**UNSUPERVISED LEARNING & RECOMMENDER SYSTEMS**

[WHAT IS CLUSTERING?](#_t6iao83vy2zz)

[APPLICATIONS OF CLUSTERING](#_skguc8t84o8n)

[PERFORMANCE METRICS FOR CLUSTERING](#_nbac6uxoqtmk)

[K-MEANS CLUSTERING](#_a8g8cswjjh7v)

[K-Means: Mathematical formulation: Objective function](#_igvwcnhv7sx1)

[K-means Algorithm : Lloyd’s algorithm](#_u68tkve25gzj)

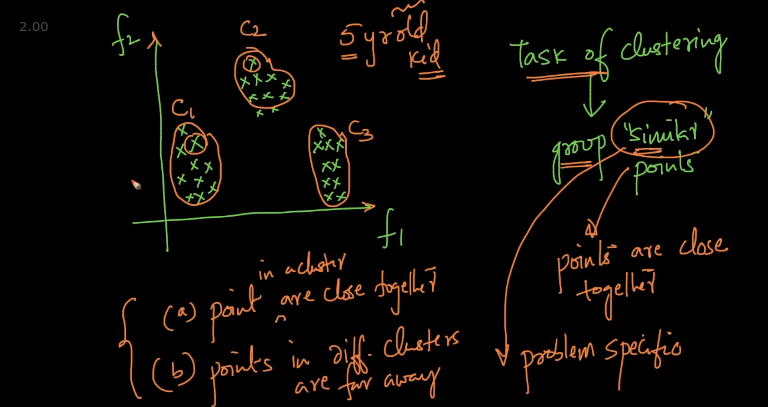
[K-Means ++](#_117j7avaok4m)

# **WHAT IS CLUSTERING?**

In Classification & Regression we’ve D = {x, y} and our job is to find the function f(x)

with x’s and labels y’s but in clustering there are no . Our task here is to cluster

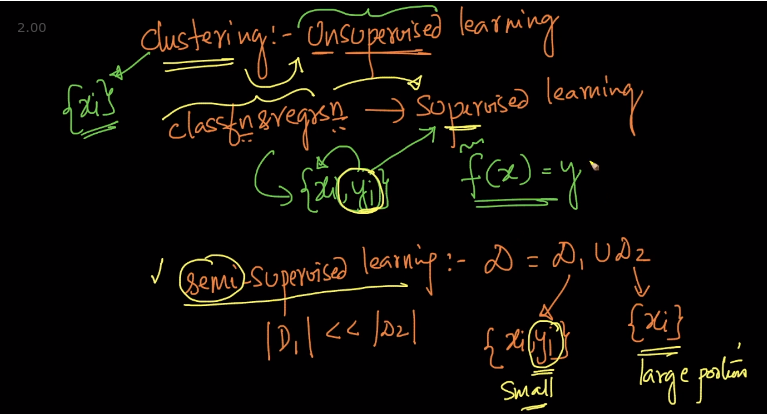
Similar data points



So we need to group similar points. In clustering we have 2 points

a) Points in a cluster are close together

b) Points in different clusters are far away

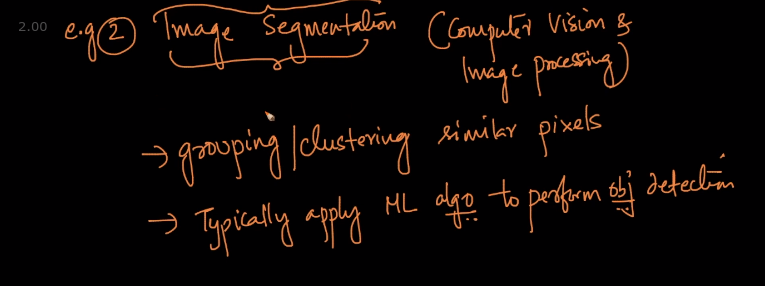


In Supervised Learning , we’ve y with x in which y helped/supervised us to get f(x)

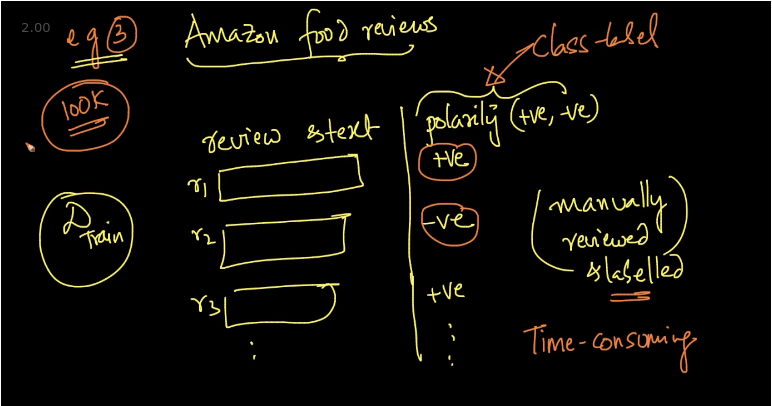
In Semi-supervised , large portion of data is just x no y + data which has both x,y

