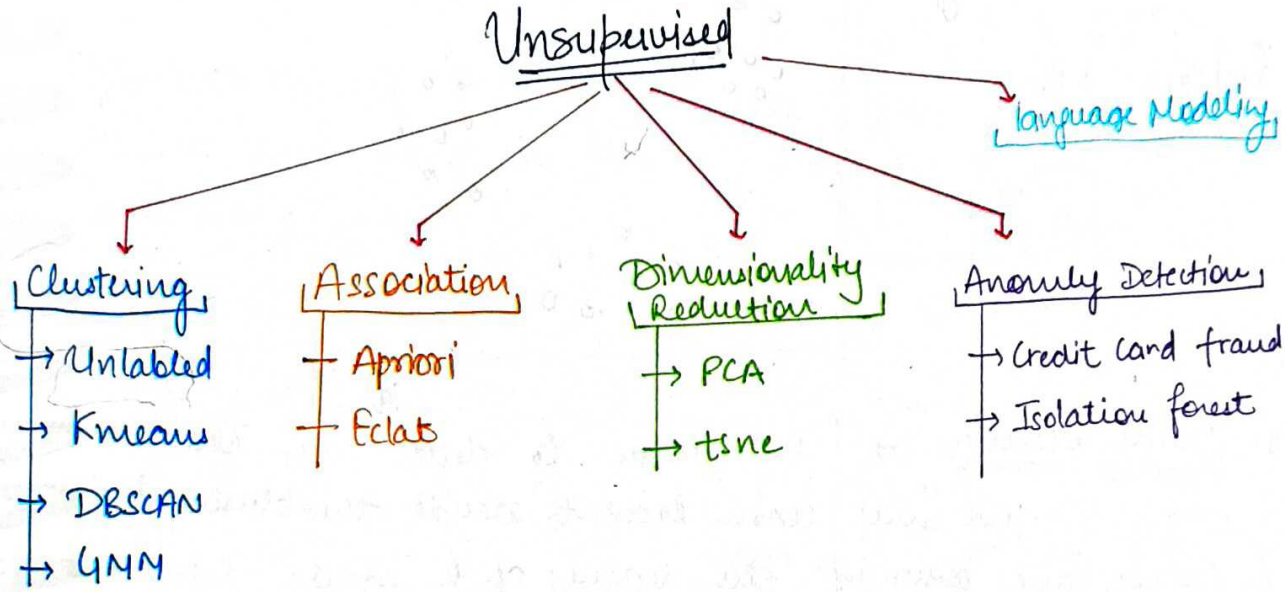


Clustering

Application of Clustering

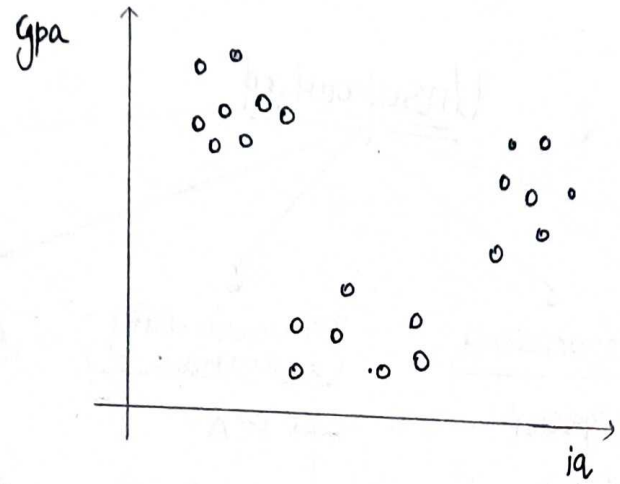
1. Customer segmentation
2. Data Analysis
3. Semi Supervised learning
4. Image Segmentation

K-Means Geometric Intution

Problem statement

1. Decide K cluster
2. Initialize centroids
3. start iterating
4. Assign clusters
   Move centroid
   Check and stop

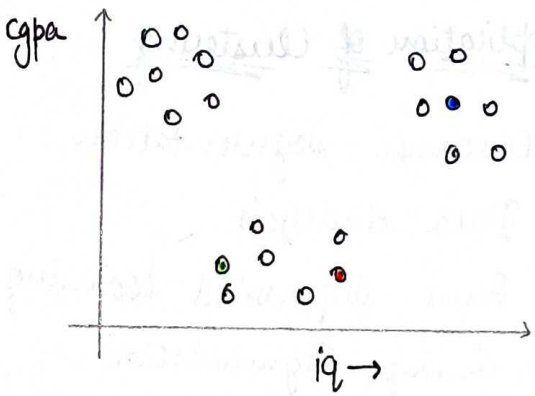
iq | Cgpa → 100



1. Decide k cluster → We have to decide the cluster. later we will learn how to decide the cluster but now, we are assuming the value of k is 3.

2. Initialize centeroids

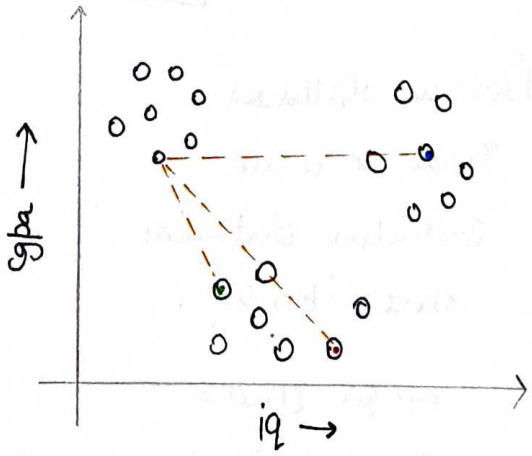
Randomly we can initialize centeroids on any 3 points.



