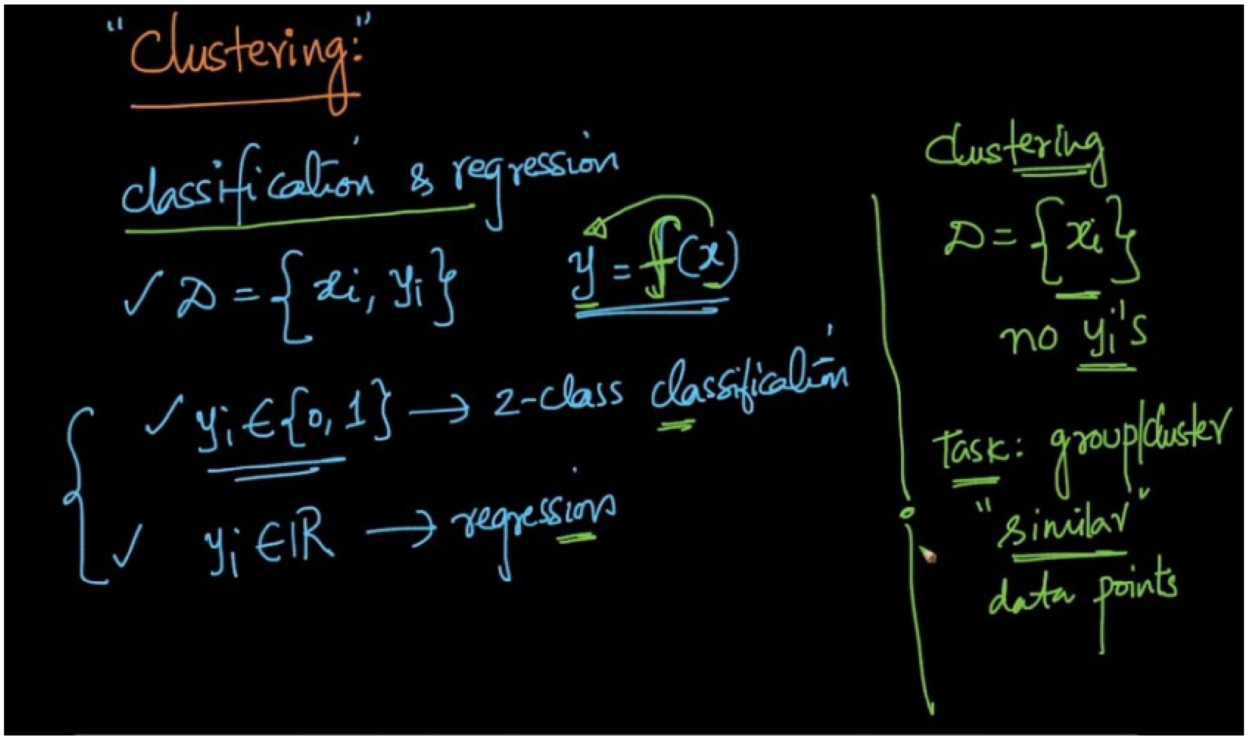
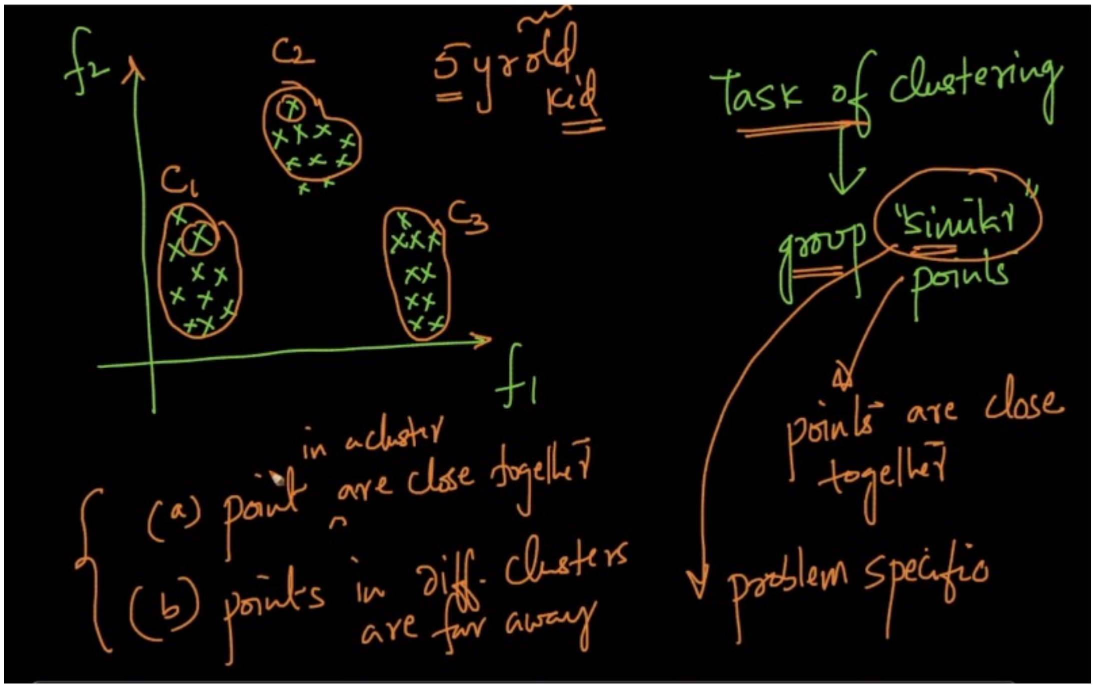
Clustering

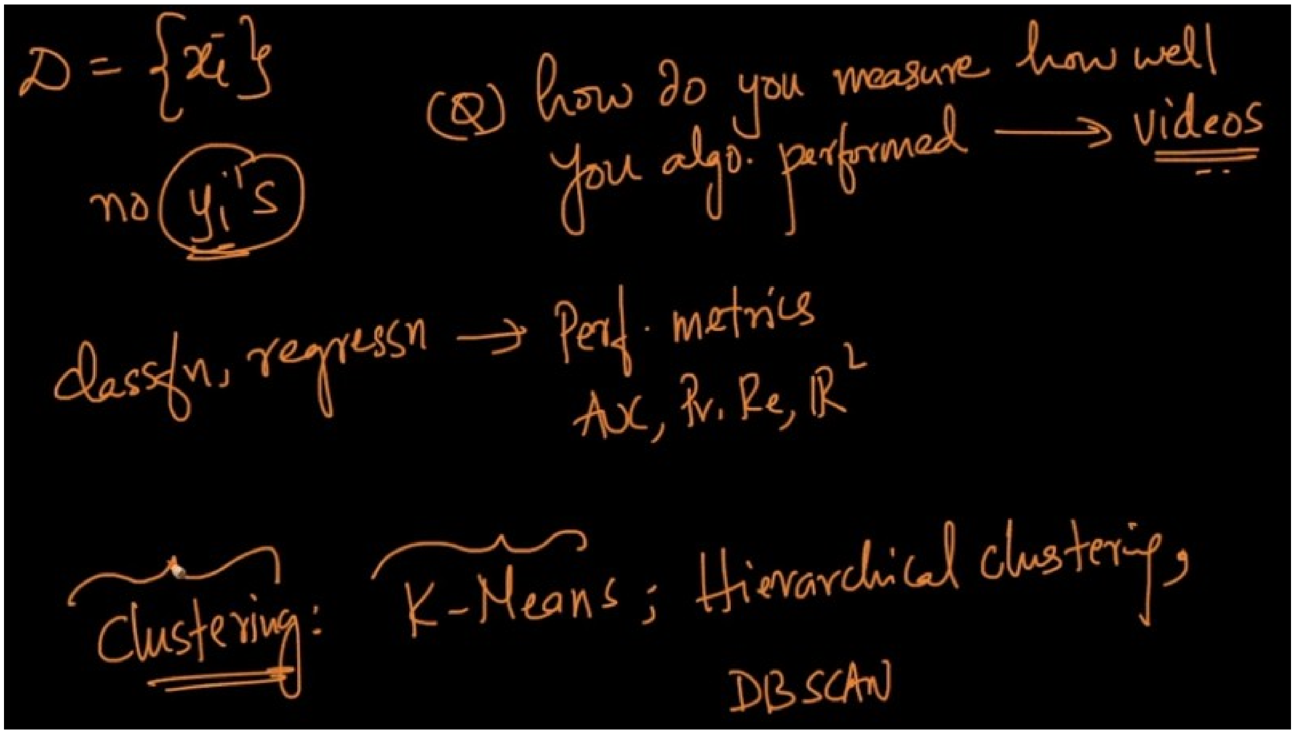
Standard problem:
Regression and classification:
We want to find the function that satisfies the training data. In Case of clustering the task is clustering.
Task: It is to group the similar data points.


"Clustering:"
classification & regression
✓ $D = \{x_i, y_i\}$
$y = f(x)$
✓ $y_i \in \{0, 1\} \rightarrow$ 2-class classification
✓ $y_i \in \mathbb{R} \rightarrow$ regression
clustering
$D = \{x_i\}$
no y_i's
Task: group/cluster "similar" data points

Consider the data points as follows:
The task of clustering is to group the similar data points. All these points are grouped together. The points are grouped together. The similar is very much task specific.


f_2
f_1
C_1
C_2
C_3
5 yr old kid
Task of clustering
group "similar" points
points are close together
problem specific
(a) point in a cluster are close together
(b) points in diff. clusters are far away

Here all the data given are the data points in the mathematical form.
Measuring the performance of the clustering algorithm. There are various metrics that can measure the performance of clustering.
Clustering algos: K- Means, Hierarchical, DBSCAN.



$D = \{x_i\}$

no y_i's

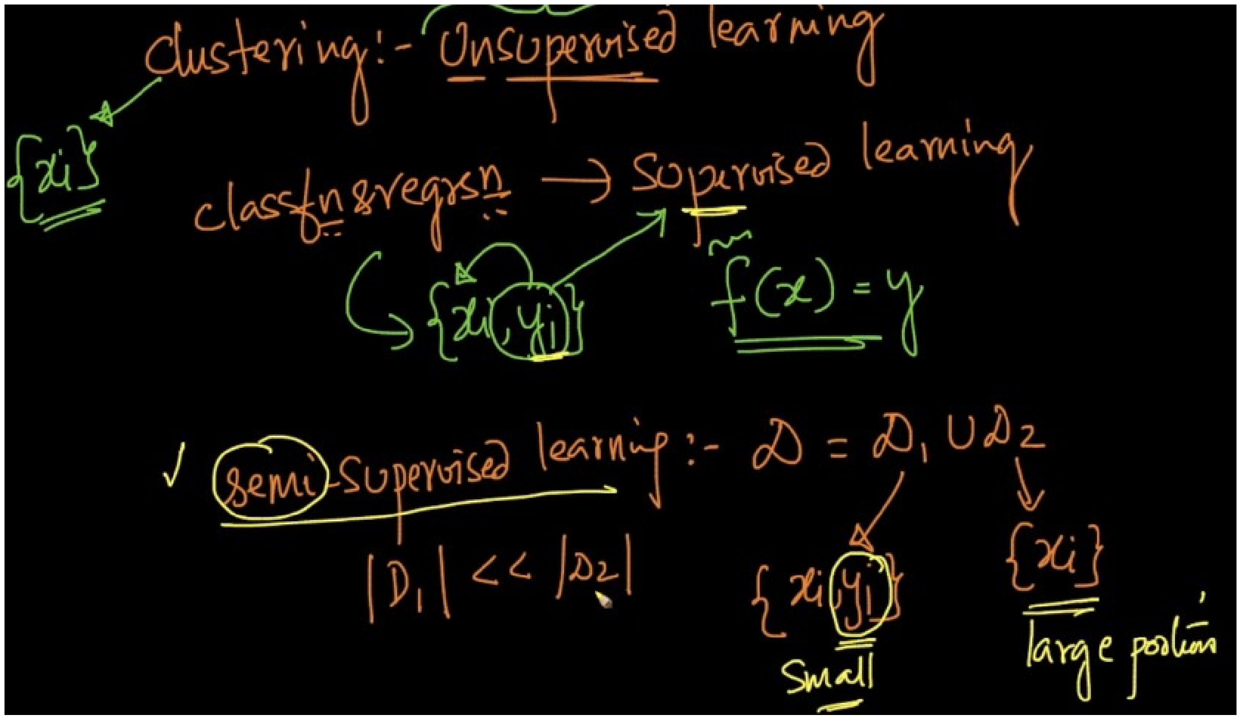
(Q) how do you measure how well you algo. performed $\rightarrow$ videos

classfn, regressn $\rightarrow$ Perf. metrics

Acc, Pr, Re, R^2

Clustering: K-Means; Hierarchical clustering, DBSCAN

These are the most used algos. There are algos for various data.
Unsupervised learning: Clustering is referred to as unsupervised learning. Both classification and regression are the supervised learning.
There is also an area called semi-supervised learning, the data sets are union of D1 and D2. This happens when the cost of labeling the data is expensive.



Clustering :- Unsupervised learning

$\{x_i\}$

classfn & regrsn $\rightarrow$ Supervised learning

$\{x_i, y_i\}$

$f(x) = y$

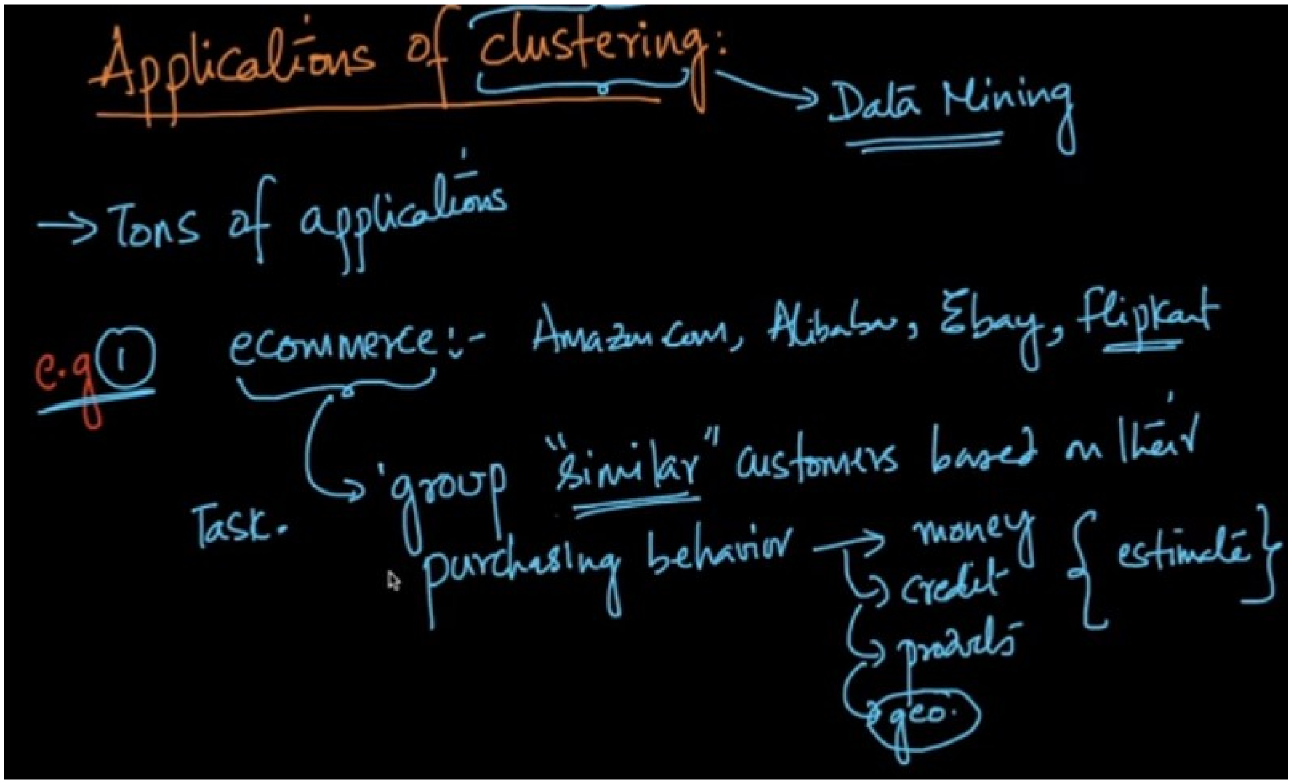
✓ semi-Supervised learning :- $D = D_1 \cup D_2$

$|D_1| << |D_2|$

$\{x_i, y_i\}$ small

$\{x_i\}$ large portion

Applications of clustering: These are more studied in Data mining, than machine learning.
There are many applications of clustering.
They want to group the similar customers based on the purchasing behavior.
They can decide the income level using the purchasing behavior.



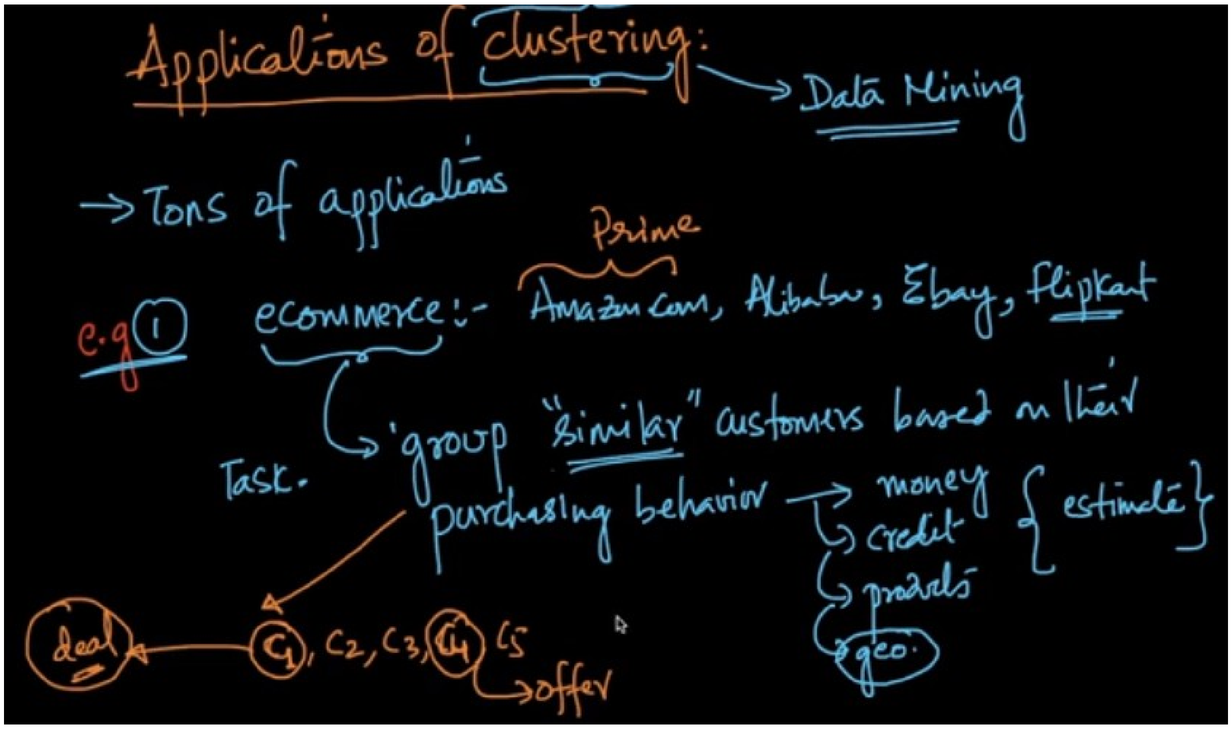
Applications of clustering: → Data Mining

→ Tons of applications

e.g ① ecommerce :- Amazon.com, Alibaba, Ebay, flipkart

Task. ↳ 'group "similar" customers based on their purchasing behavior → money ↳ credit ↳ products ↳ geo. {estimate}

Assume the clusters are grouped to various classes. By grouping customers we can give different offers.