# **APPLICATIONS OF CLUSTERING**

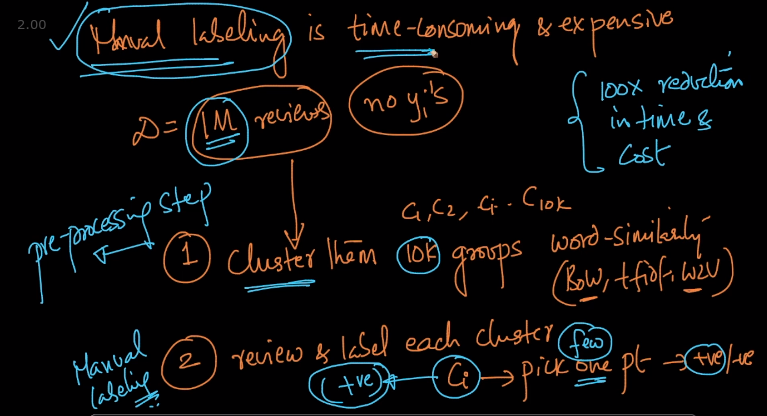
In ecommerce it can be used to cluster similar customers like in the above example has higher tendency to get stuff so they can be offered a certain deal



Like in a person sitting on a horse. Here cluster of horse, person and backgrounds would be made

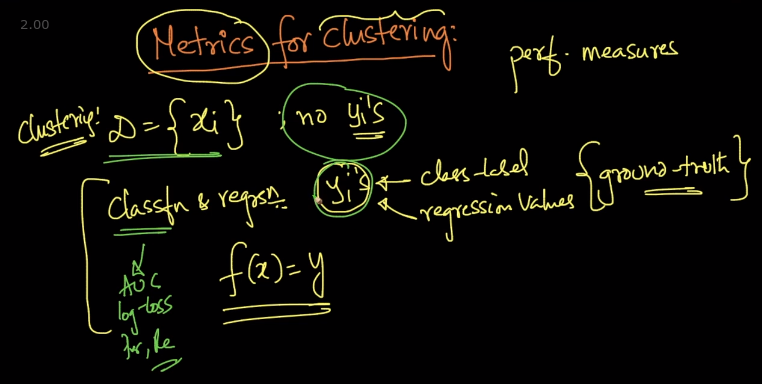


Someone must have manually reviewed and labelled in Amazon reviews. It can be very time consuming if the number of rows are lot

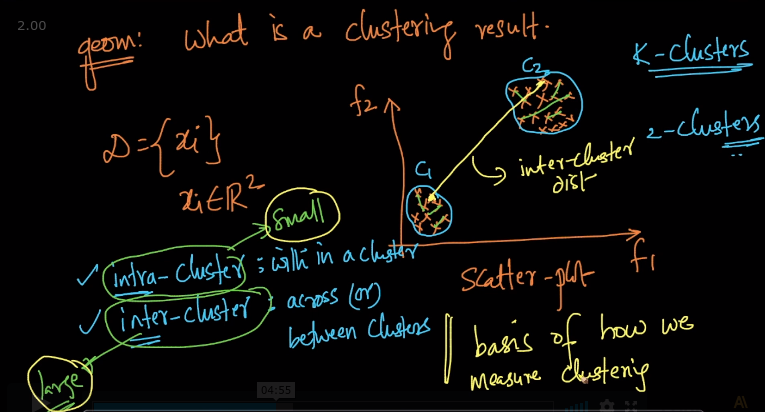


If we’ve 1M reviews with no y’s. We make clusters of 10k for each group with word similarity techniques like BOW, TF-IDF. Now we pick a cluster and 1 point from that and manually check if it’s a +ve or -ve review and if it’s +ve then we give the whole cluster positive label. Do it for every cluster and our dataset is labelled with reduction in time

# **PERFORMANCE METRICS FOR CLUSTERING**



In Classification and Regression we had class-labels to measure performance metrics like AUC, Precision and Recall but for these metrics we need y. In clustering we don’t have y’s

