

MAX (complete link) OR CLIQUE:

Complete link: largest distance between an element in one cluster and an element in the other, i.e., $d(C_i, C_j) = \max\{d(x_{ip}, x_{jq})\}$

Group Average (average link):

Average link: average distance between elements in one cluster and elements in the other, i.e., $d(C_i, C_j) = \text{avg}\{d(x_{ip}, x_{jq})\}$

EXAMPLE

Given a data set of five objects characterized by a single feature, assume that there are two clusters: C1: {a, b} and C2: {c, d, e}.

The distance matrix is as follows:

	a	**b**	**c**	**d**	**e**
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

Calculation of the distance between C1: {a, b} and C2: {c, d, e} is as follows:

Single link:

$dist(C_1, C_2)$ = min {d(a, c), d(a, d), d(a, e), d(b, c), d(b, d), d(b, e)}
= min {3, 4, 5, 2, 3, 4}
= 2

Complete Link:

$dist(C_1, C_2)$ = max {d(a, c), d(a, d), d(a, e), d(b, c), d(b, d), d(b, e)}
= max {3, 4, 5, 2, 3, 4}
= 5

Average Link:

$dist(C_1, C_2)$ = avg {d(a, c), d(a, d), d(a, e), d(b, c), d(b, d), d(b, e)}
= avg {3, 4, 5, 2, 3, 4}
= (3+4+5+2+3+4)/6
= 3.5

EXAMPLE 1:

Consider the following distance matrix for five points and generate clusters using Agglomerative Hierarchical Clustering

	a	**b**	**c**	**d**	**e**
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

Single Link (MIN)

Distance Matrix	Merged Clusters	Dendogram	Nested Cluster Diagram

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

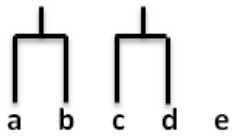
Min Distance = 1

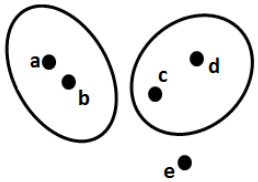
C1 :

{a, b}

C2 :

{c, d} or {d, e}



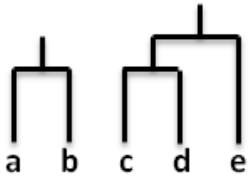


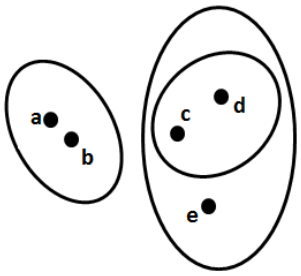
	C1 : {a, b}	C2 : {c, d}	e
C1 : {a, b}	0	2	4
C2 : {c, d}	2	0	1
e	4	1	0

d(C1, C2) = Min{ d(a,c), d(a,d), d(b,c), d(b,d)}
= Min{3, 4, 2, 3} = 2
d(C1, e} = Min{ d(a,e), d(b,e)}
= Min{5, 4} = 4
d(C2, e} = Min{d(c,e), d(d,e)}
= Min{2, 1} = 1

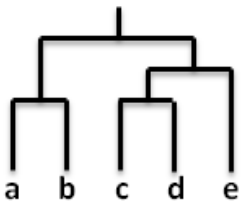
C2 and e :

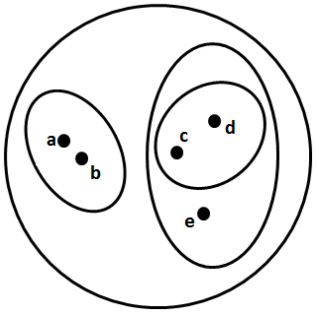
{{c, d}, e}





Final Single Cluster ->





Complete Link (MAX)

Distance Matrix

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

Min Distance = 1

Merged Clusters

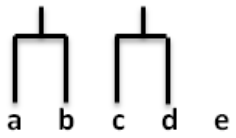
C1 :

{a, b}

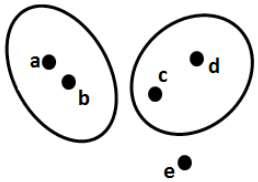
C2 :

{c, d} or {d, e}

Dendogram



Nested Cluster Diagram

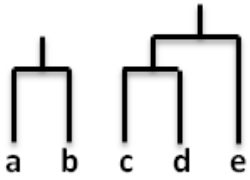


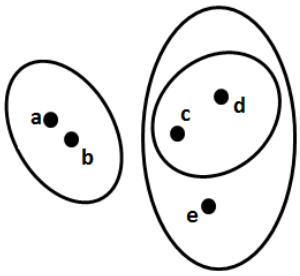
	C1 : {a, b}	**C2 : {c, d}**	**e**
C1 : {a, b}	0	4	5
C2 : {c, d}	4	0	2
e	5	2	0

d(C1, C2) = Max{ d(a,c), d(a,d), d(b,c), d(b,d)}
= Max{3, 4, 2, 3} = 4
d(C1, e} = Max{ d(a,e), d(b,e)}
= Max{5, 4} = 5
d(C2, e} = Max{d(c,e), d(d,e)}
= Max{2, 1} = 2

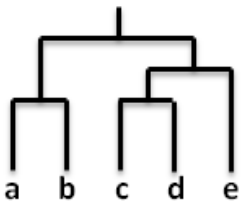
C2 and e :

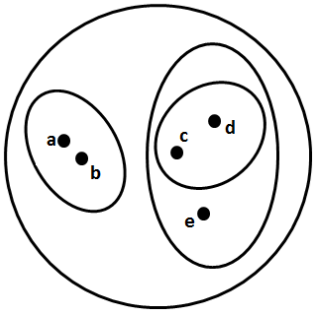
{{c, d}, e}





Final Single Cluster ->





Average Link (AVERAGE)