3. Starting Iterating → (loop)

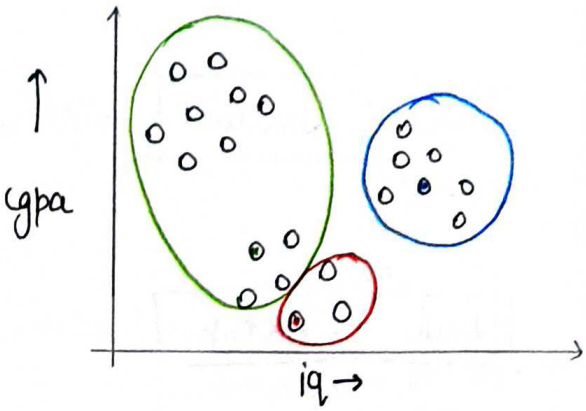
Find euclidient distance between point and centeroids.

We have to find euclidient distance of all points.



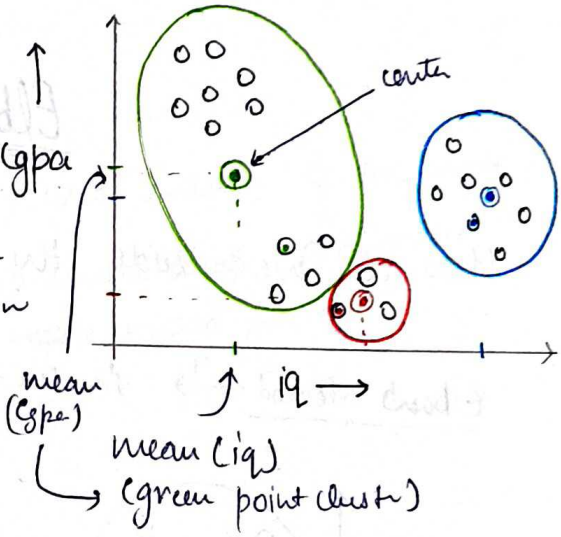
Assign Clusters

Make cluster of those points which are near to the centeroid.



Move centroid

We have to find the mean of point which are present in cluster. example:- Mean → cgpa Mean → iq, cgpa and iq → present in green cluster.

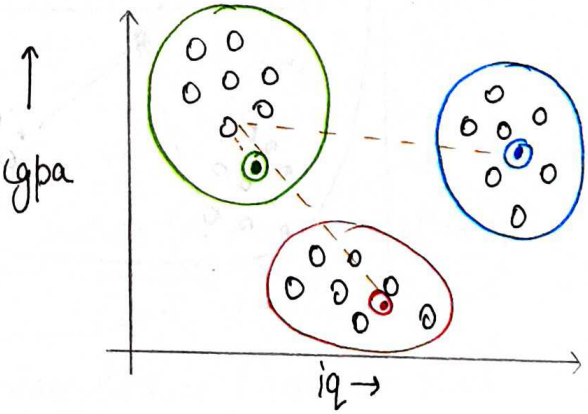


Check and stop

Condition → if previous position of centroid = current position of centroid

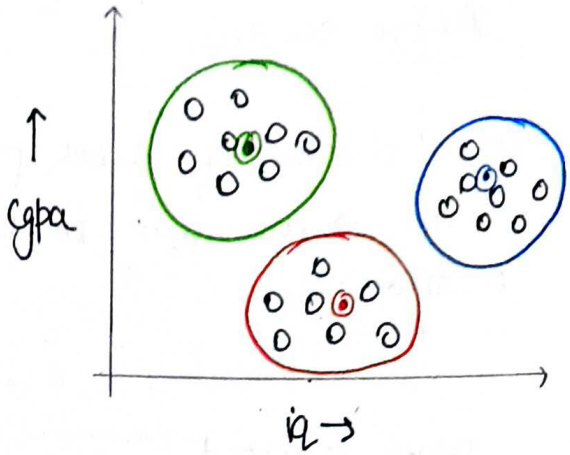
then → stop otherwise algo work in continously.

Now, Again find euclidient distance from point to centroid and again make clusturing.



Move centroid →

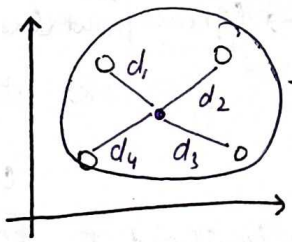
Check and stop



Elbow Method

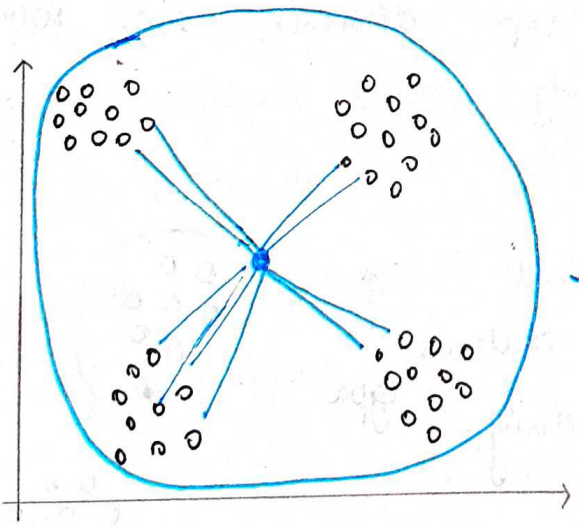
How do we decide the correct value of k?