Applications of clustering: → Data Mining

→ Tons of applications

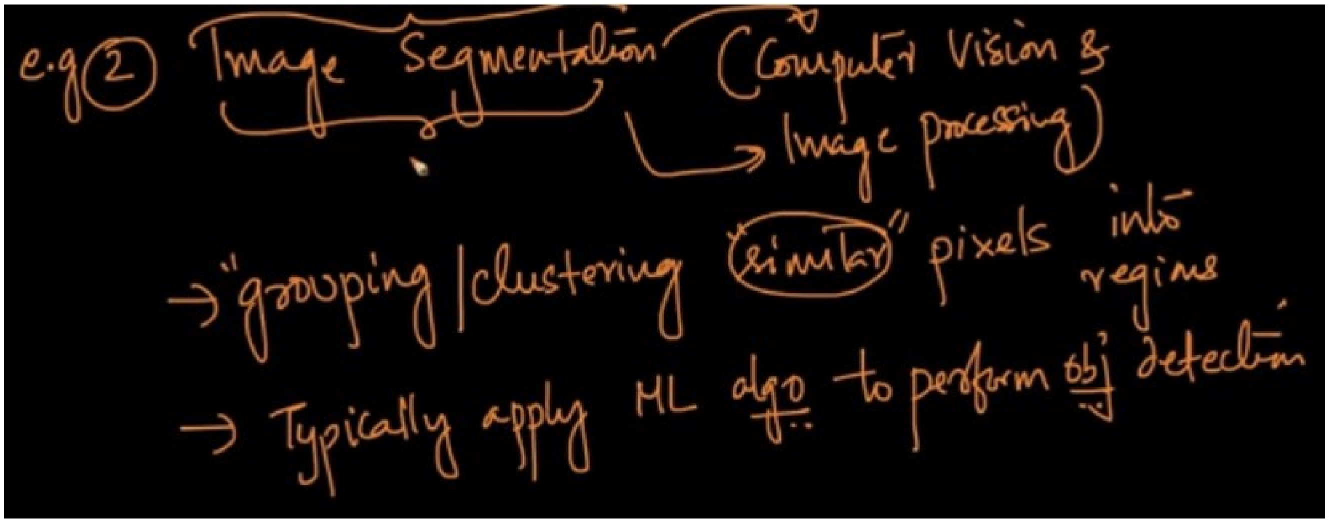
e.g ① ecommerce :- Amazon.com (Prime), Alibaba, Ebay, flipkart

Task. ↳ 'group "similar" customers based on their purchasing behavior → money ↳ credit ↳ products ↳ geo. {estimate}

deal ← C1, C2, C3, C4, C5 ↳ offer

Image segmentation: This is a problem in computer vision and image processing.

This is all about segmenting the pixels of the image. The clustering can help in seg the image.
We can apply the ML algos to apply the object detection.

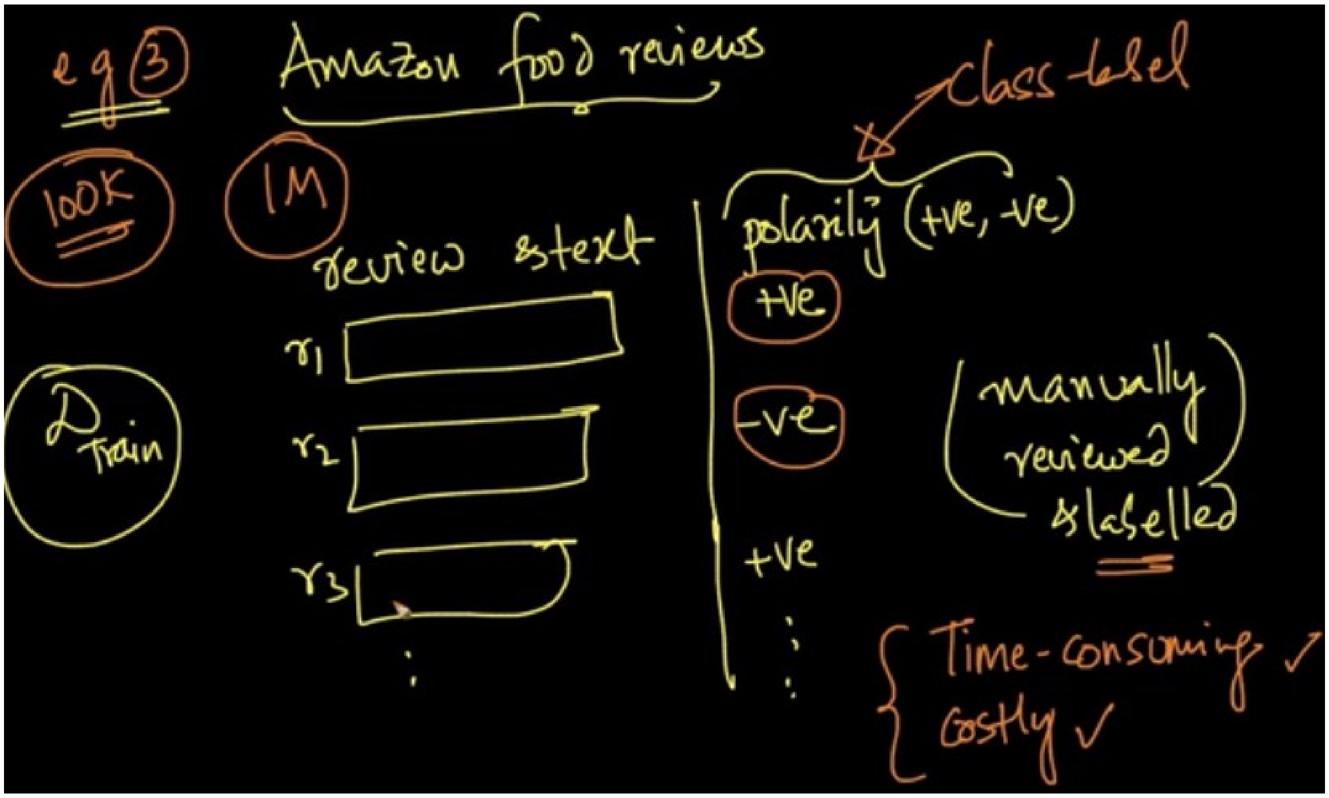


e.g ② Image Segmentation (Computer Vision & Image processing)

→ "grouping | clustering "similar" pixels into regions

→ Typically apply ML algo to perform obj detection

The polarity is the class label. This is very time consuming. We can apply the clustering algorithm to get the labels of the reviews.



eg ③ Amazon food reviews

100K

1M

D_{Train}

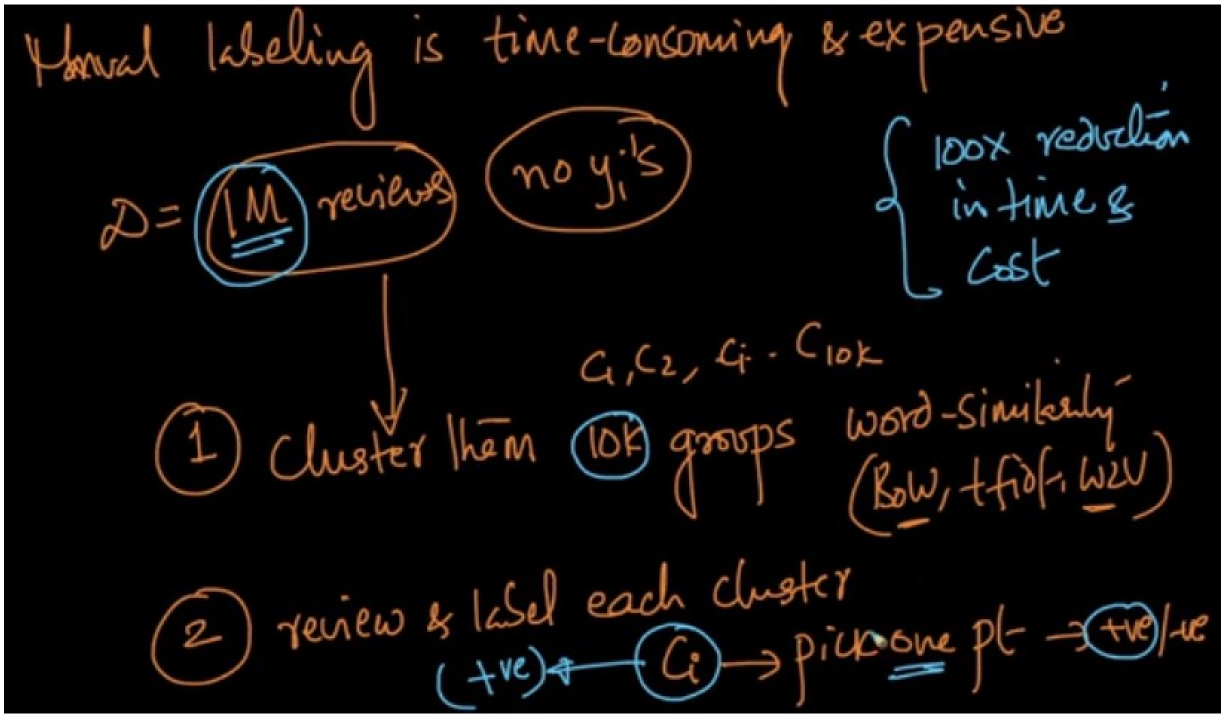
Class label

review & text	polarity (+ve, -ve)
r_1	+ve
r_2	-ve
r_3	+ve
⋮	⋮

(manually reviewed & labelled)

{ Time-consuming ✓
Costly ✓

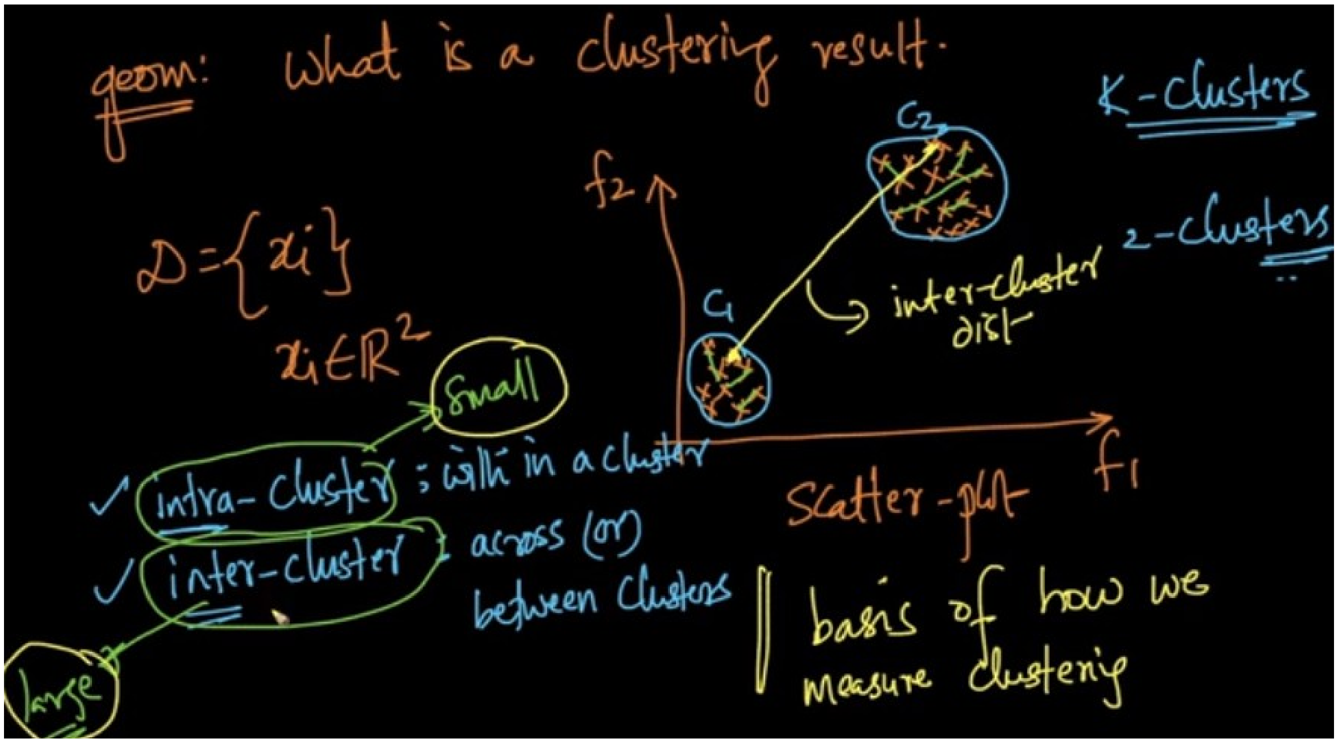
The manual labeling is time consuming and expensive. We can pick a cluster it is review and label each cluster.


Manual labeling is time-consuming & expensive
D = 1M reviews
no y_i's
100x reduction in time & cost
$C_1, C_2, C_i \cdot C_{10k}$
1 Cluster them 10k groups word-similarity (BoW, tfidf, W2V)
2 review & label each cluster
(+ve) ← C_i → pick one pt → +ve/-ve

Now, we can train the machine learning algorithms.

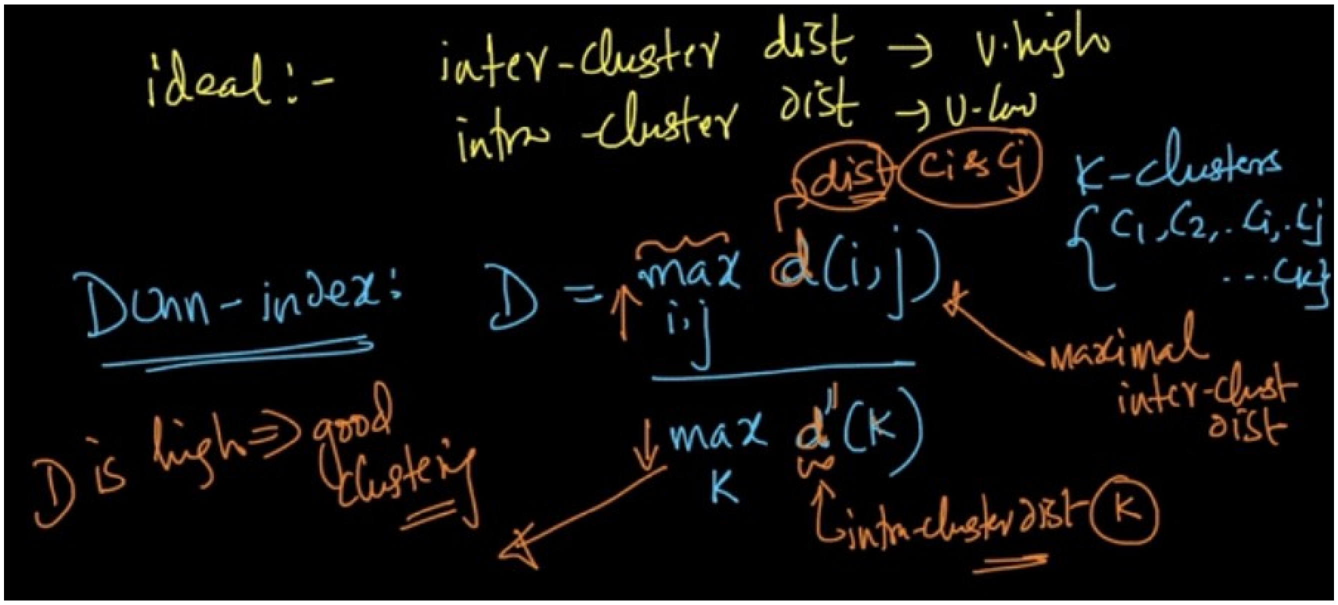
Metrics for measuring Clustering: What is a good clustering result?

