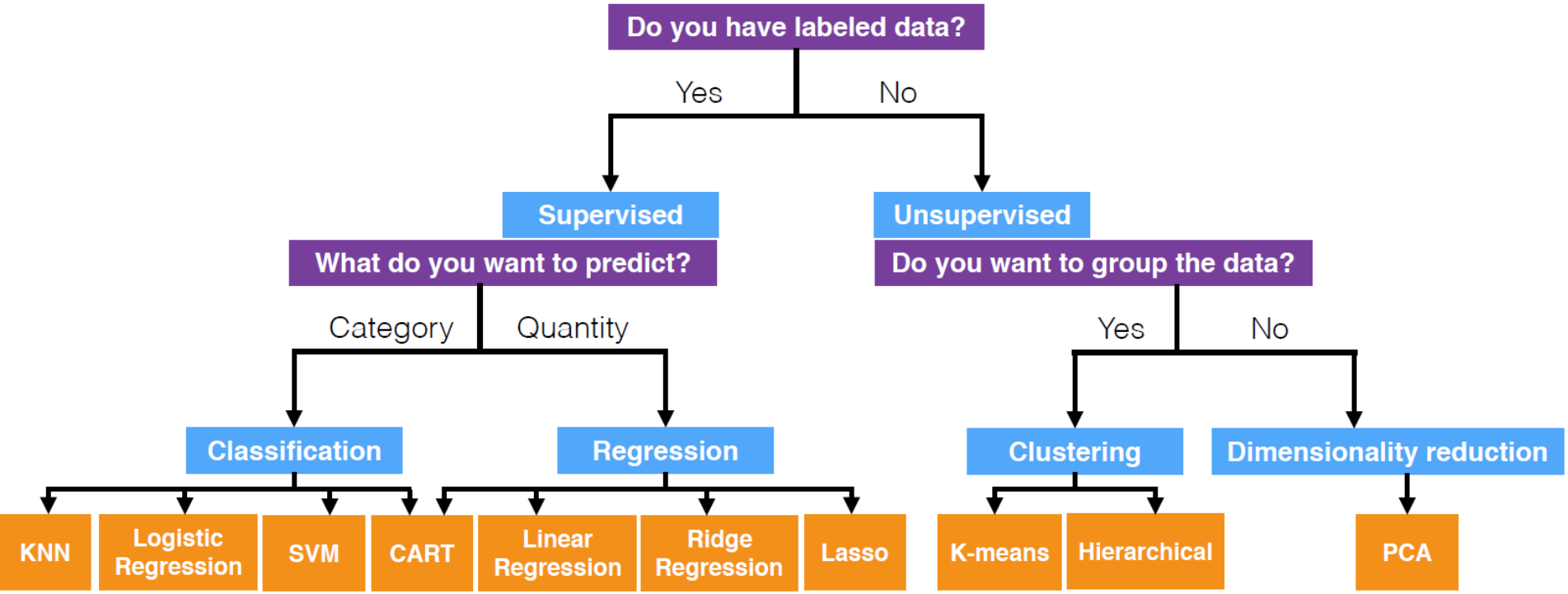
UNIT IV	UNSUPERVISED LEARNING	9+6

Clustering – Types – Applications - Partitioning Methods – K-means Algorithm – K-Medoids – Hierarchical methods – Density based methods DBSCAN – Finding patterns using Association Rules – Hidden Markov Model.

Machine Learning Methods



Unsupervised Learning

Recall: A set of statistical tools for data that only has features/input available, but no response.

In other words, we have X's but no labels y.

Goal: Discover interesting patterns/properties of the data.

- E.g. for visualizing or interpreting high-dimensional data.

Challenges of Unsupervised Learning

Why is unsupervised learning challenging?

- Exploratory data analysis — goal is not always clearly defined
- Difficult to assess performance — “right answer” unknown
- Working with high-dimensional data

Types of Unsupervised Learning

Two approaches:

- **Cluster analysis**

  - For identifying homogenous subgroups of samples

- **Dimensionality reduction**

  - For finding a low-dimensional representation to characterize and visualize the data

Clustering

- Clustering is a technique for finding similarity groups in data, called **clusters**.

I.e., it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.

- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.

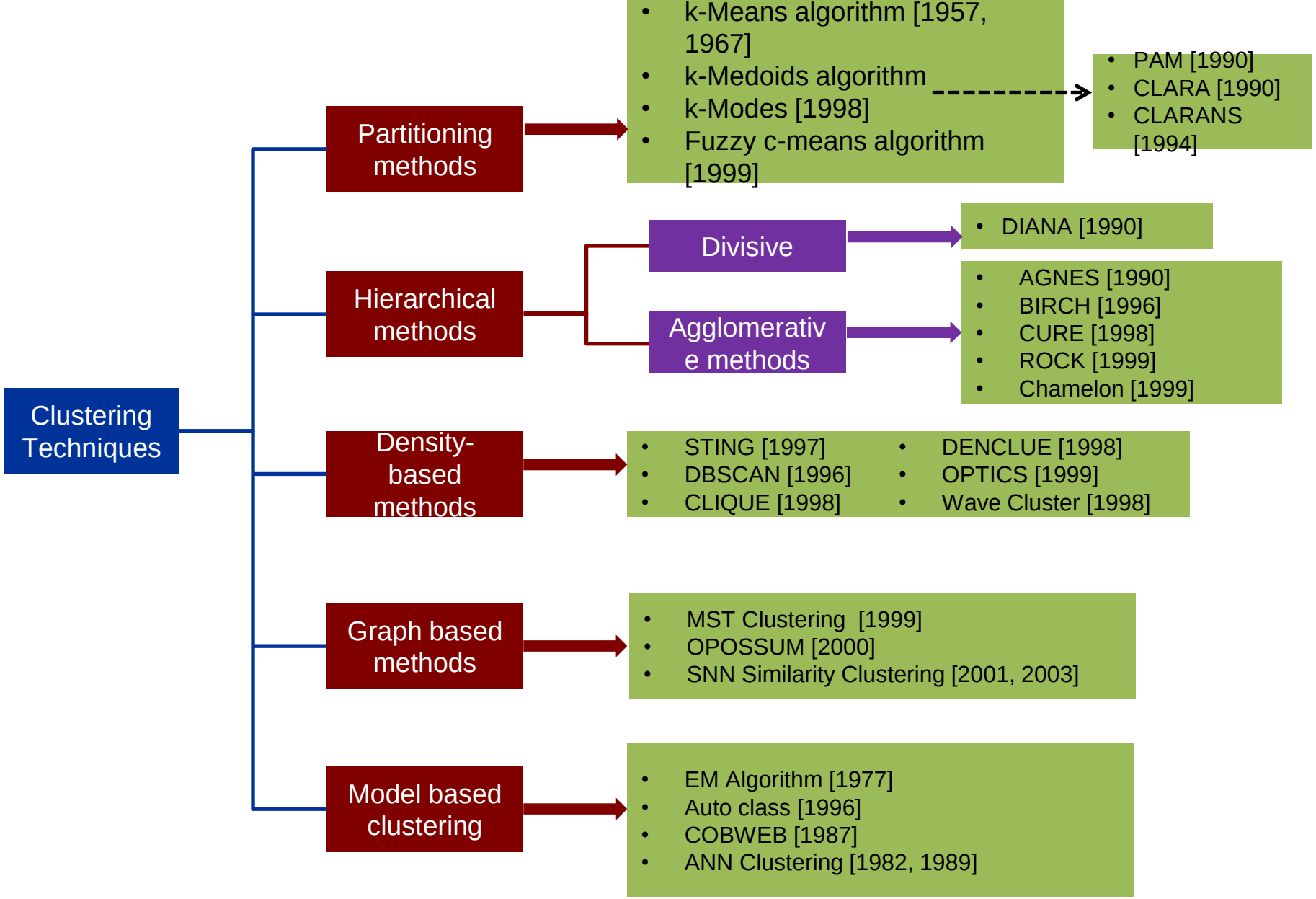

Clustering Techniques
Partitioning methods
k-Means algorithm [1957, 1967]
k-Medoids algorithm
k-Modes [1998]
Fuzzy c-means algorithm [1999]
PAM [1990]
CLARA [1990]
CLARANS [1994]
Hierarchical methods
Divisive
DIANA [1990]
Agglomerative methods
AGNES [1990]
BIRCH [1996]
CURE [1998]
ROCK [1999]
Chamelon [1999]
Density-based methods
STING [1997]
DBSCAN [1996]
CLIQUE [1998]
DENCLUE [1998]
OPTICS [1999]
Wave Cluster [1998]
Graph based methods
MST Clustering [1999]
OPOSSUM [2000]
SNN Similarity Clustering [2001, 2003]
Model based clustering
EM Algorithm [1977]
Auto class [1996]
COBWEB [1987]
ANN Clustering [1982, 1989]

Table 9.1 *Different Clustering Methods*

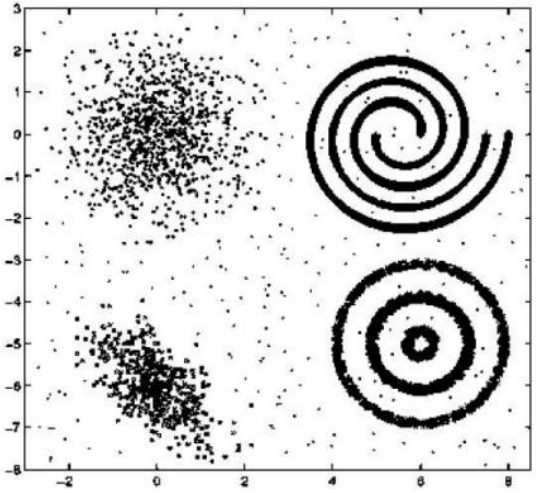
Method	Characteristics
Partitioning methods	• Uses mean or medoid (etc.) to represent cluster centre • Adopts distance-based approach to refine clusters • Finds mutually exclusive clusters of spherical or nearly spherical shape • Effective for data sets of small to medium size
Hierarchical methods	• Creates hierarchical or tree-like structure through decomposition or merger • Uses distance between the nearest or furthest points in neighbouring clusters as a guideline for refinement • Erroneous merges or splits cannot be corrected at subsequent levels
Density-based methods	• Useful for identifying arbitrarily shaped clusters • Guiding principle of cluster creation is the identification of dense regions of objects in space which are separated by low-density regions • May filter out outliers

Clustering

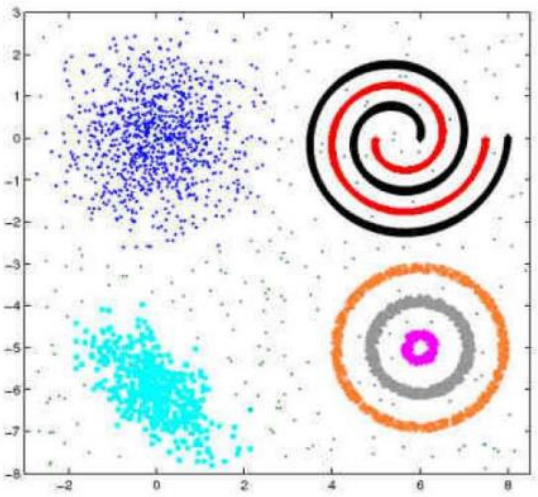
Data Clustering is an unsupervised learning problem

Given: N unlabeled examples {x1, . . . , xN }; the number of partitions K

Goal: Group the examples into K partitions



(a) Input data



(b) Desired clustering

The only information clustering uses is the similarity between examples

Clustering groups examples based of their mutual similarities

A good clustering is one that achieves: High within-cluster similarity Low inter-cluster similarity

Clustering - Similarity between Data Points

Choice of the similarity measure is very important for clustering

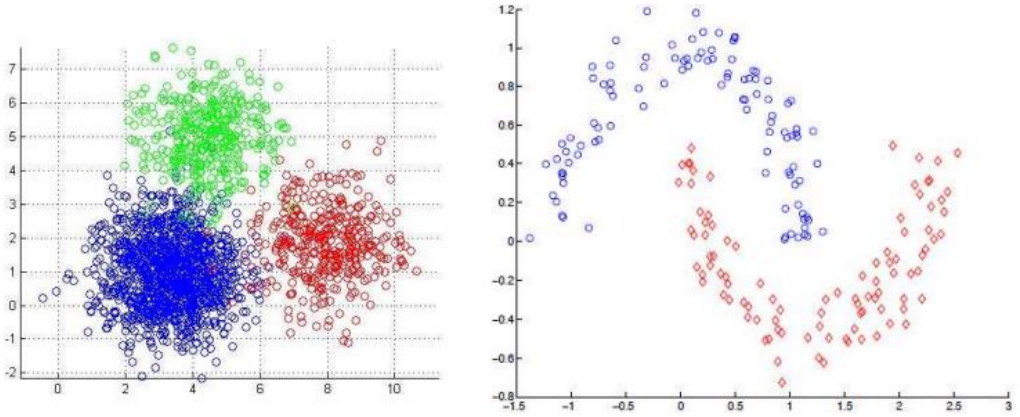
Similarity is inversely related to distance

Different ways exist to measure distances. Some examples:

Euclidean distance: $d(\mathbf{x}, \mathbf{z}) = ||\mathbf{x} - \mathbf{z}|| = \sqrt{\sum_{d=1}^{D}(x_d - z_d)^2}$

Manhattan distance: $d(\mathbf{x}, \mathbf{z}) = \sum_{d=1}^{D} |x_d - z_d|$

Kernelized (non-linear) distance: $d(\mathbf{x}, \mathbf{z}) = ||\phi(\mathbf{x}) - \phi(\mathbf{z})||$



For the left figure above, Euclidean distance may be reasonable

For the right figure above, kernelized distance seems more reasonable

Partitioning Method

Partitioning Method: This clustering method classifies the information into multiple groups based on the characteristics and similarity of the data. Its the data analysts to specify the number of clusters that has to be generated for the clustering methods. In the partitioning method when database(D) that contains multiple(N) objects then the partitioning method constructs user-specified(K) partitions of the data in which each partition represents a cluster and a particular region. There are many algorithms that come under partitioning method some of the popular ones are **K-Mean, PAM(K-Medoids), CLARA algorithm (Clustering Large Applications**) etc.

K-Mean (A centroid based Technique): The K means algorithm takes the input parameter K from the user and partitions the dataset containing N objects into K clusters so that resulting similarity among the data objects inside the group (intracluster) is high but the similarity of data objects with the data objects from outside the cluster is low (intercluster). The similarity of the cluster is determined with respect to the mean value of the cluster. It is a type of square error algorithm. At the start randomly k objects from the dataset are chosen in which each of the objects represents a cluster mean(centre). For the rest of the data objects, they are assigned to the nearest cluster based on their distance from the cluster mean. The new mean of each of the cluster is then calculated with the added data objects.

Input:

K: The number of clusters in which the dataset has to be divided
D: A dataset containing N number of objects

Output: A dataset of K clusters

Method:

1.Randomly assign K objects from the dataset(D) as cluster centres(C)
2.(Re) Assign each object to which object is most similar based upon mean values.
3.Update Cluster means, i.e., Recalculate the mean of each cluster with the updated values.
4.Repeat Step 2 until no change occurs.

Partitioning Methods

Initial Centroid:

C1 = 5

C2 = 60

C3 = 150

Select random number for iniaial centroid.

	Distance to			Cluster
Data	C1 = 5	C2 = 60	C3 = 150	
5				
10				
11				
13				
15				
35				
50				
55				
72				
92				
204				

Partitioning Methods

Initial Centroid:

C1 = 5

C2 = 60

C3 = 150

	Distance to			Cluster
Data	C1 = 5	C2 = 60	C3 = 150	
5	0	55	145	C1
10	5	50	140	C1
11	6	49	139	C1
13	8	47	137	C1
15	10	45	135	C1
35	30	25	115	C2
50	45	10	100	C2
55	50	5	95	C2
72	67	12	78	C2
92	87	32	58	C2
204	199	144	54	C3
215	210	155	65	C3

C1 = (5+10+11+13+15)/5 =10.8
C2=(35+50+55+72+92)/5=60.8
C3=(204+215)/2=209.5

Partitioning Methods

Initial Centroid:

C1 = 10.8

C2 = 60.8

C3 = 209.5

	Distance to			Cluster
Data	C1 = 10.8	C2 = 60.8	C3 = 209.5	
5	5.8	55.8	204.5	
10	0.8	50.8	199.5	
11	0.2	49.8	198.5	
13	2.2	47.8	196.5	
15	4.2	45.8	194.5	
35	24.2	25.8	174.5	
50	39.2	10.8	159.5	
55	44.2	5.8	154.5	
72	61.2	11.2	137.5	
92	81.2	31.2	117.5	
204	193.2	143.2	5.5	
215	204.2	154.2	5.5	

Partitioning Methods

Initial Centroid:

C1 = 10.8

C2 = 60.8

C3 = 209.5

New Centroid:

C1 = 14.83

C2 = 67.25

C3 = 209.5

	Distance to			Cluster
Data	C1 = 10.8	C2 = 60.8	C3 = 209.5	
5	5.8	55.8	204.5	C1
10	0.8	50.8	199.5	C1
11	0.2	49.8	198.5	C1
13	2.2	47.8	196.5	C1
15	4.2	45.8	194.5	C1
35	24.2	25.8	174.5	C1
50	39.2	10.8	159.5	C2
55	44.2	5.8	154.5	C2
72	61.2	11.2	137.5	C2
92	81.2	31.2	117.5	C2
204	193.2	143.2	5.5	C3
215	204.2	154.2	5.5	C3

Partitioning Methods

Initial Centroid:

C1 = 14.83

C2 = 67.25

C3 = 209.5

	Distance to			Cluster
Data	C1 = 14.83	C2 = 67.25	C3 = 209.5	
5	9.83	62.25	204.5	C1
10	4.83	57.25	199.5	C1
11	3.83	56.25	198.5	C1
13	1.83	54.25	196.5	C1
15	0.17	52.25	194.5	C1
35	20.17	32.25	174.5	C1
50	35.17	17.25	159.5	C2
55	40.17	12.25	154.5	C2
72	57.17	4.75	137.5	C2
92	77.17	24.75	117.5	C2
204	189.17	136.75	5.5	C3
215	200.17	147.75	5.5	C3