Distance Matrix

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
d	4	3	1	0	1
e	5	4	2	1	0

Merged Clusters

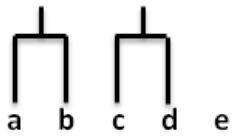
C1 :

{a, b}

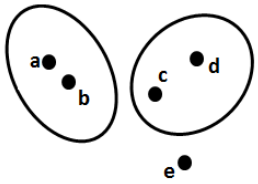
C2 :

{c, d} or {d, e}

Dendogram



Nested Cluster Diagram

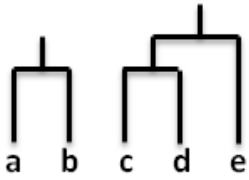


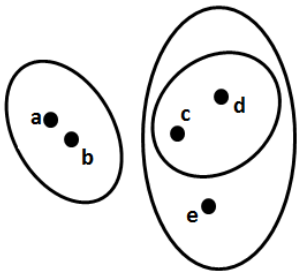
	C1 : {a, b}	**C2 : {c, d}**	**e**
C1 : {a, b}	0	3	4.5
C2 : {c, d}	3	0	1.5
e	4.5	1.5	0

d(C1, C2) = Avg{ d(a,c), d(a,d), d(b,c), d(b,d)}
= Avg{3, 4, 2, 3} = 3
d(C1, e} = Avg{ d(a,e), d(b,e)}
= Avg{5, 4} = 4.5
d(C2, e} = Avg{d(c,e), d(d,e)}
= Avg{2, 1} = 1.5

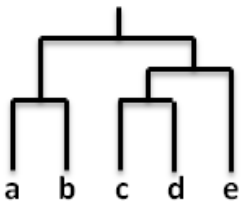
C2 and e :

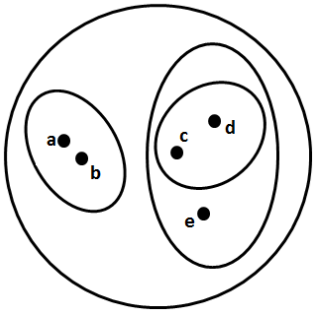
{{c, d}, e}





Final Single Cluster ->

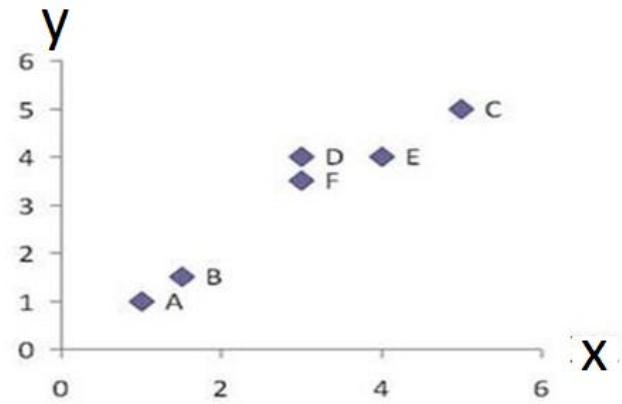




EXAMPLE 2

Given Proximity matrix can be represented as:

	x	y
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5



Distance Matrix is calculated by $d(A,B) = \sqrt{(A_X - B_X)^2 + (A_Y - B_Y)^2}$

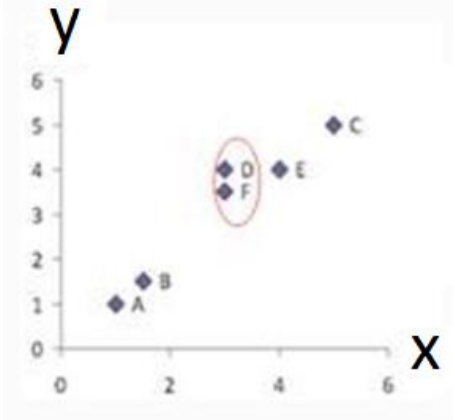
USING SINGLE LINK

Distance Matrix for the given Proximity Matrix is:

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

Minimum Distance from the above distance matrix = 0.50

Merge the clusters containing F and D

Dendogram	Nest Cluster Diagram
	

Calculate Distance Matrix for clusters A, B, C, (D, F) and E

	A	B	C	(D, F)	E
A	0.0	0.71	5.66	3.20	4.24
B	0.71	0.0	4.95	2.50	3.54
C	5.66	4.95	0.0	2.24	1.41
(D, F)	3.20	2.50	2.24	0.0	1.00
E	4.24	3.54	1.41	1.00	0.0

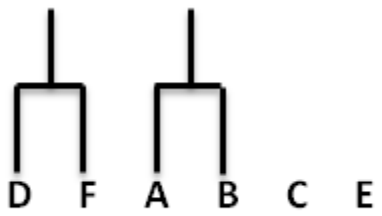
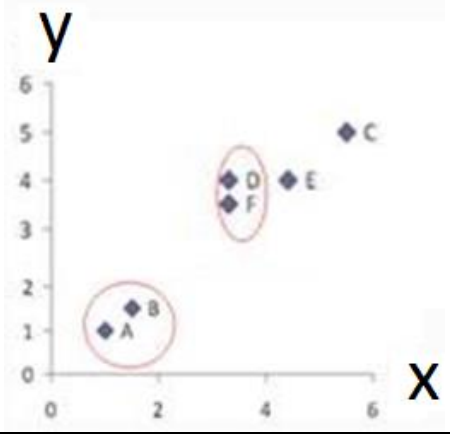
d((D,F), A) = Min{ d(D, A) , d(F, A)} = Min{3.61, 3.20} = 3.20

d((D,F), B) = Min{ d(D, B) , d(F, B)} = Min{2.92, 2.50} = 2.50

d((D,F), C) = Min{ d(D, C) , d(F, C)} = Min{2.24, 2.50} = 2.24

d((D,F), E) = Min{ d(D, E) , d(F, E)} = Min{1.00, 1.12} = 1.00

Minimum distance from the above distance matrix is 0.71. Merge the clusters containing A and B