All the points inside the cluster is called the intra clusters. All the outside the cluster is called the inter clusters. The intra cluster distance is kept small and the inter cluster distance kept large. This leads to the good clustering. **This is the basis of clustering effectiveness.**


geom: What is a clustering result.
K-Clusters
2-clusters
$D = \{x_i\}$
$x_i \in R^2$
f_2
f_1
C_1
C_2
inter-cluster dist
small
large
intra-cluster : with in a cluster
inter-cluster : across (or) between Clusters
Scatter-plot
basis of how we measure clustering

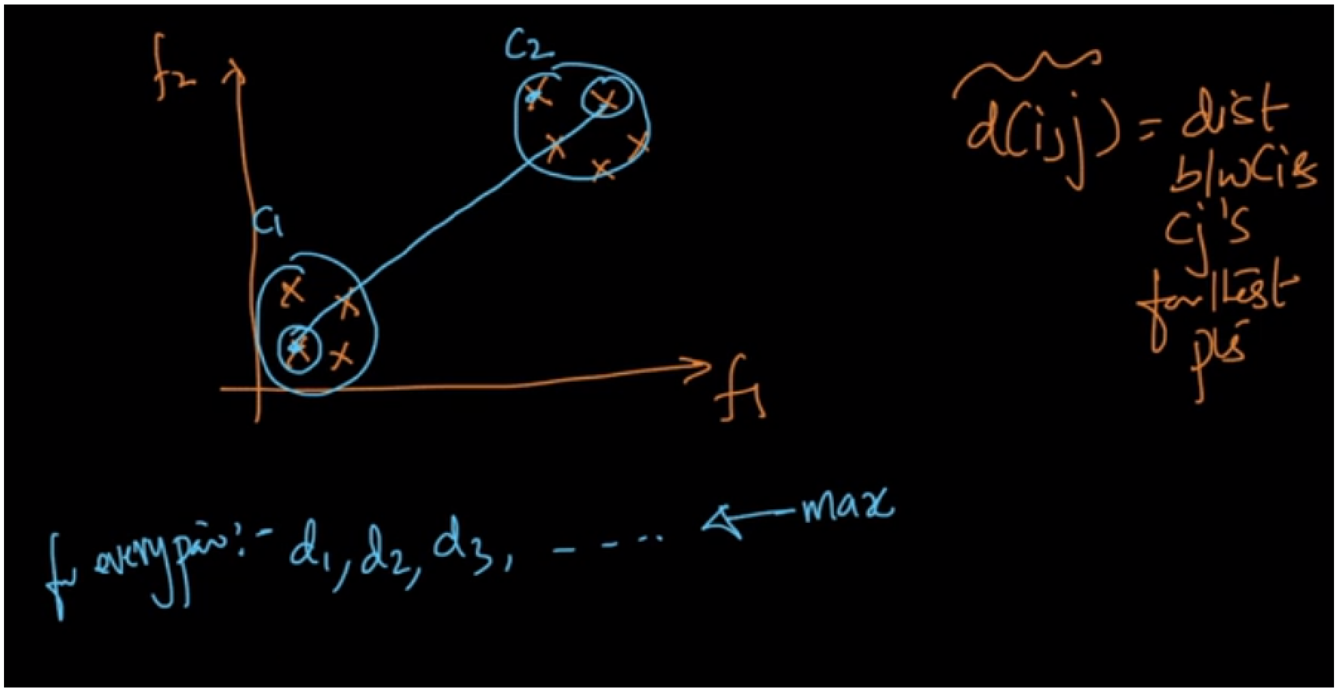
In an ideal world, we want the inter clustering distance to be very high and intra to be lvery less.

Dunn – index: The numerator is the max inter cluster distance. The denominator is the intra cluster distance.



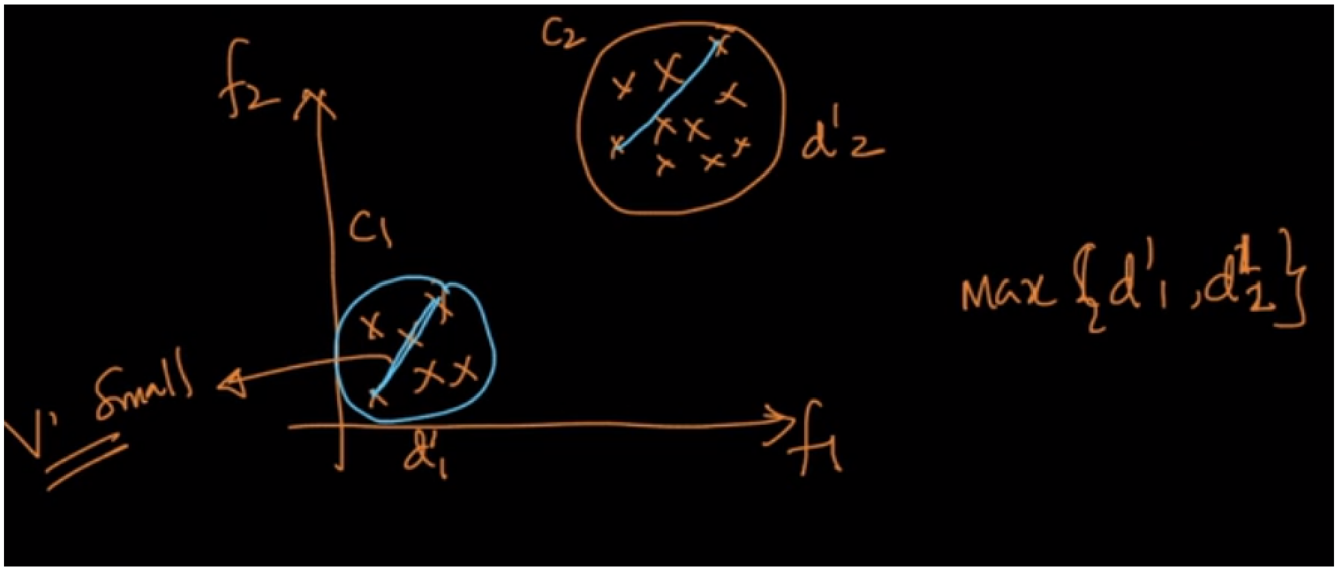
ideal :- inter-cluster dist → v.high
intra-cluster dist → v-low
dist Ci & Cj
K-clusters {C1, C2, .. Ci, .. Cj ... Ck}
Dunn-index:
D = max_{i,j} d(i,j) / max_K d'(K)
maximal inter-clust dist
D is high ⇒ good clustering
intra-cluster dist (K)

The distances are calculated as follows and the farthest points are chosen.

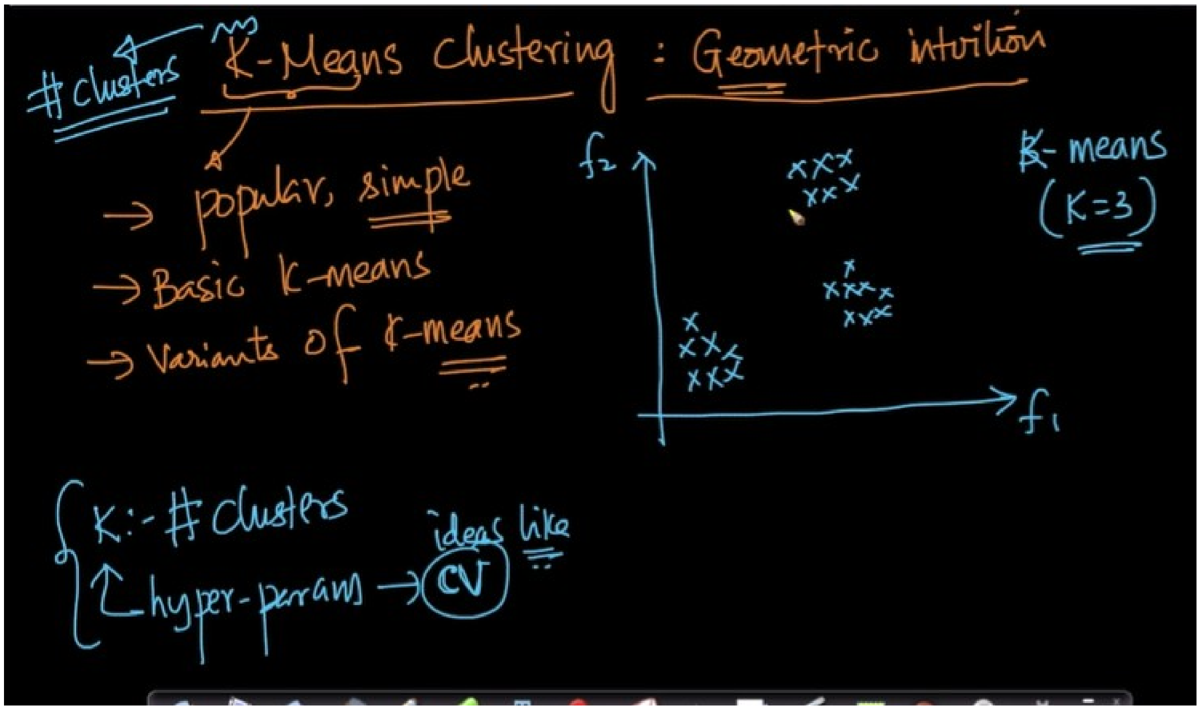


f2
f1
C1
C2
d(i,j) = dist b/w Ci's Cj's farthest pts
for every pair :- d1, d2, d3, ... ← max

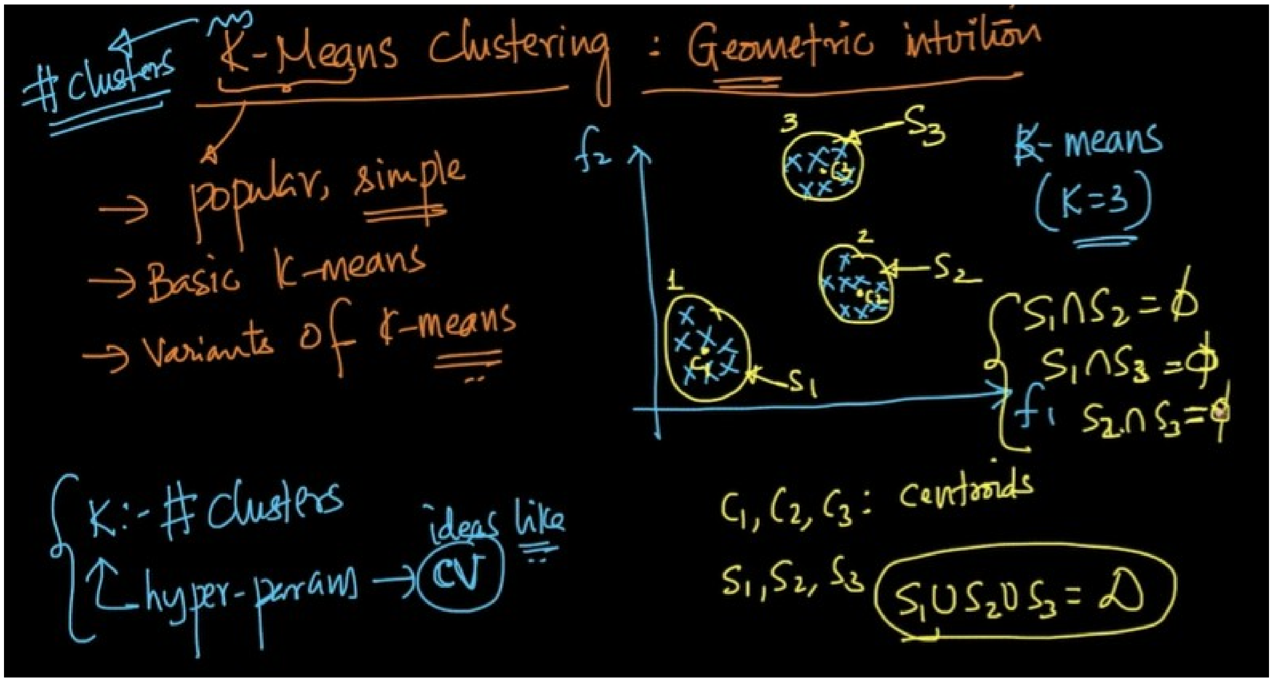
Denominator: The farthest distance of the cluster is computed, then we take the max over the distances d’1 and d’2.



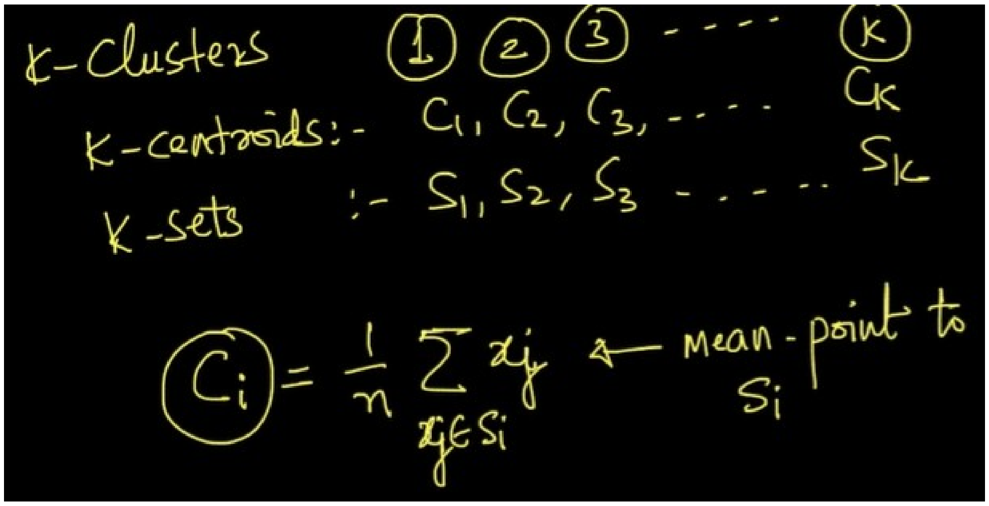
K – Means: Geometric intuition, Centroids
It is the popular clustering algorithm, variants of K- Means.
The K is the number of clusters which is the hyper parameter int K – Means.



K-Means groups every cluster, In this case we have three clusters. The intersection of the clusters is the null set.



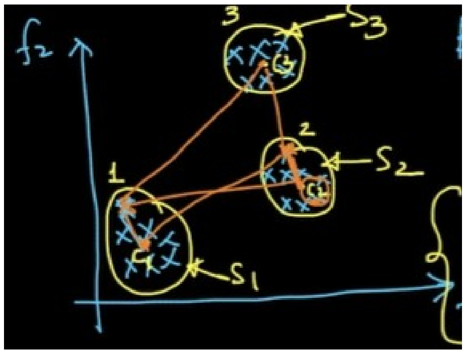
In K-Clusters, we have K centroids. K- clusters, K-sets of points. It is the geometric mean of the each cluster.



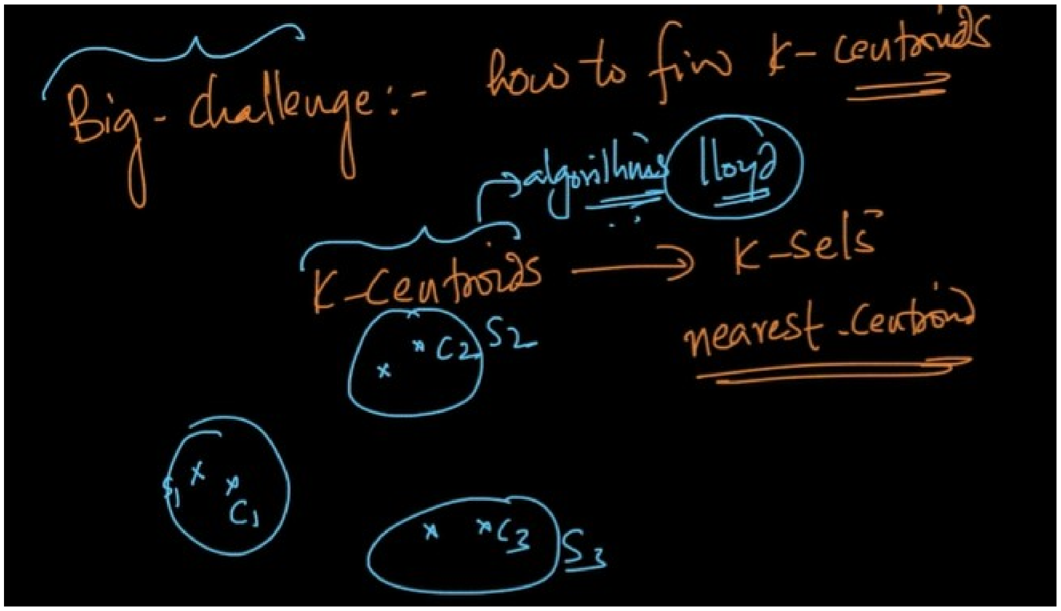
K-Clusters ① ② ③ .... Ⓚ
K-centroids :- $C_1, C_2, C_3, \ldots C_K$
K-sets :- $S_1, S_2, S_3 \ldots S_K$
$\boxed{C_i} = \frac{1}{n} \sum_{x_j \in S_i} x_j$ ← mean-point to S_i

K – Means clustering is the centroid based scheme.
There are other clustering called

Heriarical based clsutersing
DBSCAN.

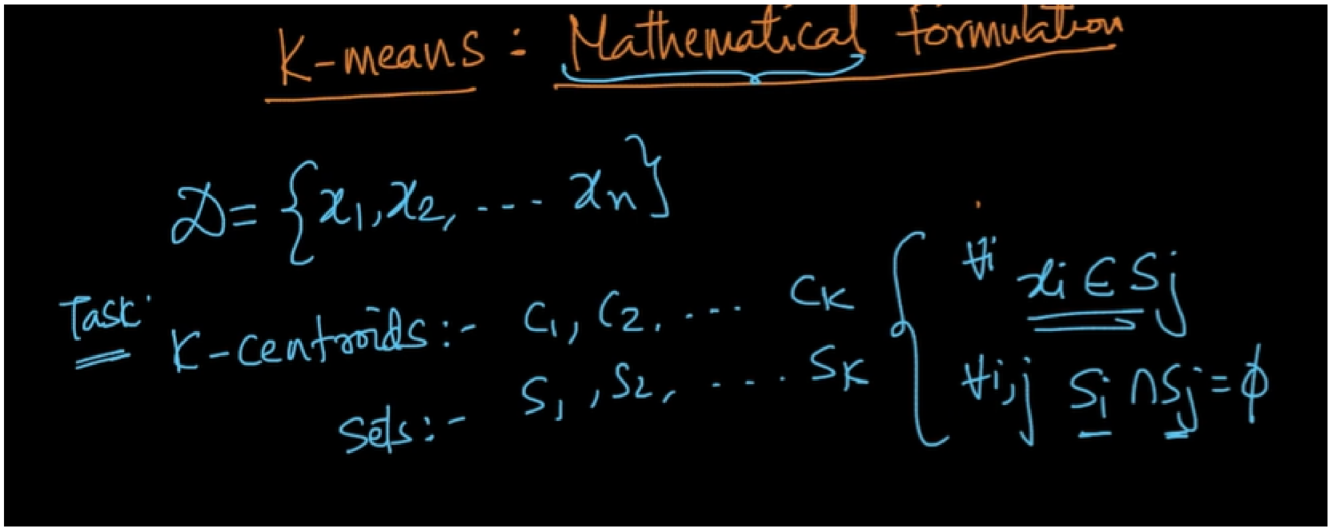


The challenge is how to find the K – Centroids. Once we get the K-centroids, we can compute the K-set nearest centroid. There are algos to find the K centroids.



To find the K central points and assign those points to the cluster.