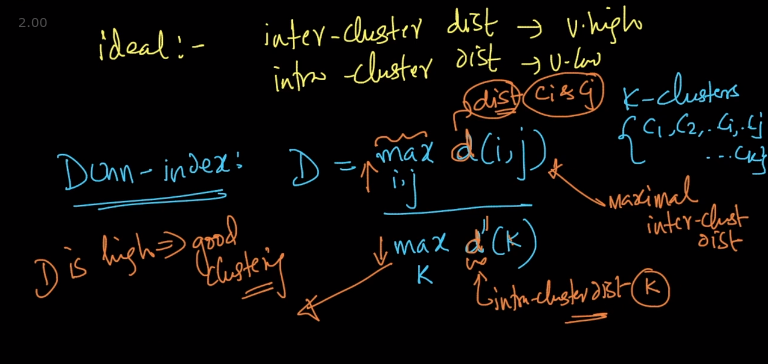


In the above 2-dimensional dataset there are 2 clusters .

Intra cluster : Within a cluster. This distance should be small

Inter Cluster : Across the clusters. It should be large.

This idea of Intra and Inter cluster forms the basis of measuring clusters

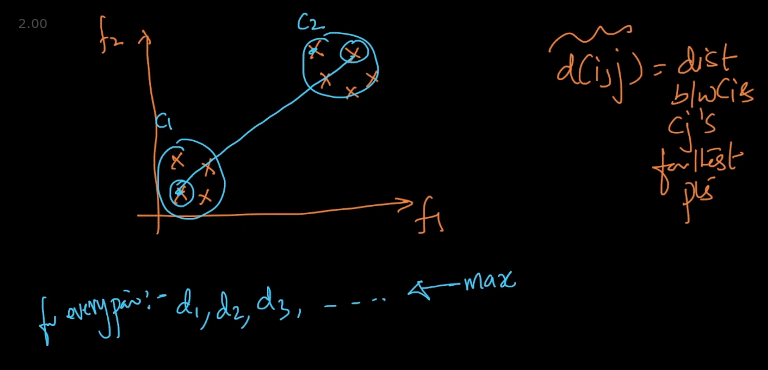


Dunn-Index : In an ideal scenario we want inter cluster distance high and intra cluster low

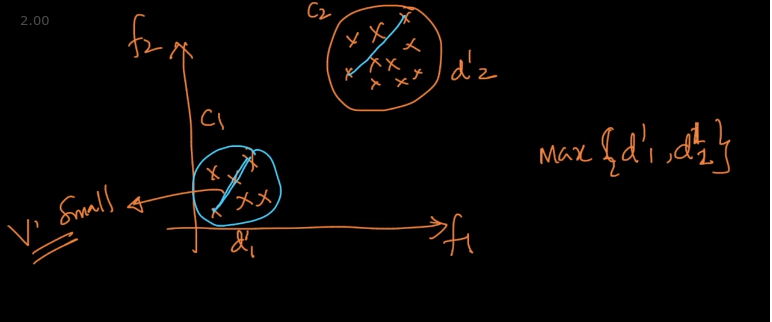
Dunn Index: D = . Suppose we’ve k- clusters {}

In Dunn Index formulae, numerator min d(i, j) is the distance between 2 clusters

and denominator max d’(k) i.e intra cluster distance within a cluster k here we look at each cluster and find the cluster which has maximal intra cluster distance

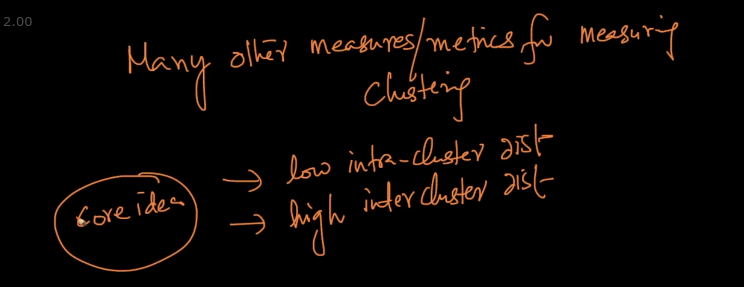
If D is high then good clustering is done

d(i, j) = distance between 2 clusters points. We calculate it by taking distance of each pair of points in which 1 point is from and another is . We take distance of every pair of points and select the one which is minimum as d(i, j) i.e numerator in Dunn



In d’(k). We take distance of points which has maximum distance for every cluster and then we select the cluster which has maximum distance

So in Clustering if min d(i, j) is large and max d’(k) is small then our Dunn Index would be higher which means our clustering is good



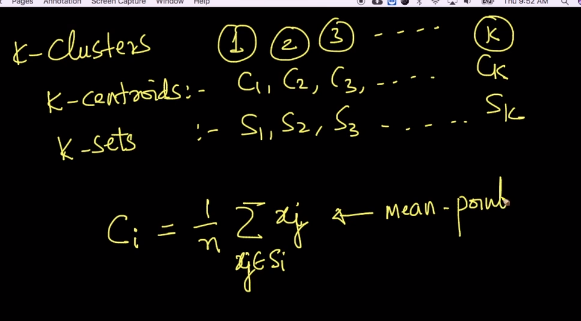
There are many other metrics for measuring clustering but the core idea mentioned above is for all metrics