Dendogram	Nest Cluster Diagram
	

Calculate Distance Matrix for clusters (A, B), C, (D, F) and E:

	(A, B)	C	(D, F)	E
(A, B)	0.00	4.95	2.50	3.54
C	4.95	0.00	2.24	1.41
(D, F)	2.50	2.24	0.00	1.00
E	3.54	1.41	1.00	0.00

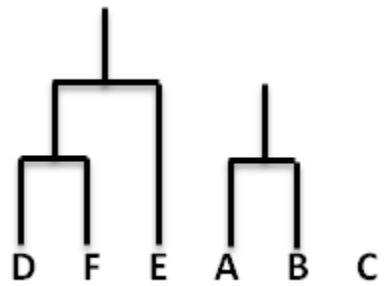
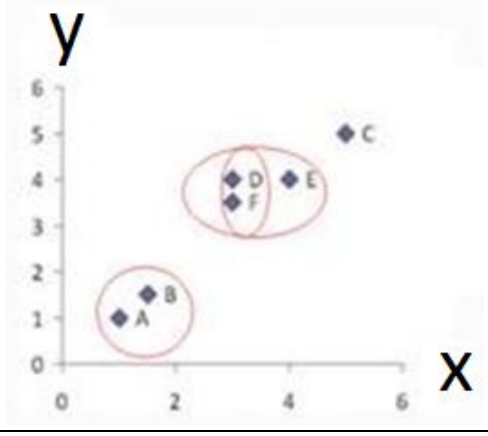
d((A,B), C) = Min{ d(A, C) , d(B, C)} = Min{5.66, 4.95} = 4.95

d((A, B) , (D,F)) = Min{ d(A, D) , d(A, F) , d(B, D) , d(B, F) } = Min{3.61, 3.20, 2.92, 2.50}=2.50

d((A,B), E) = Min{ d(A, E) , d(B, E)} = Min{4.24, 3.54} = 3.54

d((D,F), C) = Min{ d(D, C) , d(F, C)} = Min{2.24, 2.50} = 2.24

Minimum distance from the above distance matrix is 1.00. Merge the clusters containing (D, F) and E

Dendogram	Nest Cluster Diagram
	

Calculate Distance Matrix for clusters (A, B), C and ((D, F), E):

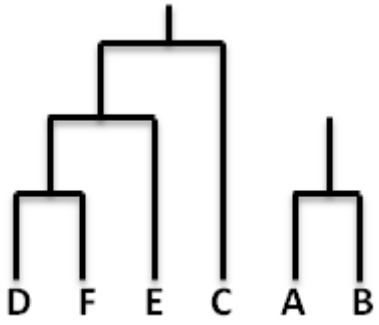
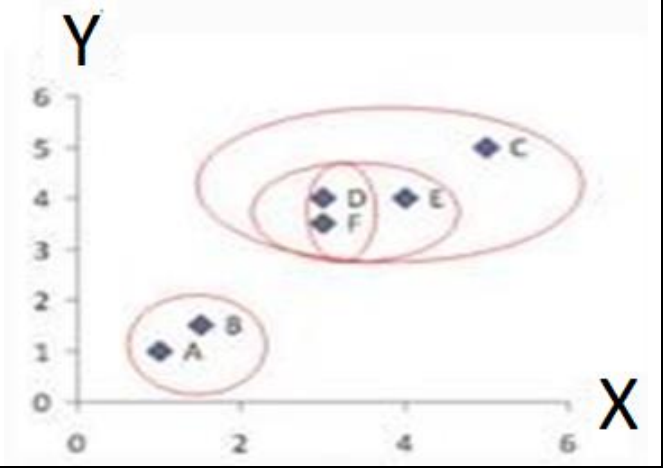
	(A, B)	C	((D, F), E)
(A, B)	0.00	4.95	2.50
C	4.95	0.00	1.41
((D, F), E)	2.50	1.41	0.00

d((A,B), C) = Min{ d(A, C) , d(B, C)} = Min{5.66, 4.95} = 4.95

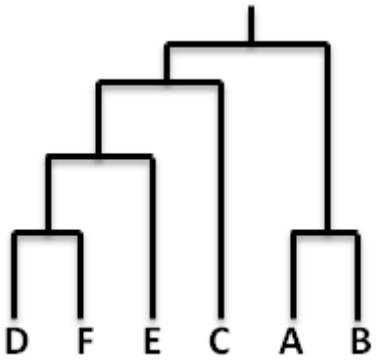
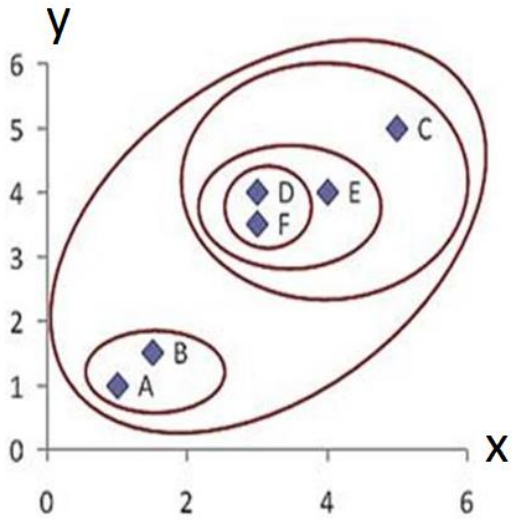
d((A, B) , ((D,F), E)) = Min{ d(A, D) , d(A, F) , d(A, E), d(B, D) , d(B, F), d(B, E) }
= Min{3.61, 3.20, 4.24, 2.92, 2.50, 3.54}=2.50

d(((D,F), E), C) = Min{ d(D, C) , d(F, C), d(E, C) } = Min{2.24, 2.50, 1.41} = 1.41