K – Means: Mathematical formulation: Objective function



K-means : Mathematical formulation

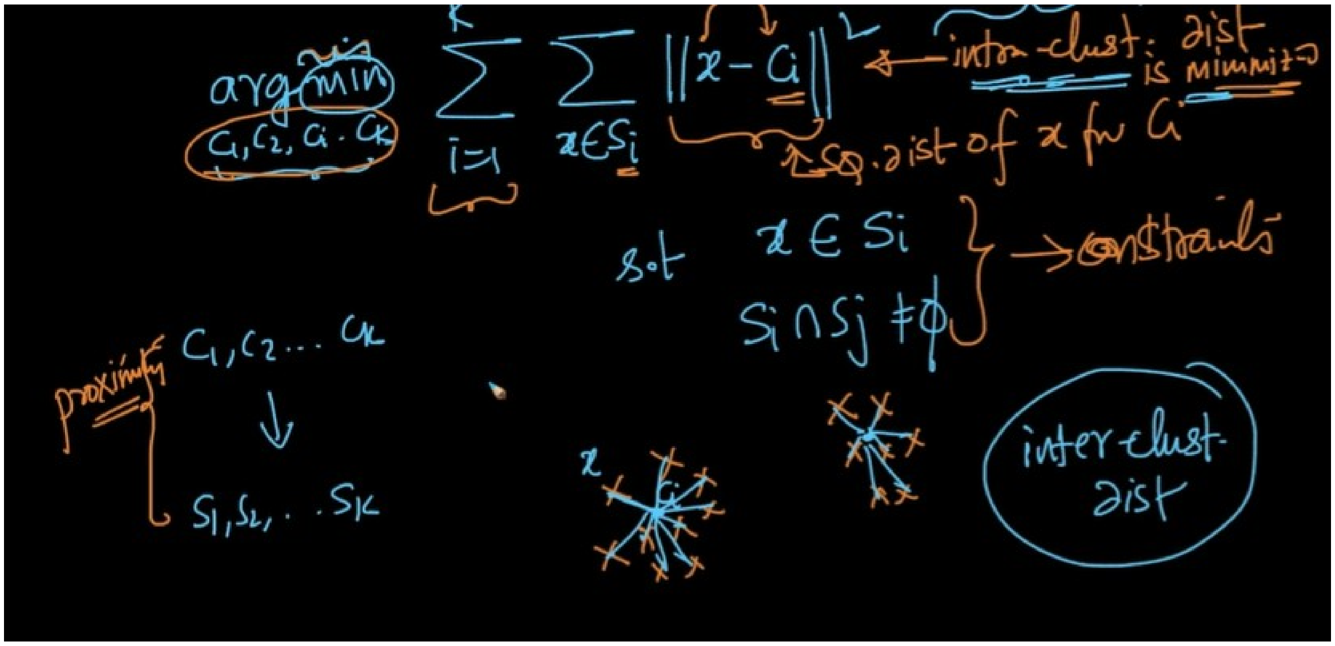
$\mathcal{D} = \{x_1, x_2, \dots x_n\}$

Task: K-centroids :- $c_1, c_2, \dots c_k$

sets :- $S_1, S_2, \dots S_k$

$\left\{ \begin{array}{l} \forall i \;\; x_i \in S_j \\ \forall i,j \;\; S_i \cap S_j = \phi \end{array} \right.$

This is the optimization problem -



$\underset{c_1, c_2, c_i \dots c_k}{\arg\min} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2$ ← intra-clust. dist is minimized

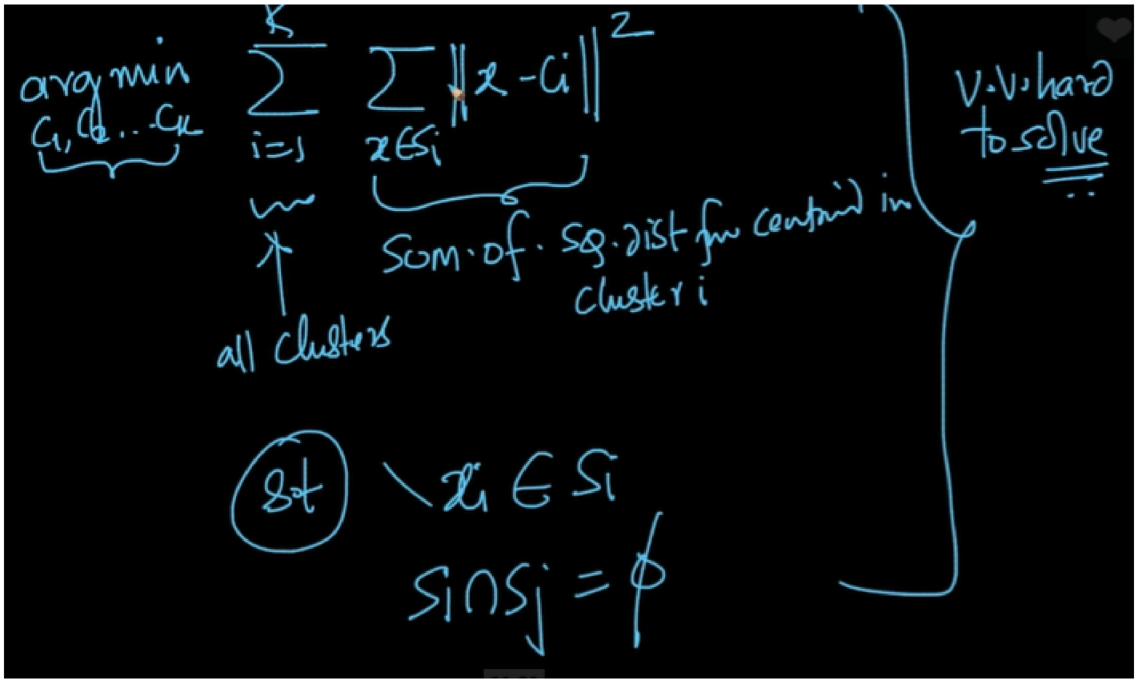
sq. dist of x fr c_i

s.t. $x \in S_i$, $S_i \cap S_j \neq \phi$ → constraints

proximity $c_1, c_2 \dots c_k$ ↓ $S_1, S_2, \dots S_k$

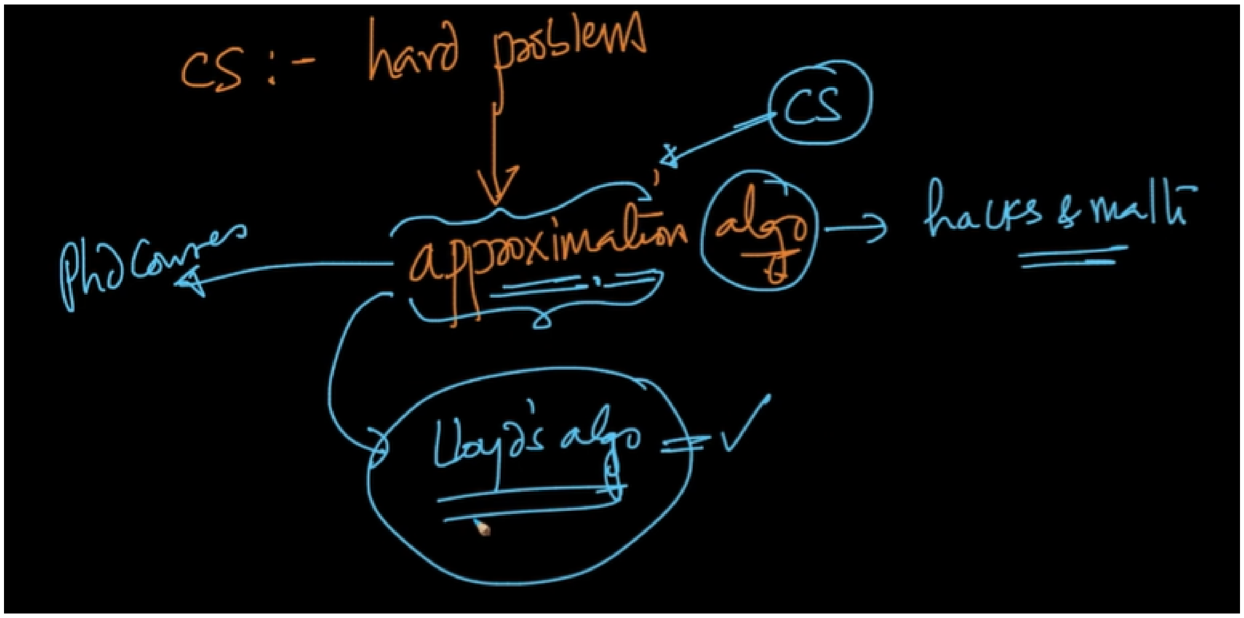
x
inter clust. dist

This is sum of squared distances in the cluster I,

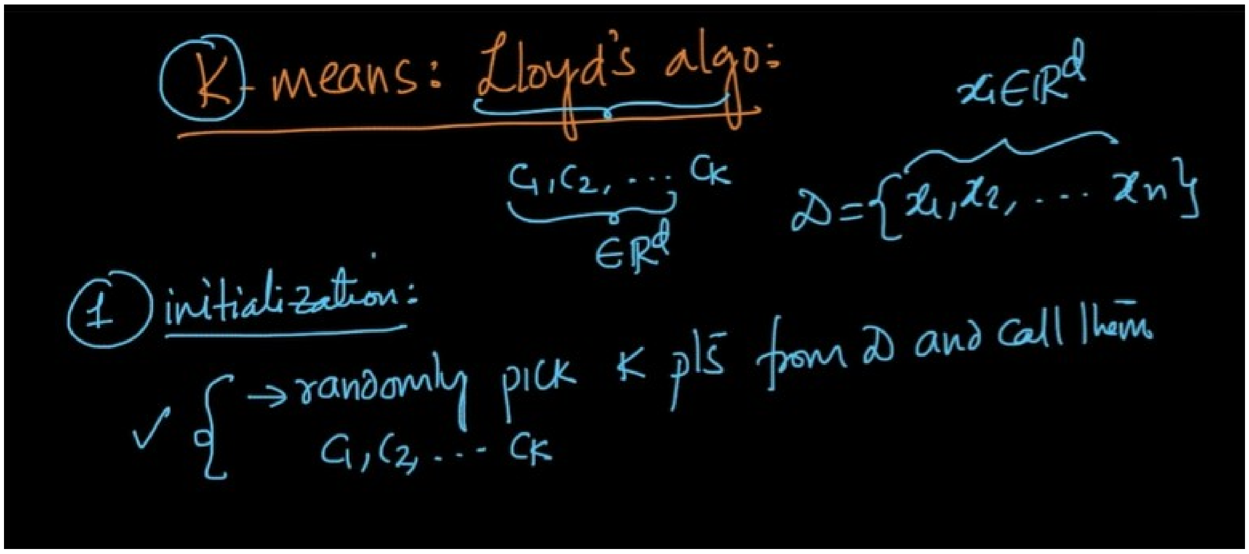


$\underset{c_1, c_2 \dots c_k}{\text{arg min}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2$
all clusters
Sum. of. sq. dist fm centroid in cluster i
st $x_i \in S_i$
$S_i \cap S_j = \phi$
V.V. hard to solve

This problem is very hard to solve. If there is a hard problem then we use the approximation of algo. Using some hacks and math.
Lloyds algorithm: This solves the problem.



Lloyd's algorithm: K-means (we find the K centroids randomly)



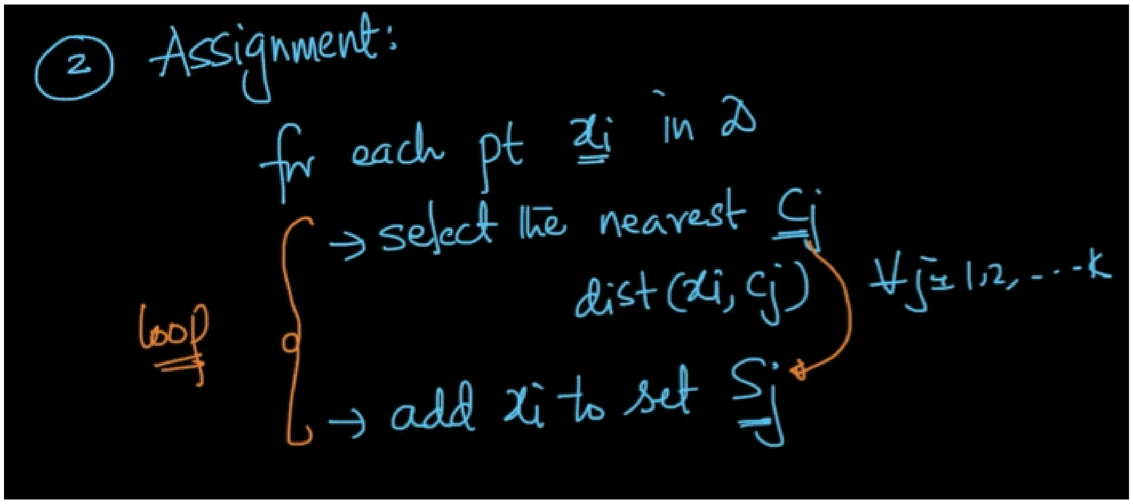
K-means: Lloyd's algo:

$\underbrace{c_1, c_2, \dots c_k}_{\in \mathbb{R}^d}$ $\quad D = \{\overbrace{x_1, x_2, \dots x_n}^{x_i \in \mathbb{R}^d}\}$

1 initialization:

✓ $\left\{ \begin{array}{l} \rightarrow \text{randomly pick } k \text{ pts from } D \text{ and call them} \\ \quad c_1, c_2, \dots c_k \end{array} \right.$

The second step is assignment -

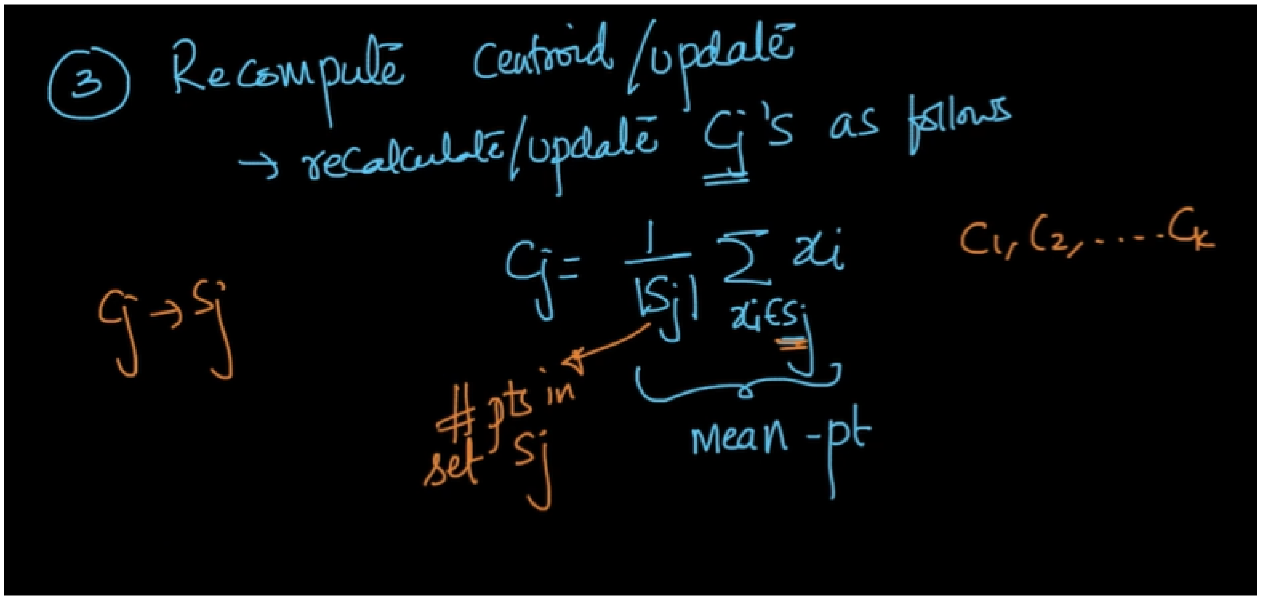


2 Assignment:

for each pt $\underline{x_i}$ in D

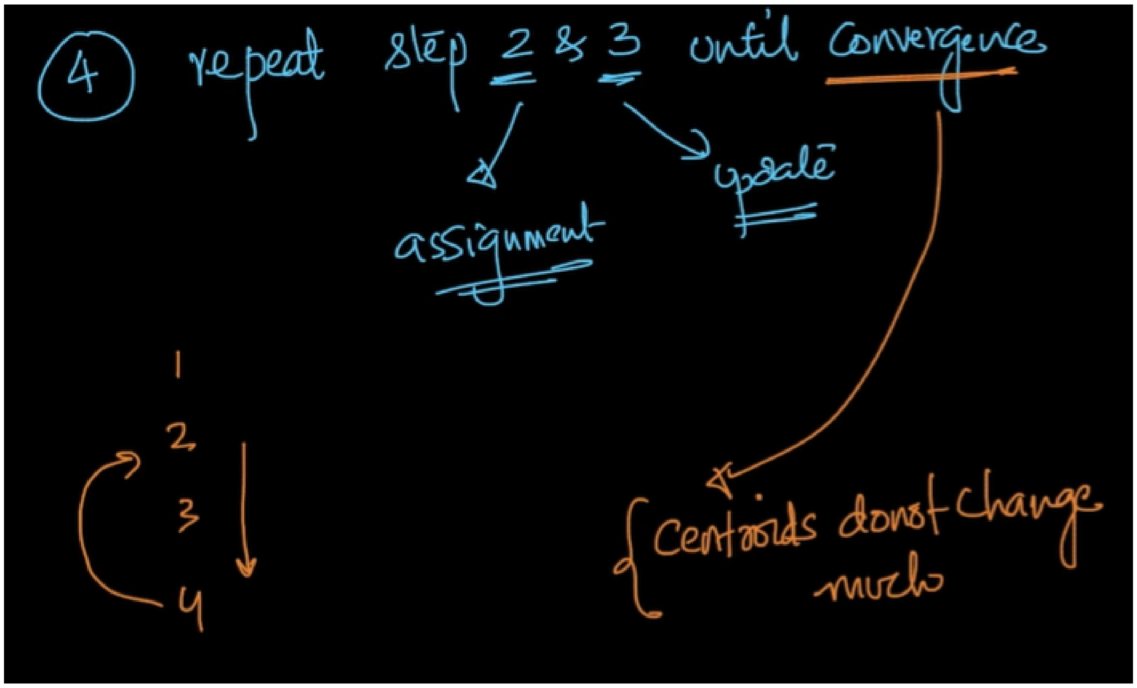
loop $\left\{ \begin{array}{l} \rightarrow \text{select the nearest } \underline{c_j} \\ \qquad \text{dist}(x_i, c_j) \quad \forall j = 1, 2, \dots k \\ \rightarrow \text{add } x_i \text{ to set } \underline{S_j} \end{array} \right.$

The third stage is called recompute centroid state: This is also called the update stage of centroids.

