Clustering HandWritten Notes

- Darshan R M

"Learn, Share, and Collaborate"

Unsupervised learning

Week - 19

clustering

↓

clustering & grouping your data similar clusters

* Supervised learning ⇒ we have both i/p and o/p [independent] [dependent]

  ↓ solve

  - classification.
  - Regression.

↓

* Consider,

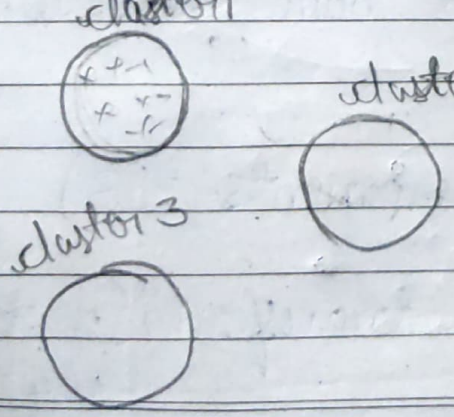
i/p			
Age	Experience	salary	No o/p feature
—	—	—	
—	—	—	
—	—	—	
—	—	—	

⇓

Unsupervised ML

⇓

group / cluster this data



Ex:

Customer segmentation

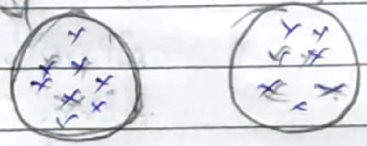
Product : data of customer bought product.

New product

Salary	Spending-Score

i/p

From existing data i may get know who may buy the product ⇒ we can give discounts to those customer and "increase" the rate of buying.



→ we give 20% them more people buy.

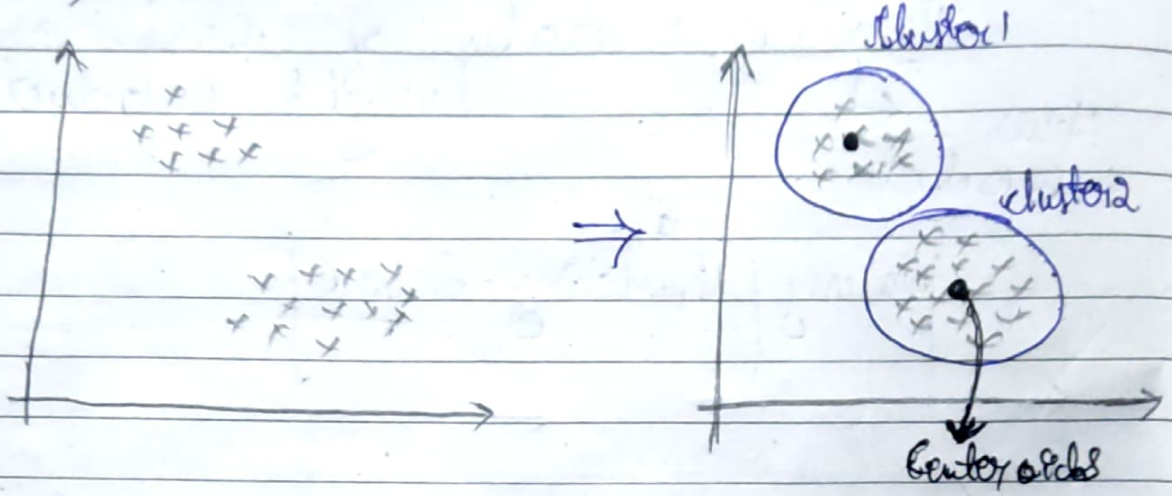
* Unsupervised ML Algorithm

Hierarchal
① K Means algorithm.
② ~~Hierarchical~~ clustering.
③ DBSCAN clustering.
④ Silhoutte scoring.

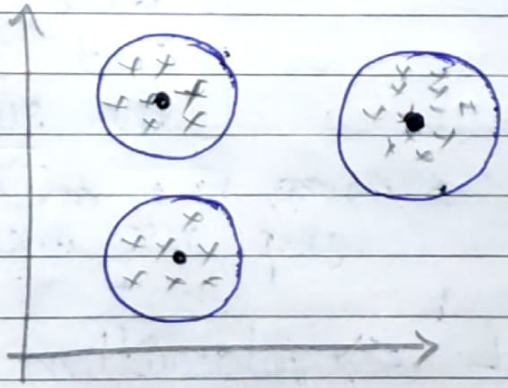
} Validate

* K-Means clustering algorithm

* consider,

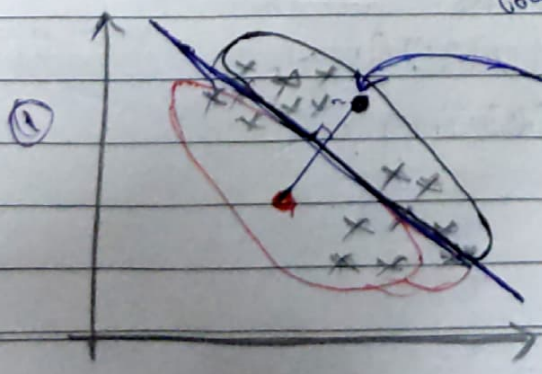


* of 3 clusters



→ we cluster similar kind of data. and each cluster have centroid.