Minimum distance from the above distance matrix is 1.41.
Merge the clusters containing ((D, F), E) and C.

Dendogram	Nest Cluster Diagram
	

Final Dengogram and Nest Cluster Diagram

Dendogram	Nest Cluster Diagram
	

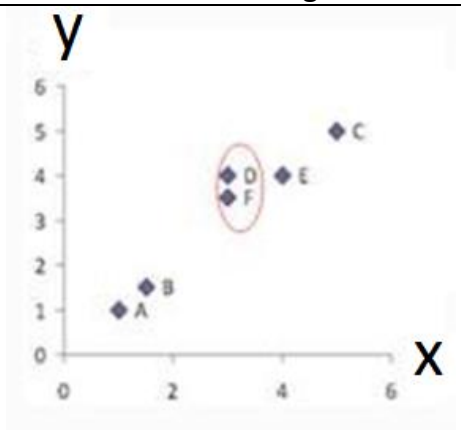
USING COMPLETE LINK

Distance Matrix for the given Proximity Matrix is:

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

Minimum Distance from the above distance matrix = 0.50

Merge the clusters containing F and D

Dendogram	Nest Cluster Diagram
	

Calculate Distance Matrix for clusters A, B, C, (D, F) and E

	A	B	C	(D, F)	E
A	0.0	0.71	5.66	3.61	4.24
B	0.71	0.0	4.95	2.92	3.54
C	5.66	4.95	0.0	2.50	1.41
(D, F)	3.61	2.92	2.50	0.0	1.12
E	4.24	3.54	1.41	1.12	0.0

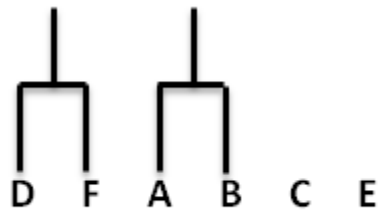
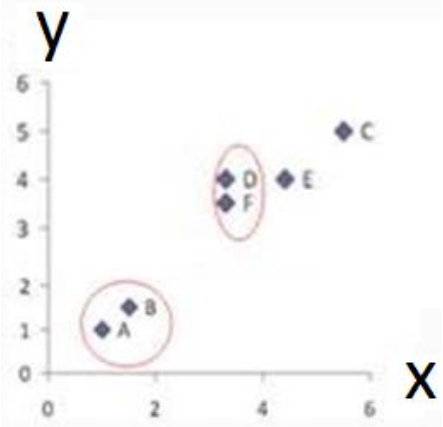
d((D,F), A) = Max{ d(D, A) , d(F, A)} = Max{3.61, 3.20} = 3.61

d((D,F), B) = Max{ d(D, B) , d(F, B)} = Max{2.92, 2.50} = 2.92

d((D,F), C) = Max{ d(D, C) , d(F, C)} = Max{2.24, 2.50} = 2.50

d((D,F), E) = Max{ d(D, E) , d(F, E)} = Max{1.00, 1.12} = 1.12

Minimum distance from the above distance matrix is 0.71. Merge the clusters containing A and B

Dendogram	Nest Cluster Diagram
	

Calculate Distance Matrix for clusters (A, B), C, (D, F) and E:

	(A, B)	C	(D, F)	E
(A, B)	0.00	5.66	3.61	4.24
C	5.66	0.00	2.50	1.41
(D, F)	3.61	2.50	0.00	1.00
E	4.24	1.41	1.00	0.00

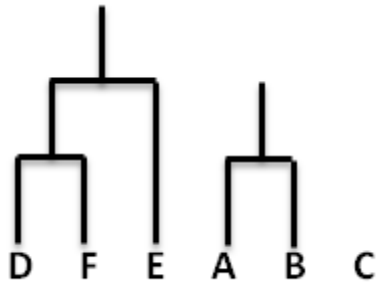
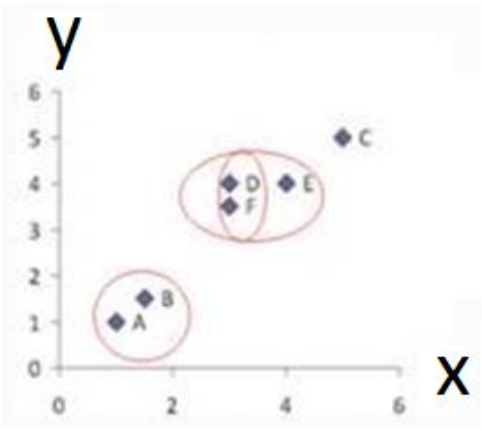
d((A,B), C) = Max{ d(A, C) , d(B, C)} = Max{5.66, 4.95} = 5.66

d((A, B) , (D,F)) = Max{ d(A, D) , d(A, F) , d(B, D) , d(B, F) } = Max{3.61, 3.20, 2.92, 2.50}=3.61

d((A,B), E) = Max{ d(A, E) , d(B, E)} = Max{4.24, 3.54} = 4.24

d((D,F), C) = Max{ d(D, C) , d(F, C)} = Max{2.24, 2.50} = 2.50

Minimum distance from the above distance matrix is 1.00. Merge the clusters containing (D, F) and E