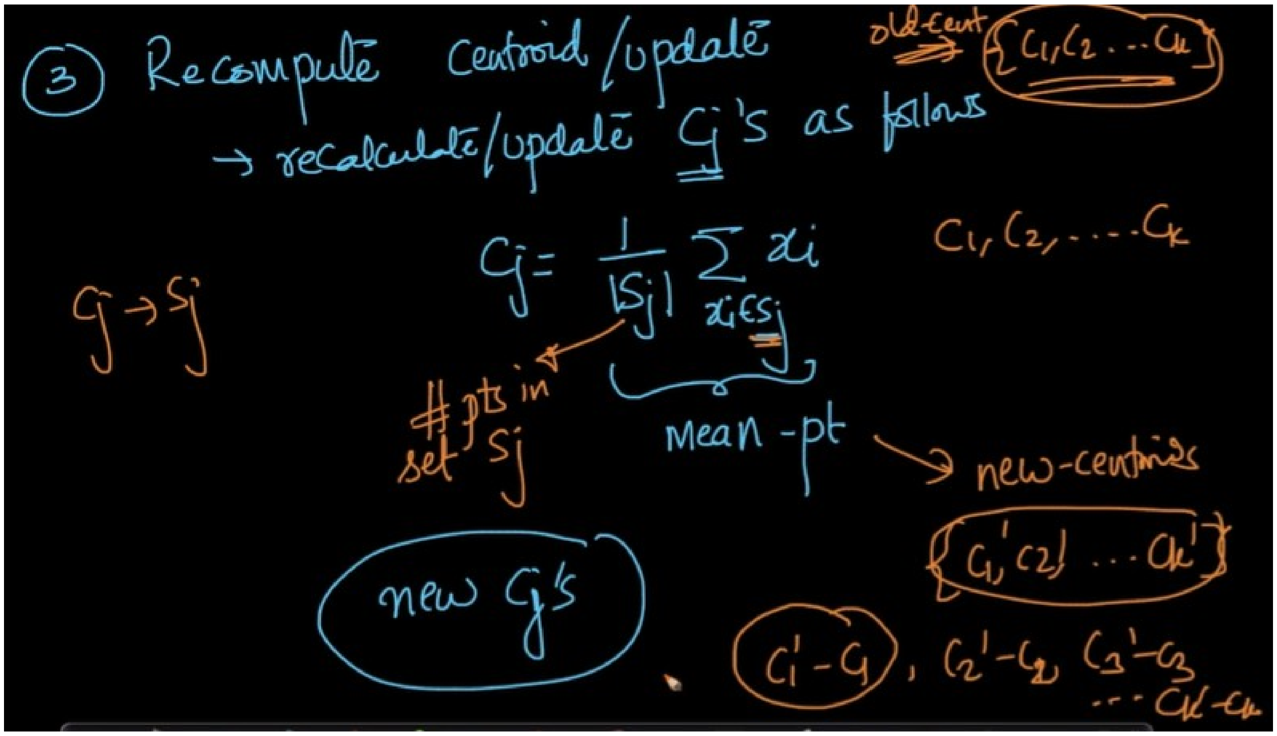

③ Recompute centroid /update
→ recalculate/update C_j's as follows
$C_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$
$C_1, C_2, \ldots C_k$
$C_j \rightarrow S_j$
# pts in set S_j
mean -pt

Step – 4:

We can repeat the step -2 and step -3, until convergence.


④ repeat step 2 & 3 until convergence
assignment
update
1
2
3
4
{centroids donot change much

Here, we have the set of centroids, of new and old.
If there are no much change in the distance of the old and new points distances.



③ Recompute centroid / update

→ recalculate/update C_j's as follows

old-cent $\rightarrow \{C_1, C_2 \ldots C_k\}$

$C_j \rightarrow S_j$

$$C_j = \frac{1}{|S_j|} \underbrace{\sum_{x_i \in S_j} x_i}_{\text{Mean-pt}} \rightarrow \text{new-centroids}$$

$C_1, C_2, \ldots C_k$

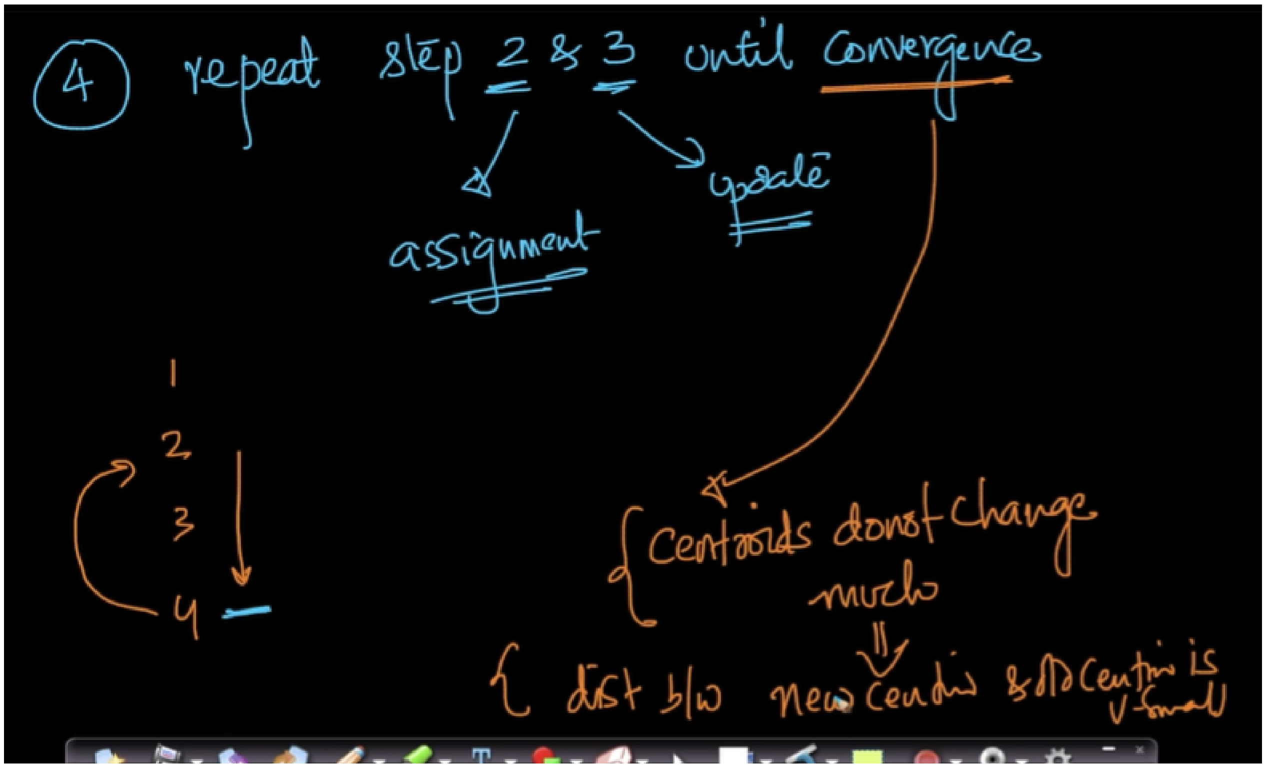
pts in set S_j

new C_j's

$\{C_1', C_2' \ldots C_k'\}$

$C_1' - C_1,\ C_2' - C_2,\ C_3' - C_3 \ldots C_k' - C_k$

Distance between new centroids and old has no change.



④ repeat step 2 & 3 until convergence

2 → assignment

3 → update

1
2
3
4

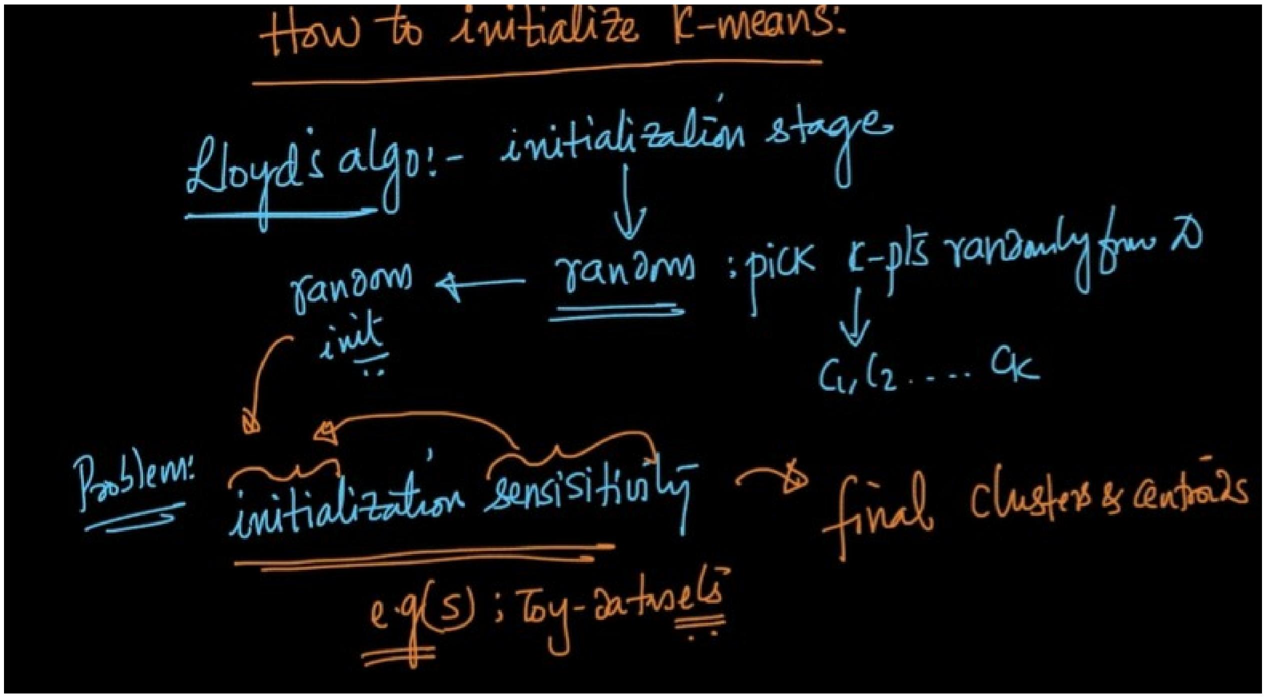
{centroids don't change much

⇓

{dist b/w new centroids & old centroids is v. small

The actual mathematical formulation is very hard. This is the lloyd algorithm.

How to initialize: K-Means++
We can do the random initialization from the dataset and make the K centroids. There is one problem, initialization sensitivity.



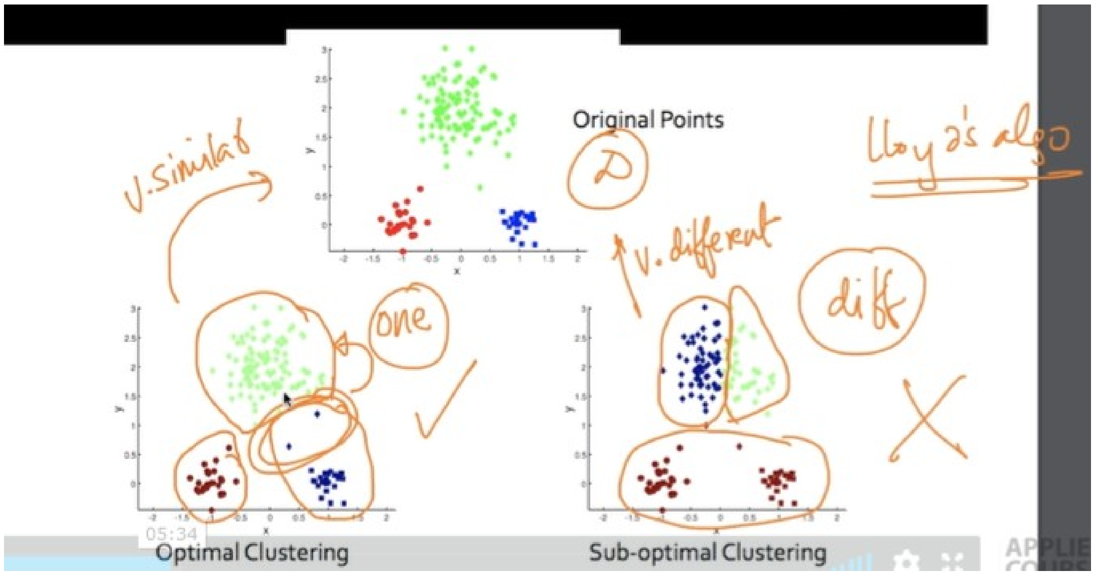
How to initialize K-means:

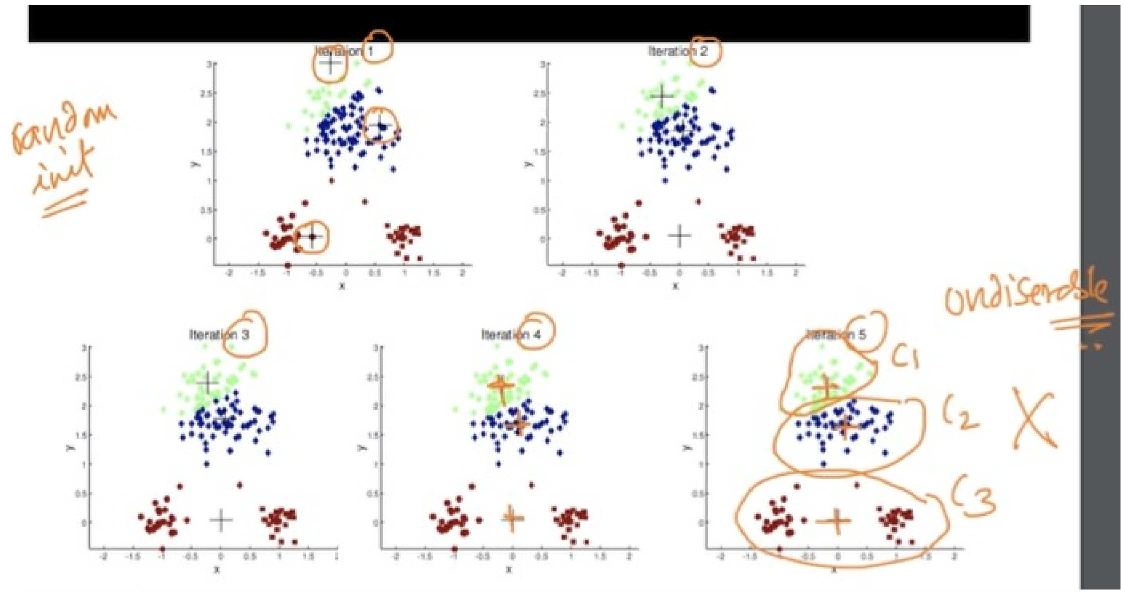
Lloyd's algo:- initialization stage

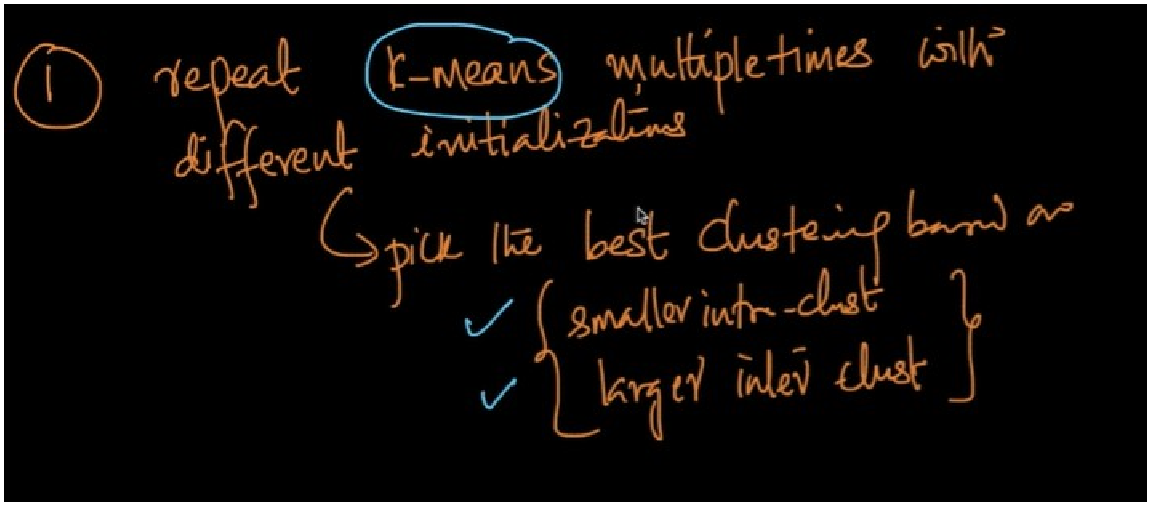
$$\begin{tikzcd}
\text{random init} \arrow[d] & \text{random} \arrow[l] & \text{: pick k-pts randomly from } D \arrow[d] \\
\text{Problem: initialization sensitivity} \arrow[rr] & & \text{final clusters \& centroids} \\
 & & c_1, c_2 \dots c_k
\end{tikzcd}$$

e.g(s): Toy-datasets

Given this ideal dataset, when we initialize the centroids. We get the optimal and sub-optmal clustering states.
Applying the same lloyd algorithm, we get the complete different solution.