# **K-MEANS CLUSTERING**

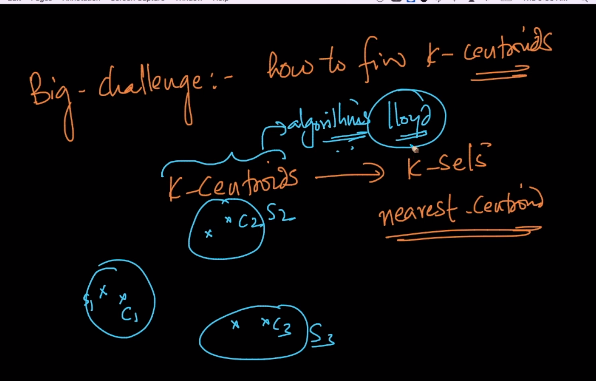


In K-means , K : No. of clusters which is a hyperparameter . K-means is based on Centroids

If there are 3 clusters i.e K = 3 then there are 3 centroids and the points in each clusters are sets and no single point is there in 2 sets i.e =



Centroid is geometric mean of all points in the set. When there are K- clusters it means there are K-centroids and K-sets



Nearest Centroid method : For ex - The point nearest to a particular Centroid let’s say C1

will belong to Set 1 instead of S2 or S3 because it’s nearest to C1.

Our big challenge is how to find K-centroids ?

# **K-Means: Mathematical formulation: Objective function**

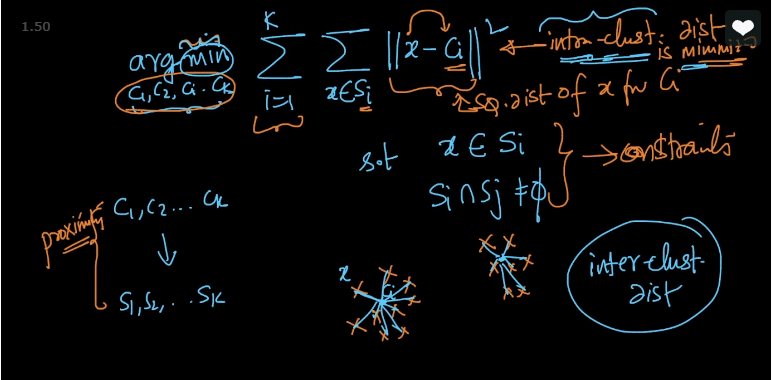
In K-means our task to find K-Centroids and their corresponding sets

Constraints : 1 ) Every point in the dataset should belong to atleast 1 set

2 ) = , i.e The intersection between any 2 sets is a null set. No

point should belong to 2 clusters

# 



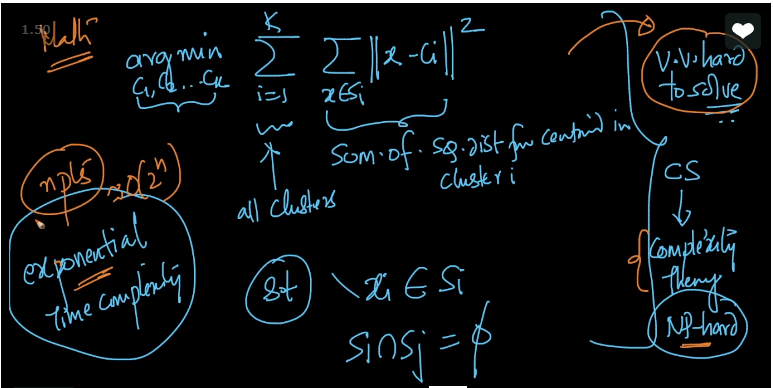
We want to find k-Centroids because once we find we can automatically find

. So the objective function is arg-min

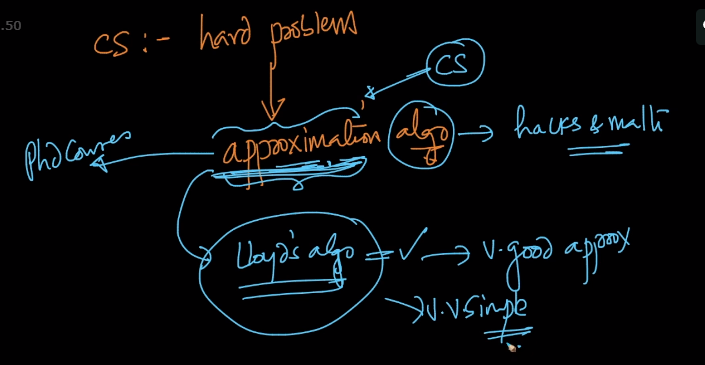
We summate for x belonging to Set , (x ) the squared distance of x from Ci

