Date - 19-Oct-2023

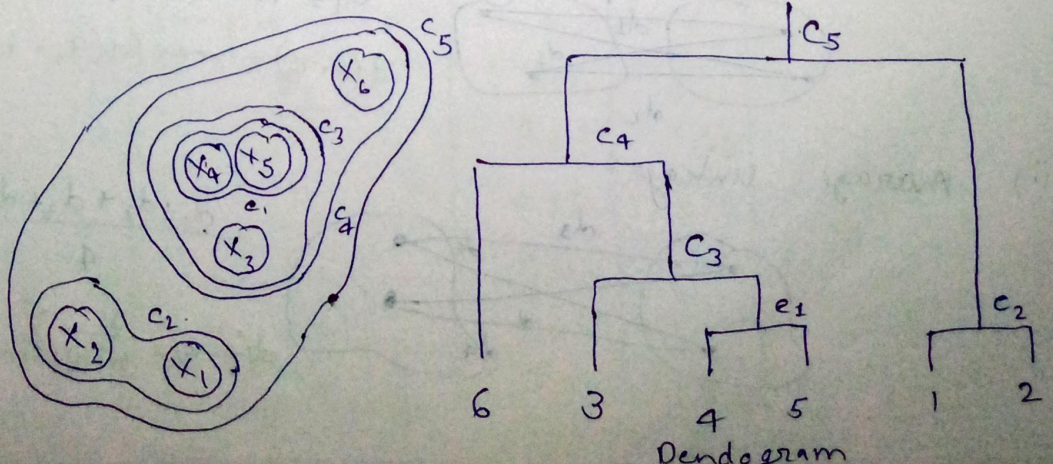
Agglomerative Hierarchical Clustering

Agglomerative clustering is a type of hierarchical clustering algorithm. it's an unsupervised machine learning technique that divides a population into clusters based on similarity. The clusters are created in a bottom up approach.

Agglomerative clustering works by:

- Treating each data point as a single cluster
- Merging the clusters based on similarity
- Repeating the process ~~until~~ untill all objects are in one big cluster

* Agglomerative clustering is good at identifying small clusters. it creates a tree-like structure that shows the relationships between clusters and their hierarchy.
* The time complexity of a naive agglomerative clustering is $O(n^3)$. This can be reduced to $O(n^2 \log n)$ using a priority queue data structure.



Dendogram

How algorithm works:

Step-1. Initialize the proximity matrix

Step-2. Make each point a cluster

Step-3. Inside a loop

(a) Merge the two closest clusters

(b) Update the proximity Matrix

Step 4. Until only one cluster is left.

Types of Agglomerative Clustering

(i) Min (single linkage)

(ii) Max (complete linkage)

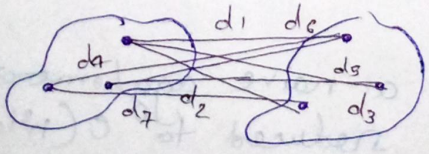
(iii) Average linkage

(iv) Ward's linkage

(v) centroid linkage.

In the above the types of agglomerative clustering is based on how we measure the distance between the two cluster.

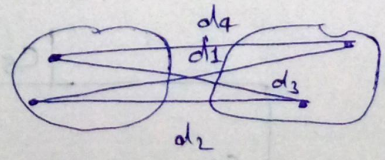
(i) single linkage agglomerative cluster.



d_1 = min (single linkage)

$d(A,B) = \min(d(a_i, b_j))$

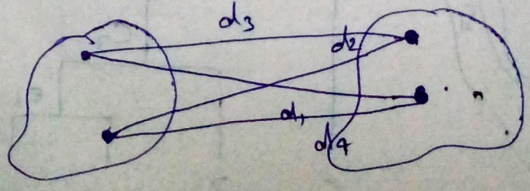
(ii) Max (complete linkage):



d_1 = max (complete linkage)

$d(A,B) = \max(d(a_i, b_j))$

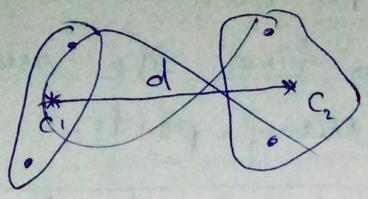
(iii) Average linkage:

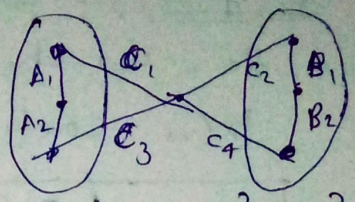


$$\frac{d_1 + d_2 + d_3 + d_4}{4} = d$$

$$d(A,B) = \frac{1}{|A||B|} \sum_{i \in A} \sum_{j \in B} d(i,j)$$

(iv) ward's ~~Centroid~~ linkage (default in sklearn)



