Unsupervised learning

→ Unsupervised algos make inferences from datasets using input vectors without referring to known outcomes.

A cluster refers to a collection of data points aggregated together because of certain similarities.

K-Means

K- refers to number of centroids, we need in DS

A centroid is imaginary or real center of cluster

Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.

means → refer to averaging data → finding centroid

K-Means

Assign and optimize

Optimize iteratively unless centroids have stabilized or defined iterations achieved.

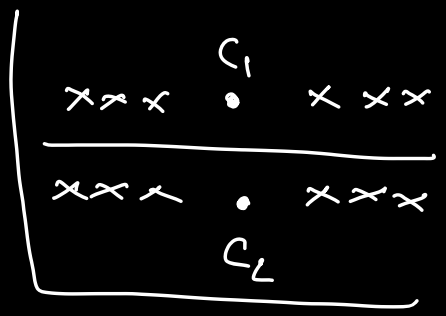
Initial position of clusters is important else may not converge for some datasets.

Hyper parameters → n_clusters, max_iter

Bad local Minima

Points above line $\in C_1$

Points below line $\in C_2$



Scikit-learn KMeans

n_clusters = 8

max_iter = 300

n_init = 10 → No. of times algo will run with different centroid seeds find output is best output

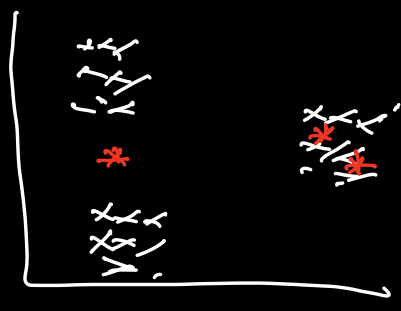
Limitations

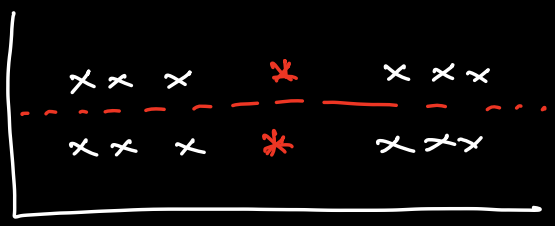
For a fixed dataset and fixed k, K-means can give different results since initial position vary

K-means is Hill climbing algo and it depens where initial cluster centers are marked.

Local Minima ⇒ actually 2 clusters

Re-run may give better solution



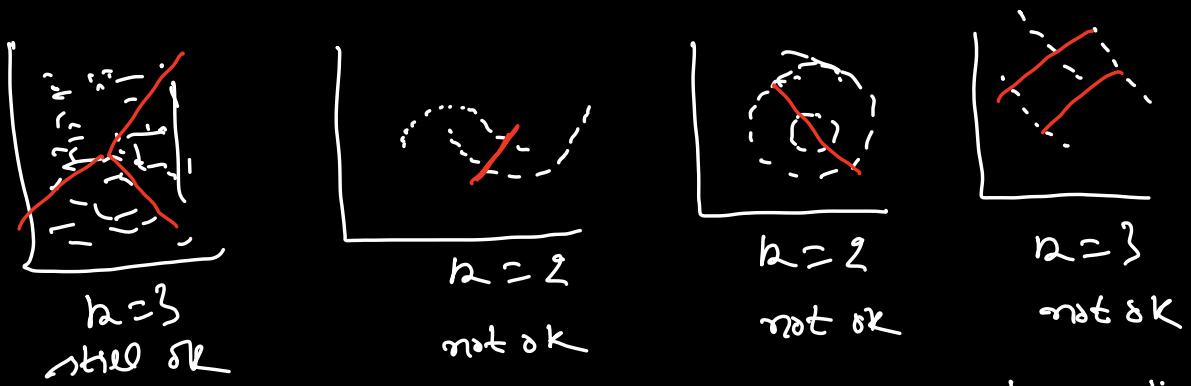


Local Minima

k Means

good if spherical and if we know how many clusters

→ depend on distance from centroids



Hierarchical Clustering

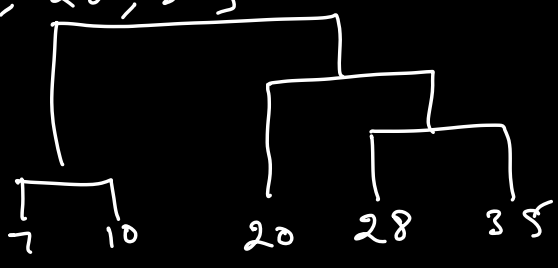
eg. pizza location

Assume each is cluster and combine

We assign each object to a separate cluster, then compute the distance between each of the cluster and join the two most similar clusters