*



Consider $K = 2$ → later we see how to select K value

calculate distance

K means

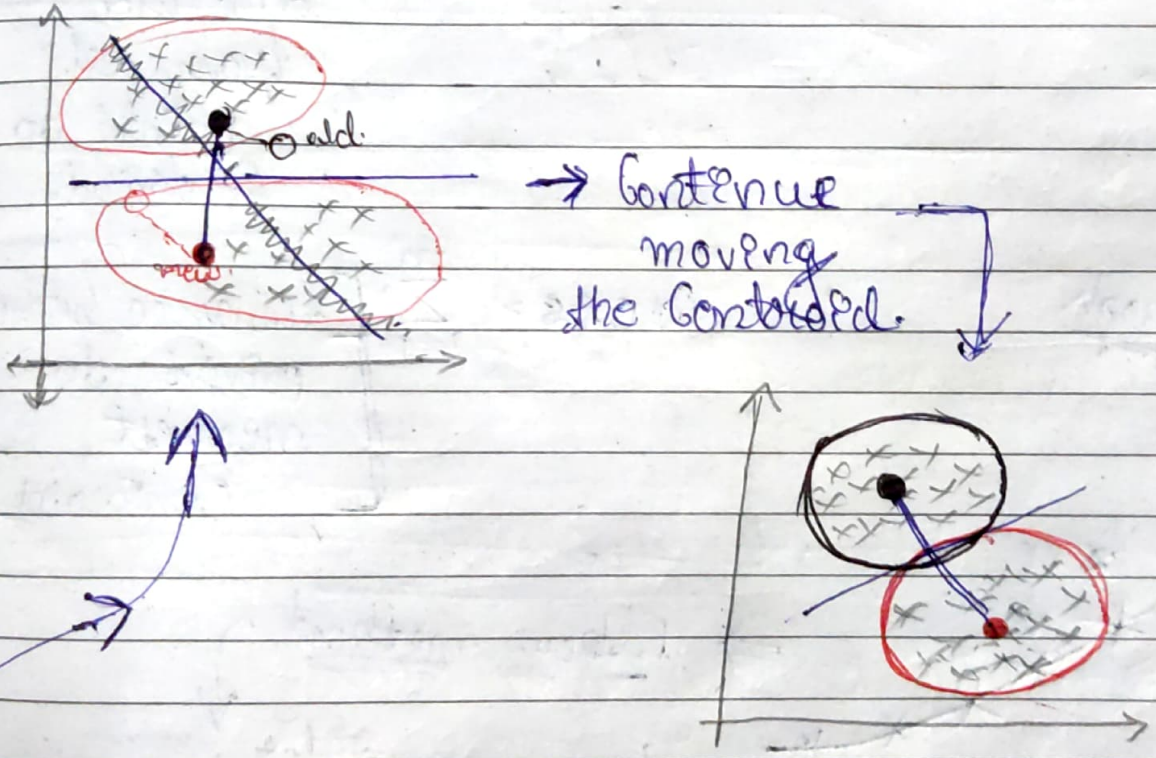
Steps

① Initialize some K → Centroids

repeat

② Find out points that are nearest to the centroids. → group them.

③ Move the centroids ⇒ average distance

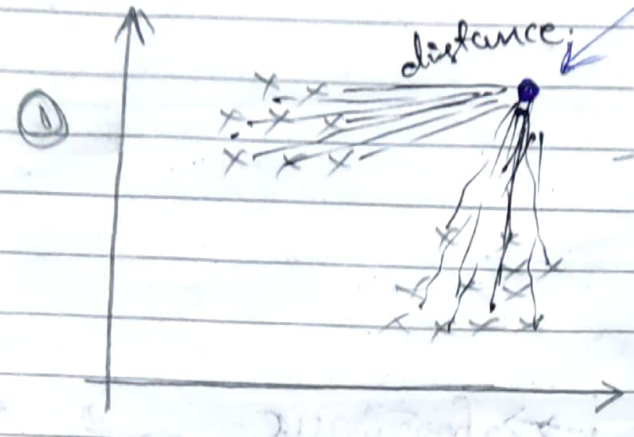


Q) How do we select K values?