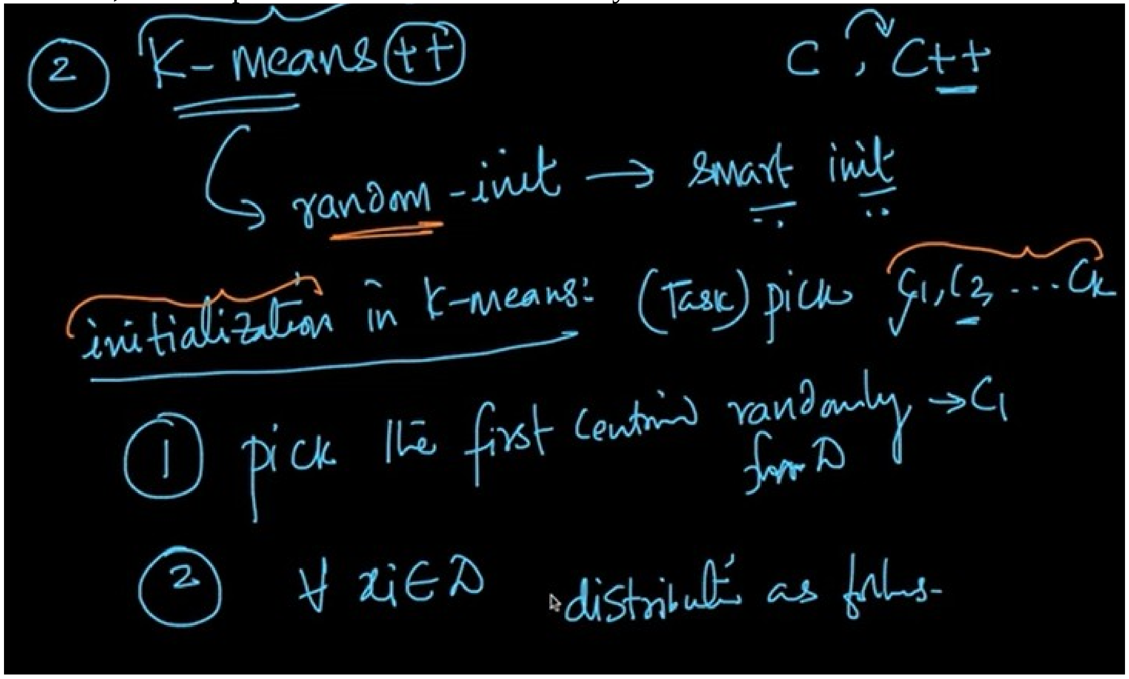


① repeat k-means multiple times with different initializations

↳ pick the best clustering based on

✓ { smaller intra-clust
✓ { larger inter clust }

The second way is K-Means ++, Instead of using the random initialization we use the smart initialization.
Step – 1:
Pick K centroids, we will pick the first cenroid randomly C1.



② K-means(++) C, C++

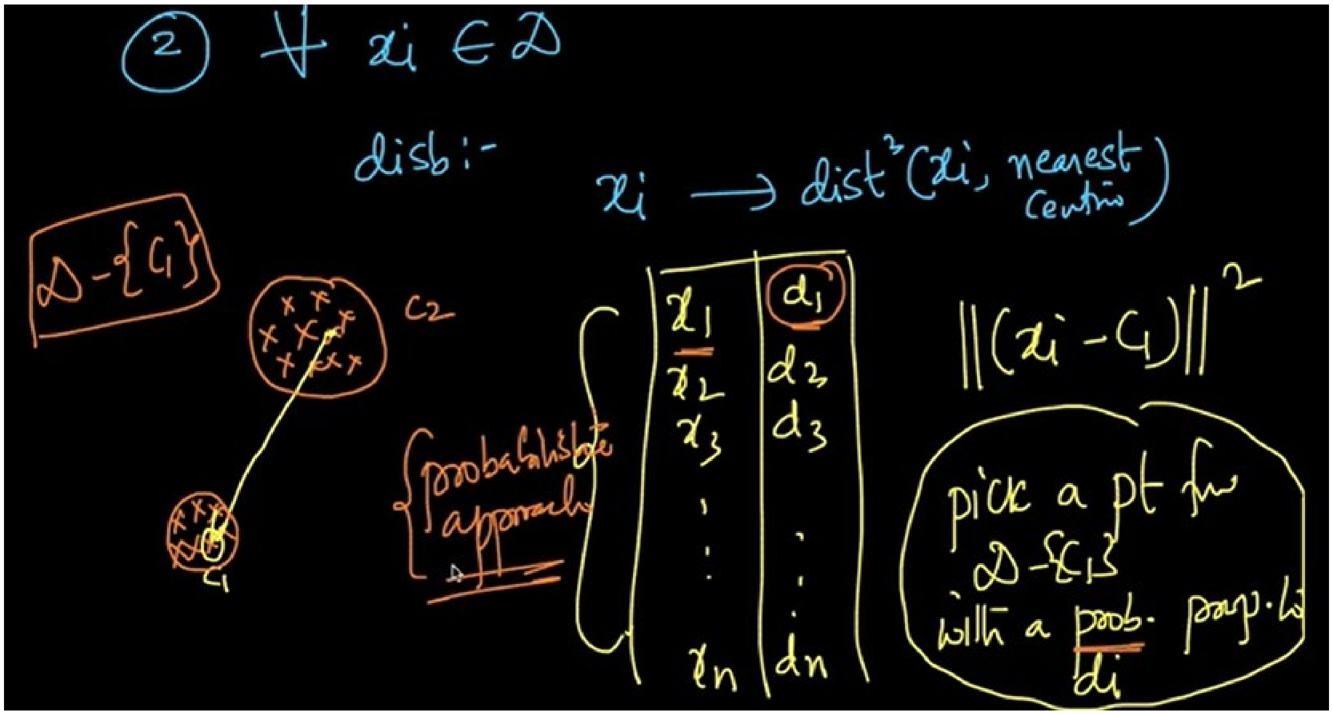
↳ random-init → smart init

initialization in K-means: (Task) pick $\{c_1, c_2 \dots c_k\}$

① pick the first centroid randomly → C_1 from D

② $\forall x_i \in D$ distribution as follows-

Step – 2:



② $\forall x_i \in D$

disb:- $x_i \rightarrow dist^2(x_i, \text{nearest centroid})$

$D - \{C_1\}$

C_2

C_1

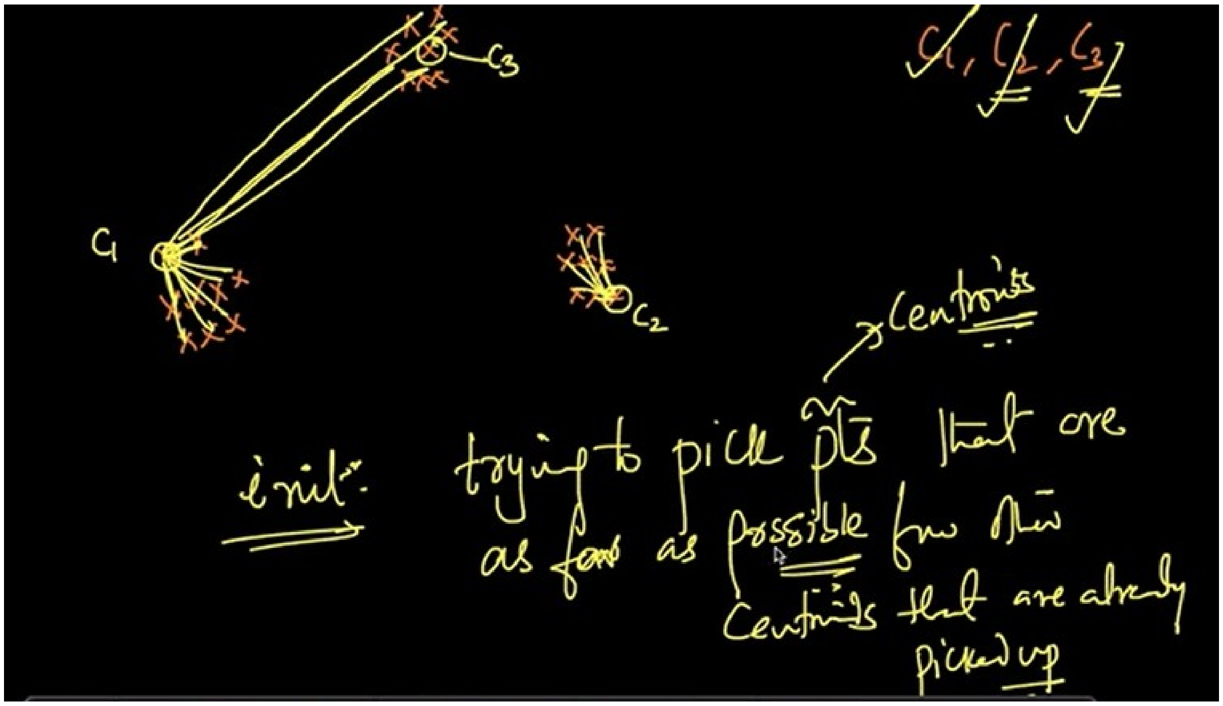
{probabilistic approach}

x_1	d_1
x_2	d_2
x_3	d_3
⋮	⋮
x_n	d_n

$\|(x_i - C_1)\|^2$

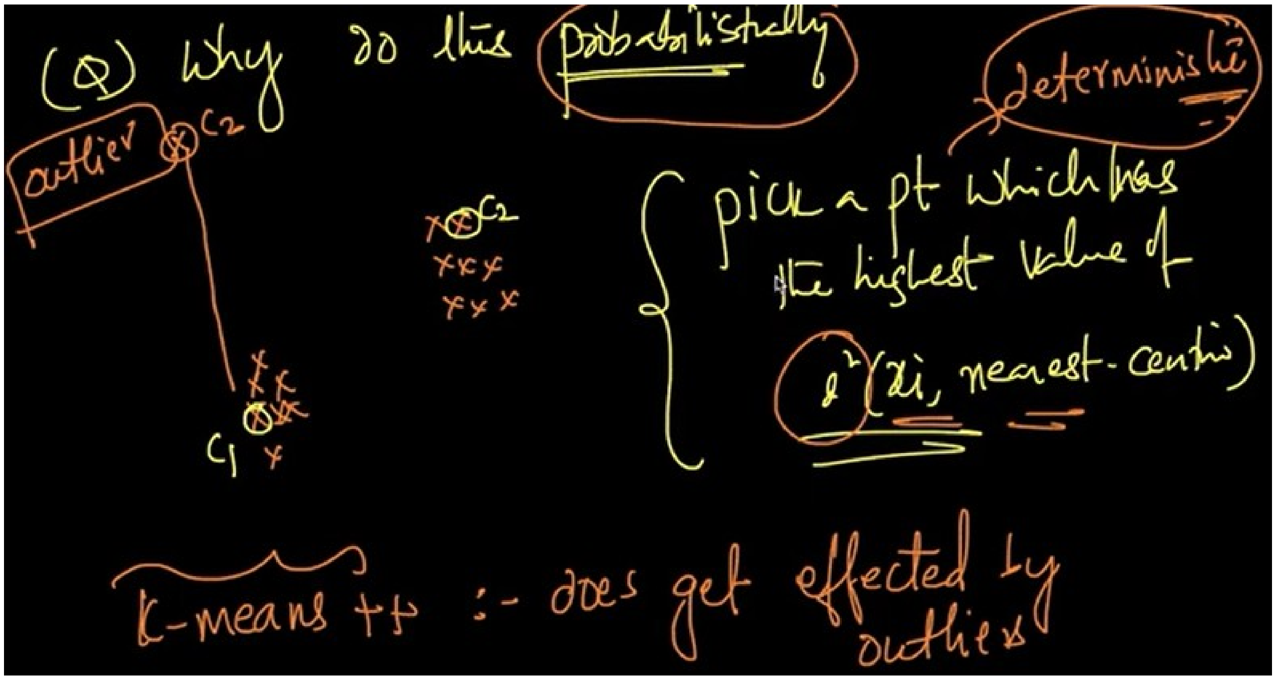
pick a pt from $D - \{C_1\}$ with a prob. prop. to d_i

The chances of the points near to the centroid is very low.
In the initialization we want to pick the points that are far as possible from other centroids picked up.


c3
c1, c2, c3
c1
c2
centroids
init:
trying to pick pts that are as far as possible from the centroids that are already picked up

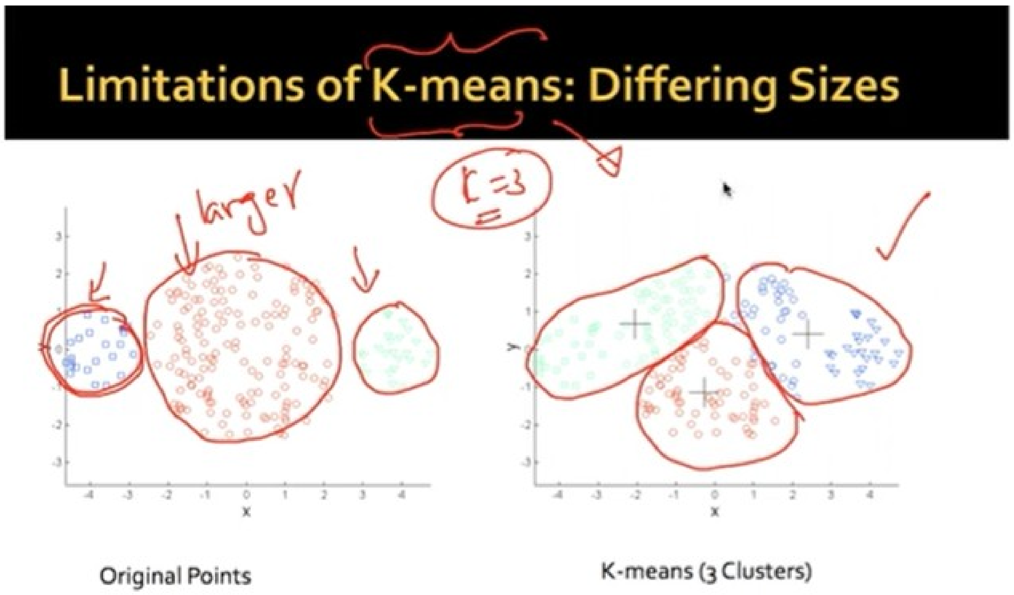
The points far to the chosen center has more probability to be the centroid.
We are trying to pick the points as far as possible that are already picked up.
Why cant I just pick the point that has highest value from the nearest center. We can have outliers.

We will pick the outliers as the centroid. K – Means ++ does get affected by outliers.


(Q) Why do this probabilistically
deterministic
outlier
c2
c2
c1
pick a pt which has the highest value of $d^2(x_i, \text{nearest-centroid})$
K-means ++ :- does get effected by outliers

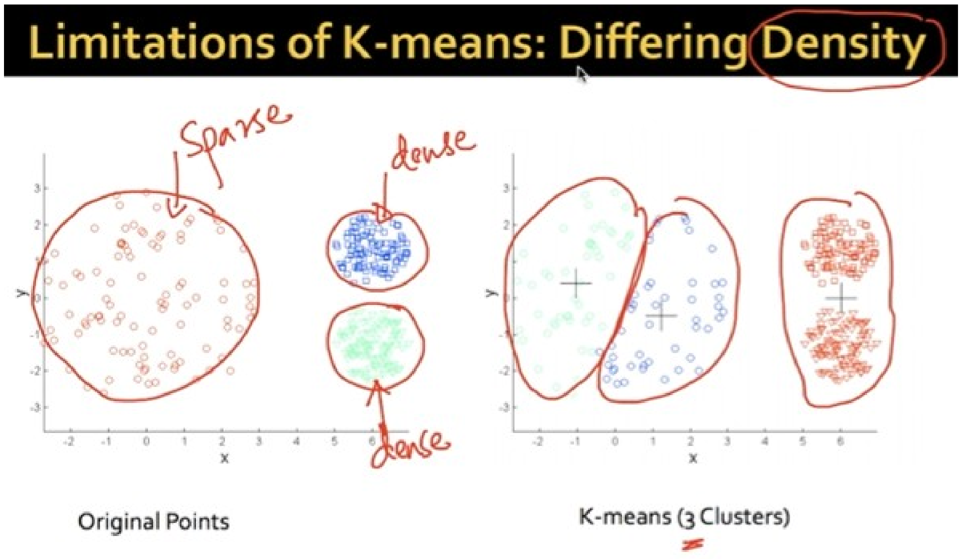
That is the whole reason we do it probabilistic-ally. To reduce the mitigation of being an outlier.

Failure cases/Limitations:
When we have clusters of different sizes, different Densities and Non – gobular shapes.


Limitations of K-means: Differing Sizes

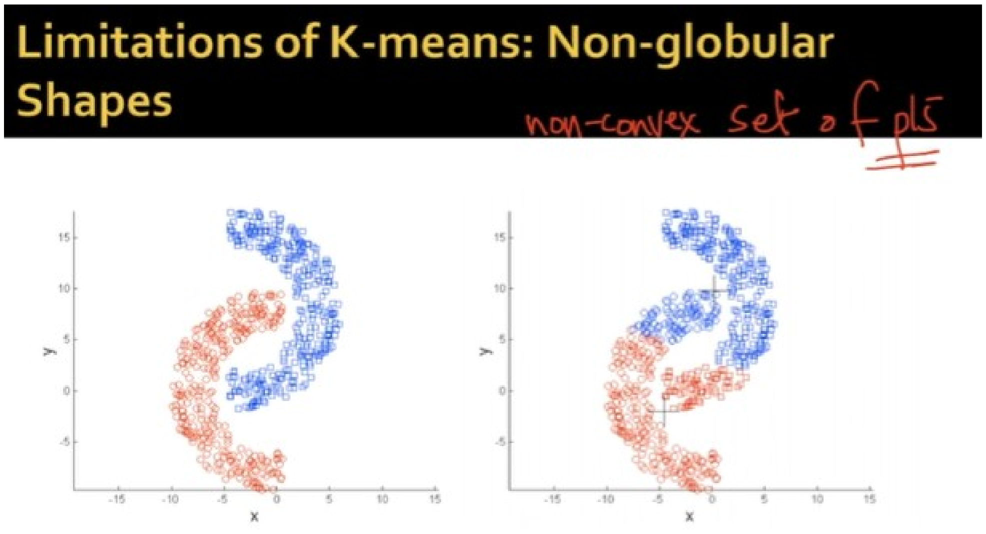
Different densities:

K means tend to fail in case of different densities.


Limitations of K-means: Differing Density

Non – gobular shapes:

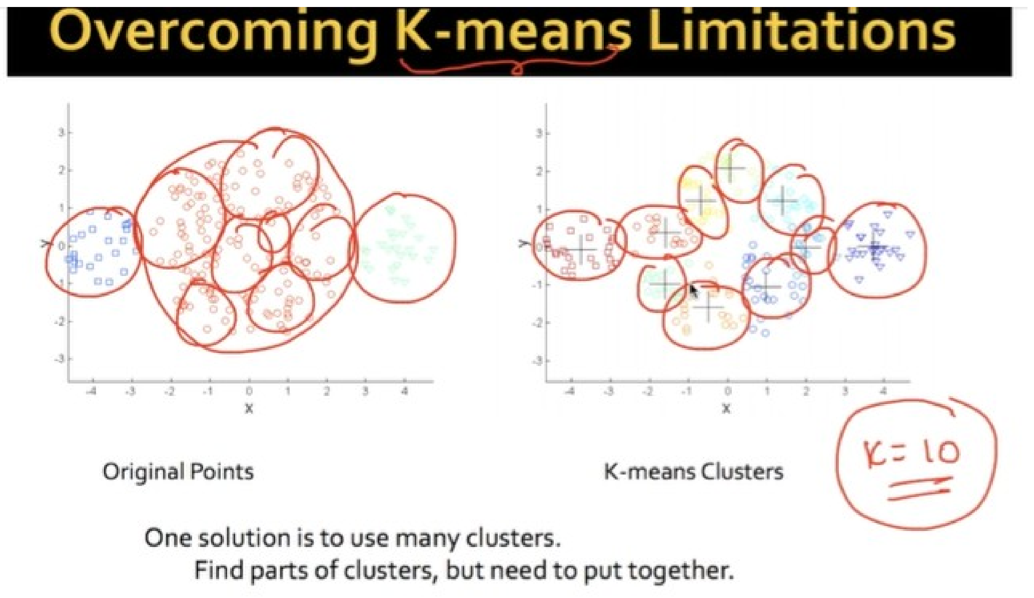
If we give this data to the K – Means, we cannot work with gobular data.