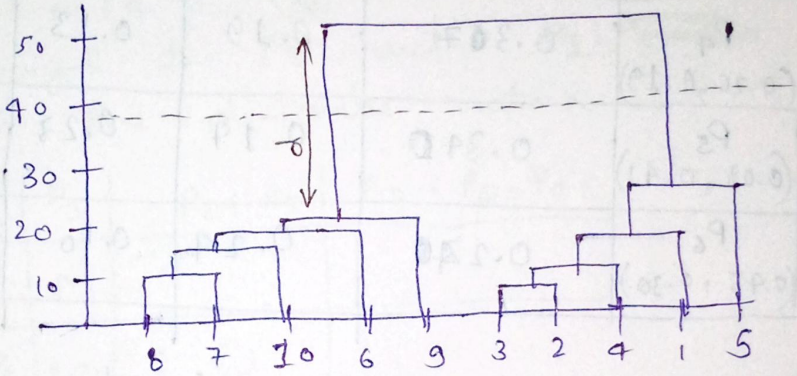


distance = $C_1^2 + C_2^2 + C_3^2 + C_4^2 - A_1^2 - A_2^2 - B_1^2 - B_2^2$

to minimize the variance.

How to find the ideal number of clusters:

for finding the ideal number of clusters in Agglomerative clustering we use dendogram of that cluster.



in the above dengram we make two cluster.

d = inter cluster similarity

Question. For a given dataset find the clusters using a single link technique. Use euclidean distance and draw the Dendogram.

Sample No.	X	Y
P_1	0.40	0.53
P_2	0.22	0.38
P_3	0.35	0.32
P_4	0.26	0.19
P_5	0.08	0.41
P_6	0.45	0.30

Step 1. Compute the distance matrix.

So, we have to find the euclidean distance b/w each and every points.

	P_1 (0.40, 0.53)	P_2 (0.22, 0.38)	P_3 (0.35, 0.32)	P_4 (0.26, 0.19)	P_5 (0.08, 0.41)	P_6 (0.45, 0.30)
P_1 (0.40, 0.53)	0	0.23	0.22	0.37	0.34	0.24
P_2 (0.22, 0.38)	0.23	0	0.14			
P_3 (0.35, 0.32)	0.22	0.14	0			
P_4 (0.26, 0.19)	0.37	0.19	0.13	0		
P_5 (0.08, 0.41)	0.34	0.14	0.28	0.23	0	
P_6 (0.45, 0.30)	0.24	0.24	0.10	0.22	0.39	0

Step 2 Merging the two closest members.

- Here the minimum value is 0.10 and hence we combine P_3 and P_6 (as 0.10 came in the P_6 raw and P_3 column)
- Now, form clusters of elements corresponding to the minimum value and update the distance matrix

Now we will update the distance matrix

min = 0.10

	P_1	P_2	P_3, P_6	P_4	P_5
P_1	0				
P_2	0.23	0			
P_3, P_6	0.22	0.14	0		
P_4	0.37	0.19	0.13	0	
P_5	0.34	0.14	0.28	0.23	0

(P_3, P_6)

minimum = 0.13

Now we will update the distance matrix P_4 is merge with (P_3, P_6)

	P_1	P_2	P_3, P_6, P_4	P_5
P_1	0			
P_2	0.23	0		
P_3, P_6, P_4	0.22	0.14	0	
P_5	0.34	0.14	0.28	0

$\{(P_3, P_6), P_4\}$

minimum = 0.14 in P_5

Now merge the P_5 with P_2.

update the distance matrix

	P_1	P_2, P_5	P_3, P_6, P_4
P_1	0		
P_2, P_5	0.23	0	
P_3, P_6, P_4	0.22	0.14	0

$\{(P_3, P_6), P_4\}$ and (P_2, P_5)

Here minimum = 0.19

Hence merge P_3, P_6, P_4 with P_2, P_5

Now update the distance matrix

	P_1	P_2, P_5, P_3, P_6, P_4
P_1	0	
P_2, P_3, P_5, P_6, P_4	0.22	0

$[\{(P_3, P_6), P_4\}, (P_2, P_5)]$

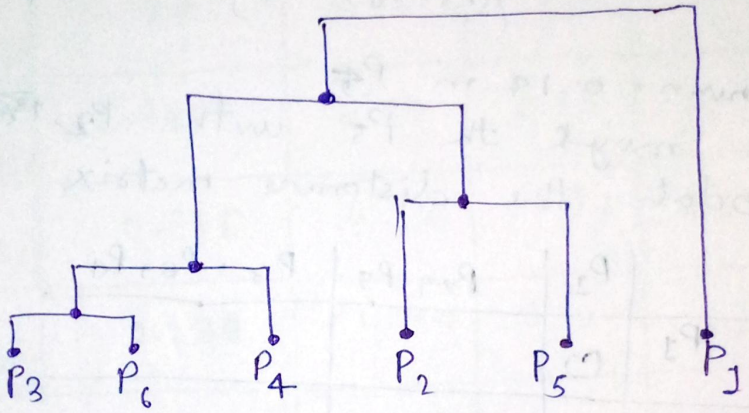
Now finally. merge P_3, P_6, P_4, P_2, P_5 with P_1

then. $[\{(P_3, P_6), P_4\}, (P_2, P_5)], P_1$

Step:-3 draw the dendogram.

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

$$[\{(P_3, P_6), P_4\}, (P_2, P_5)], P_1$$



→ computationally expensive for agglomerative clustering:- $O(N^2)$

→ computationally expensive for Divisive clustering is $O(2^N)$