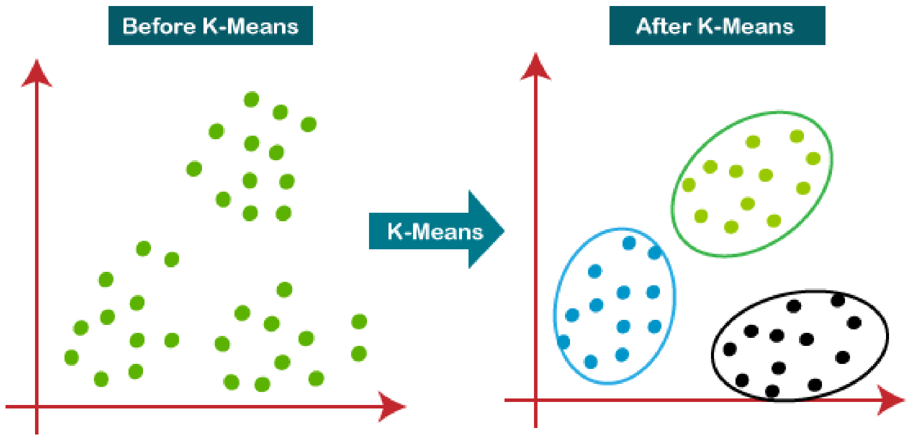
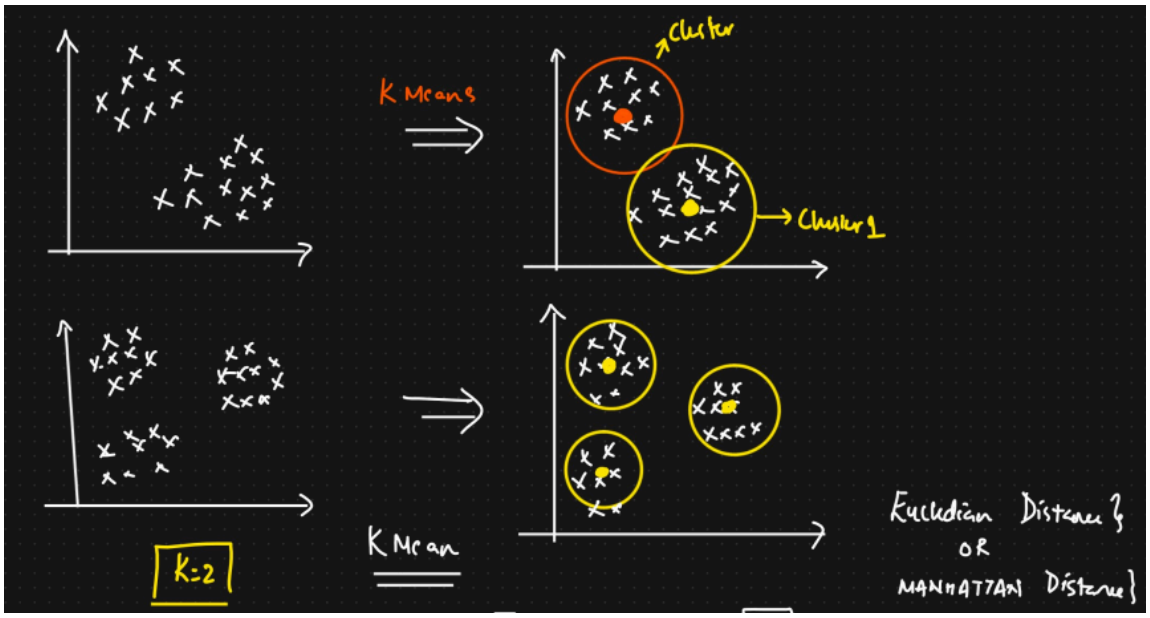
K-Means Clustering