$X = \{7, 10, 20, 28, 35\}$

Dendogram



Single Linkage

Merge two clusters whose two closest members have the smallest distance

S1 7 10 20 28 35

3 10 8 7

S2 (7,10) 20 28 35

10 8 7

S3 (7,10) 20 (28,35)

10 8

S4 (7,10) (20,28,35)

10

S5 (7,10,20,28,35)

Break the last one step to get two clusters

" " " two steps " " three "

Complete Linkage

Merge members of clusters which provide the smallest maximum pairwise distance

a.k.a Farthest Neighboring Clustering

S1 7 10 20 28 35

3 10 8 7

S2 (7, 10) 20 28 35

13 8 7

S3 (7, 10) 20 (28, 35)

13 15

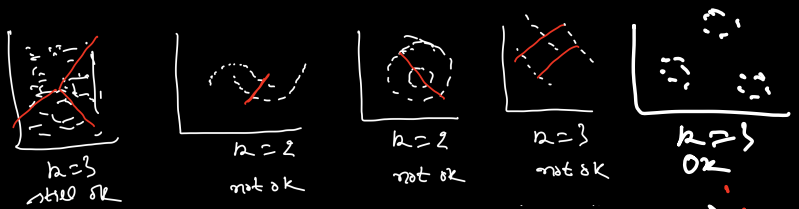
S4 (7, 10, 20) (28, 35)

28

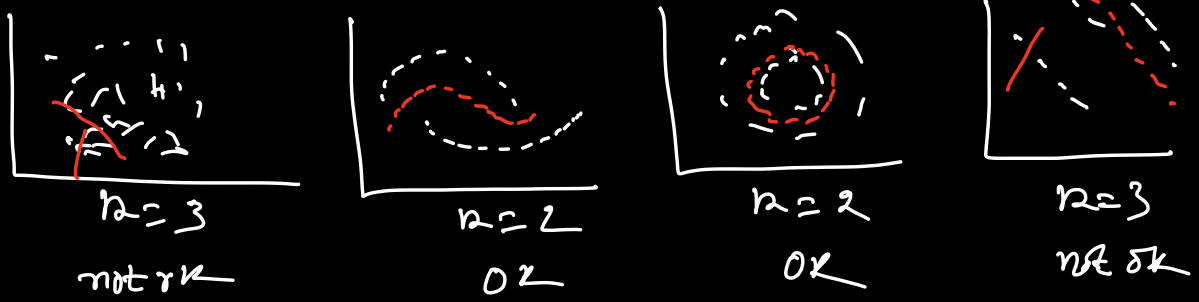
S5 (7, 10, 20, 28, 35) ⇒ compact clusters

Comparison

k-mean →



Single Link



Dendograms clearly shows how many clusters and distance between two.

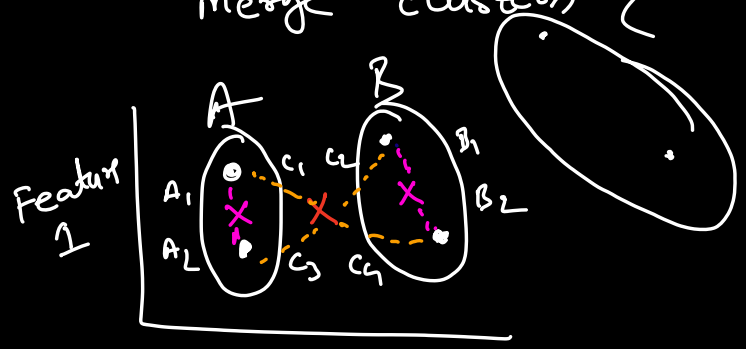
Dendograms is 2-D and show many dims clearly

Average Link

considers avg distance

Ward's Method

merge clusters C



$$\Delta(A,B) = C_1^2 + C_2^2 + C_3^2 + C_4^2 - A_1^2 - A_2^2 - B_1^2 - B_2^2$$

Find center of all points and determine distance of all points from center → square and add

$\Rightarrow C_1^2 + C_2^2 + C_3^2 + C_4^2$

Subtract Variance

Find center of each cluster and distance of each point from its cluster center

$\Rightarrow$ square and subtract

Ully find $\Delta(B,C)$ $\Delta(A,C)$

Merge cluster that have less distance

Sklearn

```
clust = cluster.AgglomerativeClustering(n_cluster=3,
                                        linkage='ward')
labels = clust.fit_predict(X)
```

Disadvantage

sensitive to noise, outliers

Computationally intensive $O(N^2)$