( ) and then summate this distances for every cluster and our objective function wants to minimize these value of these summations with our constraints

Basically, intra-cluster i.e within cluster distance is minimized



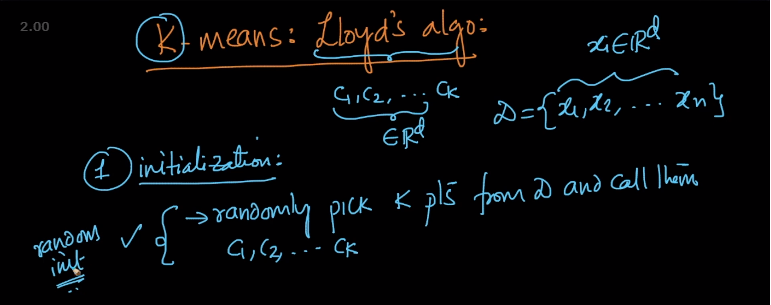
But the problem with this function is that it’s very hard to solve this problem as this is an NP hard problem



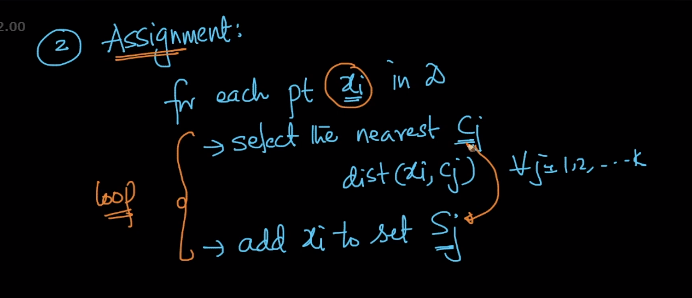
So in Computer Science whenever we het a hard problem we use approximation algorithm

i.e some hacks and math. In k-means Lloyd’s algorithm is our approximation algorithm and it works like our Objective function defined above

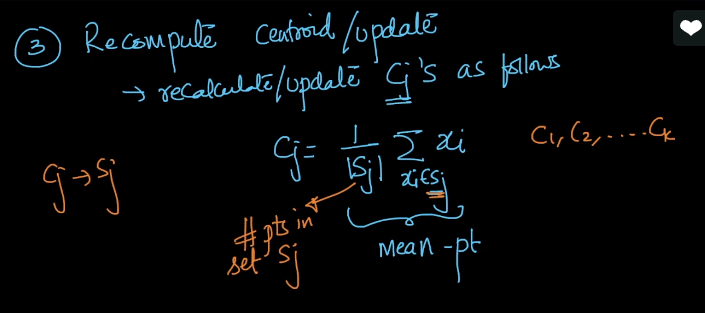
# **K-means Algorithm : Lloyd’s algorithm**



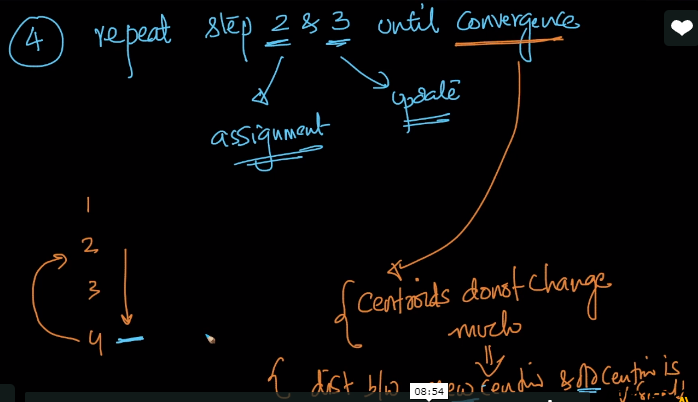
**Initialization** : From our dataset D , we select k-points ( Centroids) randomly and call them



**Assignment** : For each point in D, select the nearest centroid . To select the nearest centroid we need to calculate distance of from all Centroids and whichever is the minimum we select that. Then we add yo set



**Recompute/ Update centroid : .** Here we are taking every and summating it and then dividing it by the number of points in set ()



**Convergence :** Repeat steps assignment and update till Convergence. Convergence means

Centroids do not change much after updating i.e we can store the Centroids in a vector

