Clustering

→ K-mean Alg. aims at minimizing an objective function known as squared error function

$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} \left( \| x_i - v_j \| \right)^2$$

where $\| x_i - v_j \|$ = Euclidean distance b/w $x_i$ & $v_j$

$c_i$ = # data points in $i^{th}$ cluster

$c$ = # cluster centers.

Algorithm :-

Let $X = \{x_1, x_2, \ldots, x_n\}$ be data points & $V = \{v_1, v_2, \ldots, v_c\}$ be set of centers

① Randomly select 'c' cluster centers. (as far away as possible in cunning way)

② Calculate the distance b/w each data points & cluster center

③ Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.

④ Recalculate the new cluster center using:

$$v_i = \frac{1}{c_i} \sum_{j=1}^{c_i} x_j$$

[Take mean of all data points assigned to centroid cluster]

⑤ Recalculate the distance b/w each data point & new obtained cluster centers.

⑥ If no data point was reassigned then stop, otherwise repeat from step-③.

⑦ The results of k-mean clustering algorithm are:

① The centroids of k-cluster, which can be used to label new data

② Labels for training data (each data point is assigned to a single cluster).

Scanned by CamScanner

→ Clustering is task of dividing the data points into # groups / # clusters that data points in the same group are more similar to other data points in the same group than those in other group.

→ Application of clustering :-

- Recommendation engine
- Market segmentation
- Social N/w Analysis
- Search result grouping
- Medical Imaging
- Image segmentation
- Anomaly detection

Adv of K-mean :

① Fast, robust, easier to understand

② T.C. $O(tknd)$

$t$ = # iterations, $k$ = # clusters

$n$ = # objects, $d$ = # dimension each obj

$k < t, d < n$

③ Gives best result when data set are distinct (or) well separated to each other.

Dis of K-mean :

Scanned by CamScanner

# Hierarchical clustering :- 2 types

i) Agglomerative HCA (Bottom-up Approach)

ii) Divisive HCA (top-down approach) → Reverse of Bottom-up approach

Algo :-

(i) Given set of N items to be clustered, NXN distance (similarity) matrix.

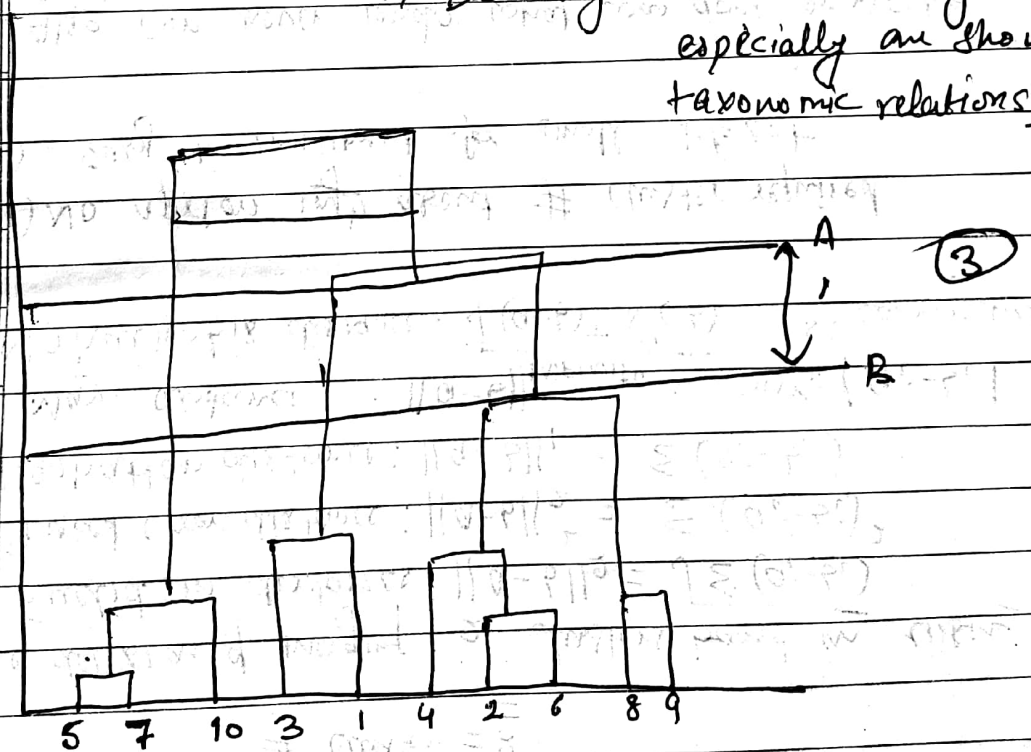
(i) Start by assigning each item to its own cluster, so if we have N items, means we have N clusters, each containing just one item. Let distance b/w the clusters is equal distance (similarity) b/w items they contain.

(ii) Find the closest (most similar) clusters and merge them into one cluster. So now we have one less cluster.

(iii) Compute distance (similarities) b/w new cluster & each of old clusters.

(iv) Repeat step-(ii) & (iii) until all items are clustered into single cluster of size N.

→ Dendrogram → Tree diagram, especially one showing taxonomic relationship.




→ The height of dendogram at which 2 clusters are merged represent distance b/w 2 clusters.

Scanned by CamScanner

→ The best choice of # clusters is # vertical lines in the dendrogram cut by a horizontal line that can traverse the max. distance vertically without intersecting a cluster.

∴ # cluster = 3

→ The decision of merging 2 clusters based on either

i) Euclidean distance: $\|a-b\|_2 = \sqrt{\Sigma (a_i-b_i)}$

ii) Squared error distance: $\|a-b\|_2^2 = \Sigma (a_i-b_i)^2$

iii) Manhattan distance: $\|a-b\|_1 = \Sigma (a_i-b_i)$

iv) Max. distance: $\|a-b\|_{INFINITY} = \max_i |a_i-b_i|$

v) Mahalanobis distance: $\sqrt{(a-b)^T S^{-1} (-b)}$ S: Convergence matrix

Adv:- i) No apriori info about # cluster required

ii) Easy to implement for small dataset

Dis :-

i) Algo can never undo what was done previously

ii) T.C = $O(n^2 \log n)$, n = # data point

iii) Sometimes difficult to identify correct # clusters

iv) HCA can't handle big data but k-mean can

HCA → $O(n^2)$ ; k-mean → $O(n)$

Scanned by CamScanner