Dendogram	Nest Cluster Diagram
	

Calculate Distance Matrix for clusters (A, B), C and ((D, F), E):

	(A, B)	C	((D, F), E)
(A, B)	0.00	5.66	4.24
C	5.66	0.00	2.50
((D, F), E)	4.24	2.50	0.00

d((A,B), C) = Max{ d(A, C) , d(B, C)} = Max{5.66, 4.95} = 5.66

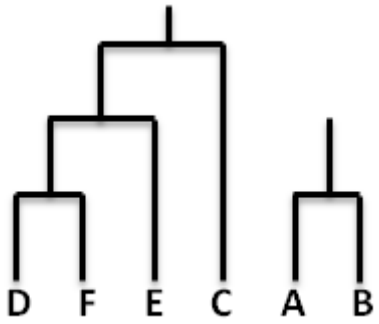
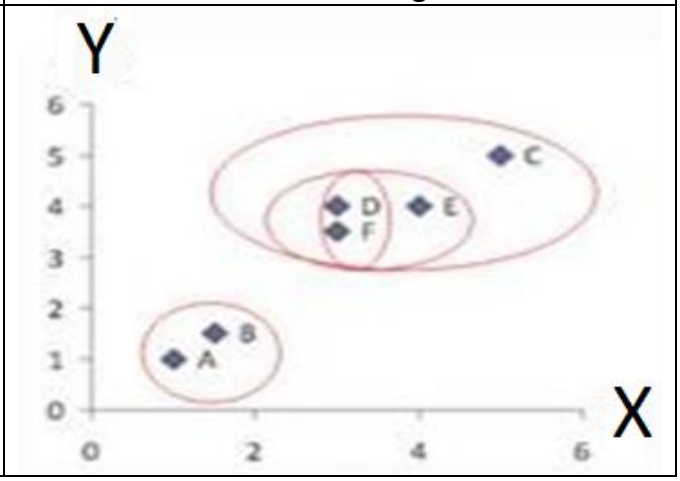
d((A, B) , ((D,F), E)) = Max{ d(A, D) , d(A, F) , d(A, E), d(B, D) , d(B, F), d(B, E) }

= Max{3.61, 3.20, 4.24, 2.92, 2.50, 3.54}=4.24

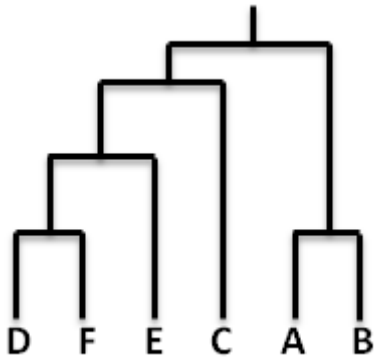
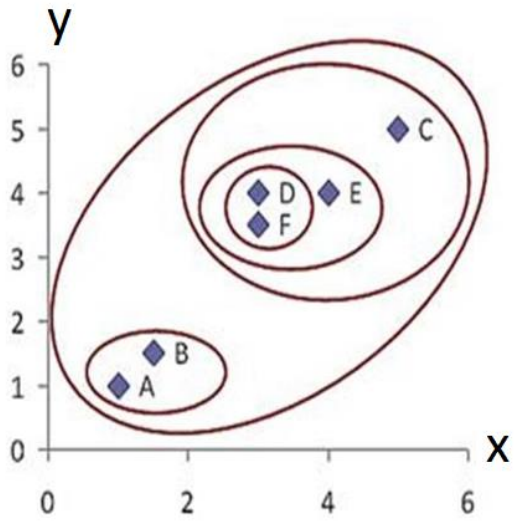
d(((D,F), E), C) = Max{ d(D, C) , d(F, C), d(E, C) } = Max{2.24, 2.50, 1.41} = 2.50

Minimum distance from the above distance matrix is 2.50.

Merge the clusters containing ((D, F), E) and C.

Dendogram	Nest Cluster Diagram
	

Final Dengogram and Nest Cluster Diagram

Dendogram	Nest Cluster Diagram
	

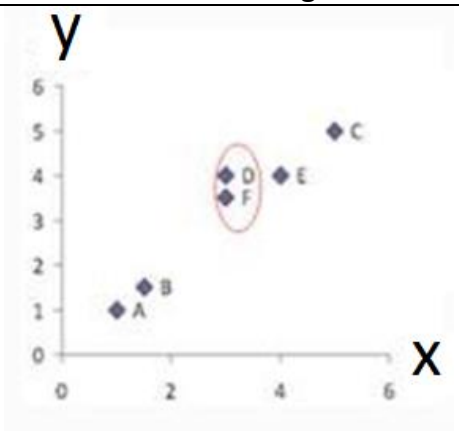
USING AVERAGE LINK

Distance Matrix for the given Proximity Matrix is:

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

Minimum Distance from the above distance matrix = 0.50

Merge the clusters containing F and D

Dendogram	Nest Cluster Diagram
	

Calculate Distance Matrix for clusters A, B, C, (D, F) and E

	A	B	C	(D, F)	E
A	0.0	0.71	5.66	3.405	4.24
B	0.71	0.0	4.95	2.71	3.54
C	5.66	4.95	0.0	2.37	1.41
(D, F)	3.405	2.71	2.37	0.0	1.06
E	4.24	3.54	1.41	1.06	0.0