I. Introduction

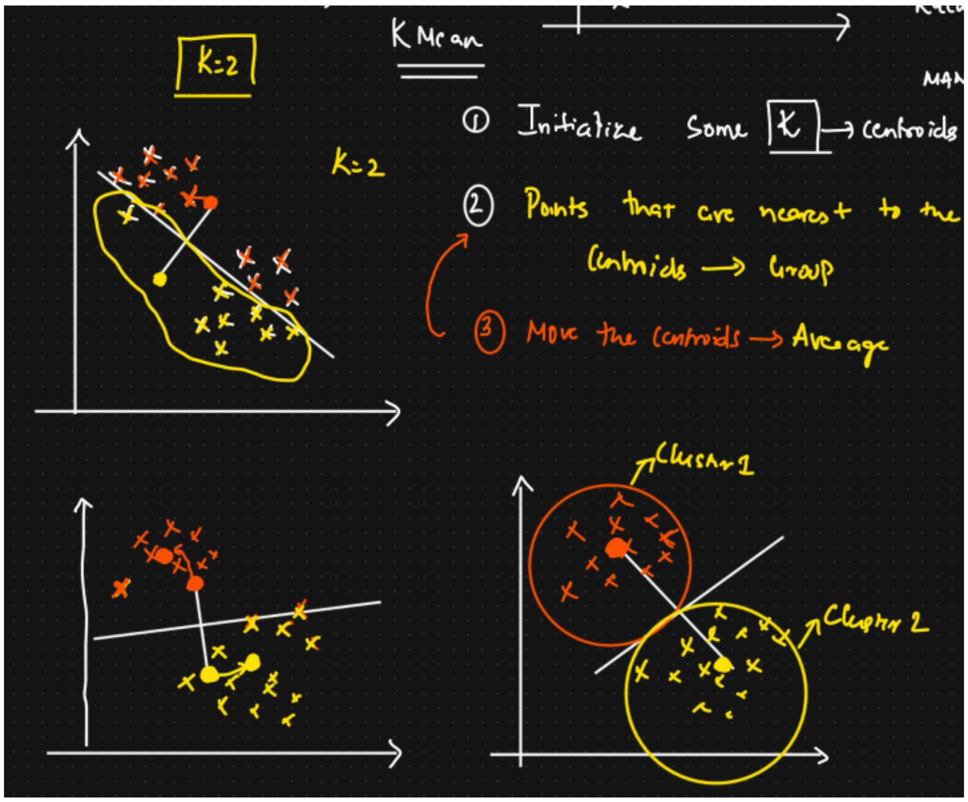
- K-Means is an **unsupervised machine learning algorithm** used to partition data into **K distinct non-overlapping clusters** based on feature similarity.
- **Goal:** To minimize the within-cluster variance (the sum of squared distances between each data point and its cluster centroid).
- **Key Idea:** Iteratively assign data points to the closest cluster centroid and then update the centroids based on the assigned points.
- **Applications:**
    - Customer segmentation
    - Image compression
    - Anomaly detection
    - Document clustering
    - Recommender systems





II. Algorithm Steps

1. **Initialization:**
    - Choose the number of clusters, k.
    - Randomly initialize k cluster centroids. Common methods include:
        - Randomly selecting k data points from the dataset.
        - Randomly assigning values within the data range for each centroid.
2. **Assignment Step:**
    - Iterate through each data point in the dataset.
    - Calculate the distance between the data point and each of the k centroids (e.g., Euclidean distance).
    - Assign the data point to the cluster whose centroid is closest.
3. **Update Step:**
    - For each of the k clusters, recalculate the new centroid by taking the mean of all the data points assigned to that cluster.
4. **Convergence:**
    - Repeat the assignment and update steps until one of the following convergence criteria is met:
        - The centroids no longer change significantly between iterations.
        - A maximum number of iterations is reached.


K=2
K Mean
K=2
① Initialize Some K → centroids
② Points that are nearest to the
Centroids → Group
③ Move the Centroids → Average
Cluster 1
Cluster 2

III. Mathematical Formulation

- **Objective Function (Within-Cluster Sum of Squares - WCSS):**

$$J(C) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where:

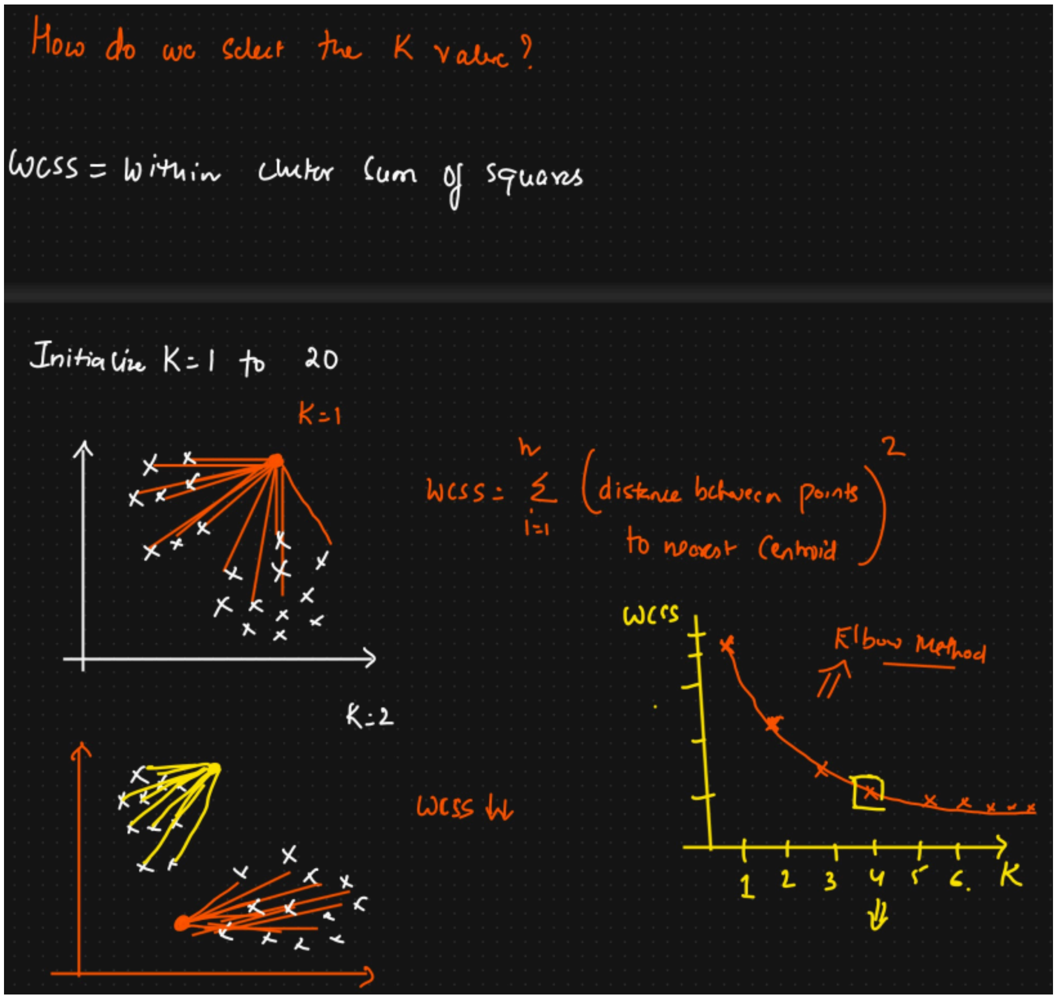
- k is the number of clusters.
- C_i is the i-th cluster.
- x is a data point in cluster C_i.
- μ_i is the centroid (mean) of cluster C_i.
- $\|x - \mu_i\|^2$ is the squared Euclidean distance between data point x and centroid μ_i.
- The algorithm aims to find cluster assignments $C = \{C_1, C_2, ..., C_k\}$ that minimize $J(C)$.