⇒

WCSS = within cluster sum of squares

Initial: K = 1 to 20



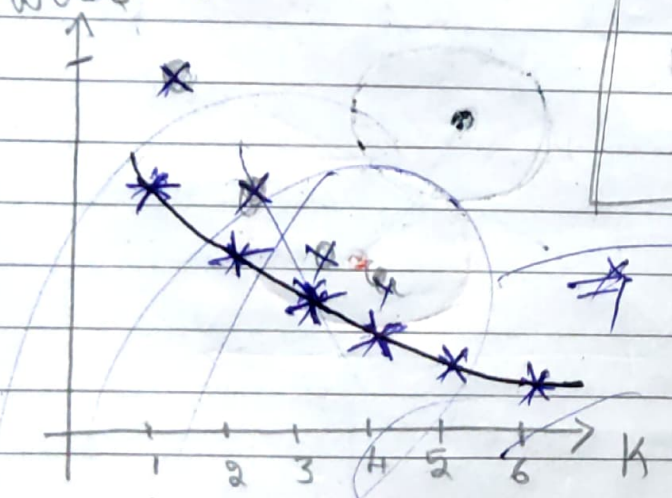
random point initialized

K=1, WCSS more

→ Compute distance from all points to Centroid.

$$WCSS = \sum_{i=1}^{n} [\text{distance between points to nearest Centroid}]^2$$

graph



↓

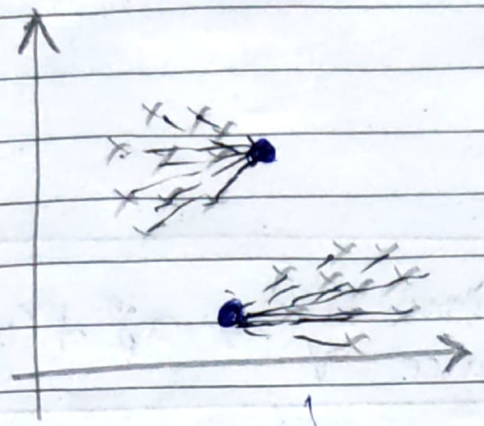
It will be high.

⇒ Elbow method ↓

we have to find a point where there is abrupt decrease in 'WCSS'.

we have to select that point.

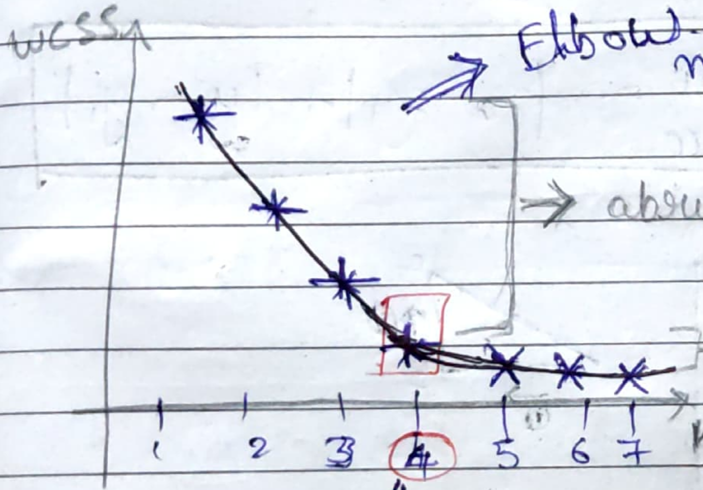
(2)



K=2, → wcss is less

goes one

~~so~~



→ Elbow method.

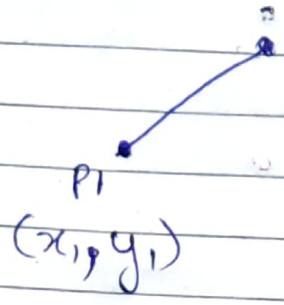
→ abrupt decrease in WCSS

→ not much [Stabalizing] change in WCSS

⇓

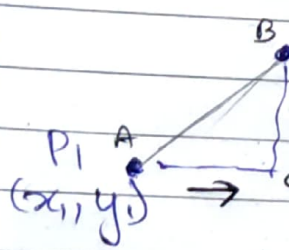
select this K value

* Euclidean distance



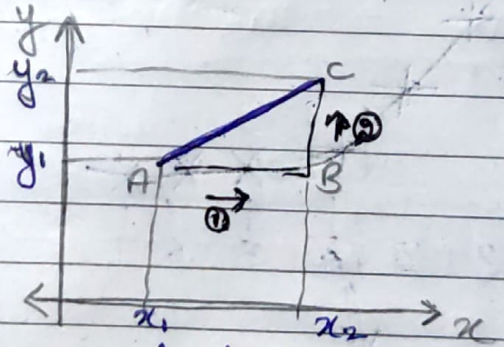
$$\text{Euclidean distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