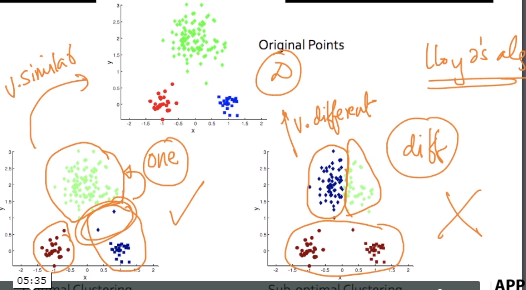
So we can check the distance of Old Vector & New Vector and if it’s very small then done

# **K-Means ++**

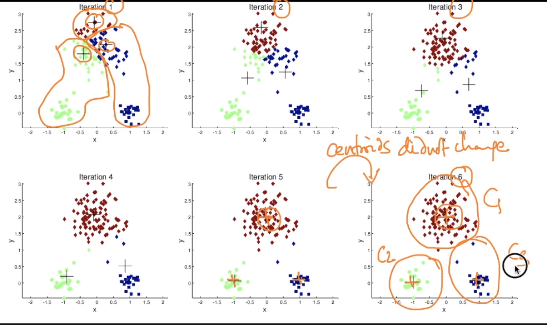
In Lloyd’s algorithm we randomly pick k-points but thr problem with random initialization

is initialization sensitivity i.e our final clusters and centroids are dependent on how we

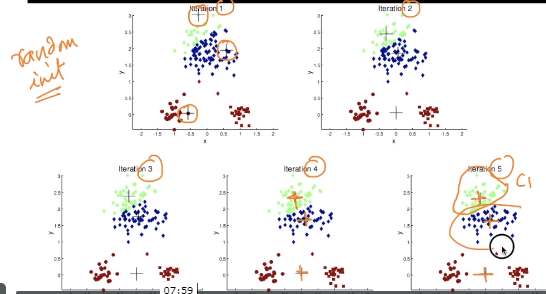
Initialize or pick our centroids



Here the top diagram is humanly labered cluster. With initialization we can get the bottom diagrams as seen. We want the bottom left one in clustering but with random initializiation we can get the bottom right cluster as well but we don’t want that

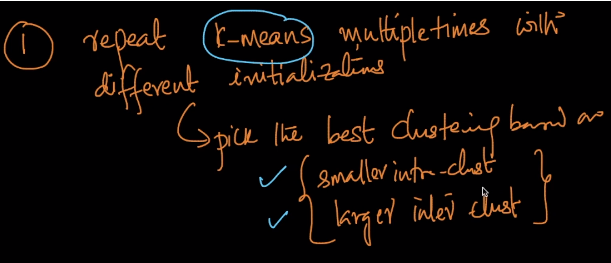


If we initialize the algorithm with the points in iteration 1 as centroids then in the 6th iteration we can get desirable results as seen

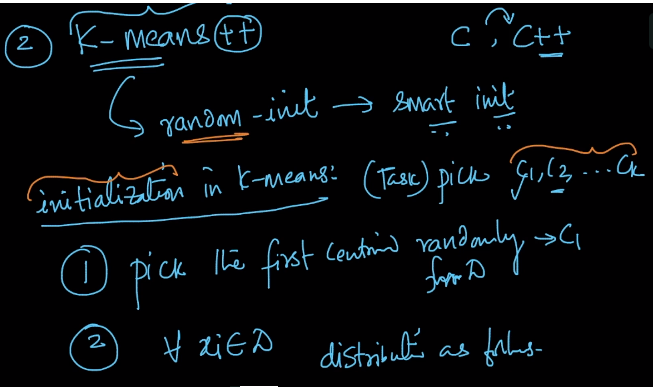


If we initialize the algorithm with the points in iteration 1 as centroids then in the 5th iteration we get results like that but that clustering result is not what we want. But since it is random initialization it can be initialized like above. So initialization sensitivity is a big problem in Lloyd’s algorithm

How this problem can be dealt?



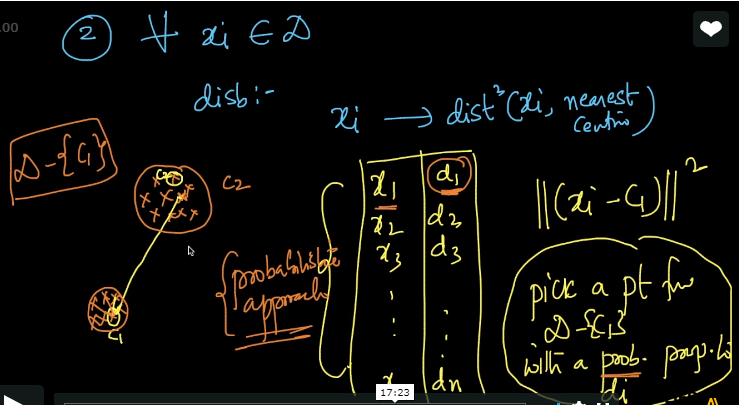
But the above method is computationally expensive