K-means Clustering

- K-means clustering is a type of unsupervised learning, which is used when you have unlabeled data(i.e., data without defined categories or groups)
- The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K(*user defined*)*
- The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided
- Data points are clustered based on feature similarity(distance)

K-means Clustering

- The k-means clustering algorithm inputs are the number of cluster *k* and data set
- The data set is a collection of features for each data point
- The algorithm starts with initial estimates for the *k centroids*, which can be either be randomly or selected from the data set
- The process of k-means clustering
  - Set the number of cluster *k*(e.g. k = 2)
  - The algorithm then iterates between following two steps until no more change in centroid point
    1. Data assignment step
    2. Centroid update step

K-means Clustering – Data assignment step

- Each centroid defines one of the clusters
- In this step, each data point is assigned to its nearest centroid, based on the Euclidean distance
- Formula

$$\arg\min \sum_{i=1}^{K} \sum_{x \in C_i} d(C_i, x)$$

  - K: The number of clusters k
  - C_i: *ith* cluster
  - X: data point in cluster
  - $d()$: is the standard(L2) Euclidean distance

K-Means Clustering

A1(2, 10), A2(2, 5),

A3(8, 4), B1(5, 8),

B2(7, 5), B3(6, 4),

C1(1, 2), C2(4, 9)

Initial Centroids:
A1: (2, 10)
B1: (5, 8)
C1: (1, 2)

Data Points			Distance to						Cluster	New Cluster
			2	10	5	8	1	2		
A1	2	10								
A2	2	5								
A3	8	4								
B1	5	8								
B2	7	5								
B3	6	4								
C1	1	2								
C2	4	9								

Initial Centroids:
A1: (2, 10)
B1: (5, 8)
C1: (1, 2)

Data Points			Distance to						Cluster	New Cluster
			2	10	5	8	1	2		
A1	2	10	0.00		3.61		8.06		1	
A2	2	5	5.00		4.24		3.16		3	
A3	8	4	8.49		5.00		7.28		2	
B1	5	8	3.61		0.00		7.21		2	
B2	7	5	7.07		3.61		6.71		2	
B3	6	4	7.21		4.12		5.39		2	
C1	1	2	8.06		7.21		0.00		3	
C2	4	9	2.24		1.41		7.62		2	

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Initial Centroids:
A1: (2, 10)
B1: (5, 8)
C1: (1, 2)

New Centroids:
A1: (2, 10)
B1: (6, 6)
C1: (1.5, 3.5)

Data Points			Distance to						Cluster	New Cluster
			2	10	5	8	1	2		
A1	2	10	0.00		3.61		8.06		1	
A2	2	5	5.00		4.24		3.16		3	
A3	8	4	8.49		5.00		7.28		2	
B1	5	8	3.61		0.00		7.21		2	
B2	7	5	7.07		3.61		6.71		2	
B3	6	4	7.21		4.12		5.39		2	
C1	1	2	8.06		7.21		0.00		3	
C2	4	9	2.24		1.41		7.62		2	

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

B1 : (8+5+7+6+4)/5 =6
(4+8+5+4+9)/5= 6

Current Centroids:
A1: (2, 10)
B1: (6, 6)
C1: (1.5, 3.5)

Data Points			Distance to						Cluster	New Cluster
			2	10	6	6	1.5	1.5		
A1	2	10	0.00		5.66		6.52		1	1
A2	2	5	5.00		4.12		1.58		3	3
A3	8	4	8.49		2.83		6.52		2	2
B1	5	8	3.61		2.24		5.70		2	2
B2	7	5	7.07		1.41		5.70		2	2
B3	6	4	7.21		2.00		4.53		2	2
C1	1	2	8.06		6.40		1.58		3	3
C2	4	9	2.24		3.61		6.04		2	1

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Current Centroids:
A1: (2, 10)
B1: (6, 6)
C1: (1.5, 3.5)

New Centroids:
A1: (3, 9.5)
B1: (6.5, 5.25)
C1: (1.5, 3.5)

Data Points			Distance to						Cluster	New Cluster
			2	10	6	6	1.5	1.5		
A1	2	10	0.00		5.66		6.52		1	1
A2	2	5	5.00		4.12		1.58		3	3
A3	8	4	8.49		2.83		6.52		2	2
B1	5	8	3.61		2.24		5.70		2	2
B2	7	5	7.07		1.41		5.70		2	2
B3	6	4	7.21		2.00		4.53		2	2
C1	1	2	8.06		6.40		1.58		3	3
C2	4	9	2.24		3.61		6.04		2	1

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Current Centroids:
A1: (3, 9.5)
B1: (6.5, 5.25)
C1: (1.5, 3.5)

Data Points			Distance to						Cluster	New Cluster
			3	9.5	6.5	5.25	1.5	3.5		
A1	2	10	1.12		6.54		6.52		1	1
A2	2	5	4.61		4.51		1.58		3	3
A3	8	4	7.43		1.95		6.52		2	2
B1	5	8	2.50		3.13		5.70		2	1
B2	7	5	6.02		0.56		5.70		2	2
B3	6	4	6.26		1.35		4.53		2	2
C1	1	2	7.76		6.39		1.58		3	3
C2	4	9	1.12		4.51		6.04		1	1

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Current Centroids:
A1: (3, 9.5)
B1: (6.5, 5.25)
C1: (1.5, 3.5)

New Centroids:
A1: (3.67, 9)
B1: (7, 4.33)
C1: (1.5, 3.5)

Data Points			Distance to						Cluster	New Cluster
			3	9.5	6.5	5.25	1.5	3.5		
A1	2	10	1.12		6.54		6.52		1	1
A2	2	5	4.61		4.51		1.58		3	3
A3	8	4	7.43		1.95		6.52		2	2
B1	5	8	2.50		3.13		5.70		2	1
B2	7	5	6.02		0.56		5.70		2	2
B3	6	4	6.26		1.35		4.53		2	2
C1	1	2	7.76		6.39		1.58		3	3
C2	4	9	1.12		4.51		6.04		1	1

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Current Centroids:
A1: (3.67, 9)
B1: (7, 4.33)
C1: (1.5, 3.5)

Data Points			Distance to						Cluster	New Cluster
			3.67	9	7	4.33	1.5	3.5		
A1	2	10	1.94		7.56		6.52		1	1
A2	2	5	4.33		5.04		1.58		3	3
A3	8	4	6.62		1.05		6.52		2	2
B1	5	8	1.67		4.18		5.70		1	1
B2	7	5	5.21		0.67		5.70		2	2
B3	6	4	5.52		1.05		4.53		2	2
C1	1	2	7.49		6.44		1.58		3	3
C2	4	9	0.33		5.55		6.04		1	1

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Problem:

Suppose we have the following 2-dimensional data points:

$$(1, 2), (1, 4), (3, 3), (5, 2), (6, 4)$$

Apply k-means clustering with $k = 2$ (i.e., divide the dataset into 2 clusters). Use the initial centroids $(1, 2)$ and $(5, 2)$. Use Euclidean distance as the distance metric.

Solution:

1. Initialize the centroids: $(1, 2)$ and $(5, 2)$.
2. Assign each point to the nearest centroid:
   - For $(1, 2)$: $(1, 2)$, $(1, 4)$
   - For $(5, 2)$: $(3, 3)$, $(5, 2)$, $(6, 4)$
3. Update the centroids by taking the mean of the points assigned to each centroid:
   - New centroid for cluster 1: $(\frac{1+1}{2}, \frac{2+4}{2}) = (1, 3)$
   - New centroid for cluster 2: $(\frac{3+5+6}{3}, \frac{3+2+4}{3}) = (4.67, 3)$
4. Repeat steps 2 and 3 until convergence (i.e., centroids do not change).

The final clusters and centroids will be:

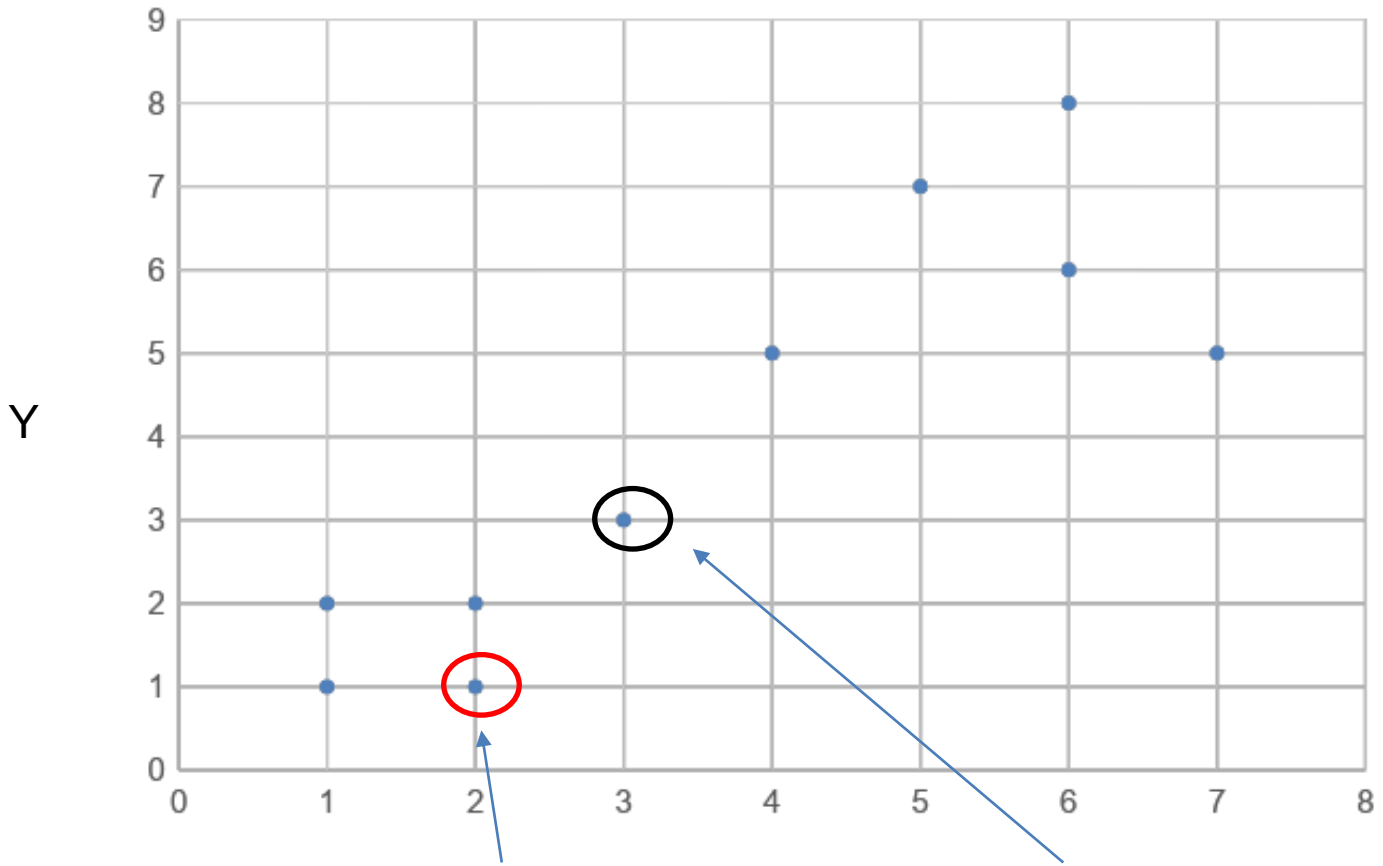
- Cluster 1: $(1, 2)$, $(1, 4)$
- Cluster 2: $(3, 3)$, $(5, 2)$, $(6, 4)$

Centroids:

- $(1, 3)$ for Cluster 1
- $(4.67, 3)$ for Cluster 2

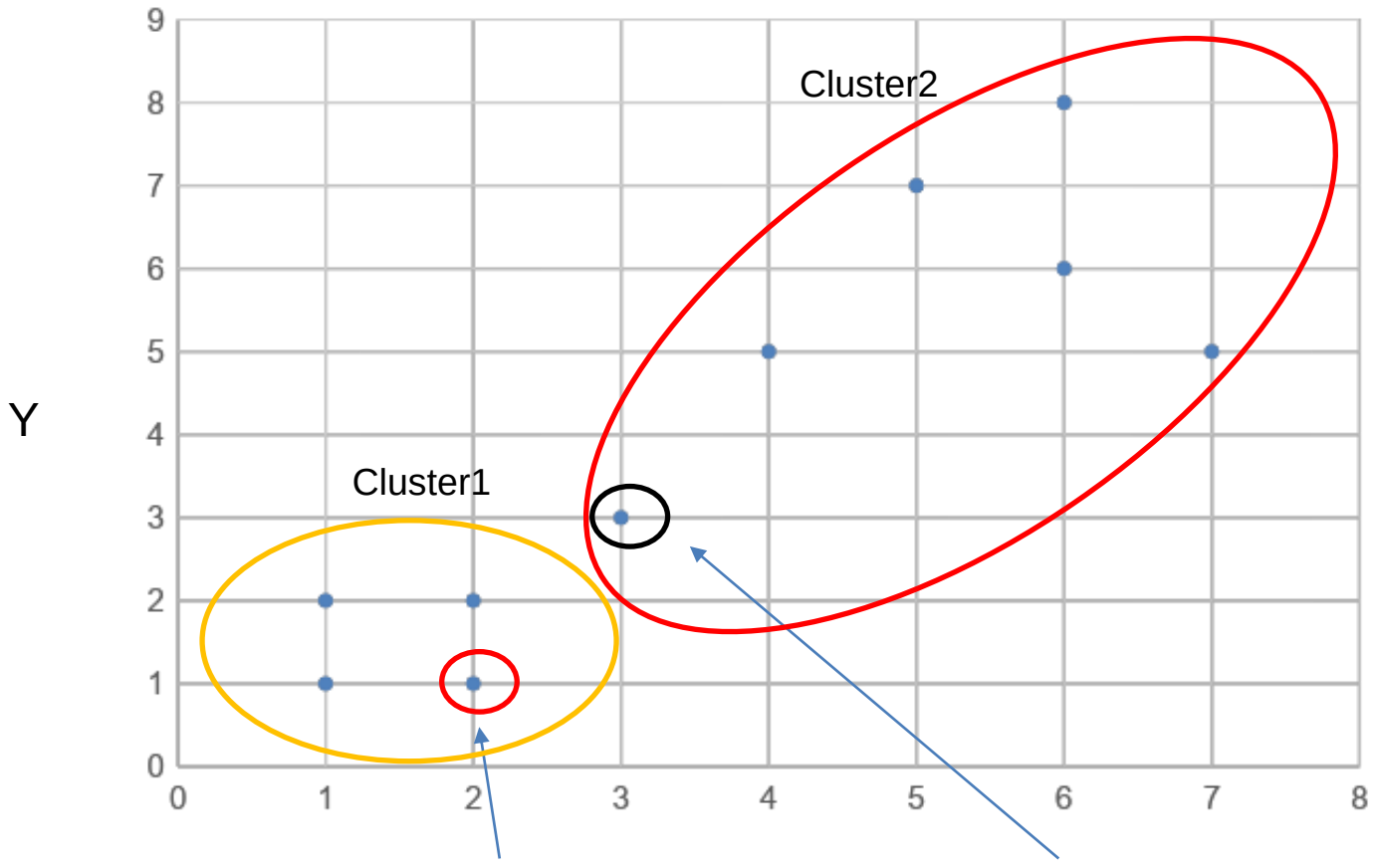
K-means Clustering – Data assignment step (Iteration 1)

- First, randomly set a point as centroid point
- For example, $k = 2$



K-means Clustering – Data assignment step(Iteration 1)

- First, randomly set a point as centroid point
- For example, $k = 2$



K-means Clustering – Centroid update step

- In this step, the centroids are recomputed by taking the mean of all data points assigned to that centroid's cluster

Data	X	Y	Cluster
1	1	1	1
2	2	1	1
3	1	2	1
4	2	2	1
5	3	3	2
6	6	6	2
7	6	8	2
8	5	7	2
9	7	5	2
10	4	5	2