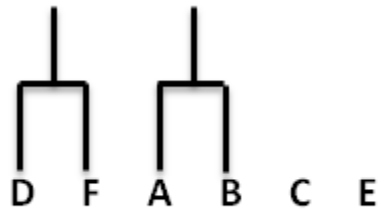
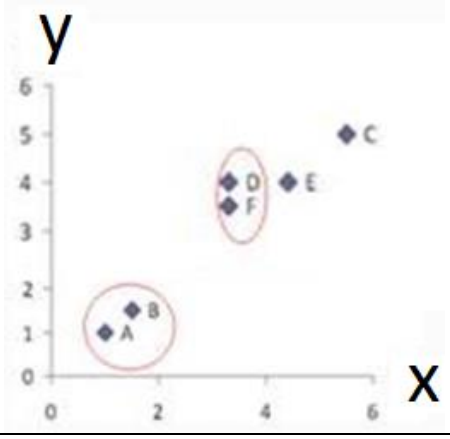
d((D,F), A) = AVG{ d(D, A) , d(F, A)} = AVG{3.61, 3.20} = 3.405

d((D,F), B) = AVG{ d(D, B) , d(F, B)} = AVG{2.92, 2.50} = 2.71

d((D,F), C) = AVG{ d(D, C) , d(F, C)} = AVG{2.24, 2.50} = 2.37

d((D,F), E) = AVG{ d(D, E) , d(F, E)} = AVG{1.00, 1.12} = 1.06

Minimum distance from the above distance matrix is 0.71. Merge the clusters containing A and B

Dendogram	Nest Cluster Diagram
	

Calculate Distance Matrix for clusters (A, B), C, (D, F) and E:

	(A, B)	C	(D, F)	E
(A, B)	0.00	5.305	3.0575	2.89
C	5.305	0.00	2.37	1.41
(D, F)	3.0575	2.37	0.00	1.00
E	2.89	1.41	1.00	0.00

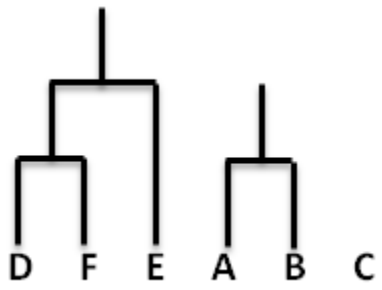
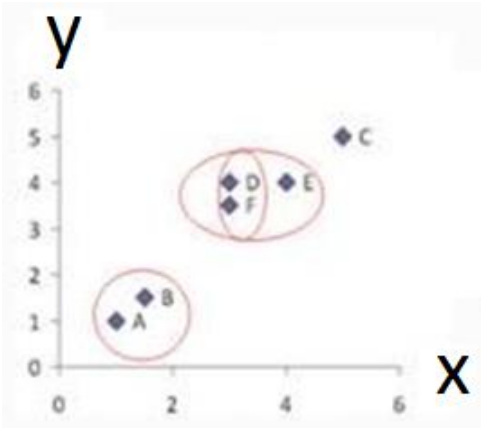
d((A,B), C) = AVG{ d(A, C) , d(B, C)} = AVG{5.66, 4.95} = 5.305

d((A, B) , (D,F)) = AVG{ d(A, D) , d(A, F) , d(B, D) , d(B, F) } = AVG{3.61, 3.20, 2.92, 2.50}=3.0575

d((A,B), E) = AVG{ d(A, E) , d(B, E)} = AVG{4.24, 3.54} = 2.89

d((D,F), C) = AVG{ d(D, C) , d(F, C)} = AVG{2.24, 2.50} = 2.37

Minimum distance from the above distance matrix is 1.00. Merge the clusters containing (D, F) and E

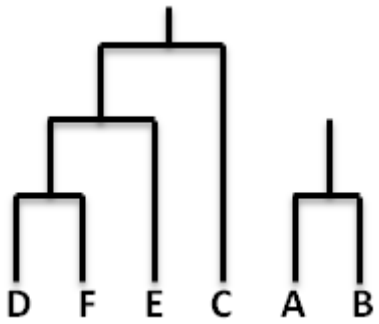
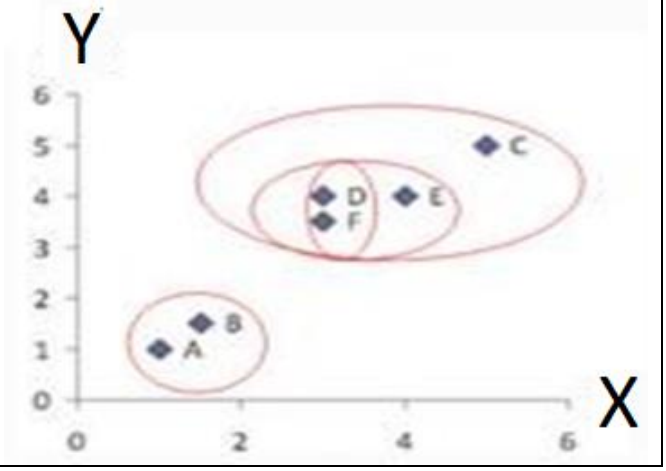
Dendogram	Nest Cluster Diagram
	

Calculate Distance Matrix for clusters (A, B), C and ((D, F), E):