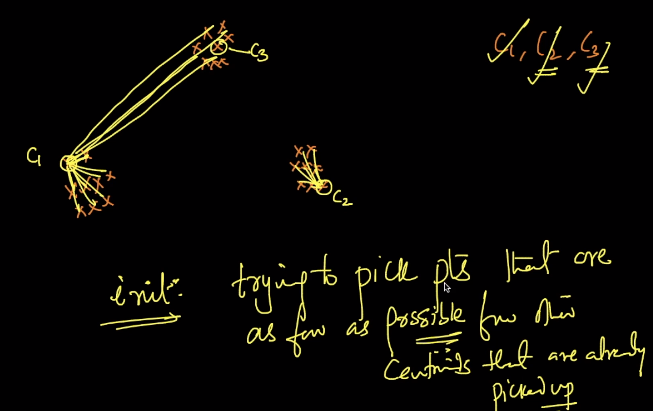
So we use K-means ++. Here instead of random initialization we use smart initialization

1 ) We pick the 1st centroid randomly from D

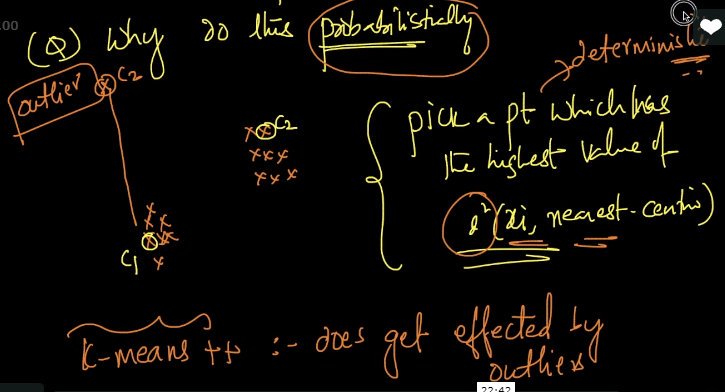
2 ) create a distribution as follows



Every point has a corresponding distance() from the nearest centroid. So we create a distribution with points and distances and we pick a point from D - {} with

probability i.e higher the distance of point from centroid higher is it’s probability to be the next centroid

Suppose we’ve picked up . We need to pick . Let’s assume is closer to the cluster for . So we calculate distance of points from and select the point as which has higher distance from since it has higher probability. Therefore in Initialization we are trying to pick centroids that are as far as possible from other centroids that are already picked up



Why can’t we simply select the point which has the highest distance from nearest Centroid?

Why use probabilistic method ? If we select the point which has highest distance then as seen above an outlier can be selected as well but we use probabilistic sampling (In Prob ch)

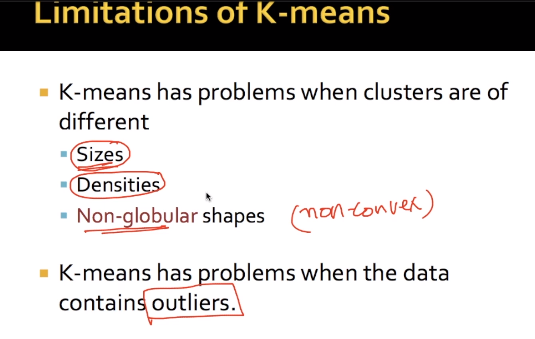
So that we select the right centroid

Let’s say the distances of points are: 1, 1.2, 1.5, 1.3, 0.8, 10, 11, 12 and 9.

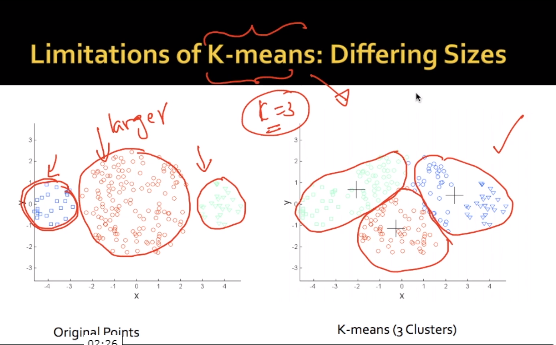
The probability values are 0.0022, 0.0031, 0.0049, 0.0037, 0.0014, 0.2207, 0.2670, 0.31 and 0.178. The max of these is 0.317 which corresponds to the one with distance = 12 from the current centroid .