* Manhattan distance



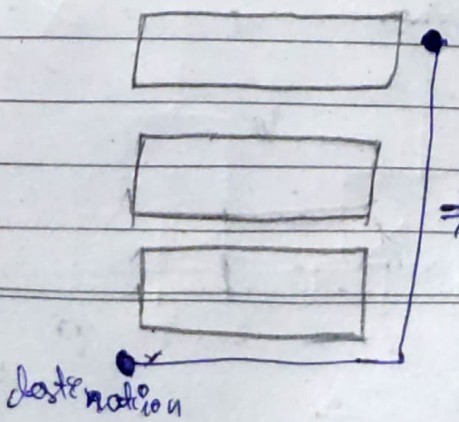
$$\text{Manhattan distance} = |x_2 - x_1| + |y_2 - y_1|$$

we have to take AC & CB



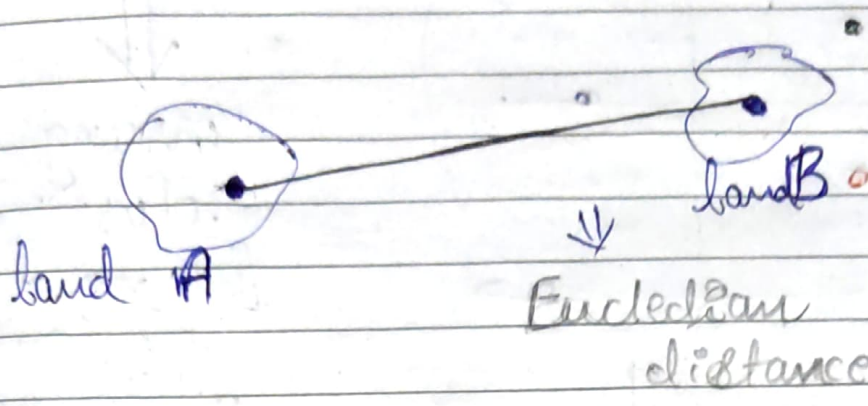
Q/ when to use ~~used~~ what?

⇒ (*) In U.S city have building



⇒ Manhattan distance

* Air traffic control:



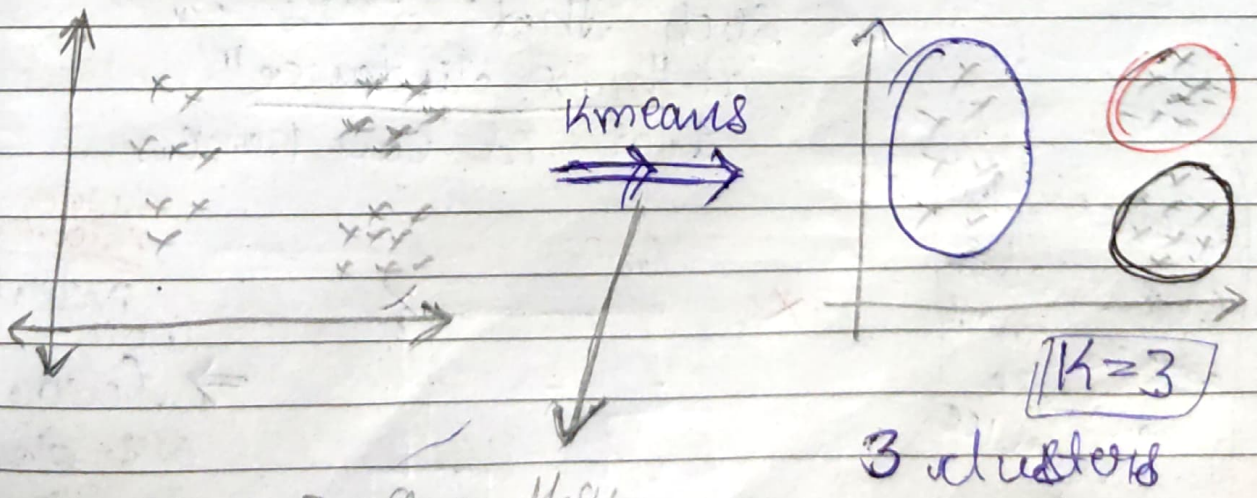
---

* what happens when 2 points are initialized near?

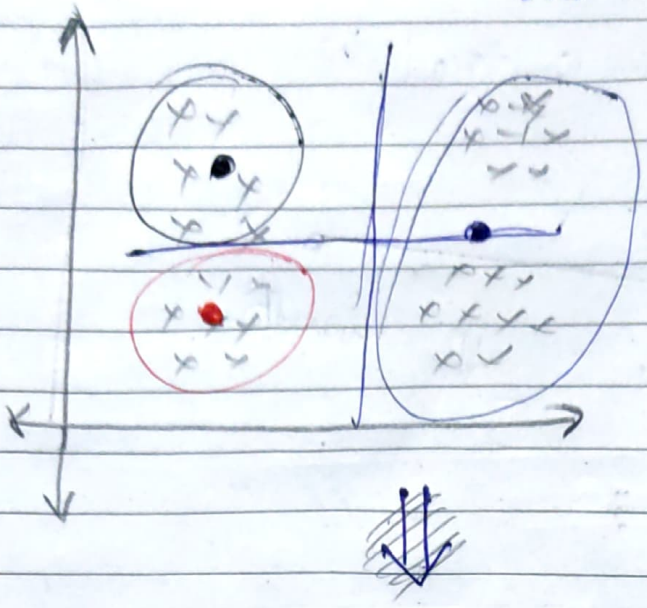
---

* Random Initialization Trap [K means ++]

* Consider, data points



During this when we do random initialization what if initialization goes "wrong."