-
- **Centroid Update:**

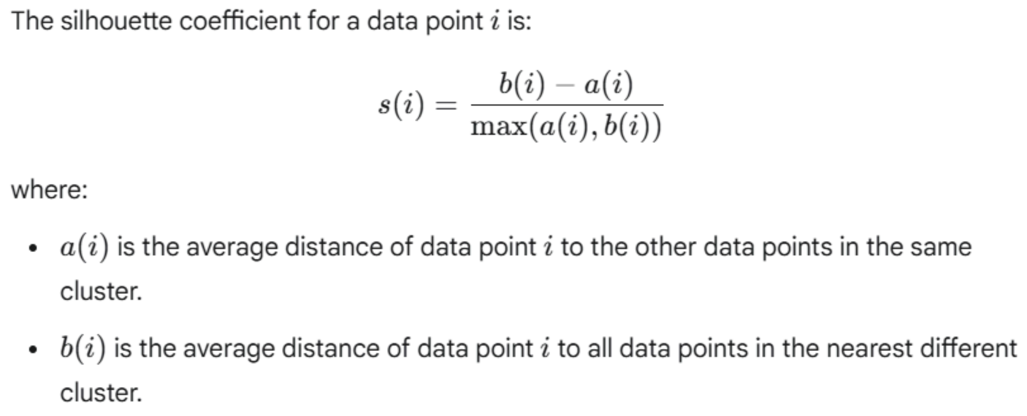
$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

- where $|C_i|$ is the number of data points in cluster C_i.

IV. Choosing the Number of Clusters (k)

- **Elbow Method:**
  - Run K-Means for a range of k values (e.g., 1 to 10).
  - For each k, calculate the WCSS.
  - Plot the WCSS against the number of clusters.
  - Look for an "elbow" point in the plot, where the rate of decrease in WCSS starts to diminish. This point is often a good estimate for k.
  - 
How do we select the K value?
WCSS = Within cluster sum of squares
Initialize K=1 to 20
K=1
WCSS = $\sum_{i=1}^{n}$ (distance between points to nearest centroid)^2
WCSS
Elbow Method
K=2
WCSS ↓
1 2 3 4 5 6 K
- **Silhouette Analysis:**
  - Calculates a silhouette coefficient for each data point, which measures how well it fits into its assigned cluster compared to other clusters.

- The silhouette coefficient for a data point i is:



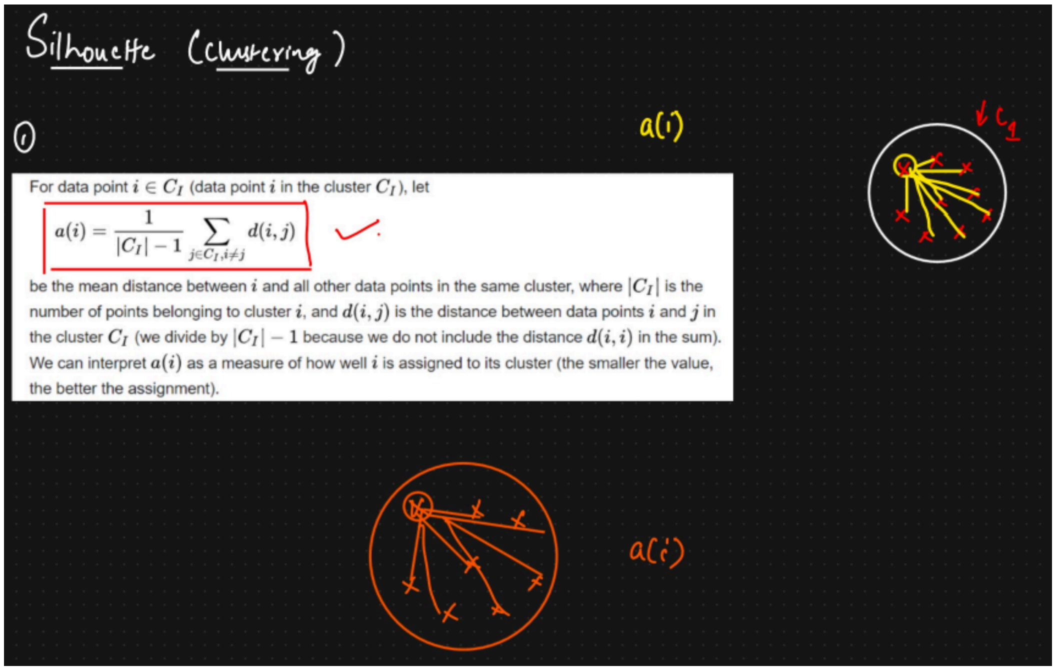
The silhouette coefficient for a data point i is:

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

where:

- $a(i)$ is the average distance of data point i to the other data points in the same cluster.
- $b(i)$ is the average distance of data point i to all data points in the nearest different cluster.