The key point here is: when we pick probabilistically, the farthest point has a higher chance of being pickup but it is NOT always guaranteed to be picked up with 100% certainty. The point which is at a distance of 12 certainly has a higher probability (0.31786676) of being picked up. But, it is not always guaranteed to be picked up as [probabilistic sampling](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/proportional-sampling-3/) will pick this point with a probability of 31.78% only. There is 100-31.78% to pick another point

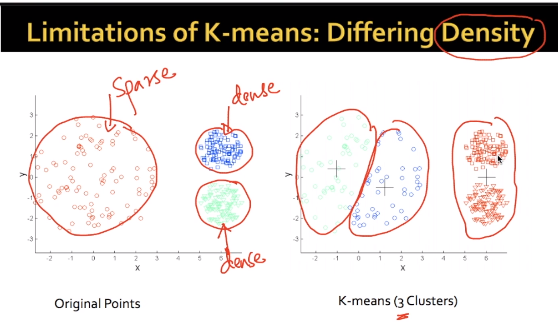
# **PROBLEMS /LIMITATIONS**



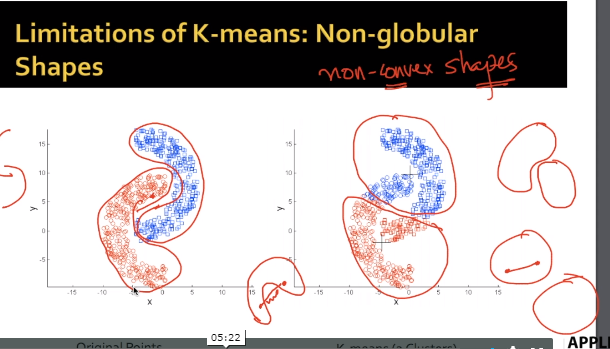
These are the problems or limitations K-means face



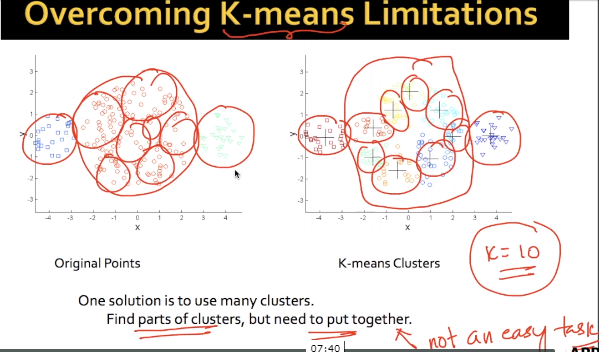
Original points are of differing sizes but K-means tends to create equally sized clusters



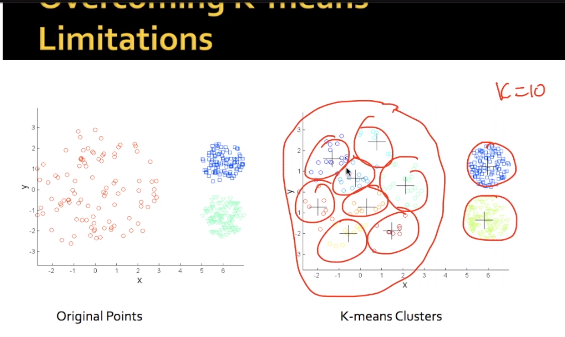
Original points are of different densities as seen but in K-means the red cluster is streching out to match the sparsity of the left clusters

****

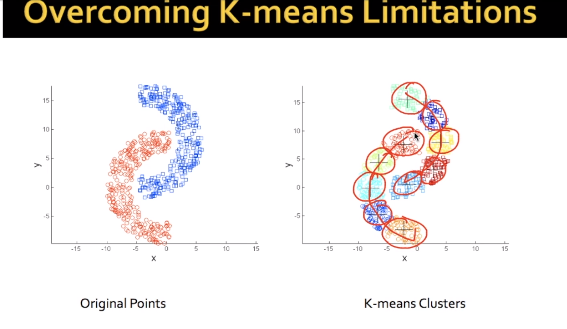
In non-convex shape it cant cluster it properly since k-means focuses more on intra cluster distances



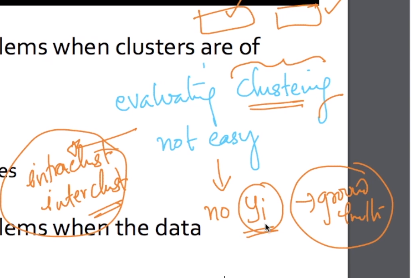
So to overcome this problem we can increase no. of k = 10 so many parts will be created but we need to put them together which is not an easy task



The idea of increasing k can work in these sparse dense clusters but again we need to put them together



Same idea as above



Ecaluation of clustering is hard since it depends on intra and inter cluster