randomly initialized centre

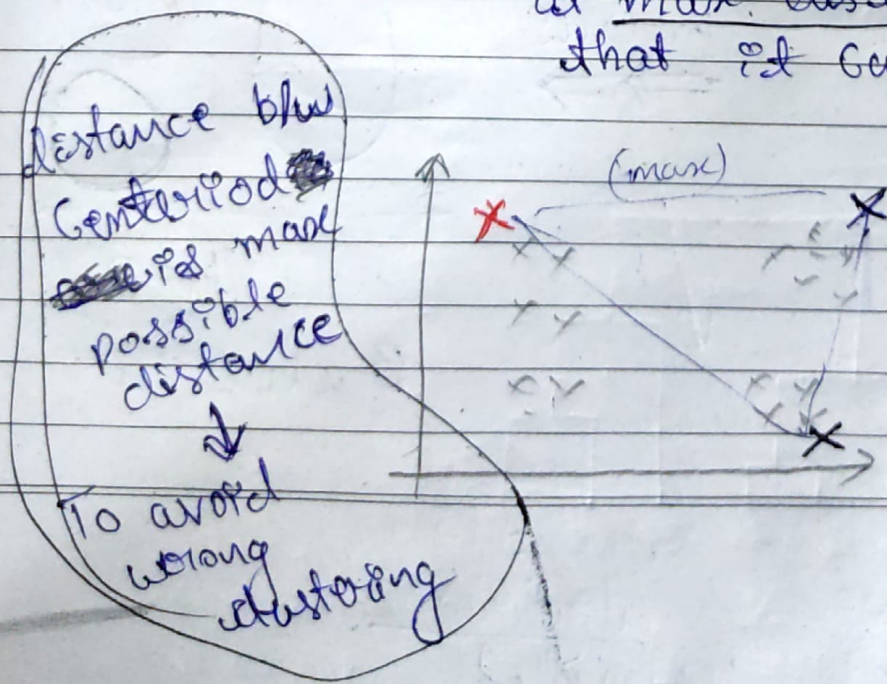
↓

wrong clustering

[We need cluster which we have behind]

So, to avoid wrong intialization.
we use "K means ++" intialization Technique.

⇒ Here, we intailize all the "Centroid" such that "it is at "max distance" that it can have.

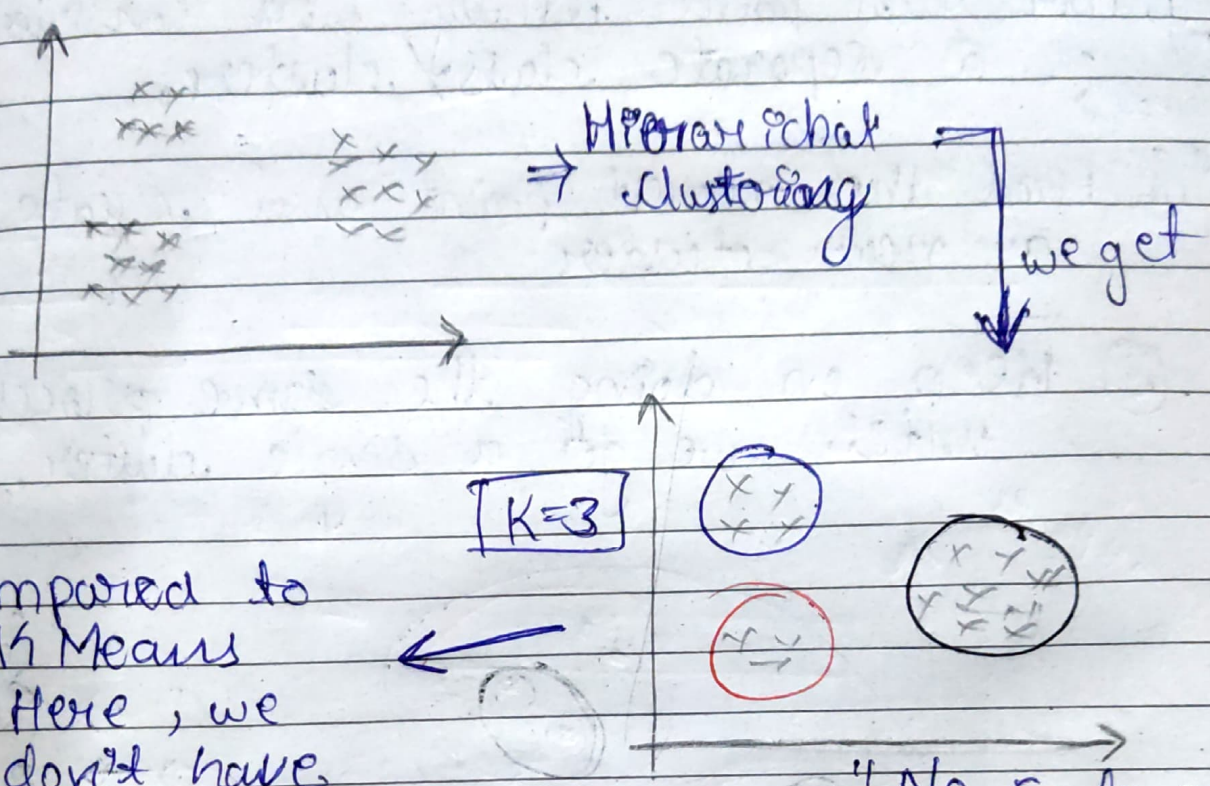


answer interview

Initially Here,

⇒ Centroid are on maximum distance possible between each other.