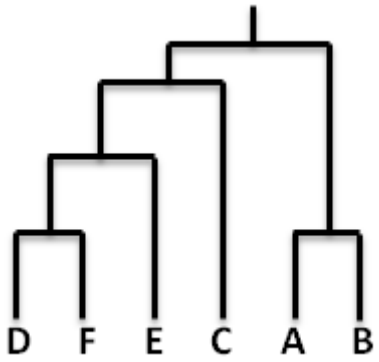
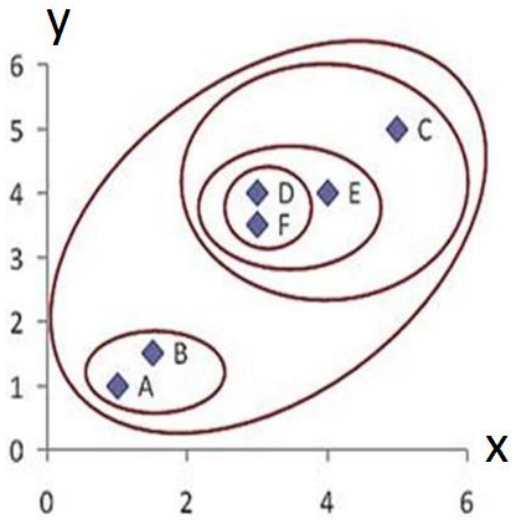
	(A, B)	C	((D, F), E)
(A, B)	0.00	5.305	4.24
C	5.305	0.00	2.05
((D, F), E)	4.24	2.05	0.00

d((A,B), C) = AVG{ d(A, C) , d(B, C)} = AVG{5.66, 4.95} = 5.305
d((A, B) , ((D,F), E)) = AVG{ d(A, D) , d(A, F) , d(A, E), d(B, D) , d(B, F), d(B, E) }
= AVG{3.61, 3.20, 4.24, 2.92, 2.50, 3.54}=3.335
d(((D,F), E), C) = AVG{ d(D, C) , d(F, C), d(E, C) } = AVG{2.24, 2.50, 1.41} = 2.05

Minimum distance from the above distance matrix is 2.05.
Merge the clusters containing ((D, F), E) and C.

Dendogram	Nest Cluster Diagram
	

Final Dengogram and Nest Cluster Diagram

Dendogram	Nest Cluster Diagram
	

Key Issues in Hierarchical Clustering

Lack of a Global Objective Function

Agglomerative hierarchical clustering cannot be viewed as globally optimizing an objective function. Instead, agglomerative hierarchical clustering techniques use various criteria to decide locally, at each step, which clusters should be merged (or split for divisive approaches). This approach yields clustering algorithms that avoid the difficulty of attempting to solve a hard combinatorial optimization problem.

Ability to Handle Different Cluster Sizes

One aspect of agglomerative hierarchical clustering that we have not yet discussed is how to treat the relative sizes of the pairs of clusters that are merged. There are two approaches: weighted, which treats all clusters equally, and unweighted, which takes the number of points in each cluster into account.

Merging Decisions Are Final

Agglomerative hierarchical clustering algorithms tend to make good local decisions about combining two clusters since they can use information about the pairwise similarity of all points. However, once a decision is made to merge two clusters, it cannot be undone at a later time. This approach prevents a local optimization criterion from becoming a global optimization criterion.

Strengths and Weaknesses

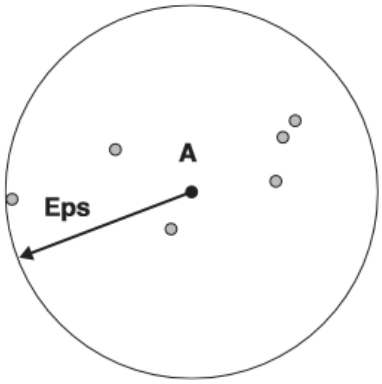
- Produce better-quality clusters
- Agglomerative hierarchical clustering algorithms are expensive in terms of their computational and storage requirements.
- All merges are final can also cause trouble for noisy, high-dimensional data, such as document data.

DBSCAN

Density-based clustering locates regions of high density that are separated from one another by regions of low density. DBSCAN is a simple and effective density-based clustering algorithm that illustrates a number of important concepts that are important for any density-based clustering approach.

Traditional Density: Center-Based Approach

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. This technique is graphically shown below. The number of points within a radius of Eps of point A is 7, including A itself.

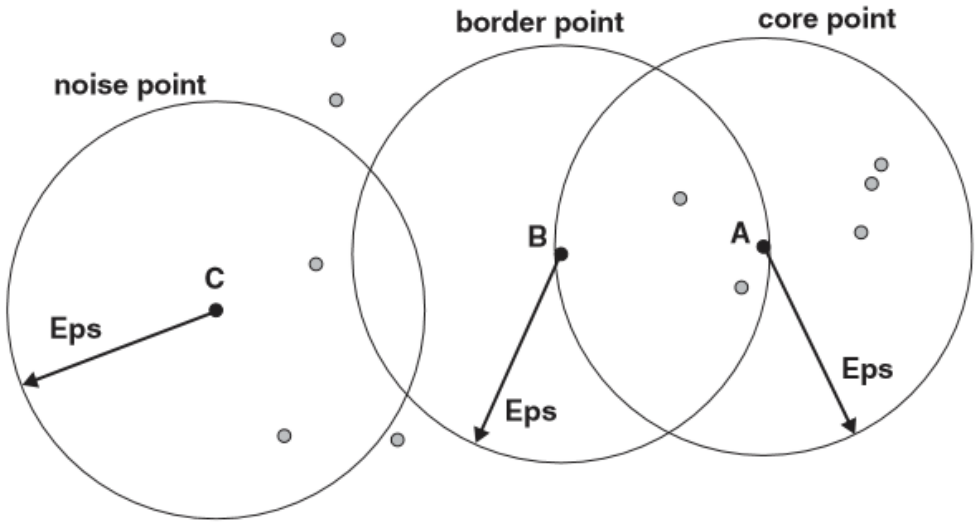


Center-based density.

