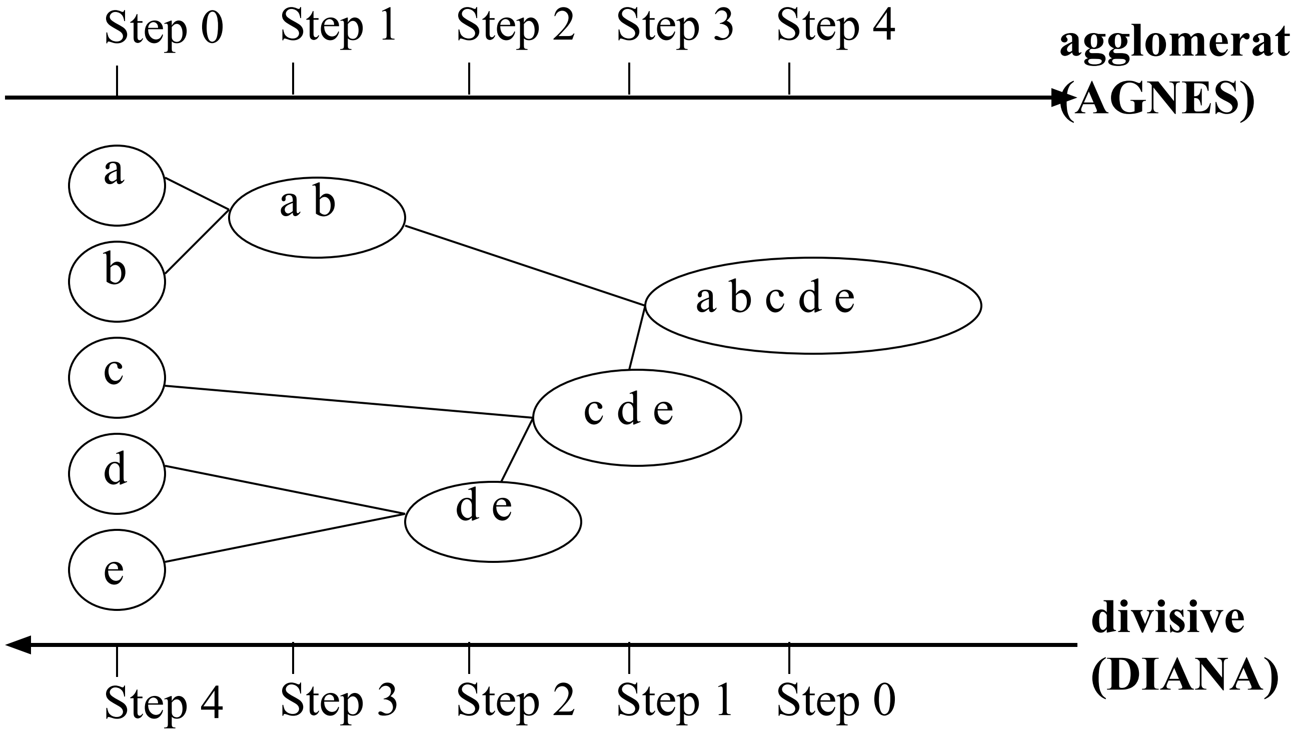
Hierarchical clustering

Hierarchical Clustering algorithms

- **Agglomerative (bottom-up):**
  - Start with each document being a single cluster.
  - Eventually all documents belong to the same cluster.
- **Divisive (top-down):**
  - Start with all documents belong to the same cluster.
  - Eventually each node forms a cluster on its own.
- Does not require the number of clusters ***k*** in advance
- Needs a termination/readout condition
  - The final mode in both Agglomerative and Divisive is of no use.