* Hierarchical clustering



*2 Types of Hierarchical clustering.

Meaning

① Agglomerative. ⇒ Combining

② Divisive. ⇒ dividing

reverse

Enough: to understand one type

P.T.O.

* Agglomerative :

Steps :

① For each point Initially we will consider as a separate class / cluster.

repeat

② Find the nearest point and create a new cluster

③ Keep on doing the same process until we get a single cluster.

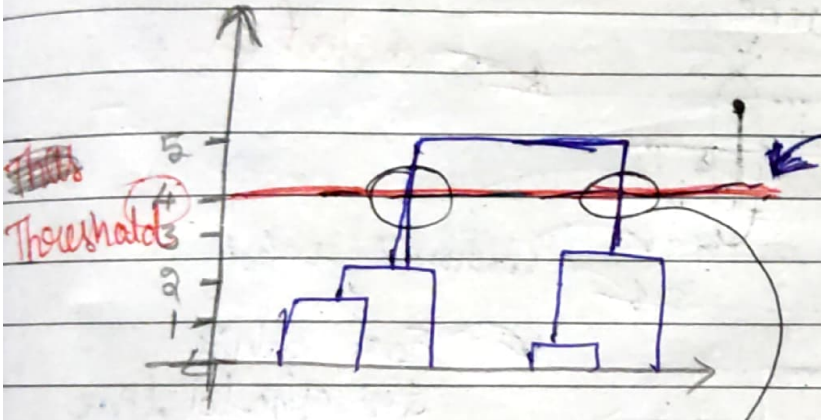
* In dendogram,

↑ bottom to top approach ⇒ Agglomerative

↓ top to bottom approach ⇒ Divisive.

* How do we find K Values?

→ found based on "threshold of euclidean distance."



create a line tells that euclidean distance b/w all the points/cluster should not be more than 4.

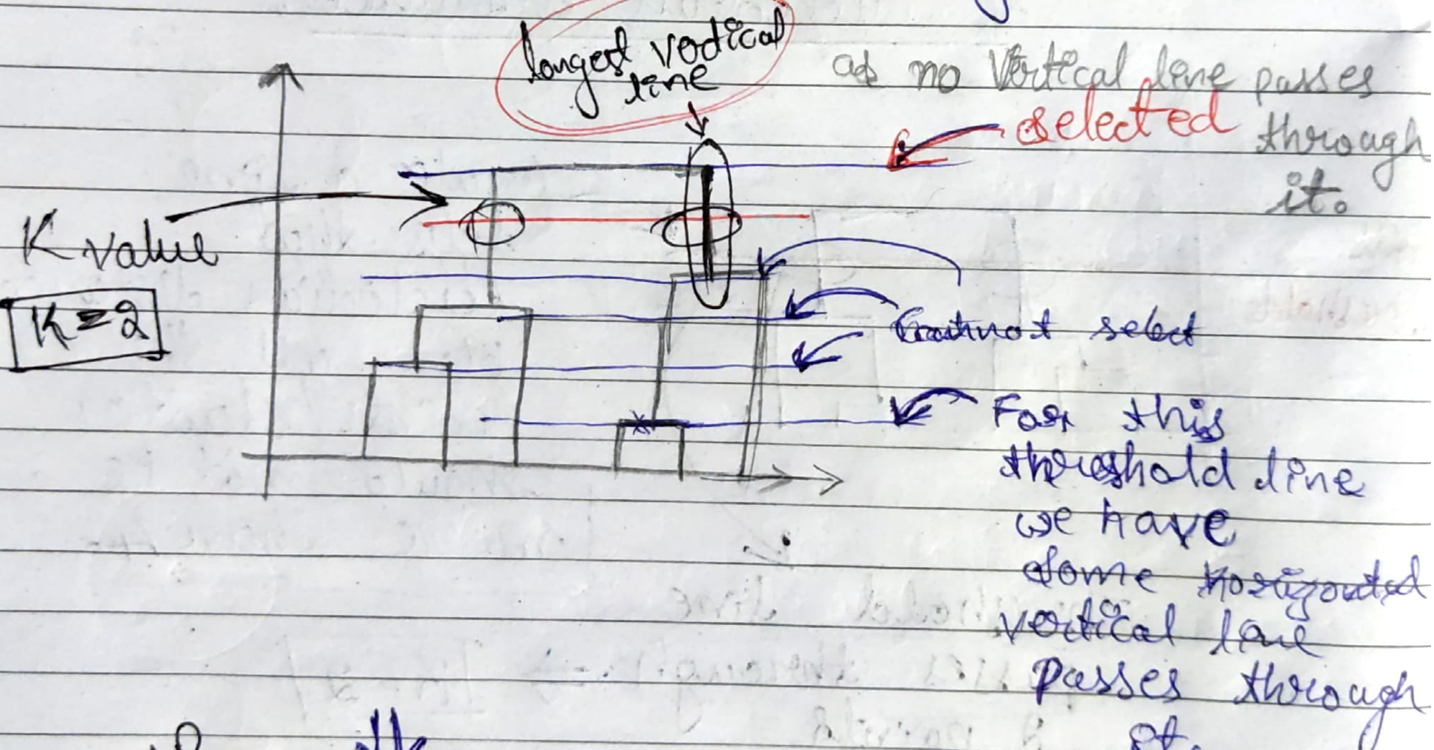
Threshold line passes through 2 points ⇒ K=2

∴ If we "decrease threshold" euclidean distance ⟶ then "No of clusters increases"

Threshold ∝ 1 / No of clusters

* But How do we find threshold value ?
↓
we the help of dendogram

⇒ we have to find the longest Horizontal ~~vertical~~ line so that none of the ~~Horizontal~~ vertical line passes through it.



⇓

imp
(*) Select the longest ~~horizontal~~ vertical line such that no ~~vertical line~~ horizontal line passes through it.
decoped

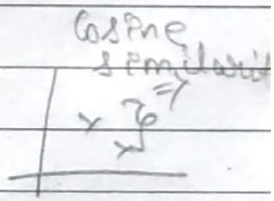
* Hierarichal v/s K Means

Based on Scalability and Flexibility

① Dataset size ⇒ Huge → K Means
Small → Hierarichal
as we have to Construct 'Dendogram'.

② K means → Numerical data

Hierarichal clustering → [Cosine similarity]
~~Variety of data~~



③ Centroids → Elobow method
↓
difficult to find No of centroid

Hierarichal → Can find easily.

* DBSCAN

- → Core point
- → border point
- → outliers.

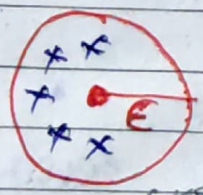
} able to handle non linear data.

Consider, min pts = 4 , ε = radius

select value using Hyper parameter tuning & also by Silhouette score

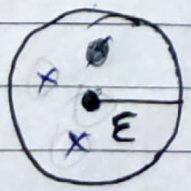
---

* Core point



⊛ No of points within the ε should be ≥ min points

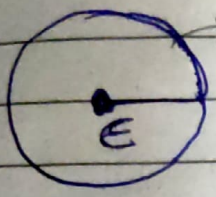
* Border point



⊛ No of points within ε will be less than min point. (≤ min point)

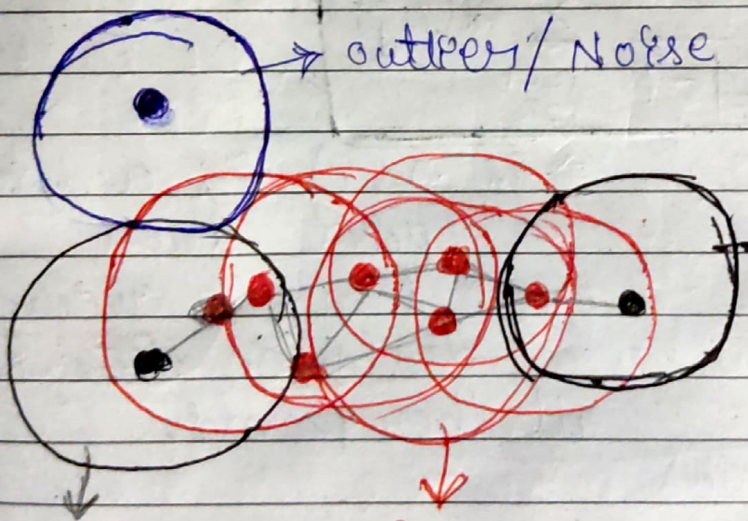
* Outliers / Noise

* No of points within ϵ is 1.



---

* In DBSCAN based on density in different regions we can group our core point, Border point and outliers.



⇓

Outliers / Noise will be removed / handled.

⇒ can handle (non linear data) + (noise)

* Silhouette clustering

How do we validate { K means
K = 4 Elbow method

Steps

(1)

$$a(i) = \frac{1}{|C_I| - 1} \sum_{\substack{j \in C_I \\ i \neq j}} d(i,j)$$

[except from a selected point]

C1

C1 cluster

⇒ if a(i) is a data point then

⇒ we compute average distance with all other points.

(2)

(nearest cluster)

C2 ← C1

average

C3

⇒ we define mean dissimilarity to point i to some cluster C_j

nearest cluster

⇒ we calculate the average distance to the points of cluster C_j

nearest cluster

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_j} d(i, j)$$

$|C_J|$ ⇒ number of points in nearest cluster.

W.K.T $a(i) \ll b(i)$ ⇒ if clustering is done well

$a(i)$: "average distance" with all other points within "cluster."

$b(i)$: "average distance" with the points of the "nearest cluster."

---

③ Calculate Silhouette score.

$$s(i) = \frac{b(i) - a(i)}{\max[a(i), b(i)]}$$

Value range ⇒ [-1 to 1]

⇒ More near to 1 the better clustering model we have created.

(or)

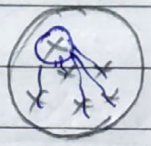
$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1, & \text{if } a(i) > b(i) \end{cases}$$

$\approx +1$; ≈ -1

Again revise 3 steps

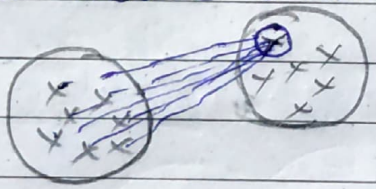
Revise

① Calculate average distance within cluster from a point.



$a(i)$

② Calculate average distance of all points of nearest cluster from same above choosed points in $a(i)$.



$b(i)$

③ Calculate silhoutte scoring. and withdraw output.

Value near to +1 ⇒ better the clustering

① $$a(i) = \frac{1}{|C_I| - 1} \sum_{\substack{j \in C_I \\ i \neq j}} d(i, j)$$

② $$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i, j)$$

③ $$s(i) = \frac{b(i) - a(i)}{\max[b(i), a(i)]}$$

* K Means Implementation → only in dependent feature

① create dataset

> from sklearn.datasets import make_blobs

binary large objects

generates isotropic gaussian blobs for clustering

2 features

> x, y = make_blobs(n_samples = 1000, centers = 3, n_features = 2)

Can give array also

as unsupervised ML we won't use y

[No of centeroid/ cluster]

x → [[[—, —], [—, —], —]

* Scatter plot

plt.scatter(x[:, 0], x[:, 1])

② Train test split.

③ clustering

> from sklearn.cluster import KMeans

[Manhattan max distance] To avoid wrong clustering

Parameters

n_clusters, init = 'k-means++',

No of Centroid

* Elbow method → Find K value

↓

wcss ↓

```
> wcss = []
for K in range(1,10):
    kmeans = KMeans(n_clusters = K, init=
    kmeans.fit(x_train)
    wcss.append(kmeans.inertia_)
```

wcss

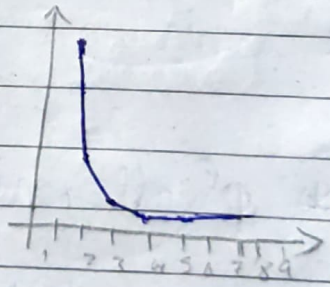
Initialize the centroid to our data

* plot elbow curve

↓

```
> plt.plot(range(1,11), wcss)
> plt.xticks(range(1,11))
> plt.show()
```

x-axis label



④ Model

```
> kmeans = KMeans(n_clusters=3,
                  init='k-means++')
> y = kmeans.fit_predict(x_train)
```

* To locate knee value for large data ↓

knee locator

'k-means++') > !pip install kneed

> from ~~kneed~~ kneed import KneeLocator

> kl = KneeLocator(range(1,10), wcss, curve='convex', direction='decreasing') Convex/Concave

> kl.elbow

(5) Silhouette score

> from sklearn.metrics import silhouette_score.

loop

> for i in

> score = silhouette_score(X_train, kmeans.labels_)

Implementation

* Hierarchical - Agglomerative clustering

① Import data

② Standardization

> from sklearn.preprocessing import StandardScaler

③ To visualize → PCA
we do

> from sklearn.decomposition import PCA

> pca_scaled = pca.fit_transform(x_scaled)

④ clustering - Agglomerative..

⑤ * Create dendogram.

> import scipy.cluster.hierarchy as sc

> sc.dendogram(sc.linkage(pca_scaled, method='ward'))

fixed

⑤ Agglomerative clustering

> from sklearn.clustering import AgglomerativeClustering

> cluster = AgglomerativeClustering(n_clusters = 2, affinity = 'euclidean', linkage = 'ward')

> cluster.fit(pca_scaled)

cluster -> .cluster.labels_

plt.scatter(pca_scaled[:,0], ——— 1],
c = cluster.labels_)

⑥ Silhouette score.

mer

* Implentation → DBSCAN

① Create dataset.
Non linear dataset
> from sklearn.datasets import
make_moons

* Make two interleaving half circles.

② > x, y = make_moons(n_samples = 250,
noise = 0.12)

* plot
> plt.scatter(x[:, 0], x[:, 1])

③ Standardization.

④ Clustering

> dbscan = DBSCAN(eps = 0.3)

> dbscan.fit(x_scaled)

> dbscan.labels_

* plot
(—, c = dbscan.labels_)