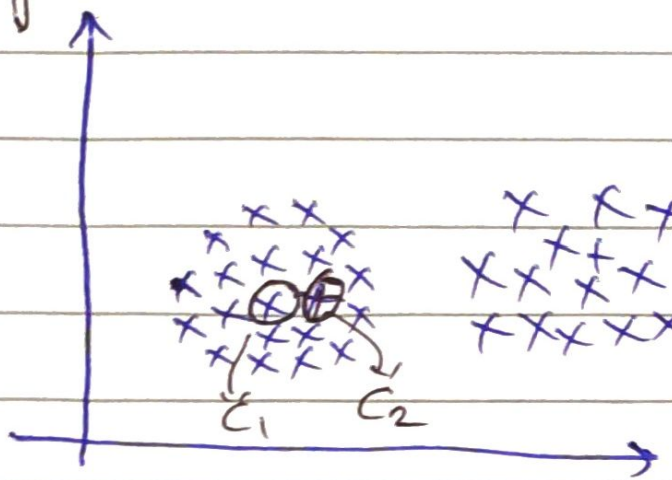
Q) What if all the selected Centroids (at initialization - randomly picking the centroids at start) are next to each other



⇒ For large datasets the probability of getting nearest (~~best~~ (centroids next to each other) is Very Very less ≈ 0

→ But for small datasets & then probability of getting Centroids next to each other & is won't ~~be~~ 0.

↳ Chances are there for small datasets

→ In real time, we have millions of data then it won't have problem (little bit chances are there)

Q) So, is there any better initialization of centroids (i.e step-1)

A) (In order to / It will solve the issue of Getting centroids next to each other)

→ Yes, there is a better initialization

⇓⇓

K-Means++ initialization.

In K-means.

Step-1: Random Intialization

↳ Decide the value of K & pick "K" points from the Dataset D & call them as $C_1, C_2, C_3, \cdots C_K$)

↳ So, Rather than randomly picking, we will replace this step with K-Means ++ initialization.

↓

→ More Computational heavy task

K-Means

Step-1. ⟶ Replace with K-Means++ initialization

Step-2 } Same

Step-3 }

Step-4 }

K-Means ++

↳ It is nothing but x k-Means algorithm by improving not selecting (doing) 1st step.

↳ Replacing with K-Means ++.

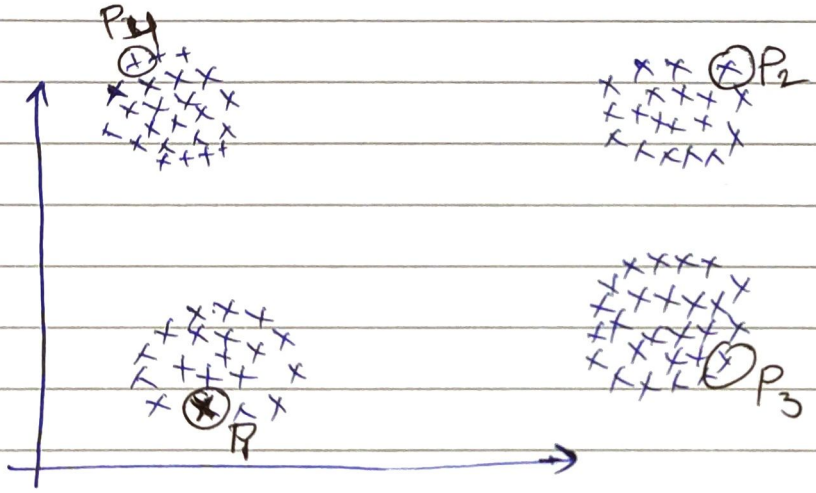
K-Means:

Step-1: Decide the value of K and randomly pick K-centroids $c_1, c_2, \ldots c_K$.

K-Means++

Step-1:- Decide the value of K and randomly pick 1 centroid

⇒ And pick K-1 Centroids as far as possible.



K=4,

⇒ I have randomly picked one point P_1

⇒ Now i will pick the 4-1=3 points as far as possible.

↳ then i will pick P2 → it is having large distance b/w them (P_1 & P_2).

→ Now i will pick P_3 as it is having high distance from P_1 & P_2.

→ In the same way we will pick P_4 as it is having high distance from P_1, P_2 & P_3.

→ Once we pick P_1, P_2, P_3 & P_4 (Lets say K=4) then our next steps will be done Quickly (step-2, 3, 4), as

→ the Centroids will Converge Quickly (sets won't Change).

(Because in the above example we got the Centroids easily as far as possible then can form the sets

→ But Getting P_1 & P_2 & P_3 & P_4 → takes Very high Computation after that it won't take much time to Converge; as it is easily clustered (In the above example points are present in each cluster)

Q) How do you decide best Value of "K"

Ans:

For this we need to calculate WCSS

↓

Within in cluster Sum of Squares.