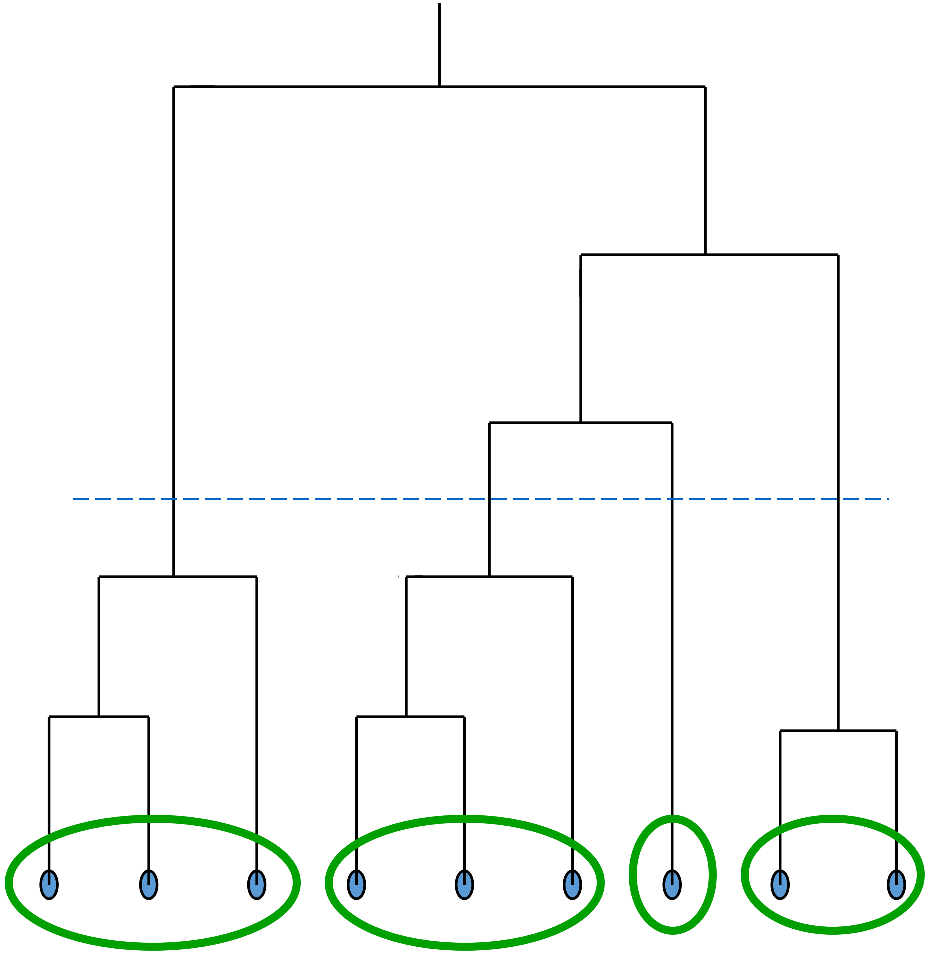
Hierarchical Clustering

- **Agglomerative: Bottom up approach**
- **Divisive :Top down approach**



Dendogram: Hierarchical Clustering

- Clustering obtained by cutting the dendrogram at a desired level
- Each connected component forms a cluster.

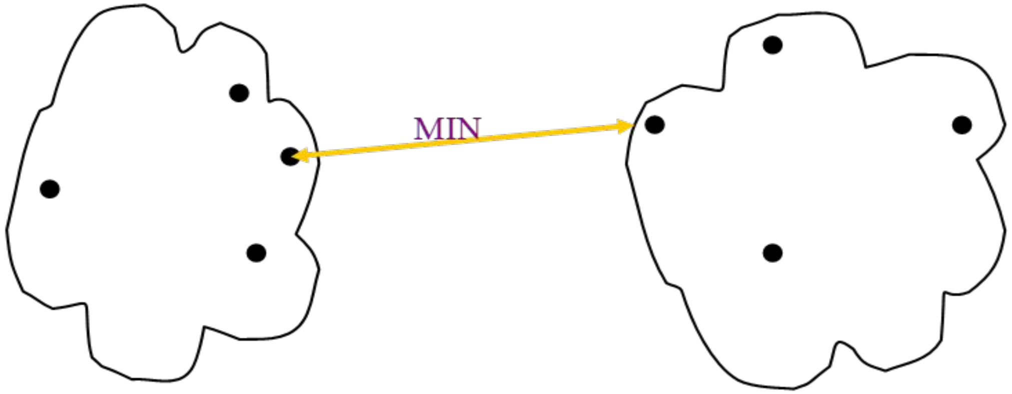


Agglomerative Hierarchical clustering method

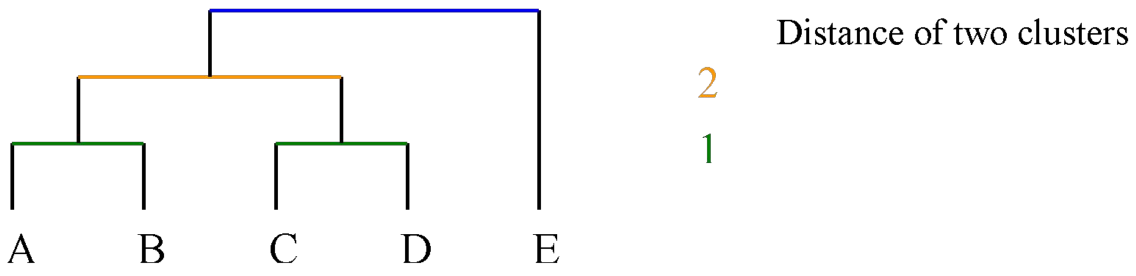
- Single link algorithm
- Complete link algorithm
- Average link algorithm