New centroid(cluster 1)

$$=\left(\frac{1+2+1+2}{4}, \frac{1+1+2+2}{4}\right)$$

$$= (1.5,\ 1.5)$$

New centroid(cluster 2)

$$=\left(\frac{3+6+6+5+7+4}{6}, \frac{3+6+8+7+5+5}{6}\right)$$

$$= (5.1,\ 5.6)$$

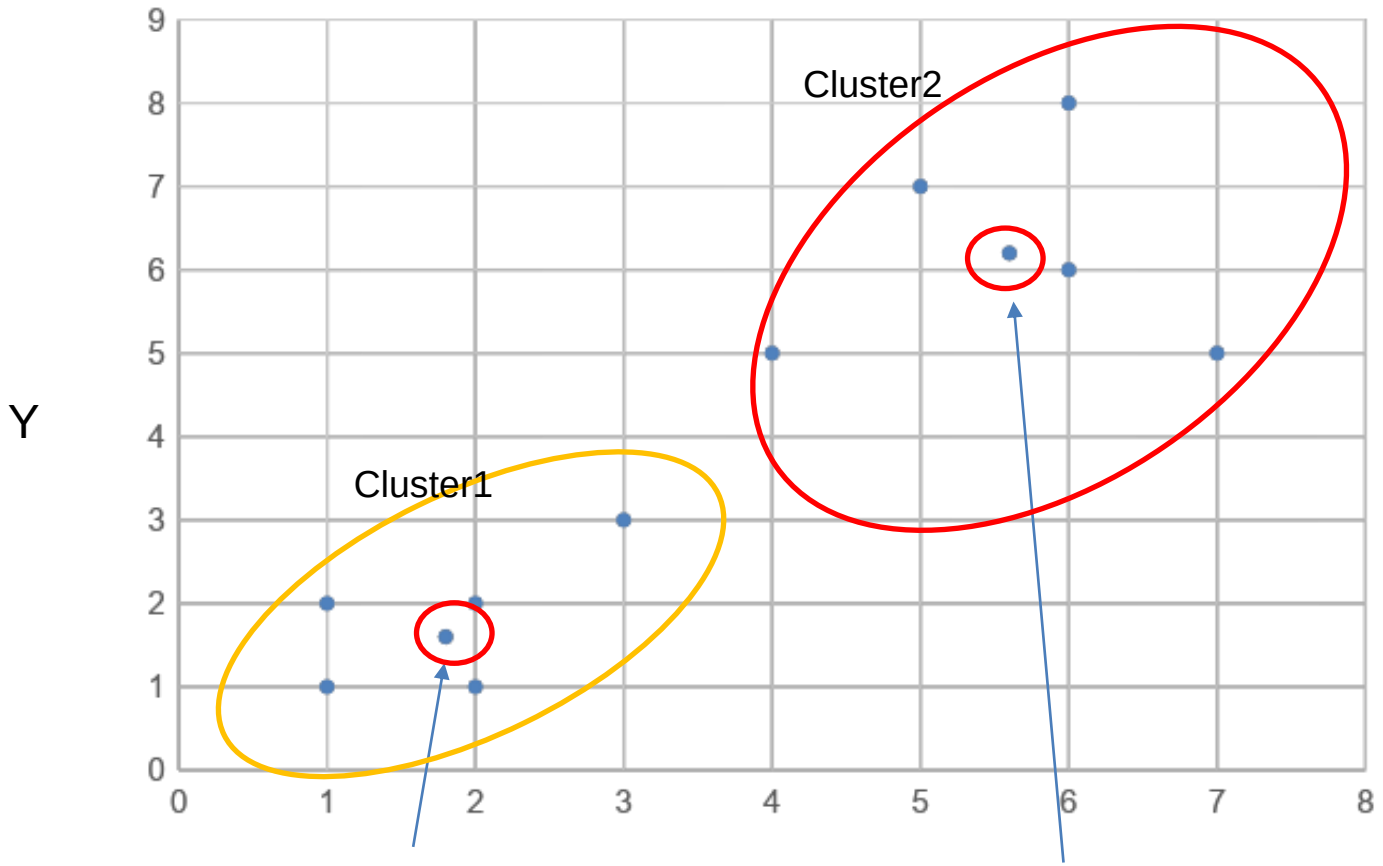
Cluster	New Centroid	Data Index
1	(1.5, 1.5)	1,2,3,4
2	(5.1, 5.6)	5,6,7,8,9,10

K-means Clustering – Data assignment step(Iteration 2)

- Now, repeat calculating distance between new centroid and each point

K-means Clustering – Data assignment step(Iteration 3)

- Now, repeat calculating distance between new centroid and each point



K-means Clustering

- Example – The animation of K-means Clustering

K-means Clustering

- Repeat data assignment step and centroid update step till no more changes of centroid point
- Advantages of K-means Clustering
  - The algorithm is simple to use
  - The algorithm can guarantee the result
- Disadvantages of K-means Clustering
  - Need to specify *k*, the number of clusters
  - Often terminates at a local optimum
    - Not suitable to discover clusters with non-convex shapes
    - Not suitable to discover clusters with different densities

Application

- Colour-Based Image Segmentation Using *K*-means

Step 1: Loading a colour image of tissue stained with hemotoxylin and eosin (H&E)

H&E image

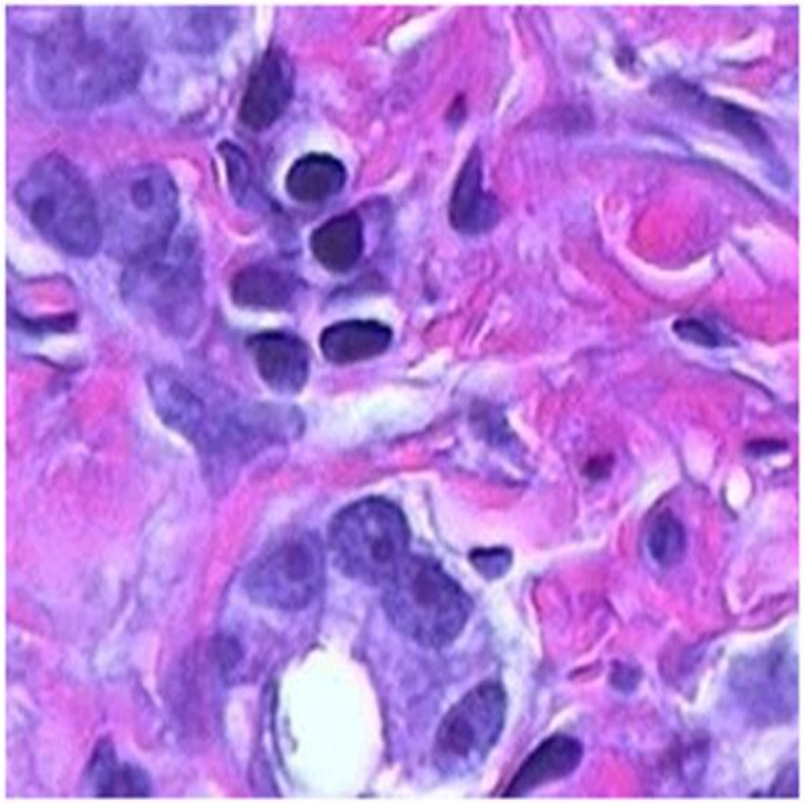


Image courtesy of Alan Partin, Johns Hopkins University

Application

- Colour-Based Image Segmentation Using *K*-means

  Step 2: Convert the image from RGB colour space to L*a*b* colour space

  - Unlike the RGB colour model, L*a*b* colour is designed to approximate human vision.
  - There is a complicated transformation between RGB and L*a*b*.

$$(L^*, a^*, b^*) = T(R, G, B).$$

$$(R, G, B) = T'(L^*, a^*, b^*).$$

Application

- Colour-Based Image Segmentation Using *K*-means

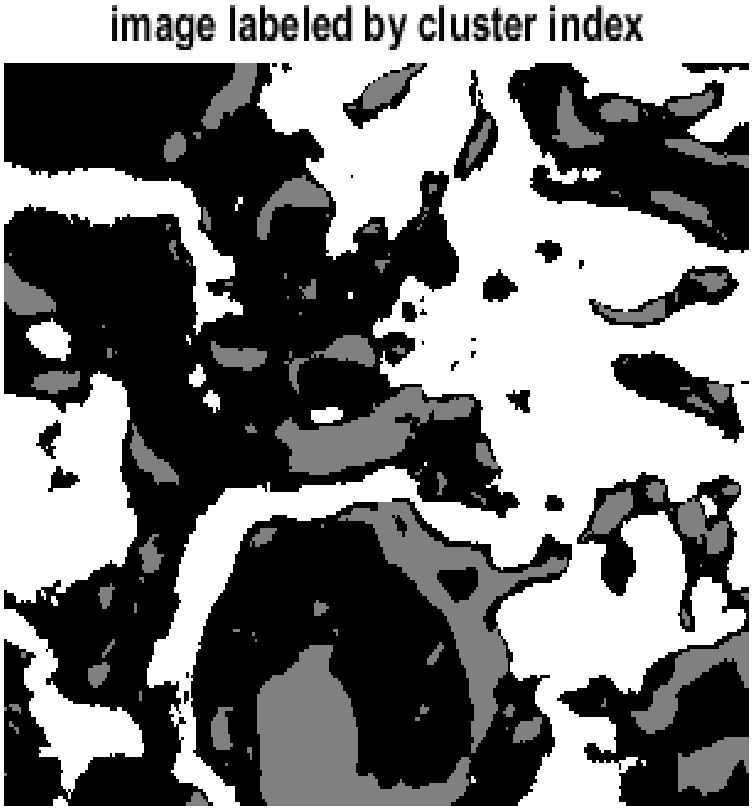
  Step 3: Undertake clustering analysis in the (a*, b*) colour space with the *K*-means algorithm

  - In the L*a*b* colour space, each pixel has a properties or feature vector: (L*, a*, b*).
  - Like feature selection, L* feature is discarded. As a result, each pixel has a feature vector (a*, b*).
  - Applying the *K*-means algorithm to the image in the a*b* feature space where $K = 3$ by applying the domain knowledge.

Application

- Colour-Based Image Segmentation Using *K*-means

  Step 4: Label every pixel in the image using the results from *K*-means clustering (indicated by three different grey levels)

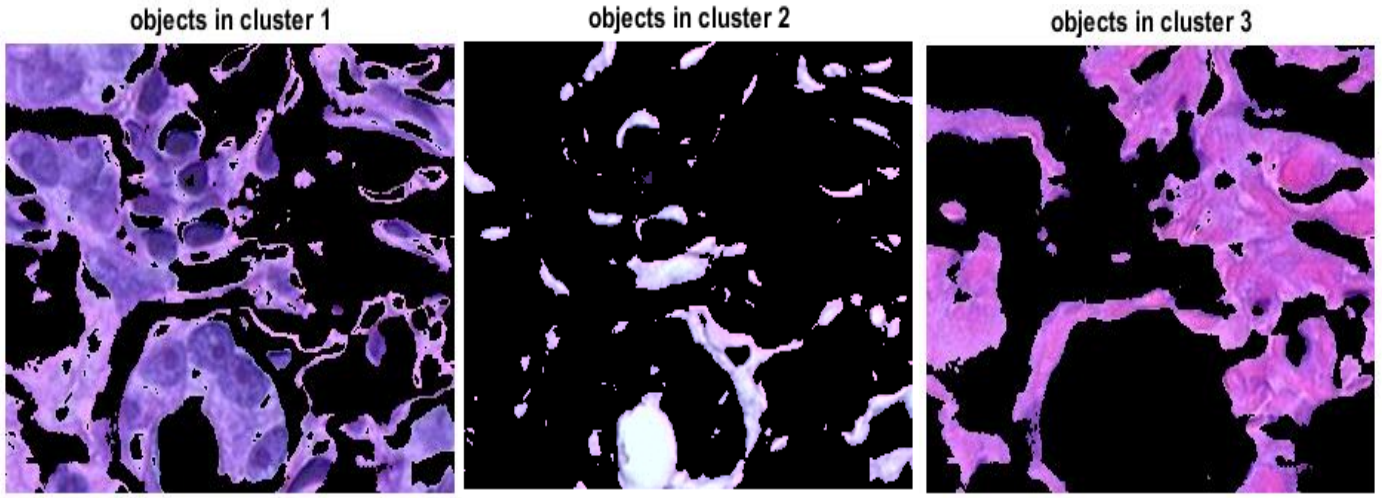


Application

- Colour-Based Image Segmentation Using *K*-means

Step 5: Create Images that Segment the H&E Image by Colour

- Apply the label and the colour information of each pixel to achieve separate colour images corresponding to three clusters.

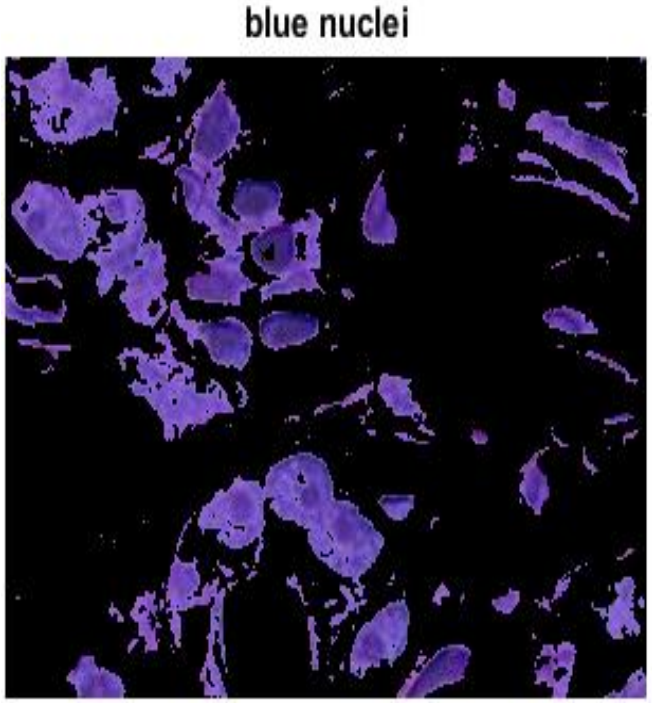


Application

- Colour-Based Image Segmentation Using *K*-means

Step 6: Segment the nuclei into a separate image with the L* feature

- In cluster 1, there are dark and light blue objects (pixels). The dark blue objects (pixels) correspond to nuclei (with the domain knowledge).
- L* feature specifies the brightness values of each colour.
- With a threshold for L*, we achieve an image containing the nuclei only.



K-Medoids clustering

- K-Medoids and K-Means are two types of clustering mechanisms in Partition Clustering.
- First, Clustering is the process of breaking down an abstract group of data points/ objects into classes of similar objects such that all the objects in one cluster have similar traits. , a group of n objects is broken down into k number of clusters based on their similarities.
- K-medoids is an unsupervised method with unlabelled data to be clustered. It is an improvised version of the K-Means algorithm mainly designed to deal with outlier data sensitivity. Compared to other partitioning algorithms, the algorithm is simple, fast, and easy to implement.

1. **The partitioning will be carried on such that:**Each cluster must have at least one object
2. An object must belong to only one cluster

- Here is a small recap on K-Means clustering:

In the K-Means algorithm, given the value of k and unlabelled data:

1. Choose k number of random points (Data point from the data set or some other points). These points are also called "**Centroids**" or "**Means**".
2. Assign all the data points in the data set to the closest centroid by applying any distance formula like **Euclidian distance, Manhattan distance,** etc.
3. Now, choose new centroids by calculating the mean of all the data points in the clusters and goto step 2
4. Continue step 3 until no data point changes classification between two iterations.

- The problem with the K-Means algorithm is that the algorithm needs to handle outlier data. An outlier is a point different from the rest of the points. All the outlier data points show up in a different cluster and will attract other clusters to merge with it. Outlier data increases the mean of a cluster by up to 10 units. Hence, **K-Means clustering is highly affected by outlier data.**

K-Medoids:

Medoid: A Medoid is a point in the cluster from which the sum of distances to other data points is minimal.

(or)

A Medoid is a point in the cluster from which dissimilarities with all the other points in the clusters are minimal.

Instead of centroids as reference points in K-Means algorithms, the K-Medoids algorithm takes a Medoid as a reference point.

There are three types of algorithms for K-Medoids Clustering:

1.PAM (Partitioning Around Clustering)

2.CLARA (Clustering Large Applications)

3.CLARANS (Randomized Clustering Large Applications)

PAM is the most powerful algorithm of the three algorithms but has the disadvantage of time complexity. The following K-Medoids are performed using PAM. In the further parts, we'll see what CLARA and CLARANS are.

Algorithm:

Given the value of k and unlabelled data:

1.Choose k number of random points from the data and assign these k points to k number of clusters. These are the initial medoids.

2.For all the remaining data points, calculate the distance from each medoid and assign it to the cluster with the nearest medoid.

3.Calculate the total cost (Sum of all the distances from all the data points to the medoids)

4.Select a random point as the new medoid and swap it with the previous medoid. Repeat 2 and 3 steps.

5.If the total cost of the new medoid is less than that of the previous medoid, make the new medoid permanent and repeat step 4.

6.If the total cost of the new medoid is greater than the cost of the previous medoid, undo the swap and repeat step 4.

7.The Repetitions have to continue until no change is encountered with new medoids to classify data points.

K-Medoid Clustering – Solved Example – 1

i	x	y
X1	2	6
X2	3	4
X3	3	8
X4	4	7
X5	6	2
X6	6	4
X7	7	3
X8	7	4
X9	8	5
X10	7	6

- Apply K-Medoid clustering algorithm to form two clusters.
- Use Manhattan distance to find the between data point and medoid.