Limitations of K-means: Non-globular Shapes

non-convex set of pts

If we are continuing this K means then, if we keep k = 10, w get clusters like as follows:

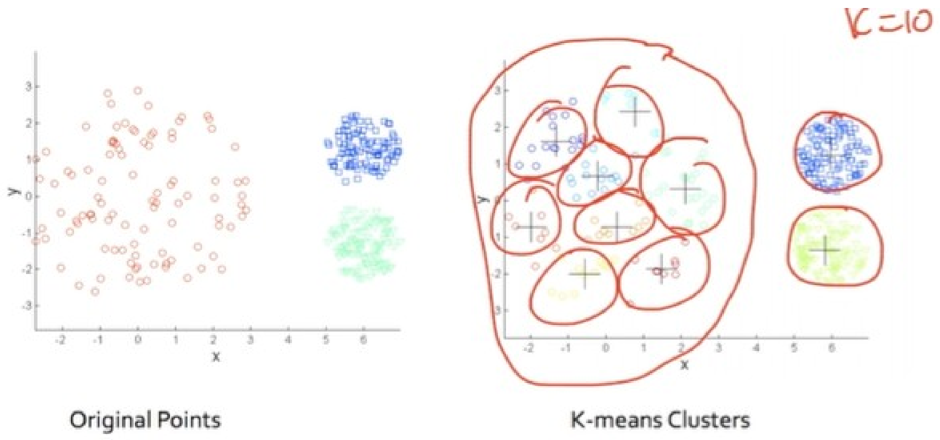


Overcoming K-means Limitations

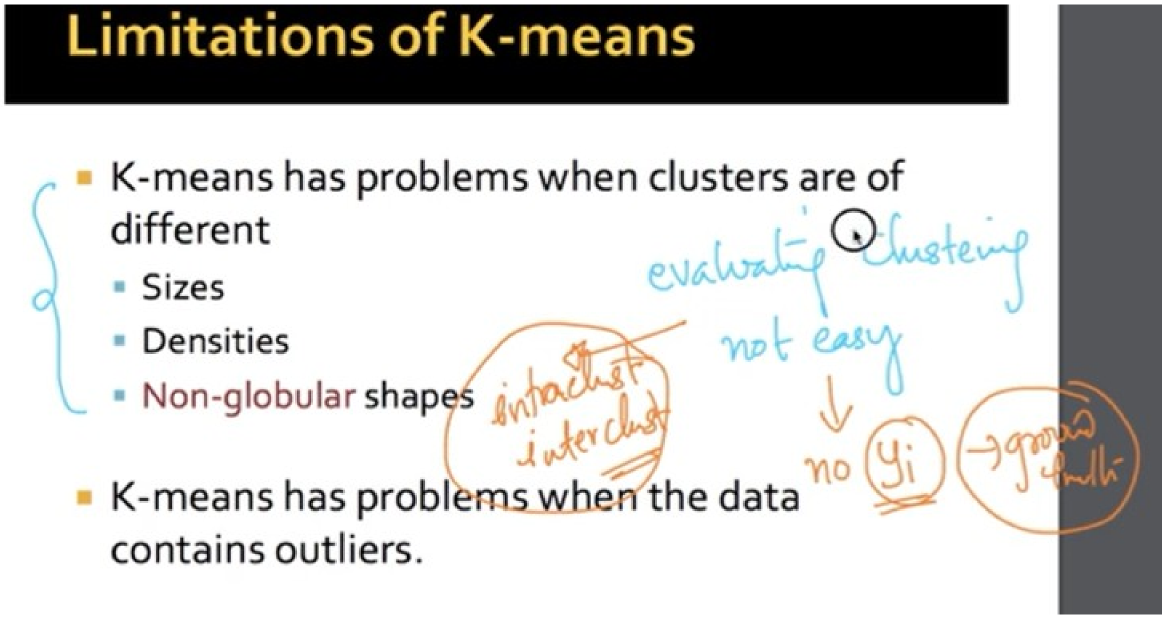
One solution is to use many clusters.
Find parts of clusters, but need to put together.

TO find the parts of cluster, but need to put together.

Clusters of different densities:



K means is never perfect algorithm.
In case of non – gobular structures, this is the best solution.
Evaluating clustering is not easy, there is no ground truth.

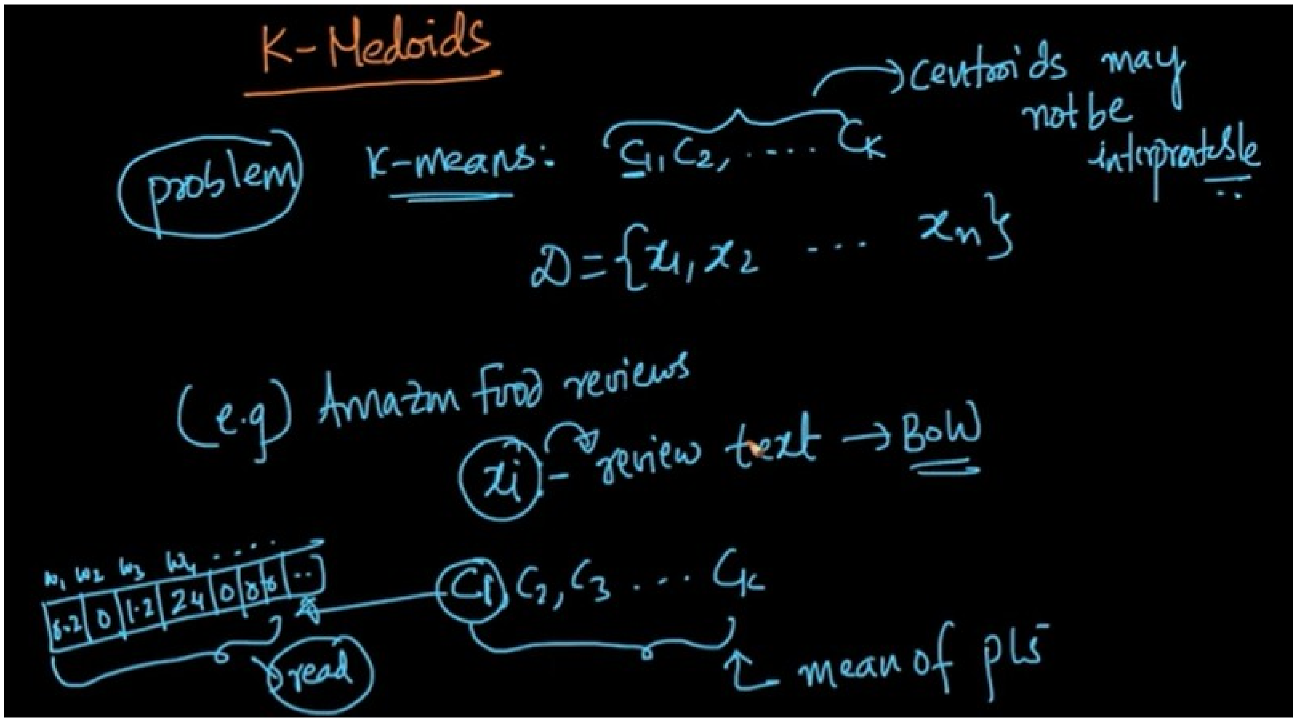


Limitations of K-means

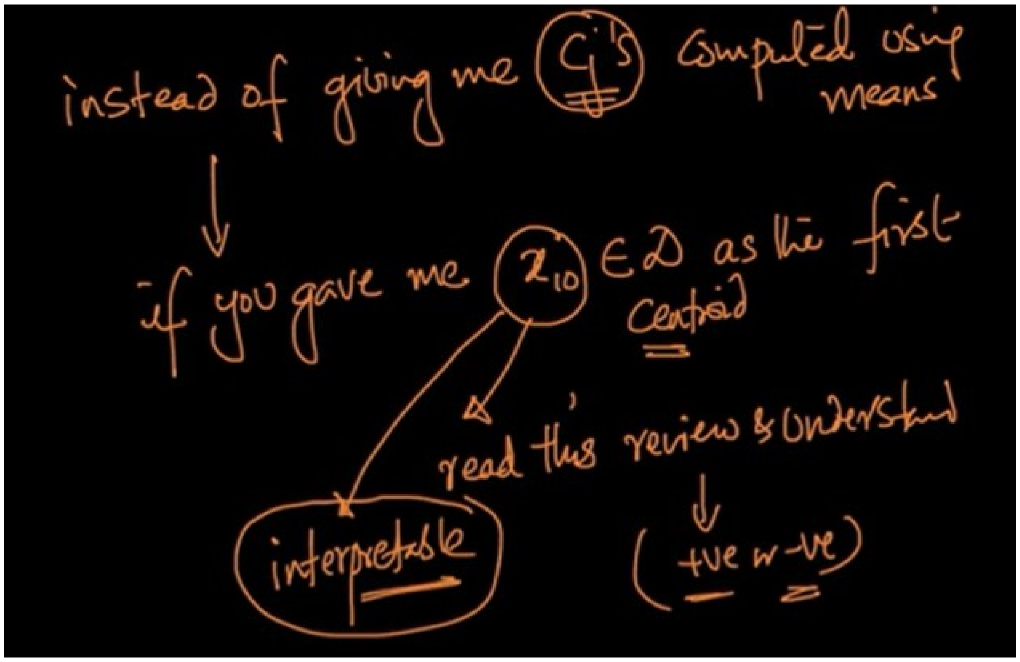
- K-means has problems when clusters are of different
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has problems when the data contains outliers.

In the case of classification and regression there is no uncertainty in the results obtained.

K – Mediods: The centroids may not be interpret-able.


K-Medoids
centroids may not be interpretable
problem
K-means: $c_1, c_2, \dots c_k$
$D = \{x_1, x_2 \dots x_n\}$
(e.g) Amazon food reviews
x_i - review text → BoW
w_1 w_2 w_3 w_4 ....
6.2 | 0 | 1.2 | 24 | 0 | 8 | 6 | ..
read
$c_1, c_2, c_3 \dots c_k$
mean of pts

That means the centroids may be not interpretable in case of K-medoids.

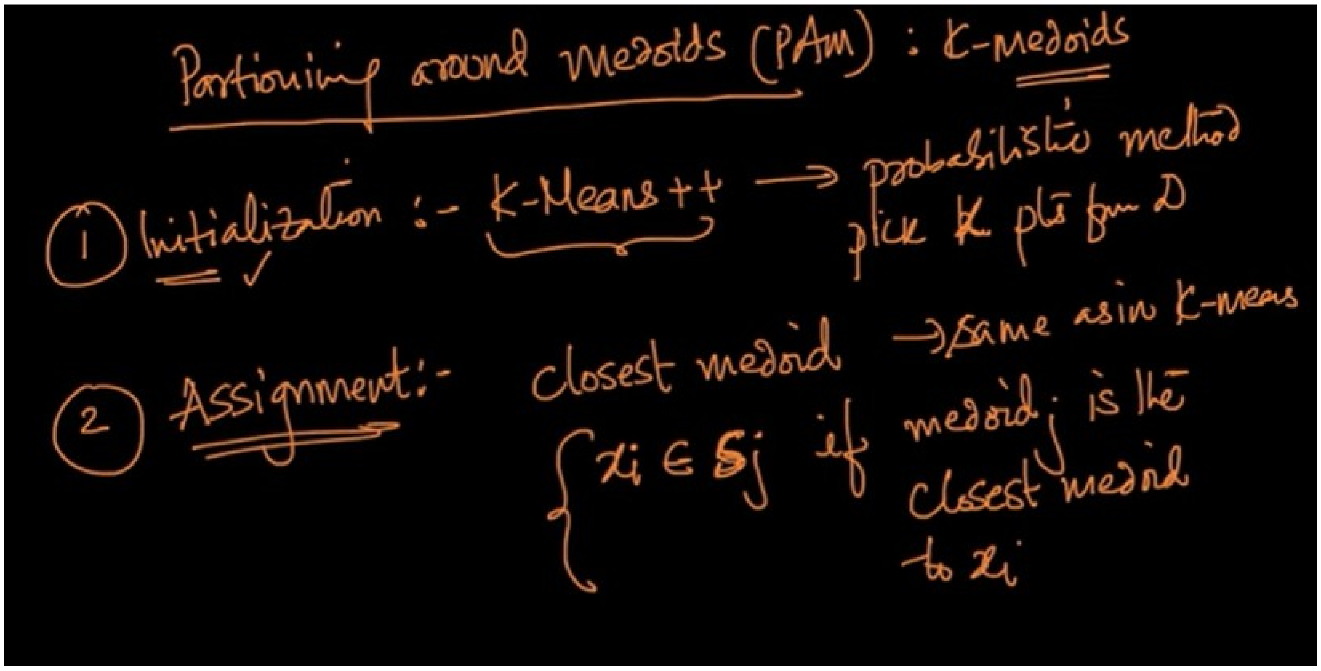

instead of giving me c_j's computed using means
if you gave me $x_{10} \in D$ as the first centroid
read this review & understand
interpretable
(+ve or -ve)

When we represent the centroid as the data point in the dataset. Then it is called K-Medoids.

Especially in case of interpretation.

K – Medoids:

Partioning around mediods(PAM) algorithm:



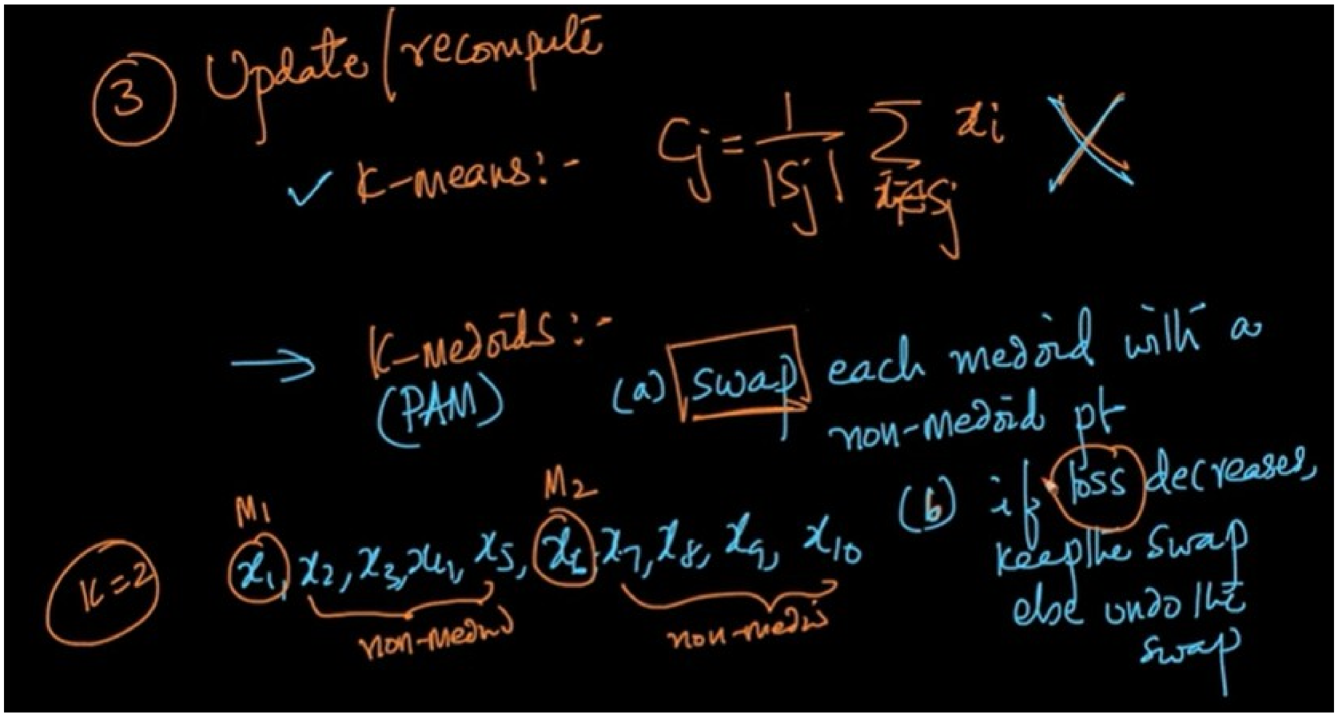
Partioning around Medoids (PAM) : K-medoids

① Initialization :- $\underbrace{\text{K-Means++}}$ $\rightarrow$ probabilistic method pick K pts from D

② Assignment:- closest medoid $\rightarrow$ same as in K-means

$\{ x_i \in S_j$ if medoid$_j$ is the closest medoid to x_i

We swap each medoid with a non-mediod point in the dataset.



③ Update/recompute

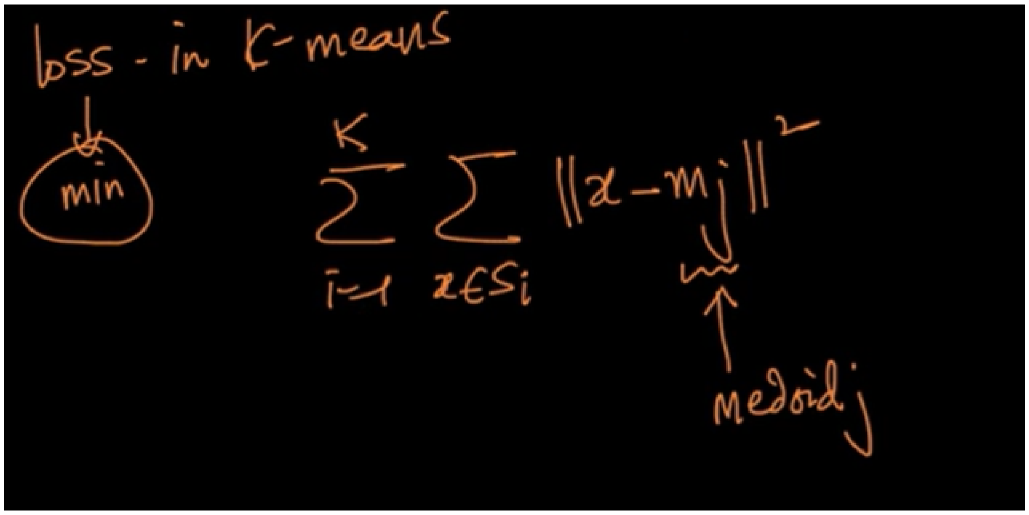
✓ K-means:- $C_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$ ✗

$\rightarrow$ K-Medoids:- (PAM) (a) $\boxed{\text{swap}}$ each medoid with a non-medoid pt

(b) if loss decreases, keep the swap else undo the swap

$\boxed{K=2}$ $\overset{M_1}{x_1}, \underbrace{x_2, x_3, x_4, x_5}_{\text{non-medoid}}, \overset{M_2}{x_6}, \underbrace{x_7, x_8, x_9, x_{10}}_{\text{non-medoids}}$

What is loss in K-Means?