Single Link Clustering

- Single link algorithm is an example of agglomerative hierarchical clustering method.
- We recall that is a bottom-up strategy: compare each point with each point. Each object is placed in a separate cluster, and at each step we merge the closest pair of clusters, until certain termination conditions are satisfied. This requires defining a notion of cluster proximity.
- For the single link, the proximity of two clusters is defined as the minimum of the distance between any two points in the two clusters.



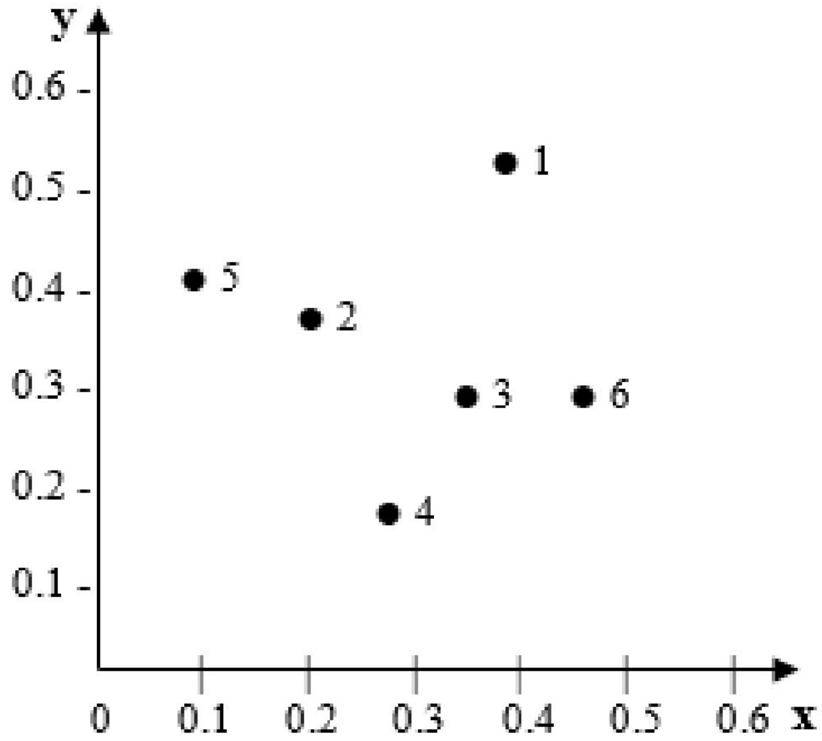
- **Dendrogram** – shows the same information as in the graph above.
- However distance threshold is vertical, and points are at the bottom (horizontal).
- The height at which two clusters are merged in the dendrogram reflects the distance of the two clusters



Example: Assume that the database D is given by the table below. Follow single link technique to find clusters in D. Use Euclidean distance measure.

	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

- Solution:
- Step 1. Plot the objects in *n*-dimensional space (where *n* is the number of attributes). In our case we have 2 attributes – x and y, so we plot the objects p1, p2 … p6 in 2-dimensional space:



- <u>Step 2.</u> Calculate the distance from each object (point) to all other points, using Euclidean distance measure, and place the numbers in a distance matrix.

$$D(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j1}|^2 + \ldots + |x_{ip} - x_{jp}|^2}$$

$$d(p1, p2) = |x_{p1} - x_{p1}|^2 + |y_{p1} - y_{p2}|^2$$

$$= \sqrt{|0.40 - 0.22|^2 + |0.53 - 0.38|^2}$$

$$= \sqrt{|0.18|^2 + |0.15|^2}$$

$$= \sqrt{0.0324 + 0.0225}$$

$$= \sqrt{0.0549}$$

$$= 0.2343$$

K means Clustering Example

Q 1) solve the following using K Means Clustering with K=2

{2, 3,4,10,11,12,20,25,30}

Solution:

Step1: M1= 2, M2=10 ………………………….> (Random centroid)

Assign the points to cluster based on minimum distance with centroid

K1= {2,3,4,}

K2= {10,11,12,20,25,30}

Step 2: New centroid points has to be calculated

M1=? M2=?