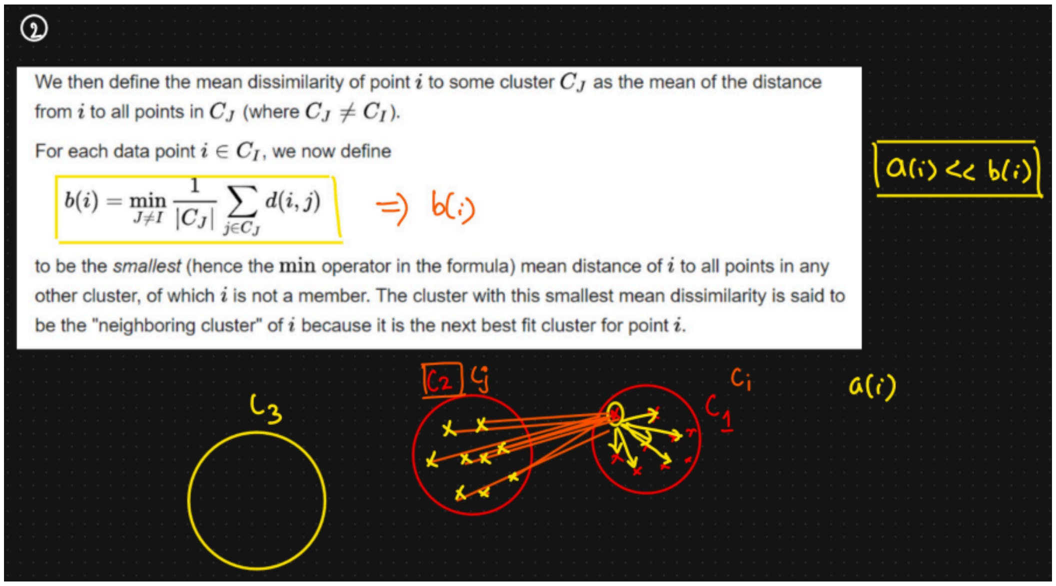
- The silhouette coefficient ranges from -1 to 1:
  - Close to 1: The data point is well-clustered.
  - Close to 0: The data point is close to the decision boundary between two clusters.
  - Close to -1: The data point might have been assigned to the wrong cluster.
- Calculate the average silhouette score for different values of k and choose the k that maximizes this score.



For data point $i \in C_I$ (data point i in the cluster C_I), let

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

be the mean distance between i and all other data points in the same cluster, where $|C_I|$ is the number of points belonging to cluster i, and $d(i, j)$ is the distance between data points i and j in the cluster C_I (we divide by $|C_I| - 1$ because we do not include the distance $d(i, i)$ in the sum). We can interpret $a(i)$ as a measure of how well i is assigned to its cluster (the smaller the value, the better the assignment).