Step 1

- Select two medoids
- C1=(3, 4)
- C2=(7, 4)
- $\boldsymbol{Manhattan\ Dist} = |x_1 - x_2| + |y_1 - y_2|$
- $\boldsymbol{Mdist}[(2,6),(3,4)] = |2-3| + |6-4| = 3$
- $\boldsymbol{Mdist}[(3,4),(3,4)] = |3-3| + |4-4| = 0$

i	x	y	C1	C2	Cluster
X1	2	6	3	7	C1
X2	3	4	0	4	C1
X3	3	8	4	8	C1
X4	4	7	4	6	C1
X5	6	2	5	3	C2
X6	6	4	3	1	C2
X7	7	3	5	1	C2
X8	7	4	4	0	C2
X9	8	5	6	2	C2
X10	7	6	6	2	C2

Step 2

- Cluster are
- C1: {(2,6), (3,4), (3,8), (4,7)}
- C2: {(6, 2), (6, 4), (7, 3), (7, 4), (8, 5), (7,6)}

- C1: {(2,6), **(3,4),** (3,8), (4,7)}
- C2: {(6, 2), (6, 4), (7, 3), **(7, 4),** (8, 5), (7,6)}
- **Calculate the Total Cost**
- $\boldsymbol{Cost(c.x) = \sum_i |c_i - x_i|}$
- $Total\ Cost = \{Cost\big((3,4),(2,6)\big) + Cost\big((3,4),(3,8)\big) + Cost\big((3,4),(4,7)\big) + Cost\big((7,4),(6,2)\big) + Cost\big((7,4),(6,4)\big) + Cost\big((7,4),(7,3)\big) + Cost\big((7,4),(8,5)\big) + Cost\big((7,4),(7,6)\big)\}$
- $\boldsymbol{Total\ Cost = 3 + 4 + 4 + 2 + 3 + 1 + 1 + 2 = 20}$

Step 3

- **Randomly select one non-medoid point and recalculate the cost.**
- C1=(3, 4) and C2=(7, 4)
- **O=(7, 3)**
- **Swap C2 with O**
- **New Medoids**
- **C1=(3, 4) and O=(7, 3)**

Step 3

- **New Medoids**
- **C1=(3, 4) and O=(7, 3)**
- $Manhattan\ Dist = |x_1 - x_2| + |y_1 - y_2|$
- $Mdist[(2,6),(7,3)] = |2-7| + |6-3| = 8$

i	x	y	C1	O	Cluster
X1	2	6	3	8	C1
X2	3	4	0	5	C1
X3	3	8	4	9	C1
X4	4	7	4	7	C1
X5	6	2	5	2	O
X6	6	4	3	2	O
X7	7	3	5	0	O
X8	7	4	4	1	O
X9	8	5	6	3	O
X10	7	6	6	3	O

- New Cluster are
- C1: {(2,6), (3,4), (3,8), (4,7)}
- O: {(6, 2), (6, 4), (7, 3), (7, 4), (8, 5), (7,6)}

i	x	y	C1	O	Cluster
X1	2	6	3	8	C1
X2	3	4	0	5	C1
X3	3	8	4	9	C1
X4	4	7	4	7	C1
X5	6	2	5	2	O
X6	6	4	3	2	O
X7	7	3	5	0	O
X8	7	4	4	1	O
X9	8	5	6	3	O
X10	7	6	6	3	O

- C1: {(2,6), **(3,4),** (3,8), (4,7)}
- O: {(6, 2), (6, 4), **(7, 3),** (7, 4), (8, 5), (7,6)}
- **Calculate the Total Cost**
- $\boldsymbol{Cost(c, x) = \sum_i |c_i - x_i|}$
- $Current\ Total\ Cost = \{Cost\big((3,4),(2,6)\big) + Cost\big((3,4),(3,8)\big) +$

  $Cost\big((3,4),(4,7)\big) + Cost\big((7,3),(6,2)\big) + Cost\big((7,3),(6,4)\big) + Cost\big((7,3),(7,4)\big) +$

  $Cost\big((7,3),(8,5)\big) + Cost\big((7,3),(7,6)\big)\}$
- $\boldsymbol{Current\ Total\ Cost = 3 + 4 + 4 + 2 + 2 + 1 + 3 + 3 = 22}$

Step 4

- Cost of Swapping of medoid C2 with O
- $S = Current\ Total\ Cost - Previous\ Total\ Cost$
- $S = 22 - 20 = 2 > 0$
- Hence Swapping C2 with O is not a good Idea.
- Final Medoids are C1 =(3, 4) and C2=(7, 4)
- Clusters are
- C1: {(2,6), (3,4), (3,8), (4,7)}
- C2: {(6, 2), (6, 4), (7, 3), (7, 4). (8. 5), (7,6)}

K-Medoids Clustering

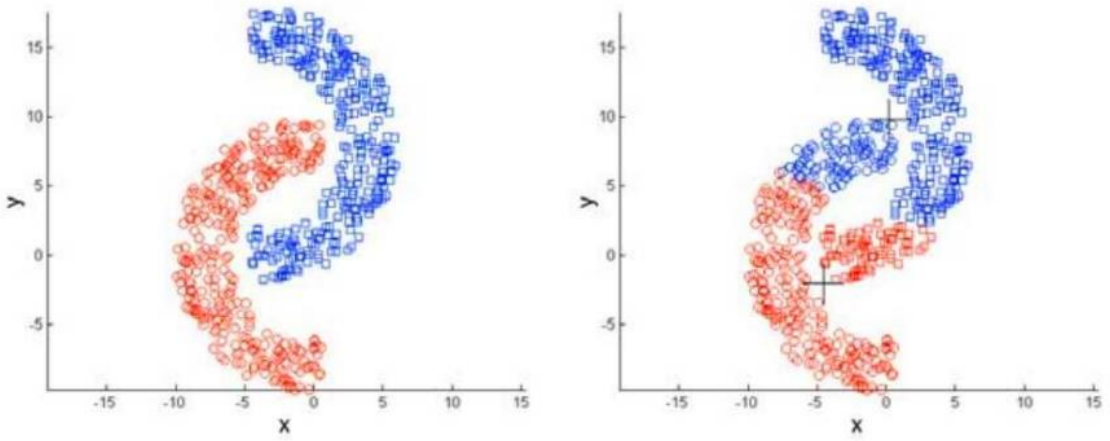
	x	y
0	5	4
1	7	7
2	1	3
3	8	6
4	4	9

If k is given as 2, we need to break down the data points into 2 clusters.

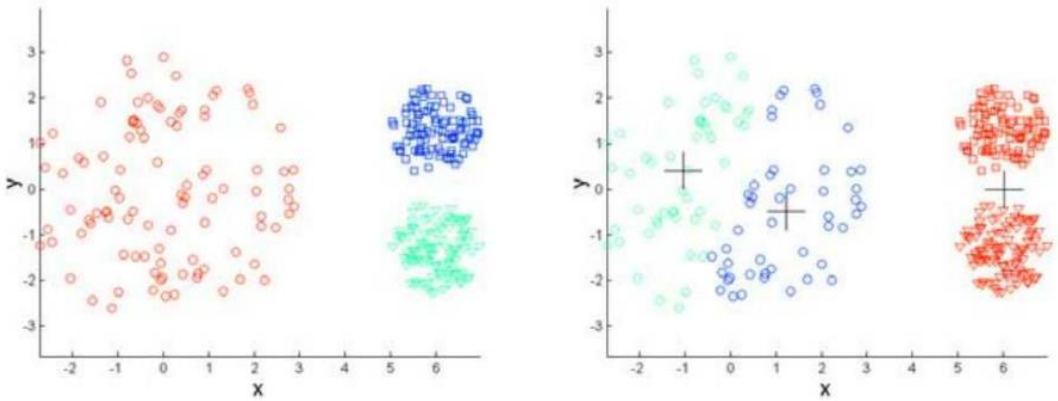
Initial medoids: M1(1, 3) and M2(4, 9).

K-means Clustering - Failures

Non-convex/non-round-shaped clusters: Standard K-means fails!



Clusters with different densities



Dissimilarity between Clusters

We know how to compute the dissimilarit $d(\mathbf{x}_i, \mathbf{x}_j)$ between two examples How to compute the dissimilarity between two clusters R and S?
Min-link or single-link: results in chaining (clusters can get very large)

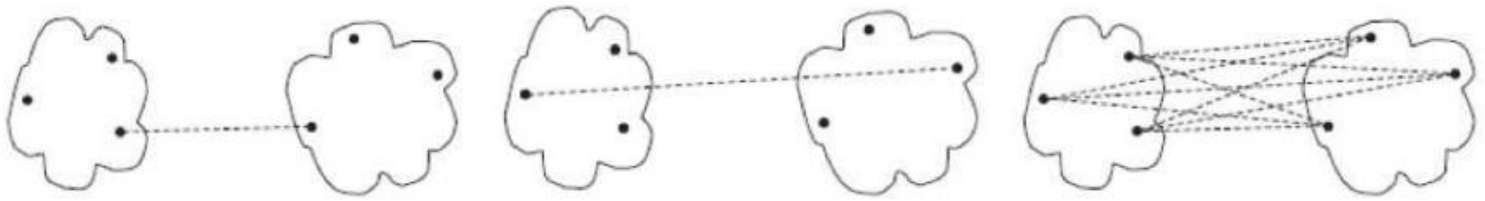
$$d(R, S) = \min_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

Max-link or complete-link: results in small, round shaped clusters

$$d(R, S) = \max_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$

Average-link: compromise between single and complete linkage

$$d(R, S) = \frac{1}{|R||S|} \sum_{\mathbf{x}_R \in R, \mathbf{x}_S \in S} d(\mathbf{x}_R, \mathbf{x}_S)$$



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

In data mining and statistics, hierarchical clustering analysis is a method of clustering analysis that seeks to build a hierarchy of clusters i.e. tree-type structure based on the hierarchy.
In machine learning, clustering is the unsupervised learning technique that groups the data based on similarity between the set of data. There are different-different types of clustering algorithms in machine learning.

Connectivity-based clustering: This type of clustering algorithm builds the cluster based on the connectivity between the data points. Example: Hierarchical clustering.
•**Centroid-based clustering:** This type of clustering algorithm forms around the centroids of the data points. Example: K-Means clustering, K-Mode clustering
•**Distribution-based clustering:** This type of clustering algorithm is modeled using statistical distributions. It assumes that the data points in a cluster are generated from a particular probability distribution, and the algorithm aims to estimate the parameters of the distribution to group similar data points into clusters Example: Gaussian Mixture Models (GMM)
•**Density-based clustering:** This type of clustering algorithm groups together data points that are in high-density concentrations and separates points in low-concentrations regions. The basic idea is that it identifies regions in the data space that have a high density of data points and groups those points together into clusters. Example: DBSCAN(Density-Based Spatial Clustering of Applications with Noise)
In this article, we will discuss connectivity-based clustering algorithms i.e Hierarchical clustering
Hierarchical clustering
Hierarchical clustering is a connectivity-based clustering model that groups the data points together that are close to each other based on the measure of similarity or distance. The assumption is that data points that are close to each other are more similar or related than data points that are farther apart.

dendrogram, while the largest clusters, which include all the data points, are located at the top. In order to generate different numbers of clusters, the dendrogram can be sliced at various heights.
The dendrogram is created by iteratively merging or splitting clusters based on a measure of similarity or distance between data points. Clusters are divided or merged repeatedly until all data points are contained within a single cluster, or until the predetermined number of clusters is attained.
We can look at the dendrogram and measure the height at which the branches of the dendrogram form distinct clusters to calculate the ideal number of clusters. The dendrogram can be sliced at this height to determine the number of clusters.

Types of Hierarchical Clustering

Basically, there are two types of hierarchical Clustering:

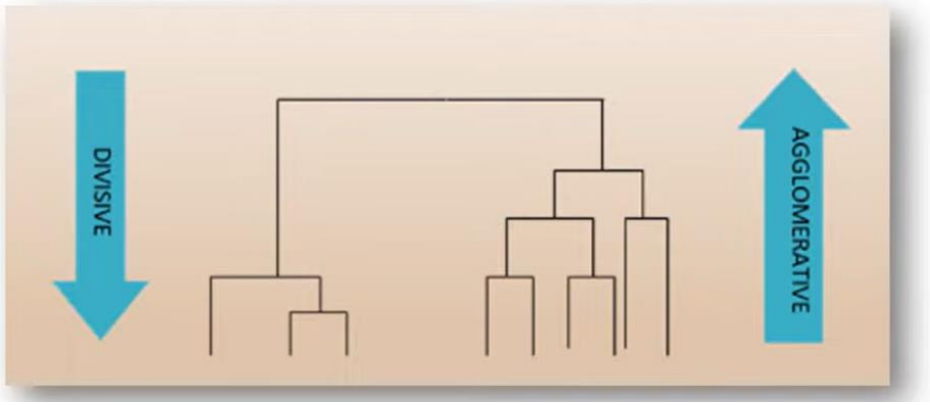
1.Agglomerative Clustering
2.Divisive clustering

Hierarchical Agglomerative Clustering

It is also known as the bottom-up approach or hierarchical agglomerative clustering (HAC). A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to prespecify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerate pairs of clusters until all clusters have been merged into a single cluster that contains all data.

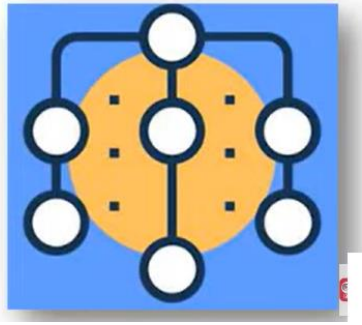
About Hierarchical Clustering

- Hierarchical clustering is another unsupervised machine learning algorithm.
- Which is used to group the unlabeled datasets into a cluster and also known as hierarchical cluster analysis or HCA.
- In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the dendrogram



Why Hierarchical Clustering?

- As we already have **K-Means Clustering Algorithm.**
- So, as we have seen in the K-means clustering that there are some challenges with this algorithm, which are a predetermined number of clusters, and it always tries to create the clusters of the same size.
- To solve these two challenges, we can option for the hierarchical clustering algorithm because in this algorithm, we don't need to have knowledge about the predefined number of clusters.



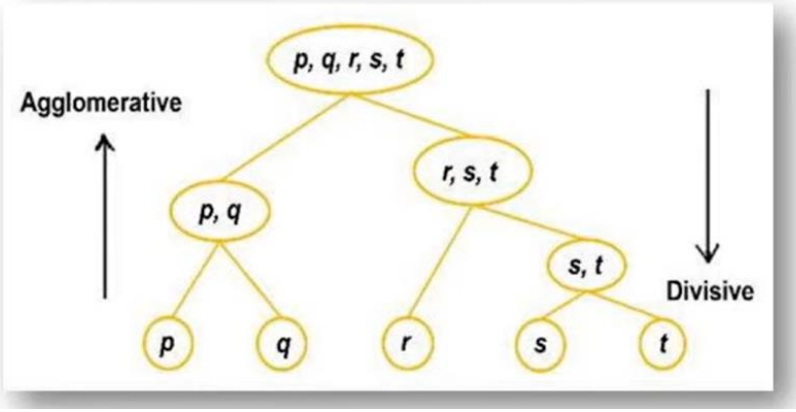
Types of Hierarchical Clustering

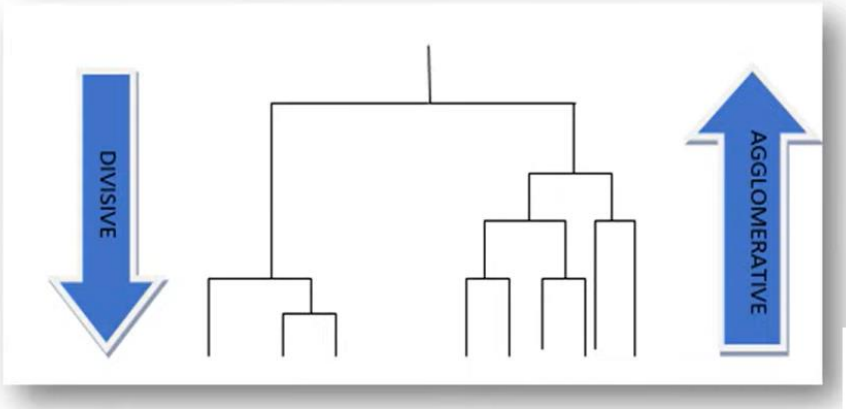
Type 1: Agglomerative Clustering:

- Agglomerative is a **bottom-up** approach, in which the algorithm starts with taking all data points as single clusters and merging them until one cluster is left.

Type 2: Divisive Clustering:

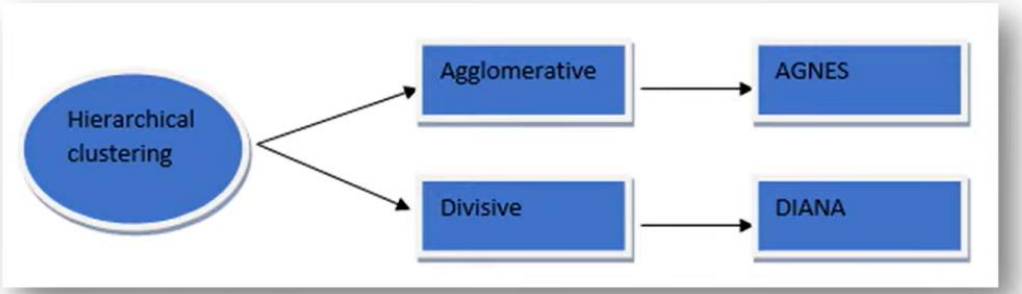
- Divisive algorithm is the reverse of the agglomerative algorithm as it is a **top-down approach.**

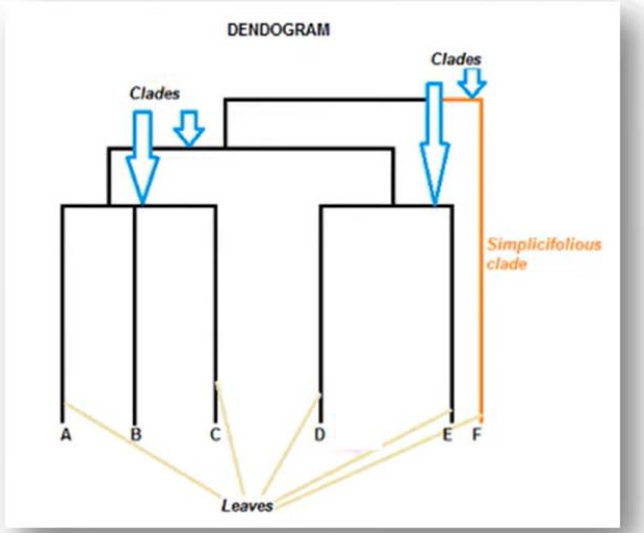




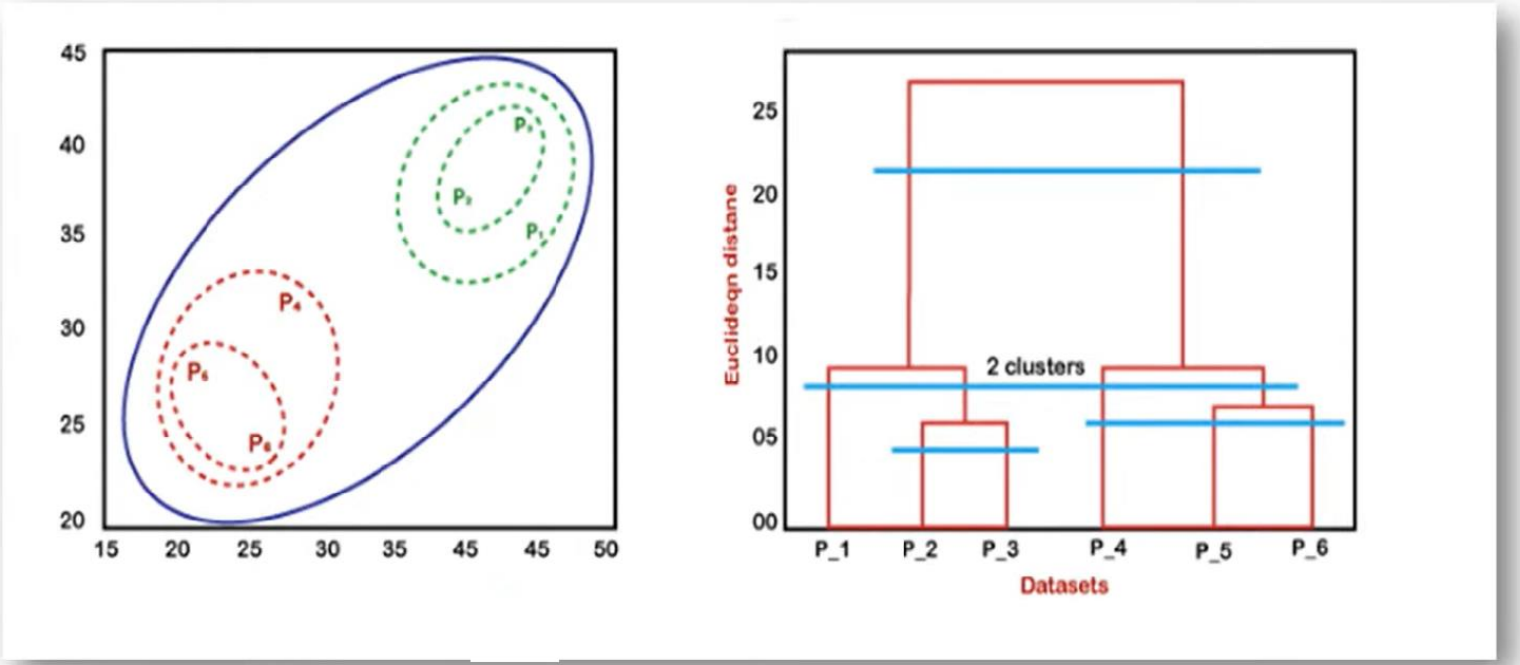
What is a Dendrogram?

- A **Dendrogram** is a type of tree diagram showing hierarchical relationships between different sets of data.
- Dendrogram contains the memory of hierarchical clustering algorithm.
- Just by looking at the Dendrogram you can tell how the cluster is formed.





Example



- In the above diagram, the **left part** is showing how clusters are created in agglomerative clustering and the **right part** is showing the corresponding dendrogram.

Hierarchical Clustering

Clustering is performed based upon Dissimilarities between Clusters

Produces a sequence or tree of clusterings

Does not need the number of clusters as input

Partitions can be visualized using a tree structure (a dendrogram)

Possible to view partitions at different levels of granularities using different K

Applications:

- Social Sciences
- Biological Taxonomy
- Medicine, Archaeology
- Computer Science
- Engineering
- etc.

Types of Hierarchical Clustering:

- Agglomerative (bottom-up) Clustering

- Divisive (top-down) Clustering

Agglomerative is more popular and simpler than divisive (but less accurate)

Algorithm :

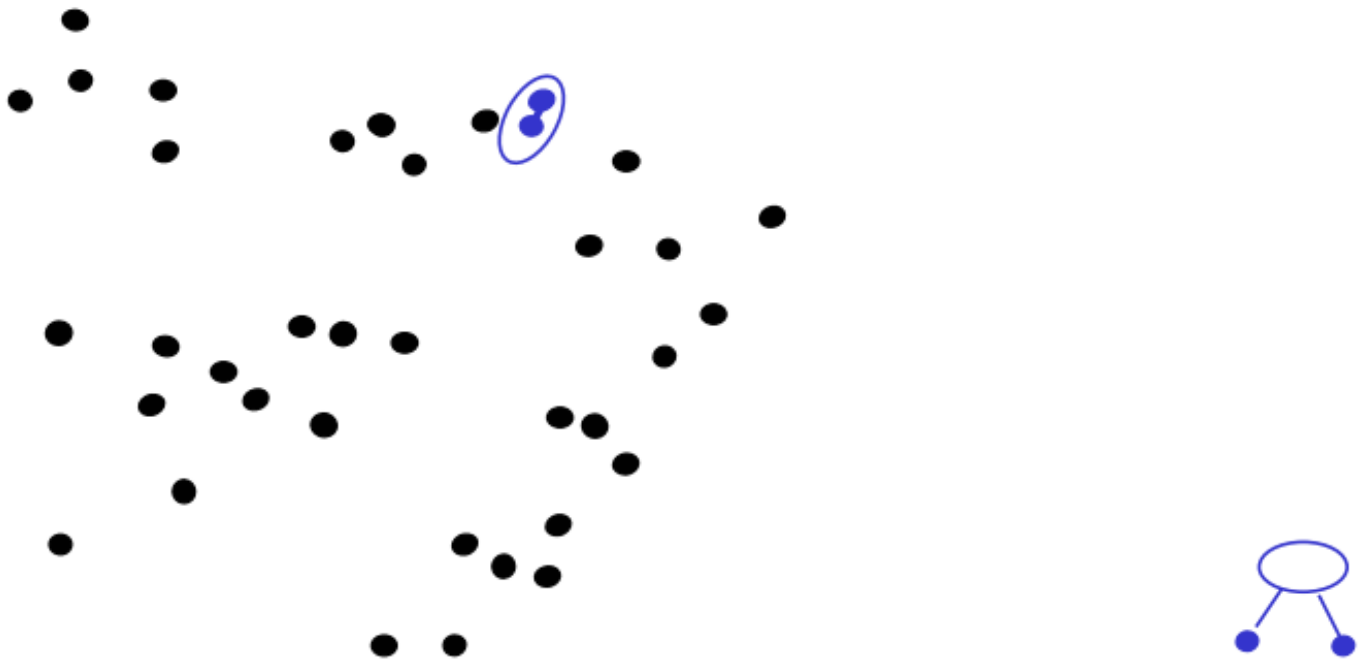
given a dataset (d_1, d_2, d_3, ...d_N) of size N
compute the distance matrix
for i=1 to N:
    # as the distance matrix is symmetric about
    # the primary diagonal so we compute only lower
    # part of the primary diagonal
    for j=1 to i:
        dis_mat[i][j] = distance[d_i, d_j]
each data point is a singleton cluster
repeat
    merge the two cluster having minimum distance
    update the distance matrix
until only a single cluster remains

Hierarchical Clustering - **Agglomerative (bottom-up) Clustering**

Say “Every point is its own cluster”

Find “most similar” pair of clusters

Merge it into a parent cluster



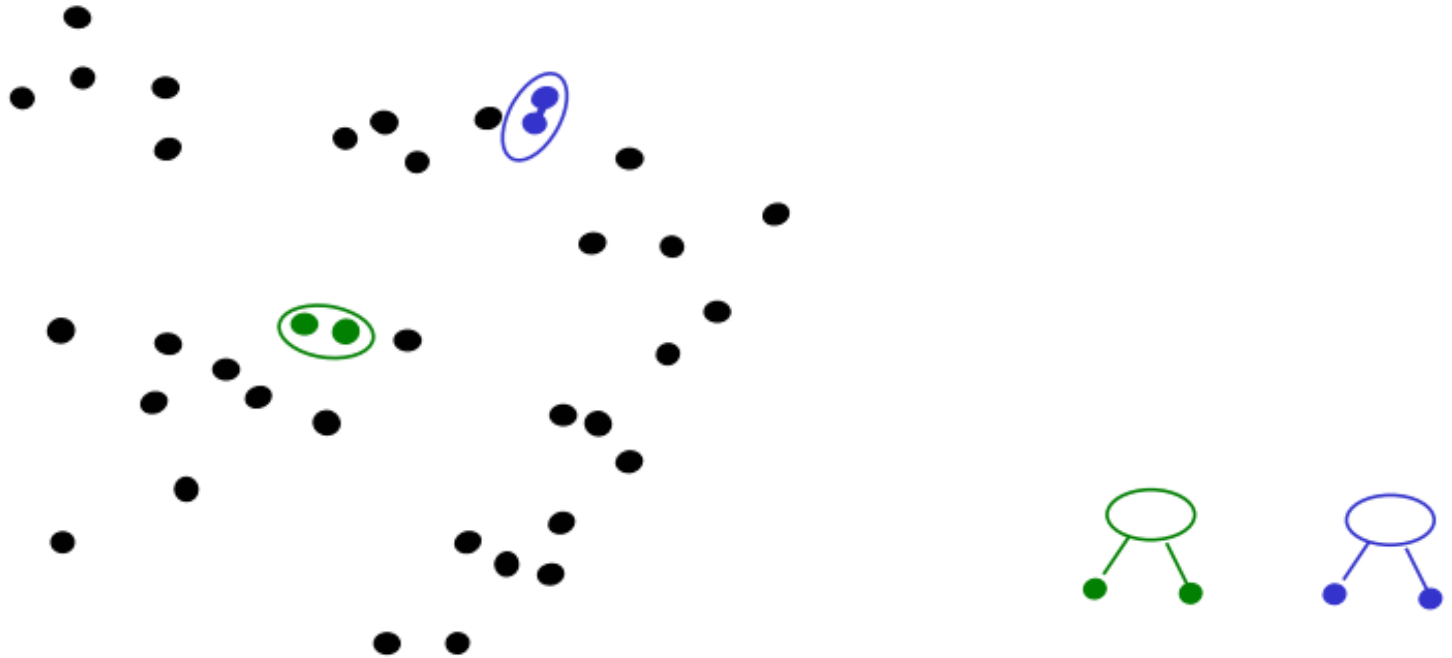
Hierarchical Clustering - **Agglomerative (bottom-up) Clustering**

Say “Every point is its own cluster”

Find “most similar” pair of clusters

Merge it into a parent cluster

Repeat



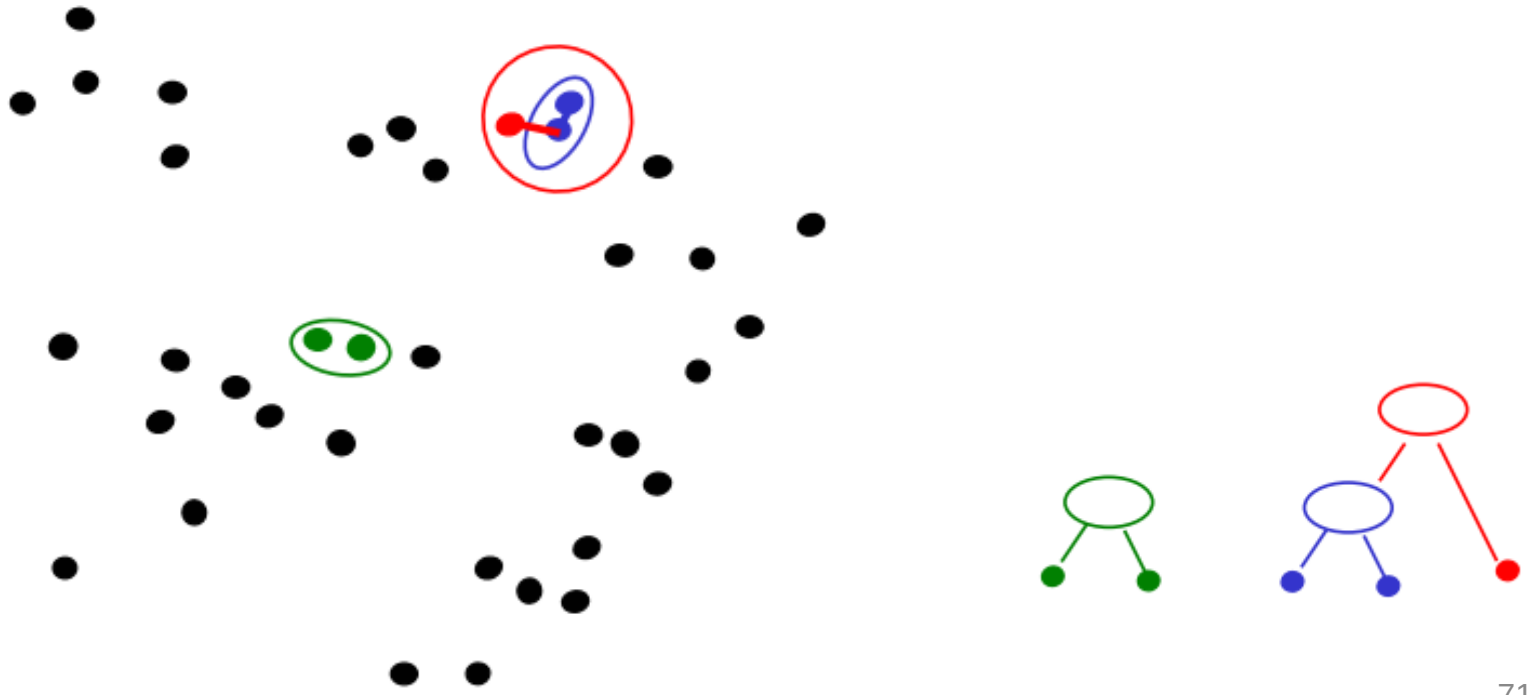
Hierarchical Clustering - **Agglomerative (bottom-up) Clustering**

Say “Every point is its own cluster”

Find “most similar” pair of clusters

Merge it into a parent cluster

Repeat until you’ve merged the whole dataset into one cluster



Hierarchical Clustering - **Agglomerative (bottom-up) Clustering**

1. Divisive Clustering is a top-down method of Hierarchical clustering

1. Start with all examples in the same cluster

1. At each time-step, remove the “outsiders” from the least cohesive cluster

1. Stop when each example is in its own singleton cluster, else go to 2

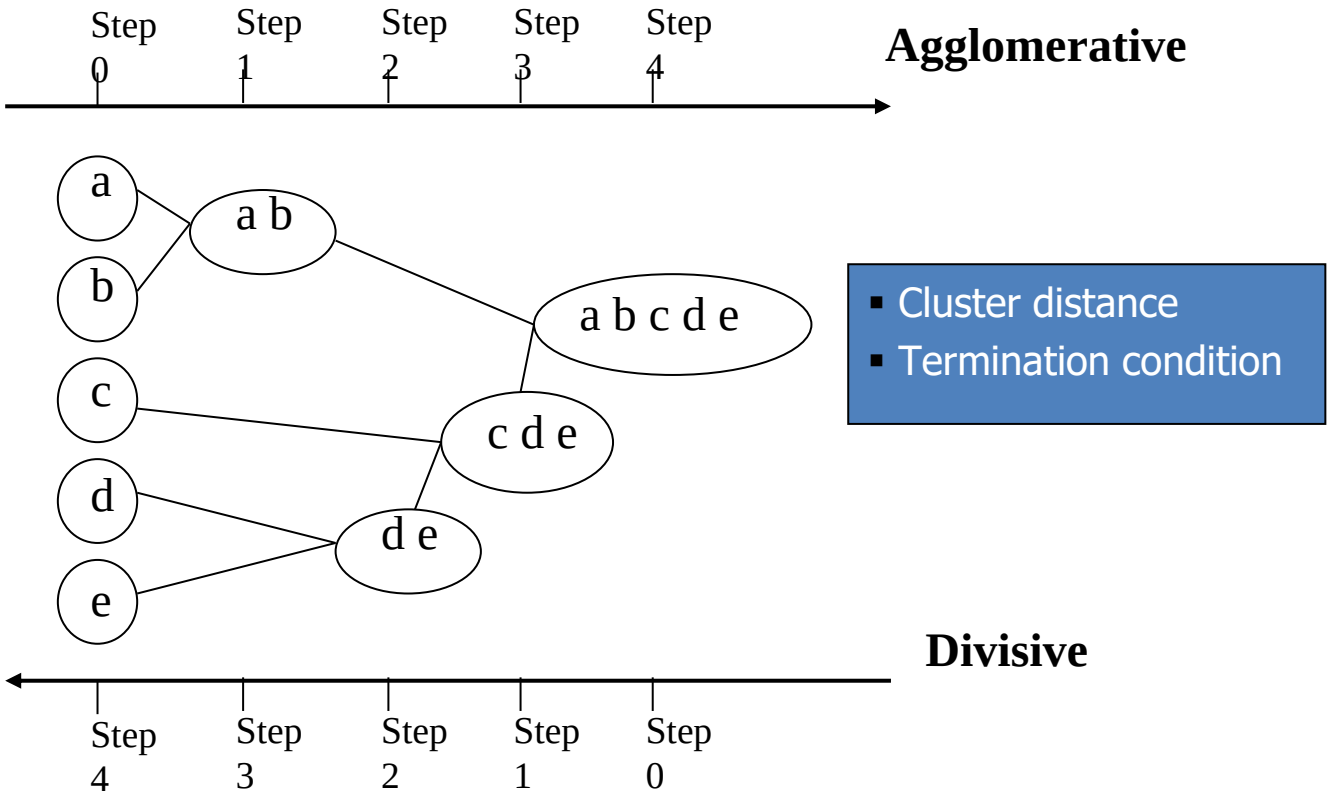
Introduction

- Hierarchical Clustering Approach
  - A typical clustering analysis approach via partitioning data set sequentially
  - Construct nested partitions layer by layer via grouping objects into a tree of clusters (without the need to know the number of clusters in advance)
  - Use (generalised) distance matrix as clustering criteria
- Agglomerative vs. Divisive
  - Agglomerative: a bottom-up strategy
    - Initially each data object is in its own (atomic) cluster
    - Then merge these atomic clusters into larger and larger clusters
  - Divisive: a top-down strategy
    - Initially all objects are in one single cluster
    - Then the cluster is subdivided into smaller and smaller clusters

Introduction: Illustration

- Illustrative Example: Agglomerative vs. Divisive

  Agglomerative and divisive clustering on the data set {a, b, c, d ,e }



Cluster Distance Measures

Example: Given a data set of five objects characterised by a single continuous feature, assume that there are two clusters: C_1: {a, b} and C_2: {c, d, e}.

	a	b	c	d	e
Feature	1	2	4	5	6

1. Calculate the 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

2. Calculate three cluster distances between C_1 and C_2.

Single link

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

Complete link

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

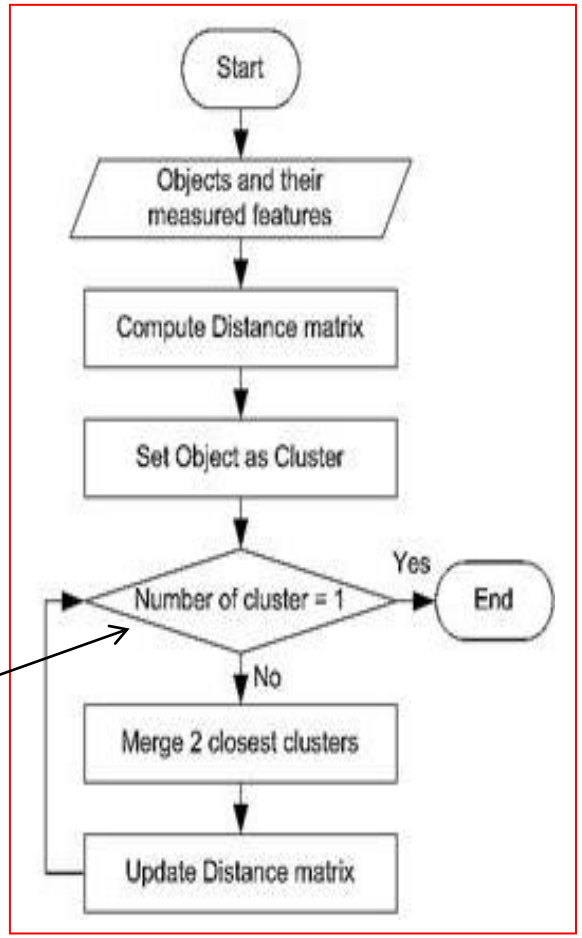
Average

$$\text{dist}(C_1, C_2) = \frac{d(a,c) + d(a,d) + d(a,e) + d(b,c) + d(b,d) + d(b,e)}{6}$$
$$= \frac{3+4+5+2+3+4}{6} = \frac{21}{6} = 3.5$$

Agglomerative Algorithm

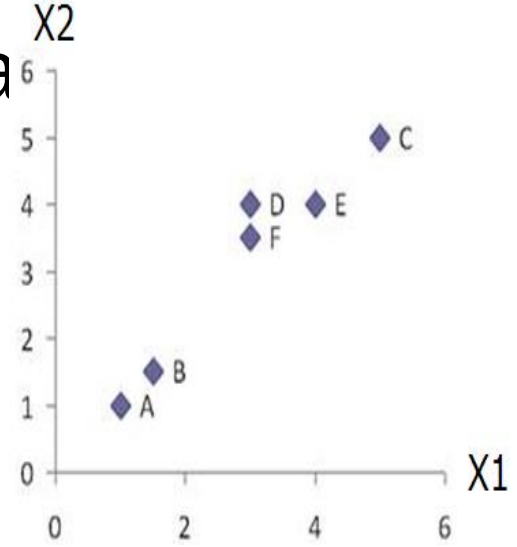
- The *Agglomerative* algorithm is carried out in three steps:

1) Convert all object features into a distance matrix
2) Set each object as a cluster (thus if we have *N* objects, we will have *N* clusters at the beginning)
3) Repeat until number of cluster is one (or known # of clusters)
   - Merge two closest clusters
   - Update "distance matrix"



Example

- Problem: clustering analysis with a algor



	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5

data matrix

$$d_{AB} = \left((1-1.5)^2 + (1-1.5)^2\right)^{\frac{1}{2}} = \sqrt{\frac{1}{2}} = 0.7071$$

$$d_{DF} = \left((3-3)^2 + (4-3.5)^2\right)^{\frac{1}{2}} = 0.5$$

Euclidean distance

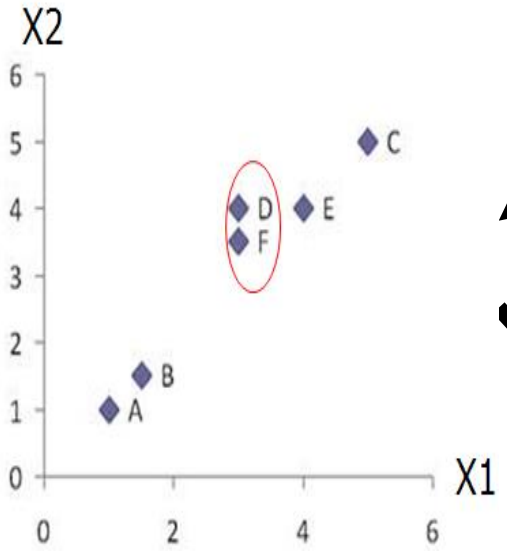
$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

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

distance matrix

Example

- Merge two closest clusters (iteration 1)



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

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

Example

- Update distance matrix (iteration

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

$$d_{(D,F)\to A} = \min\left(d_{DA}, d_{FA}\right) = \min\left(3.61, 3.20\right) = 3.20$$

$$d_{(D,F)\to B} = \min\left(d_{DB}, d_{FB}\right) = \min\left(2.92, 2.50\right) = 2.50$$

$$d_{(D,F)\to C} = \min\left(d_{DC}, d_{FC}\right) = \min\left(2.24, 2.50\right) = 2.24$$

$$d_{E\to(D,F)} = \min\left(d_{ED}, d_{EF}\right) = \min\left(1.00, 1.12\right) = 1.00$$

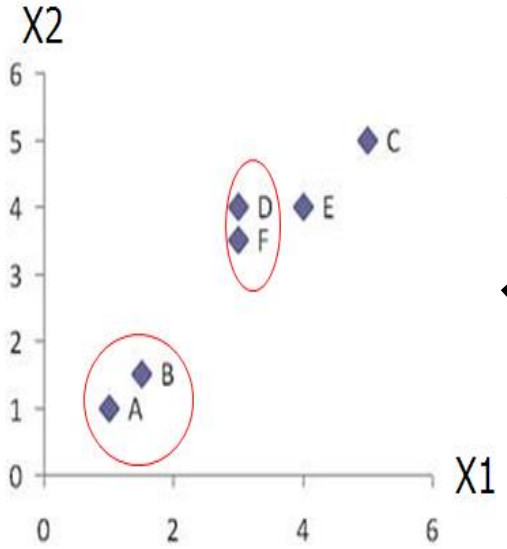
Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

Example

- Merge two closest clusters (iteration 2)



Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Example

- Update distance matrix (iteration

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

$$d_{C\to(A,B)} = \min\left(d_{CA}, d_{CB}\right) = \min\left(5.66, 4.95\right) = 4.95$$

$$d_{(D,F)\to(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}\right) = \min\left(3.61, 2.92, 3.20, 2.50\right) = 2.50$$

$$d_{E\to(A,B)} = \min\left(d_{EA}, d_{EB}\right) = \min\left(4.24, 3.54\right) = 3.54$$

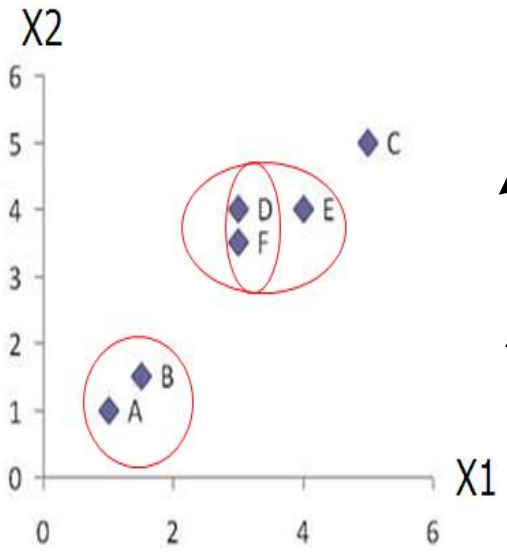
Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Min Distance (Single Linkage)

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	**4.95**	0	2.24	1.41
(D, F)	**2.50**	2.24	0	1.00
E	**3.54**	1.41	1.00	0

Example

- Merge two closest clusters/update distance matrix (



Min Distance (Single Linkage)

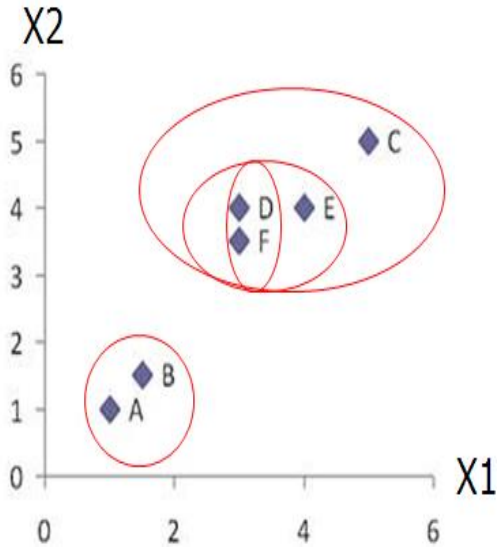
Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	**4.95**	0	2.24	1.41
(D, F)	**2.50**	2.24	0	1.00
E	**3.54**	1.41	1.00	0

Min Distance (Single Linkage)

Dist	(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

Example

- Merge two closest clusters/update distance matrix



Min Distance (Single Linkage)

Dist	(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

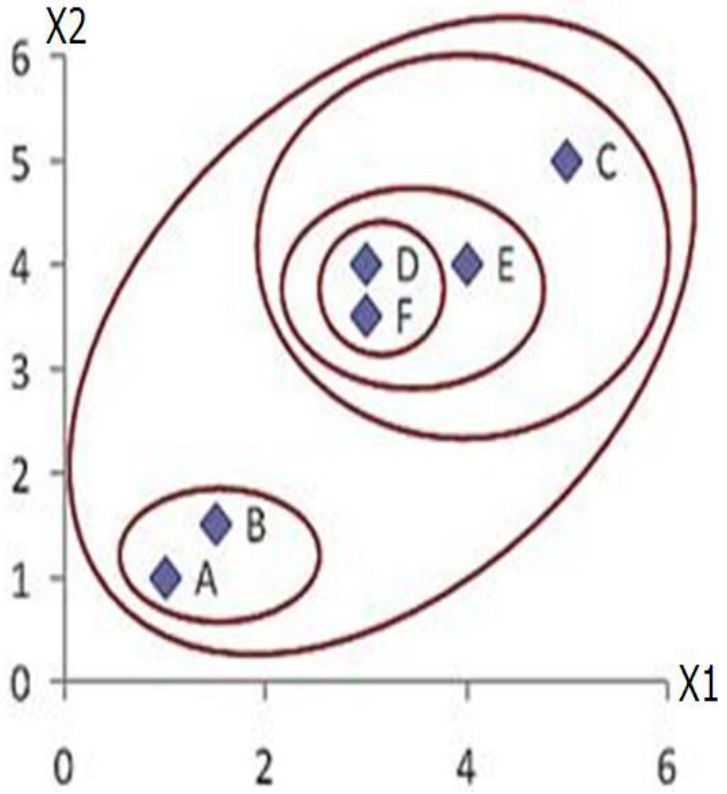
Min Distance (Single Linkage)

Dist	(A,B)	((D, F), E),C
(A,B)	0.00	2.50
((D, F), E),C	**2.50**	0.00

Example

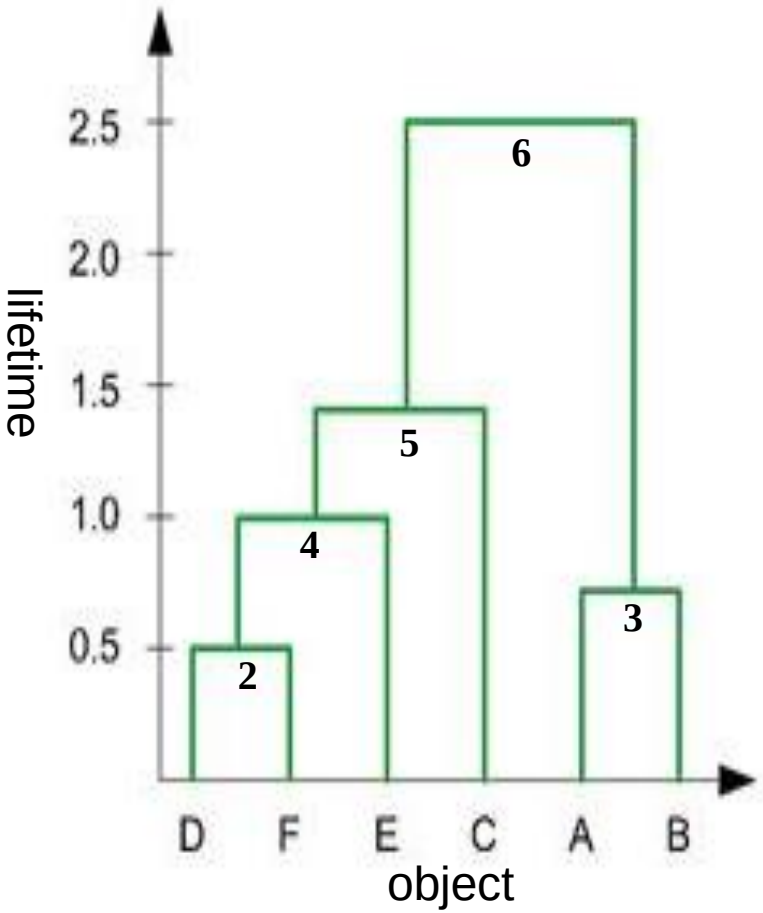
- Final result (meeting termination

	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5



Key Concepts in Hierarchal Clustering

- Dendrogram tree representation



1. In the beginning we have 6 clusters: A, B, C, D, E and F
2. We merge clusters D and F into cluster (D, F) at distance 0.50
3. We merge cluster A and cluster B into (A, B) at distance 0.71
4. We merge clusters E and (D, F) into ((D, F), E) at distance 1.00
5. We merge clusters ((D, F), E) and C into (((D, F), E), C) at distance 1.41
6. We merge clusters (((D, F), E), C) and (A, B) into ((((D, F), E), C), (A, B)) at distance 2.50
7. The last cluster contain all the objects, thus conclude the computation

Agglomerative Hierarchical Clustering

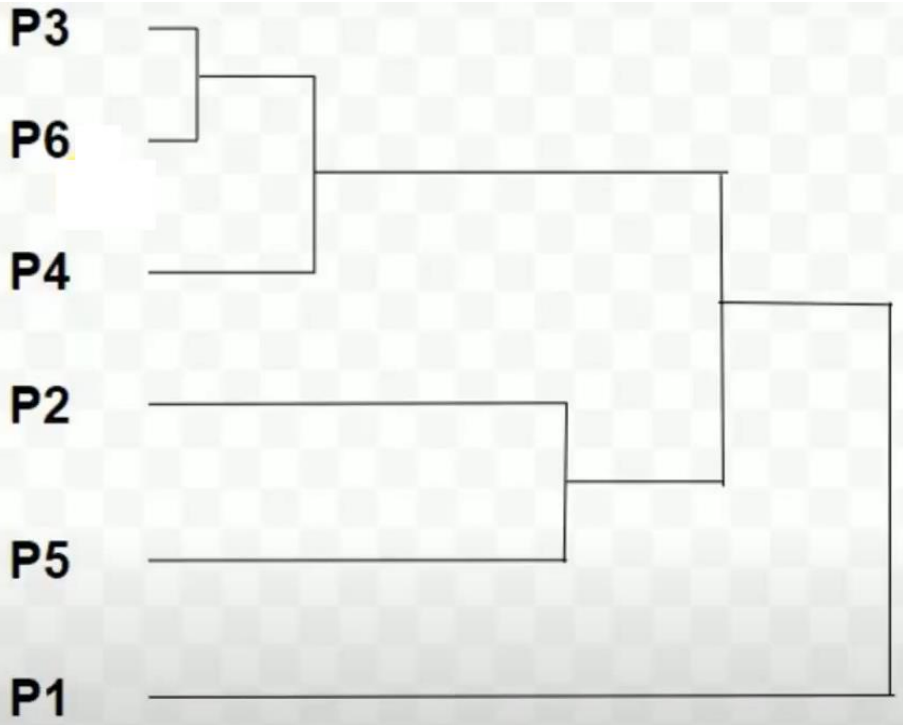
Sample No.	X	Y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30

Sample No.	X	Y
P1	0.40	0.53
P2	0.22	0.38
P3	0.35	0.32
P4	0.26	0.19
P5	0.08	0.41
P6	0.45	0.30

$$\begin{pmatrix} & P1 & P2 & P3 & P4 & P5 & P6 \\ P1 & 0 & & & & & \\ P2 & 0.23 & 0 & & & & \\ P3 & 0.22 & 0.14 & 0 & & & \\ P4 & 0.37 & 0.19 & 0.13 & 0 & & \\ P5 & 0.34 & 0.14 & 0.28 & 0.23 & 0 & \\ P6 & 0.24 & 0.24 & 0.10 & 0.22 & 0.39 & 0 \end{pmatrix}$$

So now we have reached to the solution, the dendrogram for those question will be as follows:

[{(P3, P6), P4}, (P2, P5)] , P1



Dendogram of the cluster formed

K-means clustering vs Hierarchical Clustering

- K-means clustering produces a single partitioning
- Hierarchical Clustering can give different partitionings depending on the level-of-resolution we are looking at
- K-means clustering needs the number of clusters to be specified
- Hierarchical clustering doesn’t need the number of clusters to be specified
- K-means clustering is usually more efficient run-time wise
- Hierarchical clustering can be slow (has to make several merge/split decisions)
- No clear consensus on which of the two produces better clustering

Density-based Clustering

1. DBSCAN

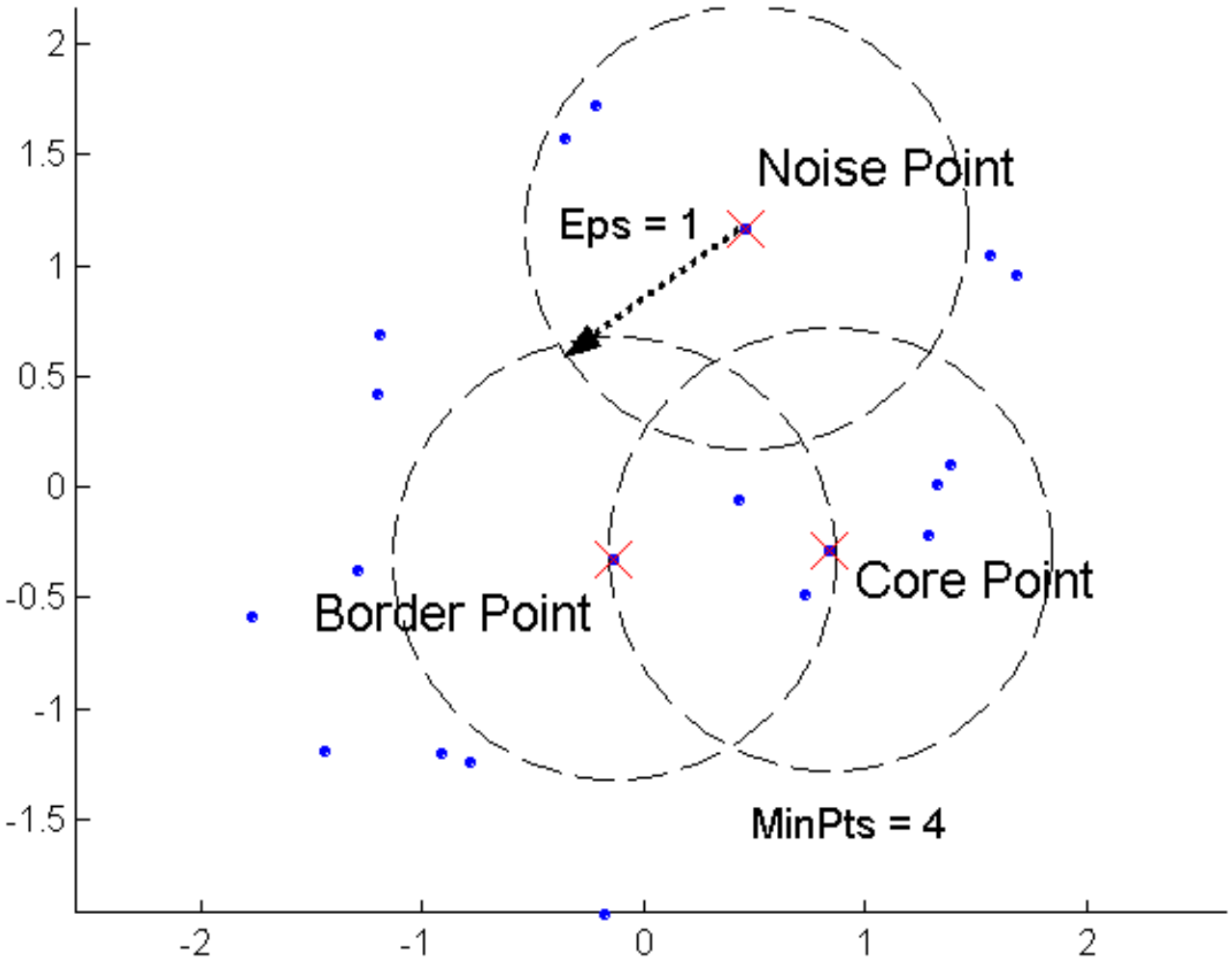
Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points or based on an explicitly constructed density function
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - DENCLUE: Hinneburg & D. Keim (KDD'98/2006)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - CLIQUE: Agrawal, et al. (SIGMOD'98)

DBSCAN

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius r (Eps)
  - 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 or a border point.

DBSCAN: Core, Border, and Noise Points



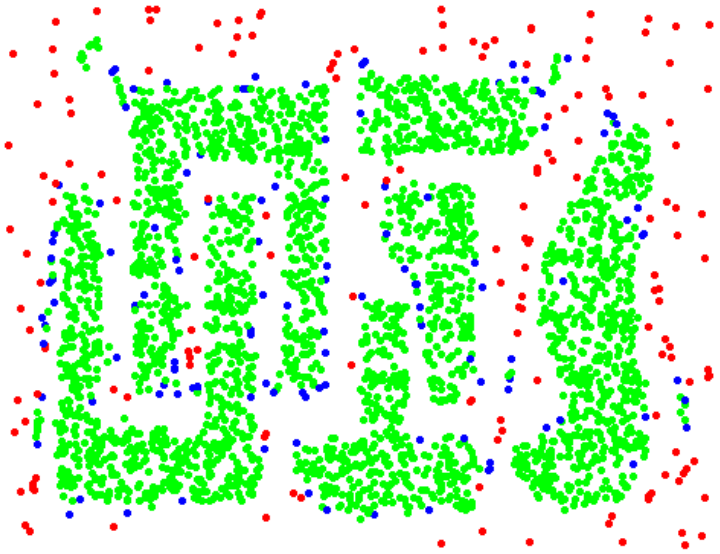
DBSCAN Algorithm (simplified view for teaching)

1. Create a graph whose nodes are the points to be clustered
2. For each core-point c create an edge from c to every point p in the ε-neighborhood of c
3. Set N to the nodes of the graph;
4. If N does not contain any core points terminate
5. Pick a core point c in N
6. Let X be the set of nodes that can be reached from c by going forward;
    1. create a cluster containing $X \cup \{c\}$
    2. $N = N/(X \cup \{c\})$
7. Continue with step 4

Remark: points that are not assigned to any cluster are outliers;

DBSCAN: Core, Border and Noise Points

Original Points

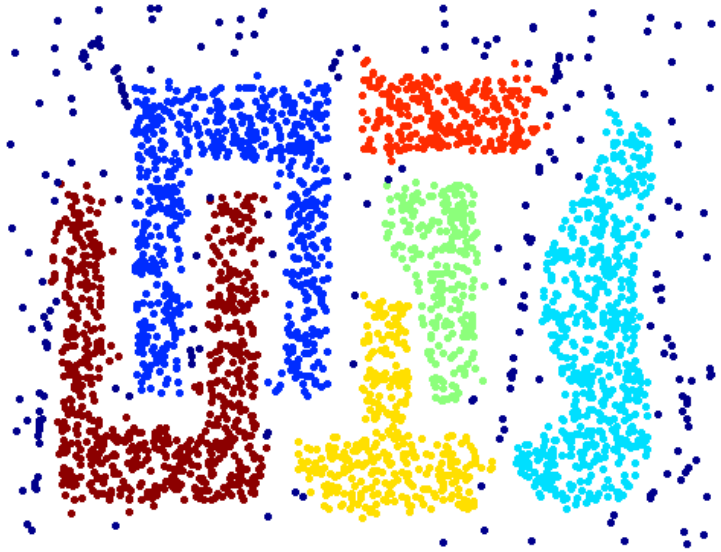


Point types: core, border and noise

Eps = 10, MinPts = 4

When DBSCAN Works Well

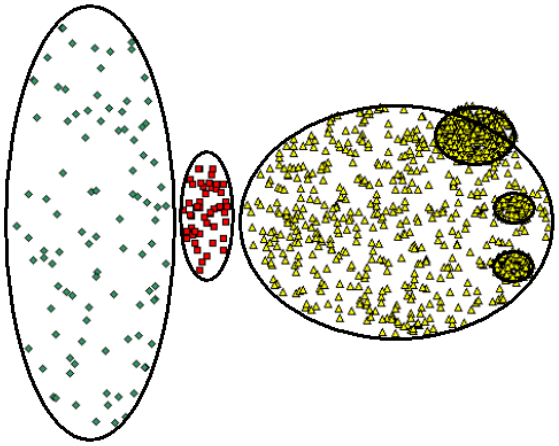
Original Points



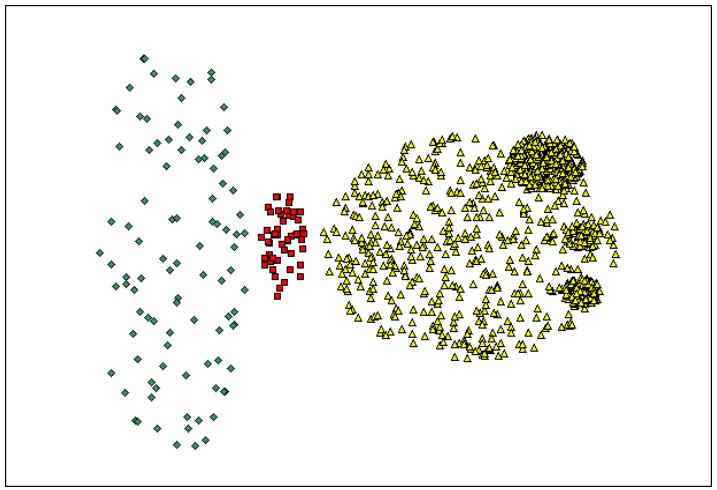
Clusters

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

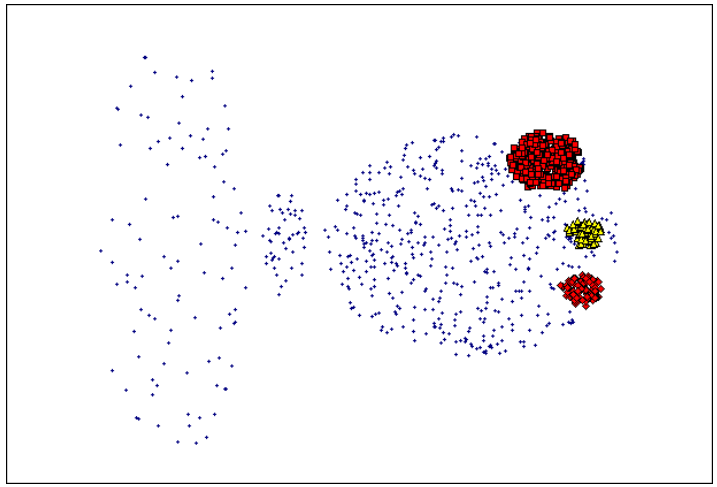
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

- Varying densities
- High-dimensional data

DBSCAN Clustering Algorithm

- Apply the DBSCAN algorithm to the given data points and
- Create the clusters with
- minPts = 4 and
- epsilon (ε) = 1.9.

Data Points:

P1: (3, 7)	P2: (4, 6)
P3: (5, 5)	P4: (6, 4)
P5: (7, 3)	P6: (6, 2)
P7: (7, 2)	P8: (8, 4)
P9: (3, 3)	P10: (2, 6)
P11: (3, 5)	P12: (2, 4)

- Use Eucladian distance and calculate the distance between each points.

$$\boldsymbol{Distance}(\boldsymbol{A}(\boldsymbol{x_1}, \boldsymbol{y_1}), \boldsymbol{B}(\boldsymbol{x_2}, \boldsymbol{y_2})) = \sqrt{(\boldsymbol{x_2} - \boldsymbol{x_1})^2 + (\boldsymbol{y_2} - \boldsymbol{y_1})^2}$$

minPts = 4 and epsilon (ε) = 1.9

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
P1	0											
P2	1.41	0										
P3	2.83	1.41	0									
P4	4.24	2.83	1.41	0								
P5	5.66	4.24	2.83	1.41	0							
P6	5.83	4.47	3.16	2.00	1.41	0						
P7	6.40	5.00	3.61	2.24	1.00	1.00	0					
P8	5.83	4.47	3.16	2.00	1.41	2.83	2.24	0				
P9	4.00	3.16	2.83	3.16	4.00	3.16	4.12	5.10	0			
P10	1.41	2.00	3.16	4.47	5.83	5.66	6.40	6.32	3.16	0		
P11	2.00	1.41	2.00	3.16	4.47	4.24	5.00	5.10	2.00	1.41	0	
P12	3.16	2.83	3.16	4.00	5.10	4.47	5.39	6.00	1.41	2.00	1.41	0

minPts = 4 and epsilon (ε) = 1.9

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
P1	0											
P2	1.41	0										
P3	2.83	1.41	0									
P4	4.24	2.83	1.41	0								
P5	5.66	4.24	2.83	1.41	0							
P6	5.83	4.47	3.16	2.00	1.41	0						
P7	6.40	5.00	3.61	2.24	1.00	1.00	0					
P8	5.83	4.47	3.16	2.00	1.41	2.83	2.24	0				
P9	4.00	3.16	2.83	3.16	4.00	3.16	4.12	5.10	0			
P10	1.41	2.00	3.16	4.47	5.83	5.66	6.40	6.32	3.16	0		
P11	2.00	1.41	2.00	3.16	4.47	4.24	5.00	5.10	2.00	1.41	0	
P12	3.16	2.83	3.16	4.00	5.10	4.47	5.39	6.00	1.41	2.00	1.41	0

P1: P2, P10

P2: P1, P3, P11

P3: P2, P4

P4: P3, P5

P5: P4, P6, P7, P8

P6: P5, P7

P7: P5, P6

P8: P5

P9: P12

P10: P1, P11

P11: P2, P10, P12

P12: P9, P11

Select position which is lesser than 1.9

P1: P2, P10

P2: P1, P3, P11

P3: P2, P4

P4: P3, P5

P5: P4, P6, P7, P8

P6: P5, P7

P7: P5, P6

P8: P5

P9: P12

P10: P1, P11

P11: P2, P10, P12

P12: P9, P11

minPts = 4 and epsilon (ε) = 1.9

Point	Status	
P1	Noise	
P2	Core	
P3	Noise	
P4	Noise	
P5	Core	
P6	Noise	
P7	Noise	
P8	Noise	
P9	Noise	
P10	Noise	
P11	Core	
P12	Noise	

P1: P2, P10

P2: P1, P3, P11

P3: P2, P4

P4: P3, P5

P5: P4, P6, P7, P8

P6: P5, P7

P7: P5, P6

P8: P5

P9: P12

P10: P1, P11

P11: P2, P10, P12

P12: P9, P11

minPts = 4 and epsilon (ε) = 1.9

Point	Status	
P1	Noise	Border
P2	Core	
P3	Noise	Border
P4	Noise	Border
P5	Core	
P6	Noise	Border
P7	Noise	Border
P8	Noise	Border
P9	Noise	
P10	Noise	Border
P11	Core	
P12	Noise	Border

P1: P2, P10

P2: P1, P3, P11

P3: P2, P4

P4: P3, P5

P5: P4, P6, P7, P8

P6: P5, P7

P7: P5, P6

P8: P5

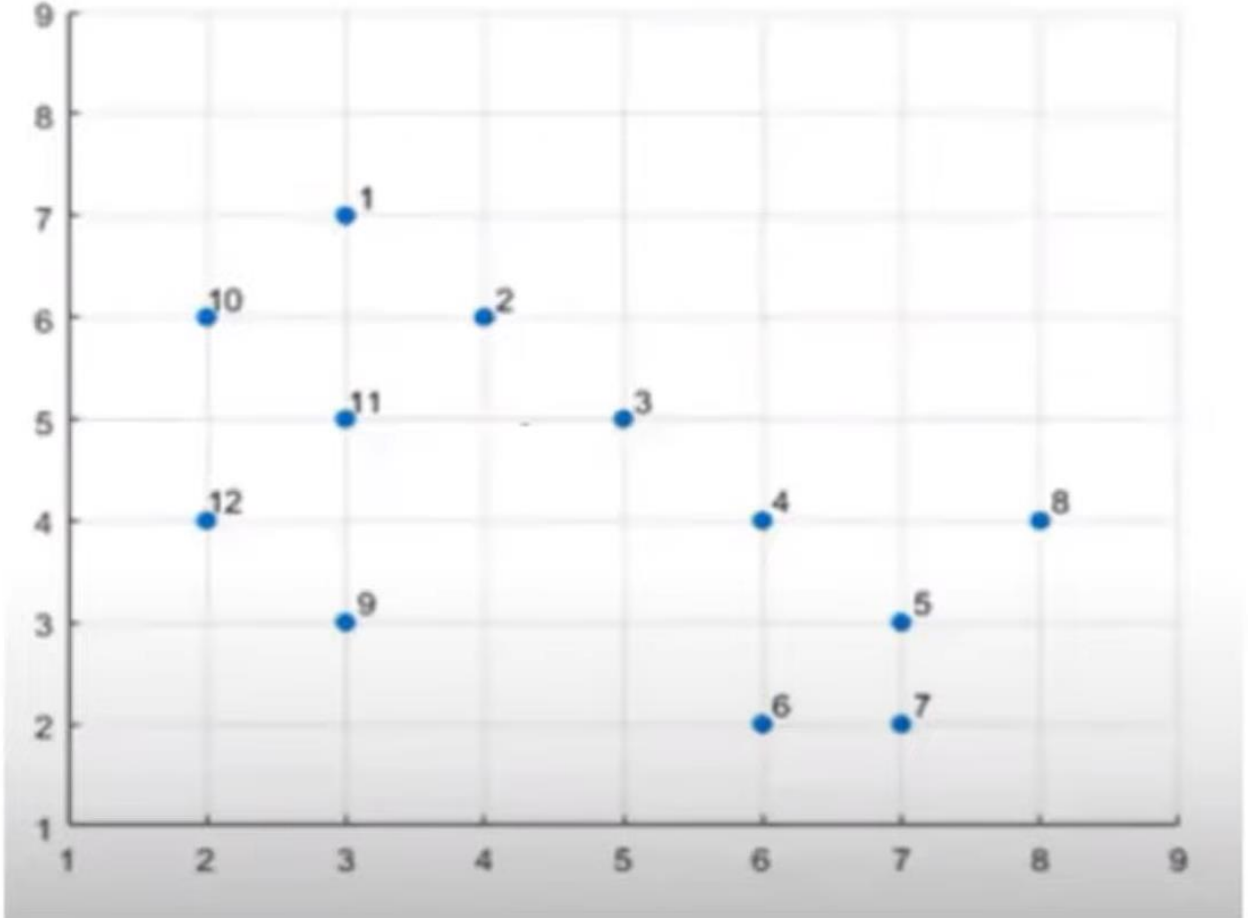
P9: P12

P10: P1, P11

P11: P2, P10, P12

P12: P9, P11

minPts = 4 and epsilon (ε) = 1.9



P1: P2, P10

P2: P1, P3, P11

P3: P2, P4

P4: P3, P5

P5: P4, P6, P7, P8

P6: P5, P7

P7: P5, P6

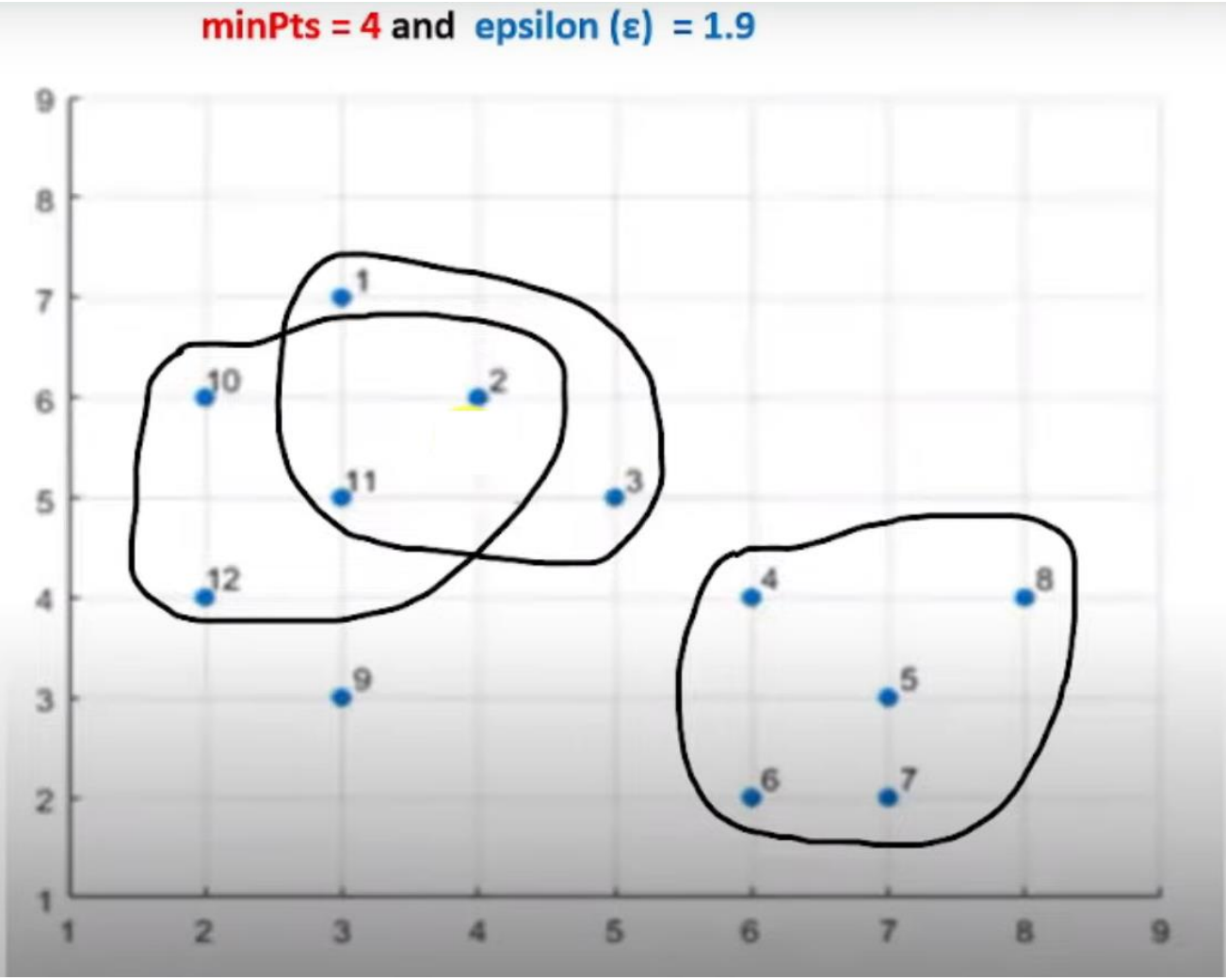
P8: P5

P9: P12

P10: P1, P11

P11: P2, P10, P12

P12: P9, P11



minPts = 2 and Similarity Index = 0.8

	P1	P2	P3	P4	P5
P1	1.00	0.10	0.41	0.55	0.35
P2	0.10	1.00	0.64	0.47	0.98
P3	0.41	0.64	1.00	0.44	0.85
P4	0.55	0.47	0.44	1.00	0.76
P5	0.35	0.98	0.85	0.76	1.00

minPts = 2 and Similarity Index = 0.8

	P1	P2	P3	P4	P5
P1	1.00	0.10	0.41	0.55	0.35
P2	0.10	1.00	0.64	0.47	0.98
P3	0.41	0.64	1.00	0.44	0.85
P4	0.55	0.47	0.44	1.00	0.76
P5	0.35	0.98	0.85	0.76	1.00

P1: -

P2: P5

P3: P5

P4: -

P5: P2, P3

Point	Status	
P1	Noise	
P2	Core	
P3	Core	
P4	Noise	
P5	Core	

No Border Points in the given dataset

Association Rule Learning

- Association rule learning is a type of **unsupervised learning technique** that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of dataset. It is based on different rules to discover the interesting relations between variables in the database.
- The association rule learning is one of the very important concepts of machine learning, and it is employed in **Market Basket analysis, Web usage mining, continuous production, etc.** Here market basket analysis is a technique used by the various big retailer to discover the associations between items. We can understand it by taking an example of a supermarket, as in a supermarket, all products that are purchased together are put together.
- For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored within a shelf or mostly nearby. Consider the below diagram:

- Association rule learning can be divided into three types of algorithms:

1. Apriori
2. Eclat
3. F-P Growth Algorithm

- Association rule learning works on the concept of If and Else Statement, such as if A then B.

- Here the If element is called **antecedent**, and then statement is called as **Consequent**. These types of relationships where we can find out some association or relation between two items is known *as single cardinality*. It is all about creating rules, and if the number of items increases, then cardinality also increases accordingly. So, to measure the associations between thousands of data items, there are several metrics. These metrics are given below:
    - **Support**
    - **Confidence**
    - **Lift**

- Support
- Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as:

$$\text{Supp}(X) = \frac{Freq(X)}{T}$$

- Confidence
- Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.
- $$\text{Confidence} = \frac{Freq(X,Y)}{Freq(X)}$$

- Lift
- It is the strength of any rule, which can be defined as below formula:

$$\text{Lift} = \frac{Supp(X,Y)}{Supp(X) \times Supp(Y)}$$

It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

•If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.

•**Lift>1**: It determines the degree to which the two itemsets are dependent to each other.

•**Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

Types of Association Rule Lerning

- Association rule learning can be divided into three algorithms:
- Apriori Algorithm
- This algorithm uses frequent datasets to generate association rules. It is designed to work on the databases that contain transactions. This algorithm uses a breadth-first search and Hash Tree to calculate the itemset efficiently.
- It is mainly used for market basket analysis and helps to understand the products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.
- Eclat Algorithm
- Eclat algorithm stands for **Equivalence Class Transformation**. This algorithm uses a depth-first search technique to find frequent itemsets in a transaction database. It performs faster execution than Apriori Algorithm.
- F-P Growth Algorithm
- The F-P growth algorithm stands for **Frequent Pattern**, and it is the improved version of the Apriori Algorithm. It represents the database in the form of a tree structure that is known as a frequent pattern or tree. The purpose of this frequent tree is to extract the most frequent patterns.

- Applications of Association Rule Learning
- It has various applications in machine learning and data mining. Below are some popular applications of association rule learning:
- **Market Basket Analysis:** It is one of the popular examples and applications of association rule mining. This technique is commonly used by big retailers to determine the association between items.
- **Medical Diagnosis:** With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.
- **Protein Sequence:** The association rules help in determining the synthesis of artificial Proteins.
- It is also used for the **Catalog Design** and **Loss-leader Analysis** and many more other applications.

Association Rules Exercise

- Here are a dozen sales transactions.
- The objective is to use this transaction data to find affinities between products, that is, which products sell together often.
- The support level will be set at 33 percent; the confidence level will be set at 50 percent.

Association Rules Exercise

$$Rule: \quad X \Rightarrow Y$$

$$Support = \frac{frq(X,Y)}{N}$$

$$Confidence = \frac{frq(X,Y)}{frq(X)}$$

Transactions List

1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	
4	Milk	Bread	Butter	
5	Bread	Butter	Cookies	
6	Milk	Bread	Butter	Cookies
7	Milk	Cookies		
8	Milk	Bread	Butter	
9	Bread	Butter	Egg	Cookies
10	Milk	Butter	Bread	
11	Milk	Bread	Butter	
12	Milk	Bread	Cookies	Ketchup

1-item Sets	Frequency
Milk	9
Bread	10
Butter	10
Egg	3
Ketchup	3
Cookies	5

Frequent 1-item Sets	Frequency
Milk	9
Bread	10
Butter	10
Cookies	5

Transactions List

1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	
4	Milk	Bread	Butter	
5	Bread	Butter	Cookies	
6	Milk	Bread	Butter	Cookies
7	Milk	Cookies		
8	Milk	Bread	Butter	
9	Bread	Butter	Egg	Cookies
10	Milk	Butter	Bread	
11	Milk	Bread	Butter	
12	Milk	Bread	Cookies	Ketchup

2-item Sets	Frequency
Milk, Bread	7
Milk, Butter	7
Milk, Cookies	3
Bread, Butter	9
Butter, Cookies	3
Bread, Cookies	4

Frequent 2-item Sets	Frequency
Milk, Bread	7
Milk, Butter	7
Bread, Butter	9
Bread, Cookies	4

1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	
4	Milk	Bread	Butter	
5	Bread	Butter	Cookies	
6	Milk	Bread	Butter	Cookies
7	Milk	Cookies		
8	Milk	Bread	Butter	
9	Bread	Butter	Egg	Cookies
10	Milk	Butter	Bread	
11	Milk	Bread	Butter	
12	Milk	Bread	Cookies	Ketchup

Milk, Bread, Butter, Cookies

3-item Sets	**Frequency**
Milk, Bread, Butter	6
Milk, Bread, Cookies	1
Bread, Butter, Cookies	3
Milk, Butter, Cookies	2

Frequent 3-item Sets	Frequency
Milk, Bread, Butter	6

Association Rule Mining - Subset Creation

- Frequent 3-Item Set = I => {Milk, Bread, Butter}
- Non-Empty subset are
  - {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}
- How to form Association Rule…?
  - For every non-empty subset S of I, the association rule is,
    - **S →(I-S)**
    - **If support(I) / support(S) >= min_confidence**

Association Rule Mining - Subset Creation

- Non-Empty subset are
  - {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}
  - Min_Support = 30% and Min_Confidence = 60%

- Rule 1: {Milk} → {Bread, Butter} {S=50%, C=66.67%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Milk) = $\frac{6/12}{9/12}$ = 6/9 = 66.67% > 60%
  - Valid

- Rule 2: {Bread} → {Milk, Butter} {S=50%, C=60%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Bread) = 6/10 = 60% >= 60%
  - Valid

Association Rule Mining - Subset Creation

- Non-Empty subset are
  - {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}
  - Min_Support = 30% and Min_Confidence = 60%

- Rule 3: {Butter} → {Milk, Bread} {S=50%, C=60%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Butter) = 6/10 = 60%>=60
  - Valid

- Rule 4: {Milk, Bread} → {Butter} {S=50%, C=85.7%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Milk, Bread) = 6/7 = 85.7% > 60%

Association Rule Mining - Subset Creation

- Non-Empty subset are
  - {{Milk}, {Bread}, {Butter}, {Milk, Bread}, {Milk, Butter}, {Bread, Butter}}
  - Min_Support = 30% and Min_Confidence = 60%

- Rule 5: {Milk, Butter} → {Bread} {S=50%, C=85.7%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Milk, Butter) = 6/7 = 85.7% >= 60%
  - Valid

- Rule 6: {Bread, Butter} → {Milk} {S=50%, C=66.67%}
  - Support = 6/12 = 50%
  - Confidence = Support (Milk, Bread, Butter)/Support(Bread, Butter) = 6/9 = 66.67%>=60
  - Valid

Hidden Markov Model

- Hidden Markov Models (HMMs) are a type of probabilistic model that are commonly used in machine learning for tasks such as speech recognition, natural language processing, and bioinformatics. They are a popular choice for modelling sequences of data because they can effectively capture the underlying structure of the data, even when the data is noisy or incomplete.

What are Hidden Markov Models?

A Hidden Markov Model (HMM) is a probabilistic model that consists of a sequence of hidden states, each of which generates an observation. The hidden states are usually not directly observable, and the goal of HMM is to estimate the sequence of hidden states based on a sequence of observations. An HMM is defined by the following components:

A set of N hidden states, S = {s1, s2, ..., sN}.

A set of M observations, O = {o1, o2, ..., oM}.

An initial state probability distribution, ? = {?1, ?2, ..., ?N}, which specifies the probability of starting in each hidden state.

A transition probability matrix, A = [aij], defines the probability of moving from one hidden state to another.

An emission probability matrix, B = [bjk], defines the probability of emitting an observation from a given hidden state.

The basic idea behind an HMM is that the hidden states generate the observations, and the observed data is used to estimate the hidden state sequence. This is often referred to as **the forward-backwards algorithm.**

- Applications of Hidden Markov Models
- Now, we will explore some of the key applications of HMMs, including speech recognition, natural language processing, bioinformatics, and finance
- **Speech Recognition**
  One of the most well-known applications of HMMs is speech recognition. In this field, HMMs are used to model the different sounds and phones that makeup speech. The hidden states, in this case, correspond to the different sounds or phones, and the observations are the acoustic signals that are generated by the speech. The goal is to estimate the hidden state sequence, which corresponds to the transcription of the speech, based on the observed acoustic signals. HMMs are particularly well-suited for speech recognition because they can effectively capture the underlying structure of the speech, even when the data is noisy or incomplete. In speech recognition systems, the HMMs are usually trained on large datasets of speech signals, and the estimated parameters of the HMMs are used to transcribe speech in real time.

•**Natural Language Processing**

Another important application of HMMs is natural language processing. In this field, HMMs are used for tasks such as **part-of-speech tagging, named entity recognition, and text classification**. In these applications, the hidden states are typically associated with the underlying grammar or structure of the text, while the observations are the words in the text. The goal is to estimate the hidden state sequence, which corresponds to the structure or meaning of the text, based on the observed words. HMMs are useful in natural language processing because they can effectively capture the underlying structure of the text, even when the data is noisy or ambiguous. In natural language processing systems, the HMMs are usually trained on large datasets of text, and the estimated parameters of the HMMs are used to perform various NLP tasks, such as text classification, part-of-speech tagging, and named entity recognition.

•**Bioinformatics**

HMMs are also widely used in bioinformatics, where they are used to model sequences of DNA, RNA, and proteins. The hidden states, in this case, correspond to the different types of residues, while the observations are the sequences of residues. The goal is to estimate the hidden state sequence, which corresponds to the underlying structure of the molecule, based on the observed sequences of residues. HMMs are useful in bioinformatics because they can effectively capture the underlying structure of the molecule, even when the data is noisy or incomplete. In bioinformatics systems, the HMMs are usually trained on large datasets of molecular sequences, and the estimated parameters of the HMMs are used to predict the structure or function of new molecular sequences.

- **Finance**
  Finally, HMMs have also been used in finance, where they are used to model stock prices, interest rates, and currency exchange rates. In these applications, the hidden states correspond to different economic states, such as bull and bear markets, while the observations are the stock prices, interest rates, or exchange rates. The goal is to estimate the hidden state sequence, which corresponds to the underlying economic state, based on the observed prices, rates, or exchange rates. HMMs are useful in finance because they can effectively capture the underlying economic state, even when the data is noisy or incomplete. In finance systems, the HMMs are usually trained on large datasets of financial data, and the estimated parameters of the HMMs are used to make predictions about future market trends or to develop investment strategies.

- Limitations of Hidden Markov Models
- Now, we will explore some of the key limitations of HMMs and discuss how they can impact the accuracy and performance of HMM-based systems.
- **Limited Modeling Capabilities**
  One of the key limitations of HMMs is that they are relatively limited in their modelling capabilities. HMMs are designed to model sequences of data, where the underlying structure of the data is represented by a set of hidden states. However, the structure of the data can be quite complex, and the simple structure of HMMs may not be enough to accurately capture all the details. For example, in speech recognition, the complex relationship between the speech sounds and the corresponding acoustic signals may not be fully captured by the simple structure of an HMM.
- **Overfitting**
  Another limitation of HMMs is that they can be prone to overfitting, especially when the number of hidden states is large or the amount of training data is limited. Overfitting occurs when the model fits the training data too well and is unable to generalize to new data. This can lead to poor performance when the model is applied to real-world data and can result in high error rates. To avoid overfitting, it is important to carefully choose the number of hidden states and to use appropriate regularization techniques.

- **Lack of Robustness**
  HMMs are also limited in their robustness to noise and variability in the data. For example, in speech recognition, the acoustic signals generated by speech can be subjected to a variety of distortions and noise, which can make it difficult for the HMM to accurately estimate the underlying structure of the data. In some cases, these distortions and noise can cause the HMM to make incorrect decisions, which can result in poor performance. To address these limitations, it is often necessary to use additional processing and filtering techniques, such as noise reduction and normalization, to pre-process the data before it is fed into the HMM.
- **Computational Complexity**
  Finally, HMMs can also be limited by their computational complexity, especially when dealing with large amounts of data or when using complex models. The computational complexity of HMMs is due to the need to estimate the parameters of the model and to compute the likelihood of the data given in the model. This can be time-consuming and computationally expensive, especially for large models or for data that is sampled at a high frequency. To address this limitation, it is often necessary to use parallel computing techniques or to use approximations that reduce the computational complexity of the model.