M1= (2+3+4)/3= 3

M2= (10+11+12+20+25+30)/6=18

K1= {2,3,4,10}

K2= {11,12,20,25,30}

Step 3: Reassign the centroid

M1=4.75   M2=19.6

K1= {2,3,4,10,11,12}

K2= {20,25,30}

Step 4: Reasign the centroid

M1= 7, M2= 25

K1= {2,3,4,10,11,12}

K2= {20,25,30}

Step 5: Reassign the centroid

M1= 7, M2= 25

Final clusters are K1={2,3,4,10,11,12 } &

K2={20,25,30 }

Q2) Cluster the given set of values {2,3,6,8,9,12,15,18,22} into three clusters.

Solution:

Step 1: M1=2, M2=3, M3=6

K1={2}

K2={3}

K3={6,8,9,12,15,18,22}

Final mean values are M1=2.5, M2=8.75, M3=18.33

K1={2,3}

K2={6,8,9,12}

K3={15,18,22}

What is K-Medoids Clustering?

K-Medoids clustering is an unsupervised machine learning algorithm used to group data into different clusters. It is an iterative algorithm that starts by selecting k data points as medoids in a dataset. After this, the distance between each data point and the medoids is calculated. Then, the data points are assigned to clusters associated with the medoid at the minimum distance from each data point. Here, the medoid is the most centrally located point in the cluster. Once we assign all the data points to the clusters, we calculate the sum of the distance of all the non-medoid data points to the medoid of each cluster. We term the sum of distances as the cost.

After calculating the cost, we select a temporary non-medoid point randomly from the dataset and swap a medoid with the selected point. Then we recalculate the cost of all the non-medoid data points to the medoid of each cluster considering the temporarily selected point as the medoid. If the newly calculated cost is less than the previous cost, we make the temporary point the permanent centroid. If the new cost is greater than the previous cost, we undo the changes. Then, we again select a non-medoid point and repeat the process until the cost is minimized.

The K-Medoids clustering is called a partitioning clustering algorithm. The most popular implementation of K-medoids clustering is the Partitioning around Medoids (PAM) clustering. In this article, we will discuss the PAM algorithm for K-medoids clustering with a numerical example.

K-Medoids Clustering Algorithm

Having an overview of K-Medoids clustering, let us discuss the algorithm for the same.

1. First, we select K random data points from the dataset and use them as medoids.
2. Now, we will calculate the distance of each data point from the medoids. You can use any of the Euclidean, Manhattan distance, or squared Euclidean distance as the distance measure.
3. Once we find the distance of each data point from the medoids, we will assign the data points to the clusters associated with each medoid. The data points are assigned to the medoids at the closest distance.
4. After determining the clusters, we will calculate the sum of the distance of all the non-medoid data points to the medoid of each cluster. Let the cost be C_i.
5. Now, we will select a random data point D_j from the dataset and swap it with a medoid M_i. Here, D_j becomes a temporary medoid. After swapping, we will calculate the distance of all the non-medoid data points to the current medoid of each cluster. Let this cost be C_j.
6. If $C_i>C_j$, the current medoids with D_j as one of the medoids are made permanent medoids. Otherwise, we undo the swap, and M_i is reinstated as the medoid.
7. Repeat 4 to 6 until no change occurs in the clusters.

K-Medoids Clustering Numerical Example

Now that we have discussed the algorithm, let us discuss a numerical example of k-medoids clustering.

The dataset for clustering is as follows.

Point	Coordinates

A1	(2, 6)
A2	(3, 8)
A3	(4, 7)
A4	(6, 2)
A5	(6, 4)
A6	(7, 3)
A7	(7,4)
A8	(8, 5)
A9	(7, 6)
A10	(3, 4)

Iteration 1

Suppose that we want to group the above dataset into two clusters. So, we will randomly choose two medoids.

Here, the choice of medoids is important for efficient execution. Hence, we have selected two points from the dataset that can be potential medoid for the final clusters. Following are two points from the dataset that we have selected as medoids.

- M1 = (3, 4)
- M2 = (7, 3)

Now, we will calculate the distance between each data point and the medoids using the Manhattan distance measure. The results have been tabulated as follows.