Elbow Method → inertia → WCSS → within Cluster Sum of squared distances



→ inertia / wcss

$$WCSS = d_1^2 + d_2^2 + d_3^3 + d_4^2$$

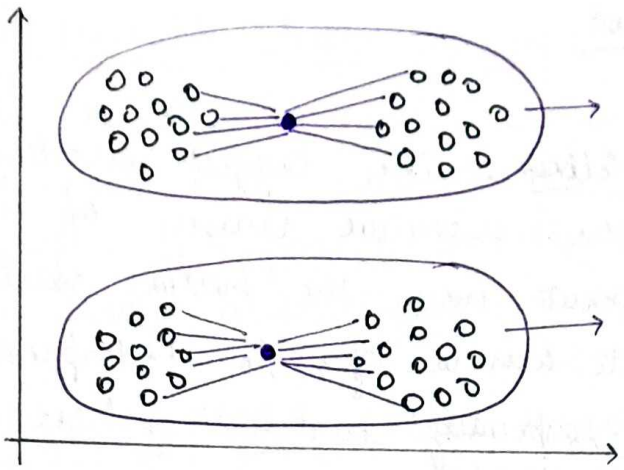


100 points

elbow method

→ Intertia (1) = X

K=1



inertia (a) → 100 points

elbow method

$K=2$

inertia (b) →

interia(2) = int(a) + int(b)

x_1 x_2

y

1) $x > y$?

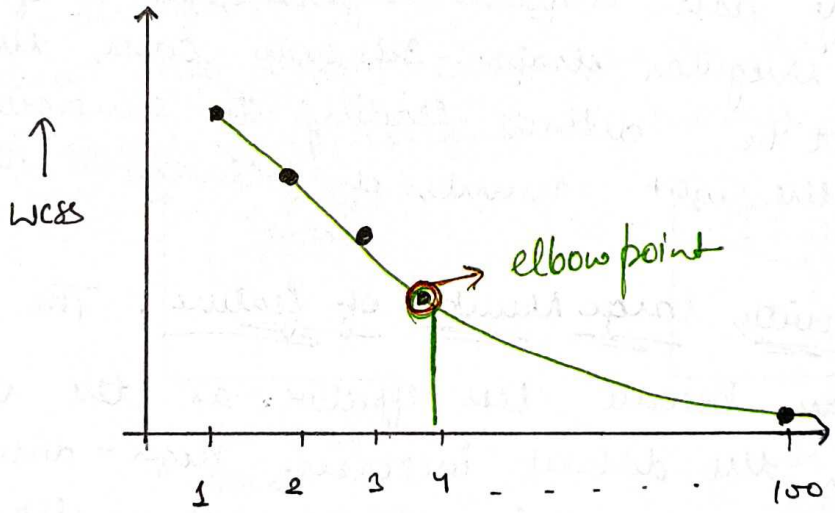
2) $y > x$?

→ If we have 100 data points and interia of 100 points is zero because 100 points → 100 centroid ⇒ 0 distance between point and centroid, so ~~radia~~ inertia is 0

$$WCSS = d_1^2 + d_2^2 + d_3^2 + \dots d_{100}^2 = 0$$

↳ $1 > 2 > 3 > 4 > 5 > \dots > 100$

↳ inertia



→ Why we Reduce WCSS?

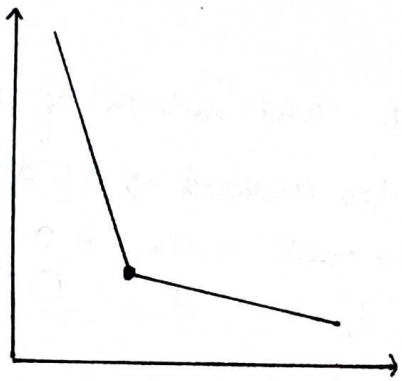
Because WCSS is like variance →

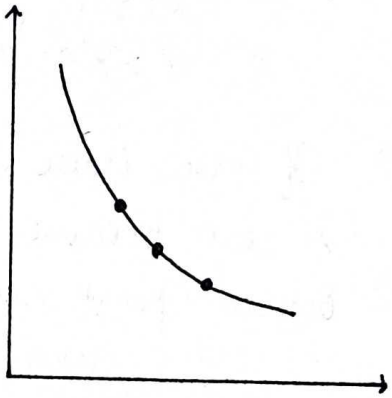
$$WCSS = \frac{d_1^2 + d_2^2 + \dots d_{100}^2}{n}$$

When WCSS Reduce → Variance also Reduce (not too much reduce) because 100 data → create 100 Cluster.

Limitation of Elbow Method

Subjectivity in identifying the Elbow: The biggest challenge with the elbow method is the subjective nature of identifying the "elbow" point. The point where the inertia starts decreasing at a slower rate can be open to interpretation and may not be clear-cut, especially in datasets where the decrease in inertia is gradual.





Not Suitable for All Dataset: The method does not work well if the data is not very clustered or if the clusters have an irregular shape. In such cases, the elbow might not be distinct leading to ambiguity in choosing the right number of clusters.