Clusters:→

(a) Intra cluster distance to be low

(b) Inter cluster distance to be high

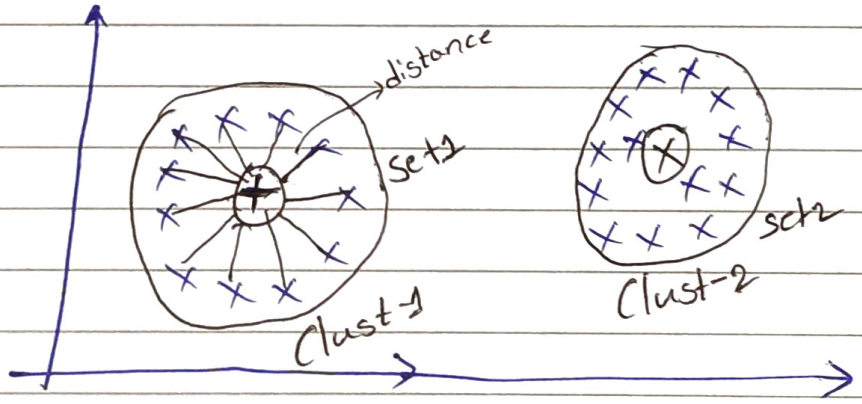
⇒ So, for good clusters, Intra cluster (distance with the Centroid inside a cluster) should be low.

⇒ So, iam calculating distance from centroid to point

↳ $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \Rightarrow \|x-c\|^2$

(x_1, y_1) (x_2, y_2)

⇒ we can represent distance as $\|x-c\|^2$



for clust-1 (centroid-1)

$$\sum_{x \in S_1} \|x-c_1\|^2$$

for clust-2 (centroid-2)

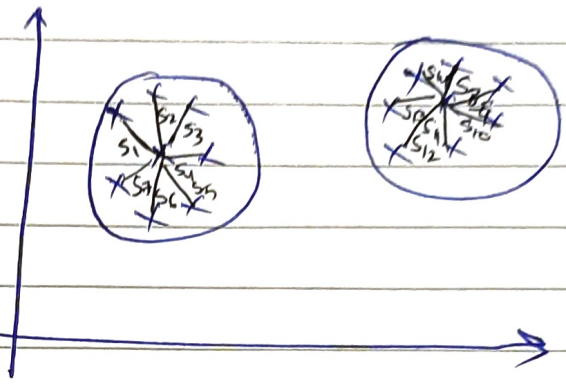
$$\sum_{x \in S_2} \|x-c_2\|^2$$

⇒ Now, i will sum up intracluster distance for the 2 centroids. (And it has to be low then only my clusters are good

→ $\sum_{x \in S_1} \|x - c_1\|^2 + \sum_{x \in S_2} \|x - c_2\|^2$ (where $S_1 + S_2 + \cdots + S_7$ and $S_8 + S_9 + \cdots + S_{14}$)

→ $\sum_{i=1}^{K} \sum_{x \in S_i} \|x - c_i\|^2$

⇒ $(S_1 + S_2 + S_3 + \cdots S_7 + S_8 + S_9 + \cdots + S_{14})$



Cluster-1 ⇒ $S_1, S_2, \cdots S_7$ → distances from centroid C_1 to points

And this sum has to be Low. then only our clusters are good.

Cluster-2 ⇒ $S_8, S_9, \cdots, S_{14}$ → distances from Centroid C_2 to points

$$C_1^*, C_2^*, \cdots C_K^* = \underset{c_1, c_2, \cdots c_K}{\arg\min} \left\{ \sum_{i=1}^{K} \sum_{x \in S_i} \|x - c_i\|^2 \right\}$$

→ Optimization Eqn.

KMeans ⇒ it is also known as LLoyd's Algorithm (4 steps)

Q) Why Can't we apply Gradient Descent on optimization Eqn so, that we Can get Centroids.

Ans)

It's because this optimization Eqn is of one of

the hardest Equation and it is really hard to solve. Thats the only reason we are going with the Lloyd's Algorithm and K Means++ Initialization.

→ Even the libraries like Sklearn, ... (anything which uses uses Lloyd's algorithm & it won't use Gradient Descent for KMeans **

(Any other library which are used for kMeans++, then it must uses Lloyd's Alg rather than Gradient Descent.

Note: And good part about this optimization Equation is that we can leverage this part to find out K.

$$c_1^*, c_2^*, \dots, c_K^* = \underset{c_1, c_2, \dots, c_K}{\arg\min} \sum_{i=1}^{K} \sum_{x \in S_i} \|x - c_i\|^2$$

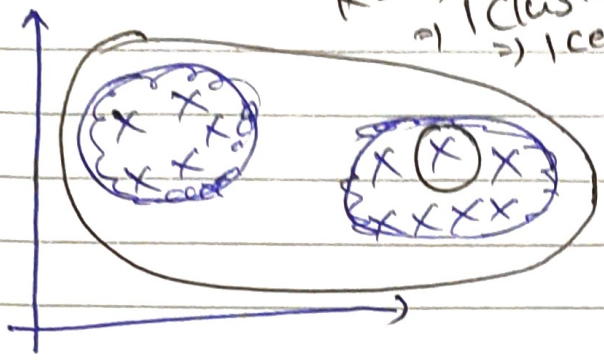
→ (This is intra Cluster Distance.

K-Means++ :→

Q) Decide the value of k.

Step-1 :→ K-Means++ initialization
Step-2 :→ Assignment (assining each data point to Corresponding set)
Step-3 :→ Recomputation (of centroids)
Step-4 :→ Repeat 2 & 3 until Convergence

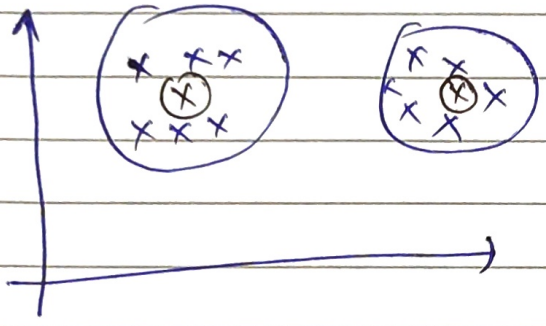
K=1, 1 cluster
⇒ 1 centroid.



Lets take K=1,
and do the
steps-1, 2, 3, 4.

→ Calculate Intra Cluster Distance
(d1)

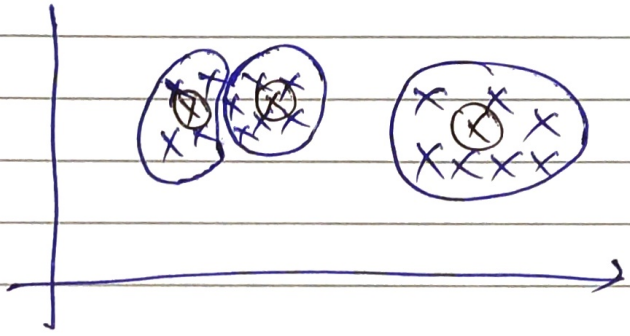
K=2



K=2, (
Now, do the step-1, 2, 3, 4)

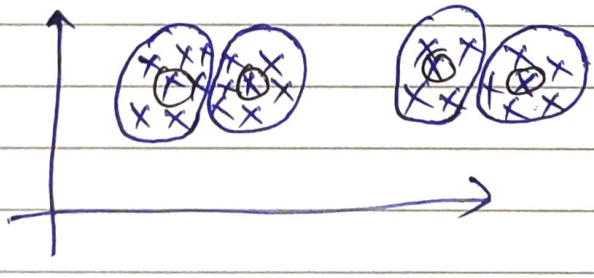
→ Cal Intracluster distance (d2)
↳ Here $d_1 > d_2$

K=3



K=3
Now, do the step-1, 2, 3, 4
→ Cal
Intracluster Distance (d3)
↳ Here $d_1 > d_2 > d_3$

K=4



K=4,
Now do the step-1, 2, 3, 4
→ Cal,
Intracluster distance (d4)
↳ Here $d_1 > d_2 > d_3 > d_4$

$$d_1 > d_2 > d_3 > d_4$$

→ My ideal K is 2

but My Equation is saying K=4 is good

because of very less Intra cluster distance (d_4)

↳ It's because we are overfitting the data

(clusters (K=4) are overly fitted)

(It's like Tight genes pant)

→ and for K=1,

My Equation is saying d_1 is very high

So, K=1 is not good.

→ 1 cluster for all the datapoints → so d_1 is very high

(for this example)

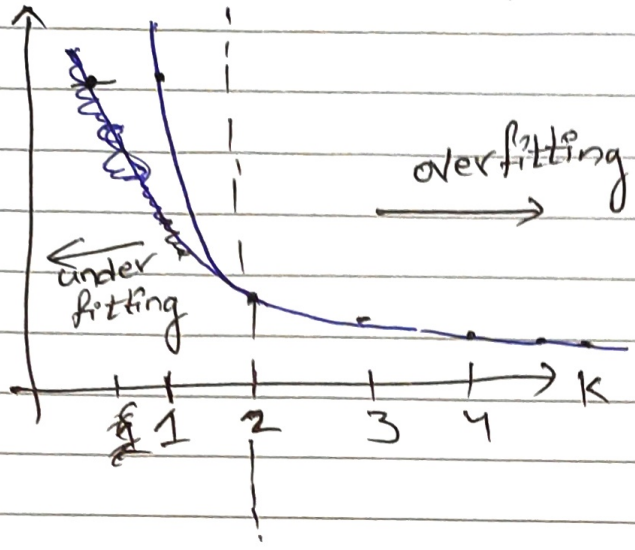
↓

It like Loose genes pant

↳ Underfitting.

→ "K" is the hyperparameter.

distance (Intracluster distance) → Elbow Graph



If i plot the distance & k then we can see sudden drop (at k=2) after that very small difference in the next K (K>2)

at K=2

↳ Intracluster distance will fall very high

for K>3 ↳ Intracluster distance will gradually reduce (slowly)

→ So in this way we use to get optimal no. of clusters.