Point	Coordinates	Distance From M1 (3,4)	Distance from M2 (7,3)	Assigned Cluster
A1	(2, 6)	3	8	Cluster 1
A2	(3, 8)	4	9	Cluster 1
A3	(4, 7)	4	7	Cluster 1
A4	(6, 2)	5	2	Cluster 2
A5	(6, 4)	3	2	Cluster 2
A6	(7, 3)	5	0	Cluster 2

A7	(7,4)	4	1	Cluster 2
A8	(8, 5)	6	3	Cluster 2
A9	(7, 6)	6	3	Cluster 2
A10	(3, 4)	0	5	Cluster 1

The clusters made with medoids (3, 4) and (7, 3) are as follows.

- Points in cluster1= {(2, 6), (3, 8), (4, 7), (3, 4)}
- Points in cluster 2= {(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}

After assigning clusters, we will calculate the cost for each cluster and find their sum. The cost is nothing but the sum of distances of all the data points from the medoid of the cluster they belong to.

Hence, the cost for the current cluster will be 3+4+4+2+2+0+1+3+3+0=22.

Iteration 2

Now, we will select another non-medoid point (7, 4) and make it a temporary medoid for the second cluster. Hence,

- M1 = (3, 4)
- M2 = (7, 4)

Now, let us calculate the distance between all the data points and the current medoids.

Point	Coordinates	Distance From M1 (3,4)	Distance from M2 (7,4)	Assigned Cluster
A1	(2, 6)	3	7	Cluster 1
A2	(3, 8)	4	8	Cluster 1
A3	(4, 7)	4	6	Cluster 1
A4	(6, 2)	5	3	Cluster 2
A5	(6, 4)	3	1	Cluster 2
A6	(7, 3)	5	1	Cluster 2
A7	(7,4)	4	0	Cluster 2
A8	(8, 5)	6	2	Cluster 2
A9	(7, 6)	6	2	Cluster 2

A10	(3, 4)	0	4	Cluster 1

The data points haven't changed in the clusters after changing the medoids. Hence, clusters are:

- Points in cluster1:{(2, 6), (3, 8), (4, 7), (3, 4)}
- Points in cluster 2:{(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}

Now, let us again calculate the cost for each cluster and find their sum. The total cost this time will be 3+4+4+3+1+1+0+2+2+0=20.

Here, the current cost is less than the cost calculated in the previous iteration. Hence, we will make the swap permanent and make (7,4) the medoid for cluster 2. If the cost this time was greater than the previous cost i.e. 22, we would have to revert the change. New medoids after this iteration are (3, 4) and (7, 4) with no change in the clusters.

Iteration 3

Now, let us again change the medoid of cluster 2 to (6, 4). Hence, the new medoids for the clusters are M1=(3, 4) and M2= (6, 4).

Let us calculate the distance between the data points and the above medoids to find the new cluster. The results have been tabulated as follows.

Point	Coordinates	Distance From M1 (3,4)	Distance from M2 (6,4)	Assigned Cluster
A1	(2, 6)	3	6	Cluster 1
A2	(3, 8)	4	7	Cluster 1
A3	(4, 7)	4	5	Cluster 1
A4	(6, 2)	5	2	Cluster 2
A5	(6, 4)	3	0	Cluster 2
A6	(7, 3)	5	2	Cluster 2
A7	(7,4)	4	1	Cluster 2
A8	(8, 5)	6	3	Cluster 2
A9	(7, 6)	6	3	Cluster 2
A10	(3, 4)	0	3	Cluster 1

Again, the clusters haven't changed. Hence, clusters are:

- Points in cluster1:{(2, 6), (3, 8), (4, 7), (3, 4)}
- Points in cluster 2:{(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}

Now, let us again calculate the cost for each cluster and find their sum. The total cost this time will be 3+4+4+2+0+2+1+3+3+0=22.

The current cost is 22 which is greater than the cost in the previous iteration i.e. 20. Hence, we will revert the change and the point (7, 4) will again be made the medoid for cluster 2.

So, the clusters after this iteration will be cluster1 = {(2, 6), (3, 8), (4, 7), (3, 4)} and cluster 2= {(7,4), (6,2), (6, 4), (7,3), (8,5), (7,6)}. The medoids are (3,4) and (7,4).

We keep replacing the medoids with a non-medoid data point. The set of medoids for which the cost is the least, the medoids, and the associated clusters are made permanent. So, after all the iterations, you will get the final clusters and their medoids.

The K-Medoids clustering algorithm is a computation-intensive algorithm that requires many iterations. In each iteration, we need to calculate the distance between the medoids and the data points, assign clusters, and compute the cost. Hence, K-Medoids clustering is not well suited for large data sets.