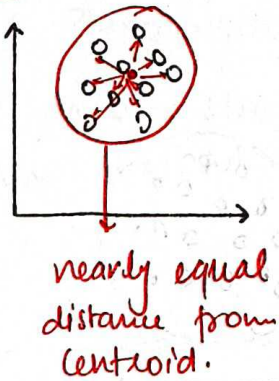
Performance with Large Number of feature: The elbow method can become less effective as the number of features in the dataset increases. High-dimensional data can make the identification of a clear elbow more difficult.

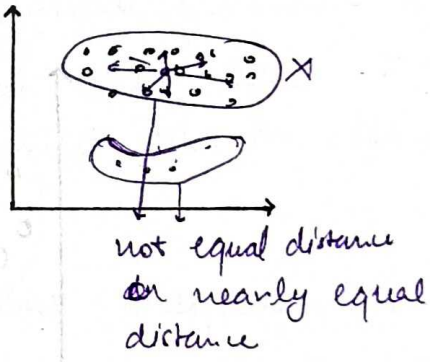
Doesn't Consider Cluster Quality: The elbow method focuses solely on the variance within the clusters and does not take into account the quality of the clusters formed. It's possible to choose a k where clusters are not meaningful or well-separated.

Sensitivity to Scaling: Like k-means clustering itself, the results of the elbow method can be sensitive to the scale of the data. Features with larger scales can dominate the results, potentially leading to suboptimal choice of k.

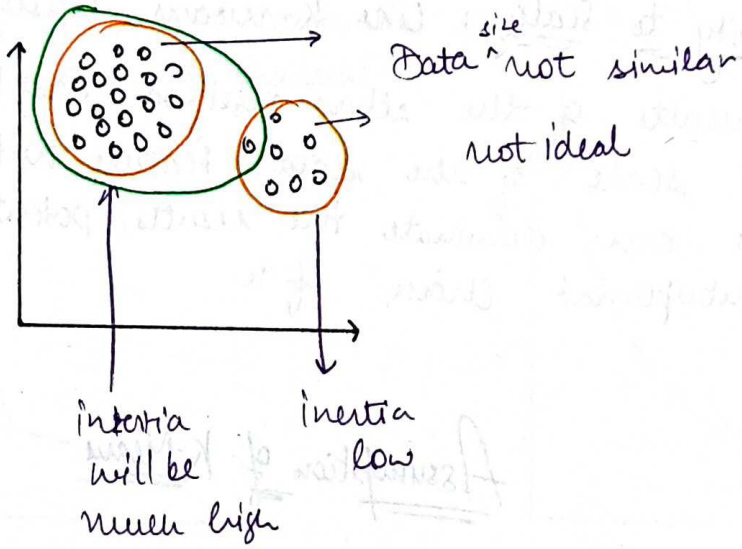
Assumption of K-Means

Spherical Cluster Shape: k-means assume that the clusters are spherical and isotropic, meaning they are uniform in all direction. Consequently, the algorithm works best when the actual cluster in the data are circular (in 2D) or spherical in (in higher dimensions).

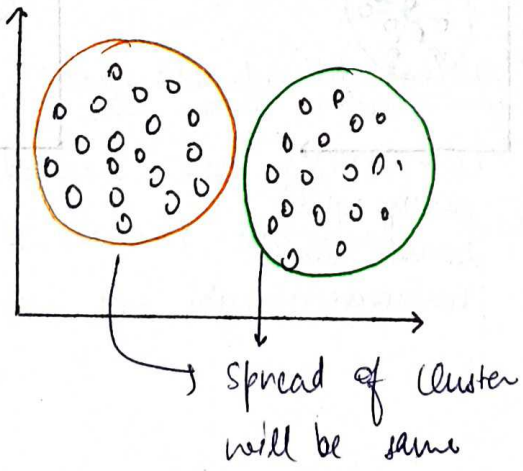




Similar Cluster Size :- The algo tends to perform better when all cluster are of approximately the same size. If one cluster is much larger than others, K-means might struggle to correctly assign the points to the appropriate cluster.



Equal Variance of Cluster : K-means assume that all clusters have similar variance. The algo uses the Euclidean distance metric, which can bias the clustering toward clusters with lower variance



Clusters are well Separated: The algorithm works best when the clusters are well separated from each other. If Clusters are overlapping or intertwined, k-means might not be able to distinguish them effectively.

Number of Cluster (k) is predefined: k-means required the number of clusters (k) to be specified in advance. Choosing the right value of k is crucial, but it is not always straight forward and typically requires domain knowledge or additional methods like the Elbow method or Silhouette analysis.

Large n, small k: k-mean is generally more efficient and effective when the dataset is large (large n) and the number of cluster is small (small k).

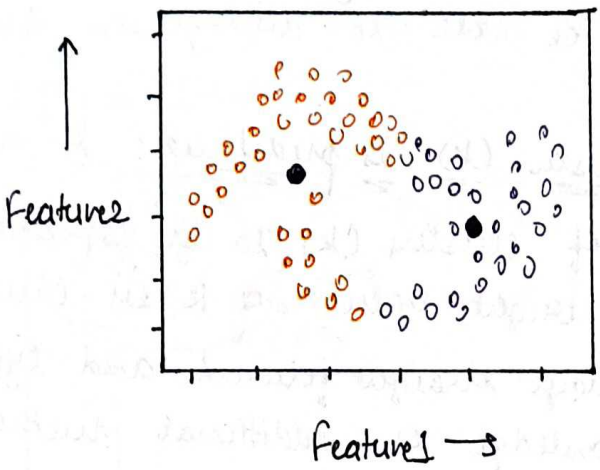
Limitation of K Means

Number of Clusters: Determining the optimal number of Clustering (k) is not straightforward and often requires domain knowledge or method like the elbow method.

Require Cluster of similar size: k-Means require the clusters to be of similar size.

Similar Variance between cluster: k means require the cluster to be of similar variance.

Assumption of Spherical Cluster: KMeans assume that clusters are spherical and of similar size, which might not be the case in real world data.



Vulnerability to outliers: Outliers can significantly distort the mean value of a cluster, leading to misleading result.

Hard Clustering: Each data point is forced into exactly one clustering, which may not be suitable for all application, especially where data can belong to multiple cluster.

Soft Clustering → Not use in Kmean

↳ Gaussian Mixture Model

↳ data 1 [0.78, 0.62] → prob of Cluster 2.

↳ prob of Cluster 1 assign data 1

High - Dimensional Challenges: In very high - dimensional spaces, the distance - between data points can be becomes less meaningful, affecting the performance of K means

Sensitive to Scale: The measure is sensitive to the scale of the features. Hence, feature scaling (like, standardization) is often recommended before applying K-means.

Silhoutte Score

Cohesion (intra cluster distance)

Defination: Cohesion refers to the degree to which element within the same cluster are close to each other. It measure how tightly grouped the data points in a cluster are.

Ideal Scenario: High cohesion means that data points in a cluster are similar or near to each other, indicating a good clustering where each cluster is distinct and meaningful.

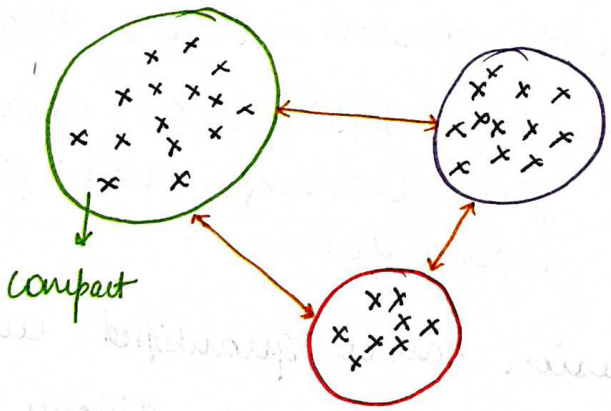
Measurement: Cohesion can be quantified using metrics such as the sum of squared distance of data points from their respective cluster centroids. In k-means this is often referred as inertia or within - cluster sum of squares.

Separation (inter cluster distance)

Defination: Separation, on the other hand, refers to how distinct or well-parted different clusters are from each other. It measures the extent to which clusters are different or distant from each other.

Ideal Scenario: High separation means that clusters as well-differentiated and far apart, indicating that the algo has done a good job in distinguish between different group in the data.

Measurement: Separation can be quantified metrics such as the distance between cluster centroids, or more complex measures like the silhoutte score which consider both cohesion and separation.



1.) cohesion = ↓ low

2.) Separation = high

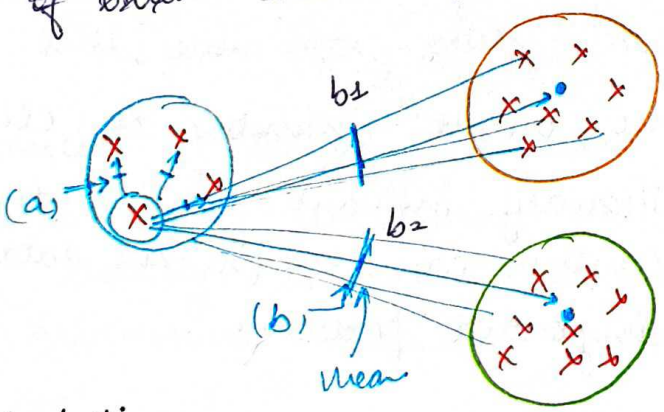
Silhoutte Score

The silhoutte score is a measure used to assess the quality of cluster created by a clustering algo. It provides a succinct graphical representation of how well each data point lies within its cluster, which is a combination of both cohesion and separation. The value of the silhoutte score range -1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters.

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

(a) = mean of distance betn selected point and all point in particular cluster.

(b) = mean of distance betn selected point and ~~centroid~~ nearest ~~of other cluster~~. point of other cluster.



if $b_1 < b_2$
↳ we consider b_1

if $b_2 < b_1$
↳ consider b_2

Interpretation:

Close to +1: Indicates that the data point is far away from the neighbouring clusters.

Close to 0: Indicates that the data point is on or very close to the decision boundary beth two neighbouring clusters.

Close to -1: Indicates that the data point may have been assigned to the wrong cluster.

Kmeans Hyperparameter

class sklearn.cluster.kmeans(n_clusters=8, init='k-means++', n_init='warn', max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm="lloyd")

Number of Clusters (k):

Description: This is the number of cluster you want the algorithm to form, as well as the number of centroids to generate.

Importance: Choosing the right number of cluster is crucial as it significantly influence the clustering results. Too many clusters can overfit the data, while too few can miss important patterns.

Initialization Method

Description: This parameter specifies the method for initializing the centroids. Common methods include

'random' (randomly choosing k data points as initial centroids) and 'k-means++' (a smarter way of intializing centroids to improve convergence.

Importance: Good intialization can lead to faster convergence and better clustering. k-mean++ is generally preffered over random intialization.

Number of Initialization Runs (n_init):

Description: This is the number of times the KMeans algo will be run with different centroid initializations. The final results will be the best output of n_init consecutive runs in term of inteia.

Importance: Multiple initialization can prevent the algo from falling into sub-optimal solutions, but increase computational cost.

Maximum Number of Iteration (max_iter):

Description: The maximum number of iterations the algo will run for each intialization.

Importance: A higher number of iteration allows more time for convergence but increase computational time. Usually, KMeans converges well before reaching the maximum number of iteration.

Tolerance (tol):

Description: This is the tolerance to declare convergence if the centroids do not move significantly (as

defined by this parameter) in consecutive iterations, the algorithm stops.

Importance: A smaller tolerance can lead to a more precise solution, but might increase computation time. A large tolerance might speed up the algo but can lead to less precise clustering.

Algorithm

Description: The computational algo to use. Options typically include 'auto', 'full', or 'Elkan', 'full' corresponds to the classic EM-style algo, while 'elkan' is an optimized variant that is generally more efficient.

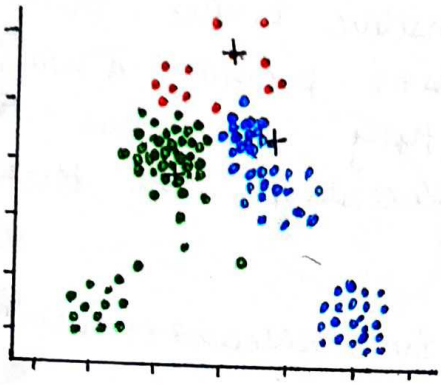
Importance: The choice of algo affects the computa-tional efficiency. 'Elkan' is often faster but works only with Euclidean distance.

Random State:

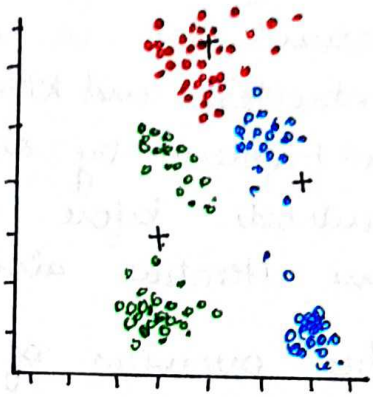
Description: Determines the random number generation for centroid initialization, which influences the behaviour of the algo.

Importance: Setting a random state ensures reproducibility, as KMeans can produce different results on different runs due to its stochastic nature.