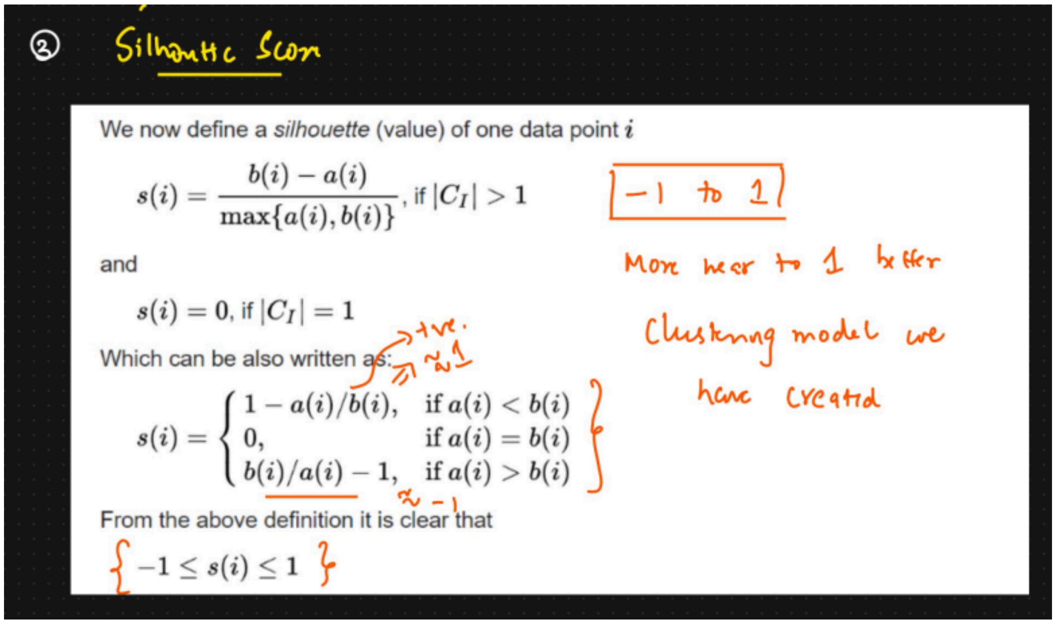


We then define the mean dissimilarity of point i to some cluster C_J as the mean of the distance from i to all points in C_J (where $C_J \neq C_I$).

For each data point $i \in C_I$, we now define

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

to be the *smallest* (hence the $\min$ operator in the formula) mean distance of i to all points in any other cluster, of which i is not a member. The cluster with this smallest mean dissimilarity is said to be the "neighboring cluster" of i because it is the next best fit cluster for point i.



More near to 1 better clustering model we have created

We now define a *silhouette* (value) of one data point i

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_I| > 1$$

and

$$s(i) = 0, \text{ if } |C_I| = 1$$

Which can be also written as:

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \quad \xrightarrow{} +ve.\ \approx 1 \\ 0, & \text{if } a(i) = b(i) \\ \underline{b(i)/a(i) - 1}, & \text{if } a(i) > b(i) \quad \approx -1 \end{cases}$$

From the above definition it is clear that

$$\left\{ -1 \leq s(i) \leq 1 \right\}$$

- **Domain Knowledge:** Prior understanding of the data can sometimes suggest a reasonable number of clusters.

V. Advantages of K-Means

- **Simple and easy to understand and implement.**
- **Relatively efficient, especially for a large number of data points.** The time complexity is generally <3>O(n · k · I · <4>d), where n is the number of data points, k is the number of clusters, I is the number of iterations, and d is the number of features.
- **Often works well in practice for many types of data.**

VI. Disadvantages and Limitations of K-Means

- **Sensitive to the initial placement of centroids.** Different initializations can lead to different clustering results (local optima). Running the algorithm multiple times with different initializations can mitigate this.
- **Assume clusters are spherical, equally sized, and have similar densities.** It may not perform well on clusters with complex shapes, varying sizes, or densities.
- **Sensitive to outliers.** Outliers can significantly affect the position of the centroids.
- **Requires specifying the number of clusters (k) beforehand.** Choosing the optimal k can be challenging.
- **Not guaranteed to converge to the global optimum.**

VII. Important Considerations and Best Practices

- **Feature Scaling:** It's often beneficial to scale the features (e.g., using standardization or normalization) before applying K-Means, as it is distance-based and features with larger scales can dominate the distance calculations.
- **Running Multiple Initializations (k-means++)**: Algorithms like k-means++ can help in choosing better initial centroids, leading to faster convergence and potentially better results.
- **Evaluating Cluster Quality:** Use metrics like WCSS, silhouette score, or domain-specific metrics to assess the quality of the obtained clusters.
- **Understanding Data Characteristics:** Before applying K-Means, it's crucial to understand the characteristics of your data and whether the assumptions of K-Means are likely to hold.

I. What is Random Initialization in K-Means?

- As we discussed, the K-Means algorithm starts by randomly selecting k initial centroid points within your dataset's feature space.
- These initial centroids act as the starting points around which the algorithm will try to form clusters.