This method is simple to implement, but the density of any point will depend on the specified radius. For instance, if the radius is large enough, then all points will have a density of m, the number of points in the data set. Likewise, if the radius is too small, then all points will have a density of 1. Selecting appropriate radius is crucial.

Classification of Points According to Center-Based Density

The center-based approach to density allows us to classify a point as being (1) in the interior of a dense region (a core point), (2) on the edge of a dense region (a border point), or (3) in a sparsely occupied region (a noise or background point). The following figure graphically illustrates the concepts of core, border, and noise points using a collection of two-dimensional points.



Core, border, and noise points.

Core points:

These points are in the interior of a density-based cluster. A point is a core point if the number of points within a given neighborhood around the point as determined by the distance function and a user specified distance parameter, Eps, exceeds a certain threshold, MinPts, which is also a user-specified parameter. In the above figure, point A is a core point, for the indicated radius (Eps) ifMinPts ≤ 7.

Border points:

A border point is not a core point, but falls within the neighborhood of a core point. In the above figure, point B is a border 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. In the above figure, point C is a noise point.

The DBSCAN Algorithm

- Any two core points that are close enough—within a distance Eps of one another—are put in the same cluster.
- Likewise, any border point that is close enough to a core point is put in the same cluster as the core point.
- Ties may need to be resolved if a border point is close to core points from different clusters.
- Noise points are discarded.

Algorithm 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.

NOTE:

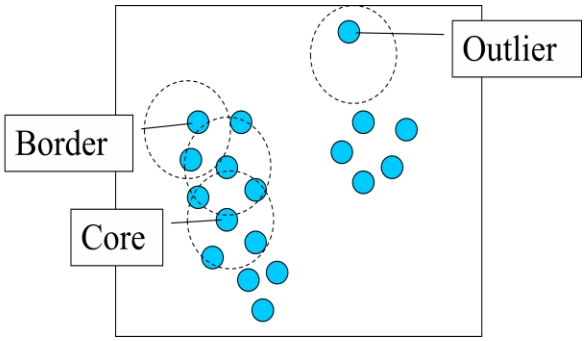
- DBSCAN Clustering is based on density (local cluster criterion)
- It is used to discover clusters of arbitrary shape
- DBSCAN handles noise very well.
- DBSCAN needs only one scan to cluster the data points
- We can provide termination condition as density parameters.
- The Two parameters that we can specify are:
  - ***Eps:*** Maximum radius of neighborhood
  - ***MinPts:*** Minimum number of points in an Eps-neighborhood of a point
- The neighborhood of the data point is represented as:
  - $N_{Eps}(p) = \{q \in D \mid dist(p,q) <= Eps\}$
- The above statement can be represented as, if the distance between q and p is less than or equal to the given radius, then the data point belongs to cluster.

Different measurements of the DBSCAN are:

- Identifying core point, border point and outlier.
- Directly density-reachable.
- Density Connectivity.

Identifying core point, border point and outlier:

- A point is a core point if it has more than a specified number of points (MinPts) within Eps. These are points that are at the interior of a cluster.
- A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.
- A noise point is any point that is not a core point nor a border point.

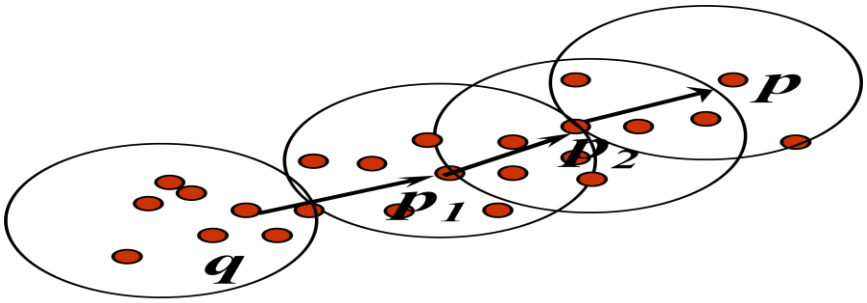


Directly density-reachable:

A point ***p*** is directly density-reachable from a point ***q*** wrt. ***Eps***, ***MinPts*** iff

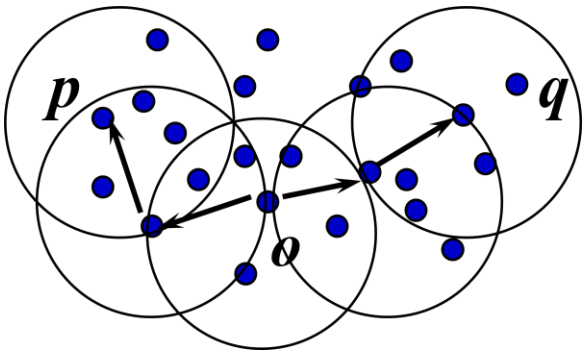
1) p belongs to $N_{Eps}(q)$

2) q is a core point: $|N_{Eps}(q)| >= MinPts$



Density-Connectivity:

- A pair of points p and q are density-connected if they are commonly density-reachable from a point o.
- Density-connectivity is symmetric



Strengths and weakness of DBSCAN:

Strengths:

- Resistant to Noise
- Can handle clusters of different shapes and sizes

Weakness:

- DBSCAN is sensitive to Parameters like *€' (*Radius) and minpoints.
- The size and shape of the clusters vary from one another based on the parameters given.
- DBSCAN does not work well with varying densities.
- It does not work well with High-dimensional data.