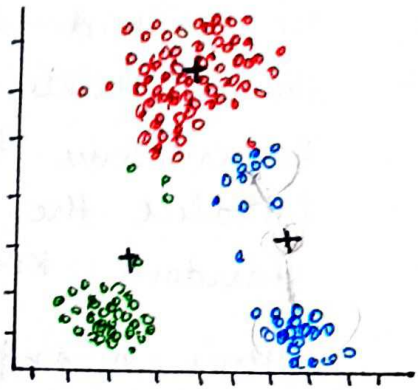
Kmeans++



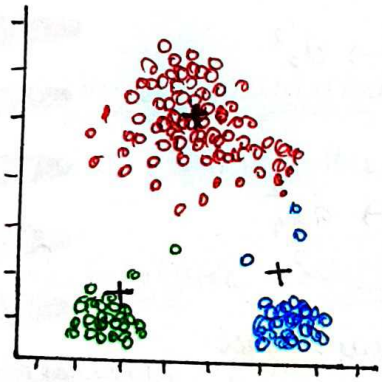
iteration 1



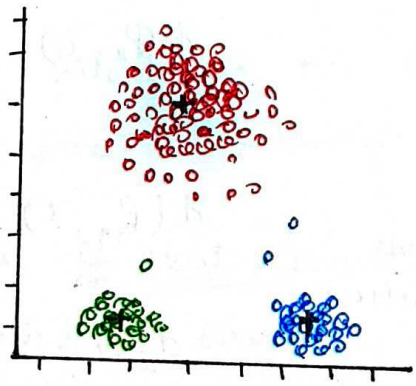
iteration 2



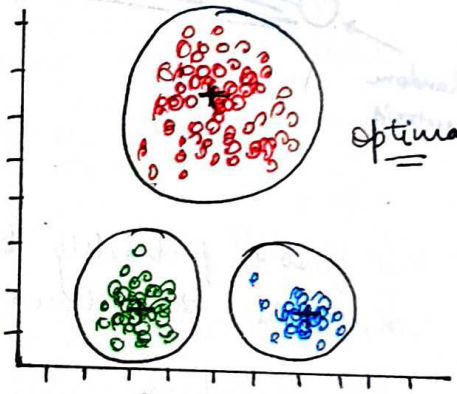
iteration 3



iteration 4

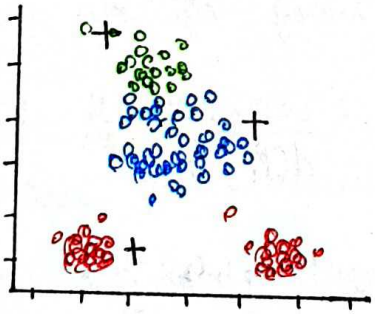


iteration 5

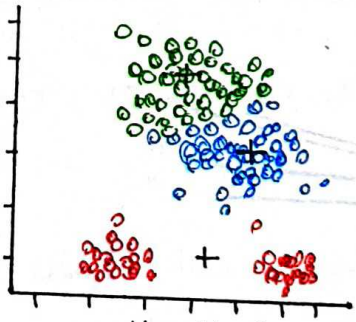


optimal

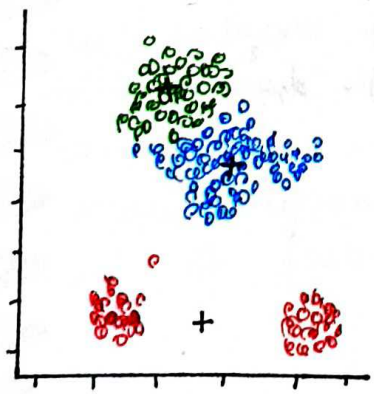
iteration 6



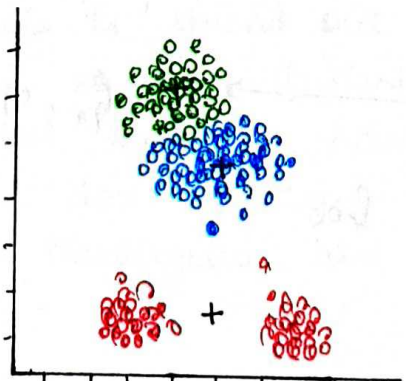
iteration 1



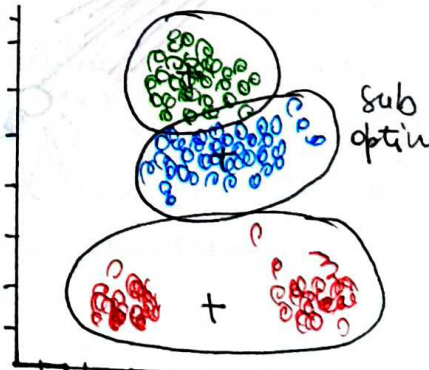
iteration 2



iteration 3



iteration 4

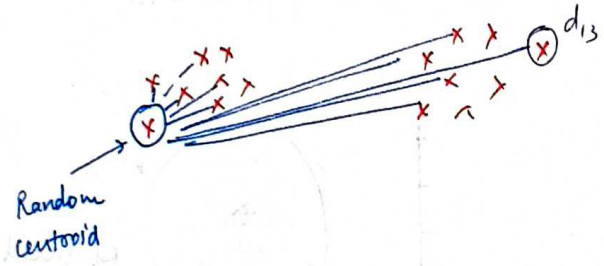


sub optimal

iteration 4

KMeans++ is an algorithm for choosing the initial values (or "seeds") for the KMeans clustering algorithm. The standard KMeans algo is sensitive to the initial starting points (centroids), and KMeans++ provide a way to overcome this problem by specifying a procedure to initialize the centroids before proceeding with the standard KMeans iterative algo.

Here's a simplified overview of how KMeans++ works:



↗ Point

$d(P_1, C) \rightarrow d_1^2$

$d(P_2, C) \rightarrow d_2^2$

⋮

$d(P_{99}, C) \rightarrow d_{99}^2$

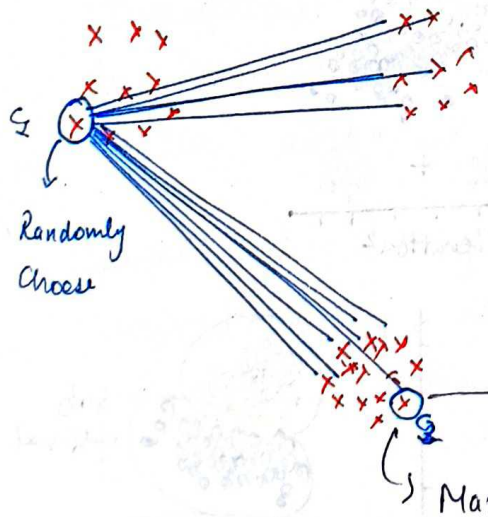
d_{13} → distance higher

↓ → Probability higher

Select chance high

* We finding probability because some time data have outliers

For 3 Cluster

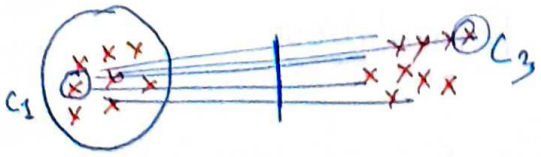


$d(P_1, C_1) \rightarrow d_1^2$

$d(P_2, C_1) \rightarrow d_2^2$

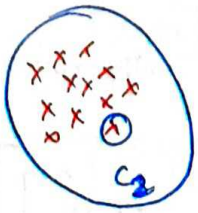
⋮

$d(P_{99}, C_1) \rightarrow d_{99}^2$



$C_3 \rightarrow$ every point $\rightarrow$ nearest centroid

if C_2 is nearest then we can find the distance bet^n C_2 and every point.



Initial Centroid Selection: The first centroid is chosen uniformly at random from the data points that are being clustered.

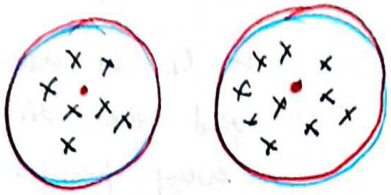
Distance Calculation: Calculation the distance of each data point from the nearest, previously chosen centroid.

Probabilistic Selection of Next Centroids: Choose the next centroid from the data points with a probabilities proportional to the square of the distance from the point to its nearest centroid. This step biases the algo to select data points that are far from the existing centroids.

Repeat Until K Centroids: Repeat steps 2 and 3 until K centroids have been chosen.

Proceed with Standard KMeans: Once the intial centroids are chosen, proceed with the standard KMeans clustering algo. This methods tends to spread out the initial centroids, which can lead to better clustering results compared to selecting the initial centroids randomly, as the standard KMeans algorithm does. By doing so, KMeans++ can often lead to faster convergence and better clustering.

Kmeans Mathematical formulation

Kmeans 

K=2

[Loyd's Algo] jyad

↓

Approximation

$$J = \underset{\mu_1 \mu_2 \cdots \mu_K}{\operatorname{argmin}} \left[\sum_{i=1}^{k} \left[\sum_{x \in S_i} \left[\|x - \mu_i\|^2 \right] \right] \right] \rightarrow \text{Inertia}$$

Such that

$S_i \cap S_j = \{\}$

↳ Not common point betn 2 cluster or more.

$\mu_1 \mu_2 \cdots \mu_k \rightarrow$ centroid

k centroid

$S_i \rightarrow$ current cluster

→ We have to find the value of $\mu_1 \mu_2 \cdots \mu_k$ which reduce the value of Inertia.

→ This is not best soln because everytime we take new intial ~~data~~ then everytime new solution form through Loyd's Algo.

KMeans Time Complexity

Kmeans → time complexity

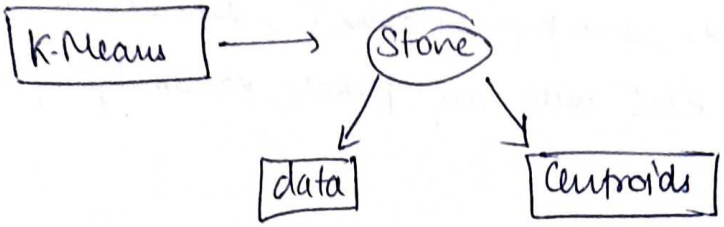
$O(n k d i)$ → iteration

n → no. of rows

k → centroids

d → dimension

KMeans Space Complexity



$$O(nd + kd)$$

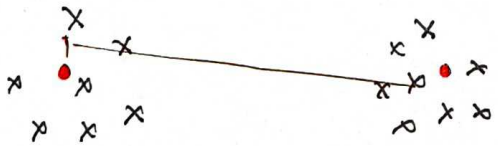
- nd: no. of rows, cols → data
- kd: no. of centroids, cols → centroids

Mini Batch KMeans

Variation → KMeans

↳ Mini Batch KMeans

↳ Gradient Descent

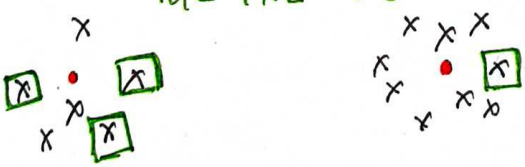


→ $n \times k$ → distances

↓

batches

$n = 4 \times 2 \rightarrow 8$



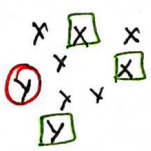
batch size = 4

mini batch

- Speed ↑
- Accuracy ↓

Algorithm 1 Mini-batch k-Means

1. Given: k, mini-batch size b, iteration t, data set X
2. Initialize each $c \in C$ with an picked randomly from X
3. $v \leftarrow 0$
4. for i = 1 to t do
5. $M \leftarrow$ b examples picked randomly from X (mini batch)
6. for $x \in M$ do
7. $d[x] \leftarrow f(C, x)$ // Cache the center nearest to x
8. end for
9. for $x \in M$ do
10. $c \leftarrow d[x]$ // Get cached center for this x
11. $v[c] \leftarrow v[c] + 1$ // Update per-center counts
12. $\eta \leftarrow \frac{1}{v[c]}$ // Get per-center learning rate
13. $c \leftarrow (1 - \eta)c + \eta x$ // Take gradient step
14. end for
15. end for



X = 200 points (n)

k = no. of cluster

b = 10

i = 10 (iteration)

Step 3: V = [0, 0] (because k = 2]

if k = 3 V = [0, 0, 0]

Step 4: = loop → 1 - t / 10 times

Step 5: - [Randomly 10 → 200 points] mini batch

Step 6 → Nested loop → 10 times

$x_1 \rightarrow c_1$ Step: 7 dict = (x, c)

$x_2 \rightarrow c_1$

$x_3 \rightarrow c_2$

⋮

$x_{10} \rightarrow c_2$

Step 8: loop end

Step 9: loop → 10 times

Step 10: $c \leftarrow d[x]$ ⇒ We have to find nearest centroid with the help of step 7: dict = (x, c)

Step 11: $v[c] \leftarrow v[c] + 1$ ⇒ (update value)

c_1

$v = [1, 0]$

↳ centroid near the point 1.

Step 12: $\eta \leftarrow \frac{1}{\eta[c]} = \frac{1}{1} = 1$ (learning rate)

Step 13: $c_1 = (1-\eta) c_1 + \eta x_1$

$c_1 = \underbrace{(1-1)}_{0} c_1 + 1 x_1$

$c_1 = x_1$

another example

$x_2 \rightarrow c_1$ ↗ 2 point near centroid

$v_2 = [2, 0]$

$\eta = \frac{1}{2} = 0.5$

$c_1 = (1 - 0.5) c_1 + 0.5 x_2$