II. The "Trap" - Why is Random Initialization Problematic?

The problem arises because the *initial random placement* of these centroids can heavily influence the final clustering outcome. Here's why this is a "trap":

1. **Suboptimal Clustering (Local Optima):**
   - Depending on the initial positions, the algorithm might converge to a *local optimum* rather than the *global optimum*.
   - A local optimum is a clustering where the within-cluster variance is minimized for *that specific initialization*, but it might not be the overall best possible clustering for the data.
2. **Inconsistent Results:**

    - Running K-Means multiple times on the same dataset with different random initializations can yield significantly different cluster assignments and centroid positions.
    - This lack of consistency makes it difficult to trust the results of a single run, especially if the clusters aren't well-separated.
3. **Slow Convergence:**
    - Poorly placed initial centroids can lead to more iterations being required for the algorithm to converge.
4. **Empty Clusters:**
    - In some unfortunate initializations, one or more of the randomly chosen centroids might not have any data points assigned to them during the assignment step. This can lead to errors or require specific handling within the algorithm.

IV. Visualizing the Trap

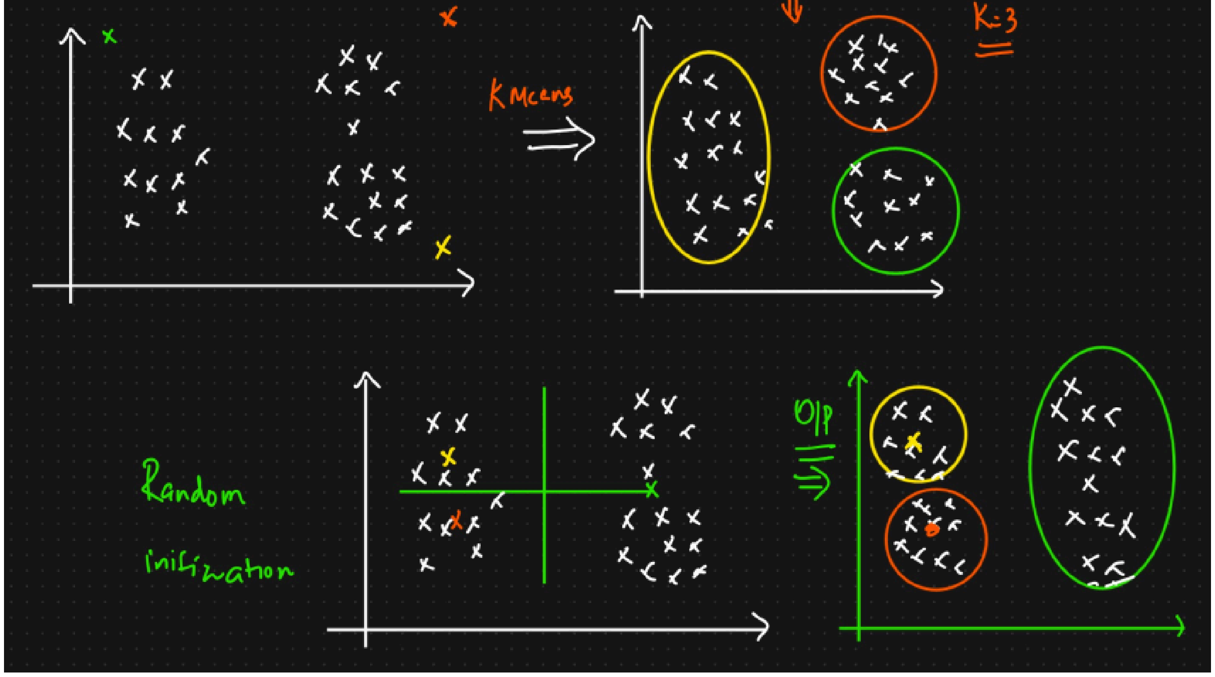
Imagine a dataset with three natural clusters.

- **Good Initialization:** If the initial three centroids happen to be placed roughly within each of these clusters, the algorithm is likely to converge to a good representation of the underlying structure.
- **Bad Initialization:** If two initial centroids are placed very close to each other within one natural cluster, and the third is far away, the algorithm might incorrectly split one natural cluster and leave another poorly represented.

V. Basic Mitigation Strategies:

- **Multiple Initializations:** Run the K-Means algorithm multiple times with different random seeds and choose the solution with the lowest inertia (WCSS). This increases the chances of finding a better local optimum.
- **Smarter Initialization Techniques (like k-means++):** These techniques aim to select initial centroids that are more spread out and less likely to fall into suboptimal configurations.

Random Initialization TRAP (Kmeans ++)


K=3
Kmeans
Random
initization
O/p