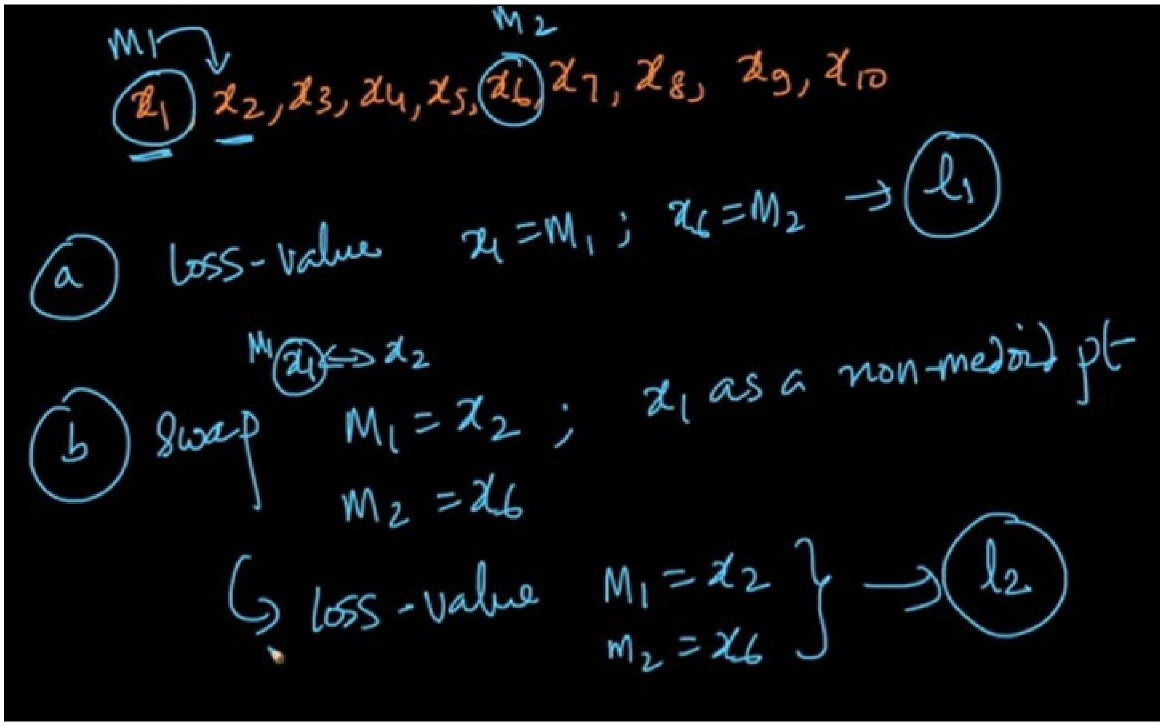


loss - in K-means

min

$$\sum_{i=1}^{K} \sum_{x \in S_i} \|x - \underbrace{m_j}_{\text{medoid } j}\|^2$$

After swapping again compute the loss values.



$$\overset{M_1}{x_1}, x_2, x_3, x_4, x_5, \overset{M_2}{x_6}, x_7, x_8, x_9, x_{10}$$

(a) loss-value $x_1 = M_1$; $x_6 = M_2 \rightarrow l_1$

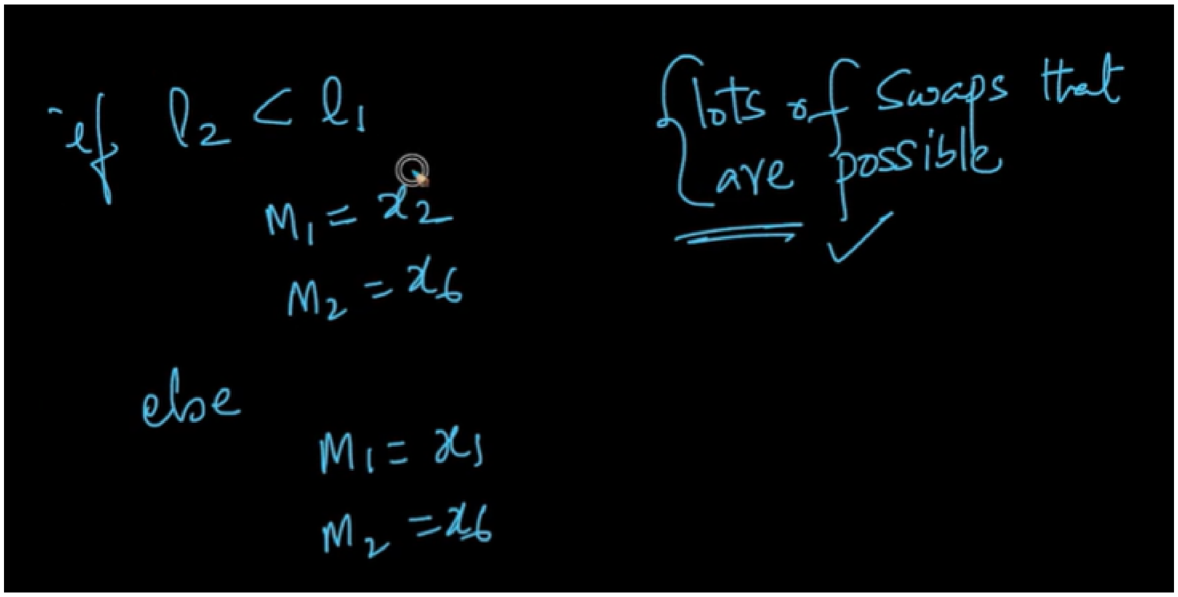
M_1 $x_1 \Leftrightarrow x_2$

(b) swap $M_1 = x_2$; x_1 as a non-medoid pt

$M_2 = x_6$

↳ loss-value $\left.\begin{matrix} M_1 = x_2 \\ M_2 = x_6 \end{matrix}\right\} \rightarrow l_2$

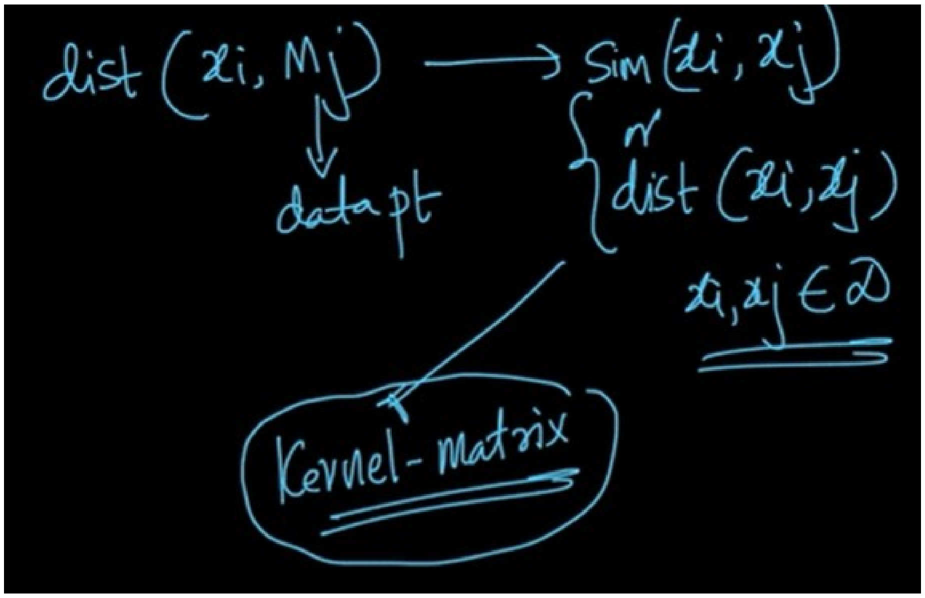
Is loss decreases by swapping then we keep the point that decreases the loss. There are lot of swaps possible.



The swap is successful, when the loss decreases. Medoid is also a data point.

We can use the kernel matrix or distance matrix and apply K – Mediods. The massive advantage is

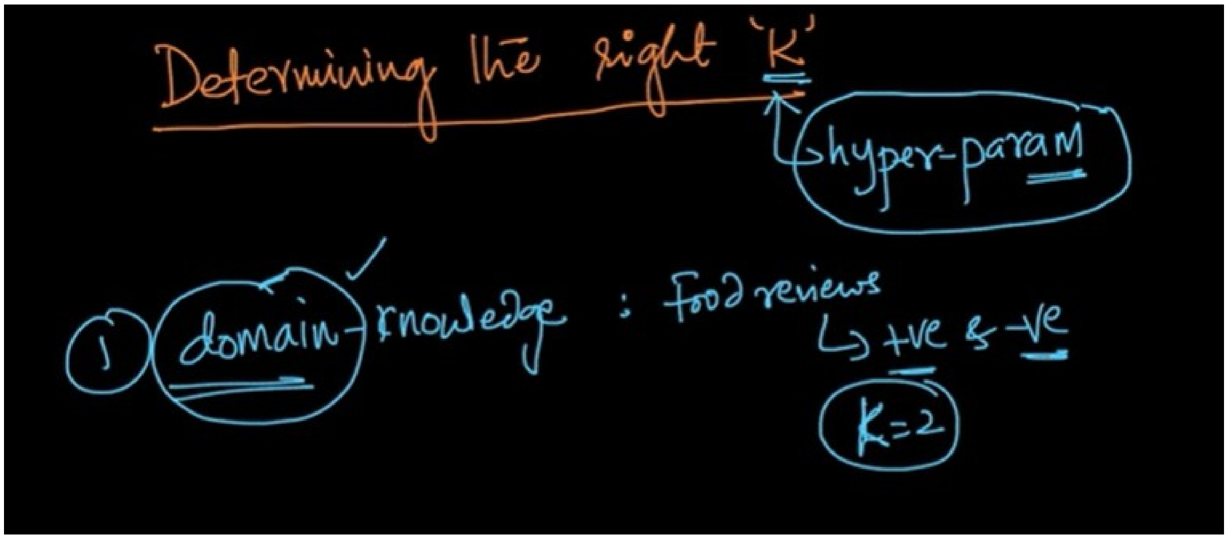
1. More interpretation.
2. Kearnelization.
It is trivially kernalizable.



Determining the right K:

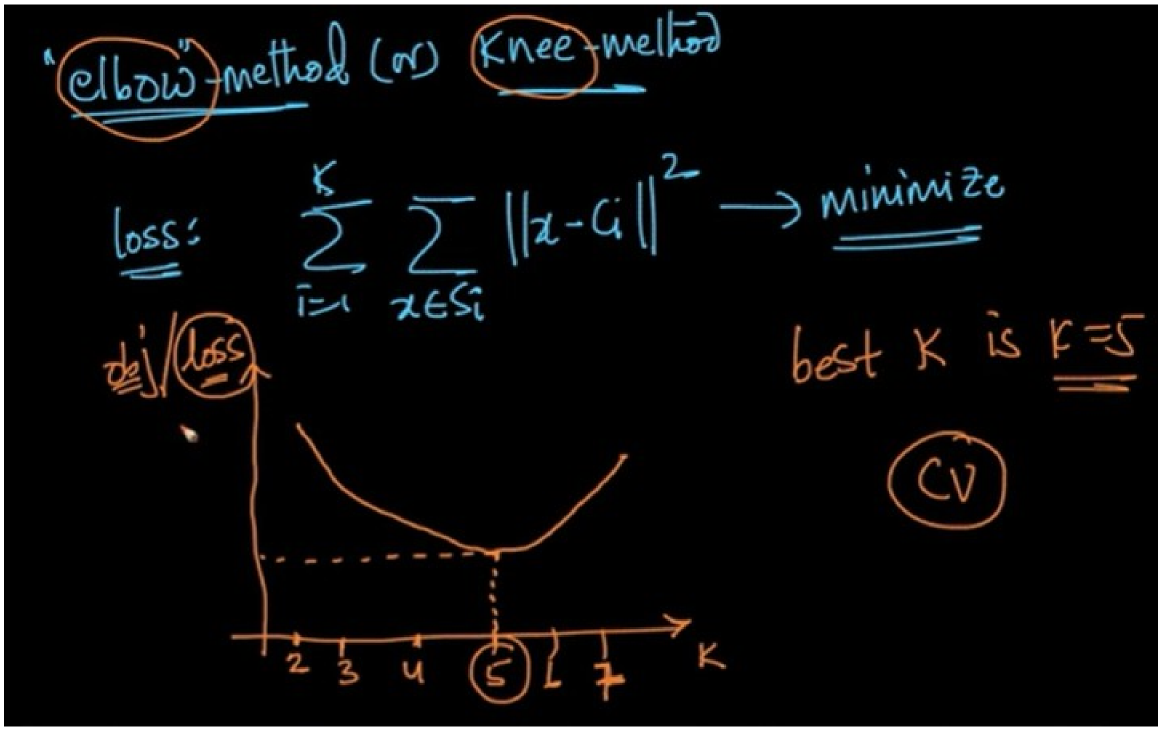
In K – means, K is the hyper parameter,

1. Domain knowledge – we can know the clusters.



Then we use the elbow-method (or) Knee-method -

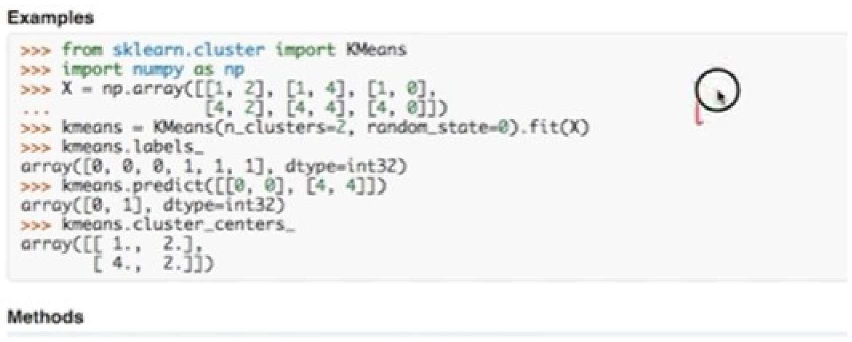
We want to minimize the loss function is k means.



We only have data D={xi}.

Code examples:

Sklearn – sklearn.cluster.Kmeans



Examples

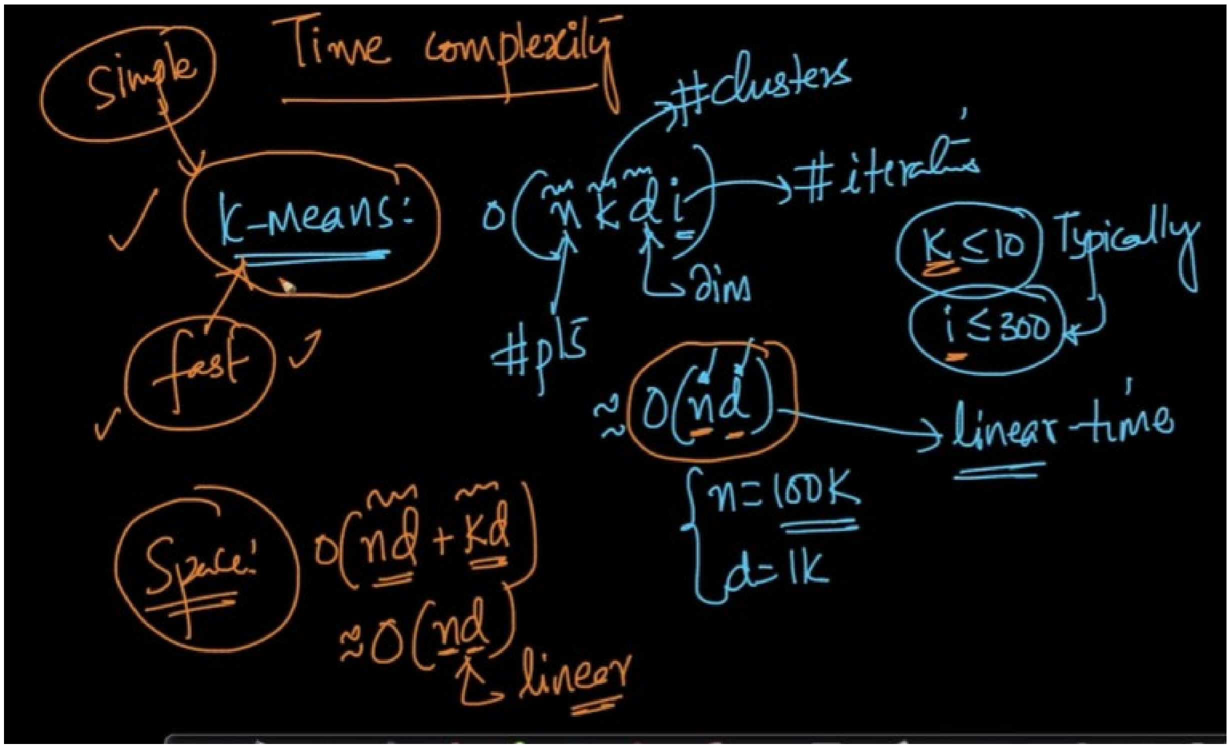
```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [4, 2], [4, 4], [4, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([0, 0, 0, 1, 1, 1], dtype=int32)
>>> kmeans.predict([[0, 0], [4, 4]])
array([0, 1], dtype=int32)
>>> kmeans.cluster_centers_
array([[ 1.,  2.],
       [ 4.,  2.]])
```

Methods

Time and space complexity:

K-Means: O(nkdi) This is still a linear in time.
As far as space concerned we need to store nd + kd. Which is also linear.

It is fast and simple to understand.



Cluster Amazon reviews: