# K-Means Clustering

K Means clustering is an unsupervised machine learning algorithm used to partition a data into k distinct clusters. The algorithm identifies similar data points and assigns them to clusters based on their features. The key idea is to group data points in such a way that items in the same (group) cluster are more similar to each other than to those in other clusters.

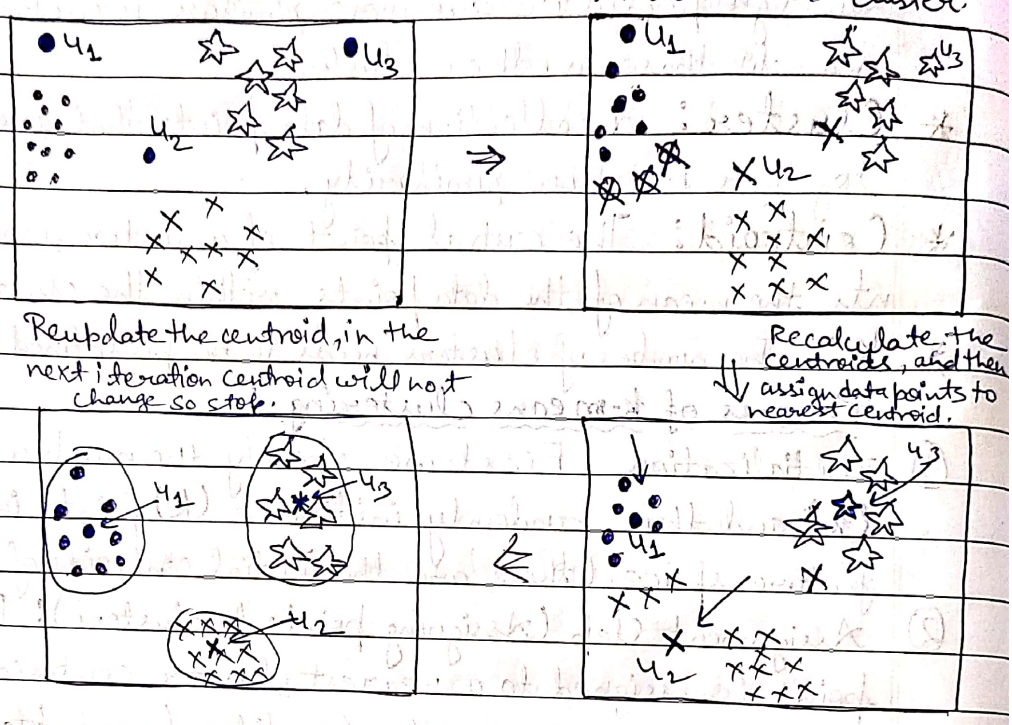
* **Cluster:** A collection of data points that are grouped together based on similarity.
* **Centroid:** The central point of a cluster, which represents the mean of the data points within the cluster.
* **K:** The number of clusters, needs to be predefined by user.

## Process of k-means clustering:

1. **Initialization:** First we specify the number of clusters (k) and then randomly initialize (k) points in the feature space (these are the initial centroids of the cluster).
2. **Assignment Step (Assigning points to clusters):** Each data point is assigned to a nearest centroid based on a distance metric usually (euclidean / manhattan distance). For each point $X_i$, its distance to all centroids is calculated, and it is assigned to the centroid having nearest distance.
3. **Update step (Recalculating Centroids):** Once the data points are assigned to the clusters, the centroid of each cluster is recalculated as the mean of all the data points in that cluster. The new centroid is the point at the center of group of assigned points.
4. **Repeat steps 2 and 3:** After updating the centroids, reassign all data points to the nearest centroid and recalculate centroids. This process is repeated iteratively until the centroids no longer change significantly (i.e., the algorithm converges) or a predefined number of iterations is reached.
5. **Stopping Criteria:** The algorithm stops when the centroids

stabilize (i.e., when the assignment of data points to clusters do not change between iterations). Alternatively, the algorithm stops after a fixed number of iterations.

* Geometrical Intuition




Initially, clusters may be formed randomly depending on the starting centroids. With each iteration, points are reassigned to clusters and centroids are adjusted, leading to more refined clusters.

Ultimately, K-Means tries to minimize the Within-Cluster Sum of Squares (WCSS) i.e., the sum of squared distances between each points and its assigned centroid within the cluster. The goal is to make each cluster compact, with points closed to their centroid.

## Loss function used in K-Means

In K-Means clustering, the loss function used is the Within-Cluster Sum of Squares (**WCSS**) or **interia**. This loss function measures the compactness of the clusters by calculating the sum of squarred distances between each data point and the centroid of its assigned cluster.

The loss function for K-Means is written as:

$$WCSS = \sum_{i=1}^{K} \sum_{x \in C_i} ||x - \mu_i||^2$$ where,

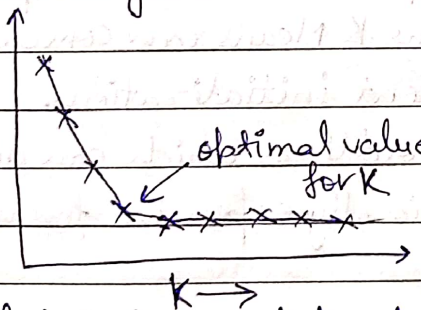
→ K is the number of clusters
→ $C_i$ represents the set of data points in the i-th cluster.
→ $x$ is a data point.
→ $\mu_i$ is the mean of i-th cluster.
→ $||x - \mu_i||^2$ is the Euclidean distance between the data point $x$ and the centroid $\mu_i$

The goal of K-Means is to minimize this WCSS by adjusting the centroids and reassigning points to clusters, so that the data point within each cluster are as close as possible to the centroid.

**Choosing the value of K in K-Means (Elbow Method)**

One common method for selecting K is the elbow method, which involves:

1. Running K-Means for different values of K.
2. Plotting the WCSS (sum of squared distances from each point to its centroid) for each value of K.
3. Looking for an "elbow" point on the graph, where the rate of decrease sharply slows down. This is considered as the optimal value of K.




As the value of K increases, the inertia decreases.
The most value of K can be equal to the number of data points in the dataset, at that point each data point will be centroid of a cluster itself and the WCSS=0.

* In real world scenario, the elbow method might not be super useful for getting the optimal value of K as we may not get the elbow point, in such cases we may take the help of domain knowledge for getting an idea what should be my ideal K.

**KMeans vs KMeans++ :** In naive-KMeans, the initialization of K points (centroids) happens randomly and in case of poor initialization in the early stages, it can lead to suboptimal clustering and it can even take longer time to converge. In case of naive-KMeans all initial centroids might fall into one region of the data space, leading to poor cluster formation. On the other hand, KMeans++ addresses this problem by choosing initial centroids by probabilistic approach, i.e., points that are farther from the existing centroids are more likely to be chosen as next centroids. Thus KMeans++ ensures that the initial centroids are much spread out, reducing the chances of poor convergence with faster speed (in less iterations) and often shows more stable results as compared to naive KMeans.

**Advantages of KMeans Clustering :**

1) It is easier to understand and implement.
2) It often works well even on larger datasets.
3)

**Disadvantages of KMeans Clustering :**

1) The number of clusters, K, needs to be specified beforehand, which can be difficult if the optimal number is unknown.
2) The final clusters depend on the initial placement of centroids, which means K-Means can converge to different clusters with different initializations.
3) It is sensitive to outliers which can distort the centroids.
4) It can be computationally expensive for very large datasets.

**Applications :**

1) Can be used for segmentation tasks like - Image segmentation, customer segmentation, etc.
2) Anomaly detection tasks.
3) Document clustering : grouping similar text documents for topic modelling.

classmate
Date
Page 05

# Interview questions on KMeans Clustering

Ques 1. How do we handle categorical data in KMeans clustering?

Ans→ KMeans clustering is primarily designed for numerical data, where the concept of distance is well defined. However, when dealing with categorical features, K-Means in its original form struggles to calculate distances. For categorical data, we cannot directly compute distances like we would with continuous numerical data. This poses a challenge when trying to cluster datasets that have categorical attributes.

Several approaches can be used to adapt K-Means to handle categorical data or mixed data (both categorical and numerical). Here's how it can be done:

(1.) One-Hot encoding for categorical data → It converts categorical features into multiple binary features. These binary features can be treated like numerical features for calculating distance. However, there are some caveats:

(a) One Hot Encoding can increase the dimensionality of data significantly, which might make K-Means less efficient.

(b) The euclidean distance calculated for categorical features might not capture the true relationship between categorical values (e.g. the distance between "Red" and "Blue" might not be meaningful) thus resulting cluster will not make any sense.

(2) Label Encoding for Ordinal Categorical data: For ordinal categorical data (where the categories have an inherent order such as low, medium, high), we can apply label encoding, where each category is assigned an integer. Low→1, Medium→2, High→3. In this case, KMeans can handle these numerical features effectively because the ordering of the categories makes sense because in terms of distance. However, this approach is not suitable for nominal (unordered) categorical data because the algorithm will treat the values as having a numerical relationship that doesn't actually exist.

(3) K-Modes for purely categorical data: For datasets that contain only categorical features, the **K-Modes** algorithm is a variant of K-Means which instead of using Euclidean distances uses a matching dissimilarity measure, which counts the number of mismatches between feature values. Instead of updating centroids as the mean of data points (as in K-Means), K-Modes updates centroids to be the **mode** (most frequent value) of the categorical variable in each cluster. In K-Modes, the distance between two categorical data points is calculated as the number of feature mismatches

$$\text{Dissimilarity}(x,y) = \sum_{i=1}^{d} \delta(x_i, y_i)$$ where,

$\delta(x_i, y_i) = 0$ if $x_i = y_i$, otherwise $\delta(x_i, y_i) = 1$

This allows the algorithm to handle categorical data naturally

(4) K-Prototypes Algorithm for mixed data (Numerical + Categorical)
The K-Prototypes algorithm is an extension of both K-Means and K-Modes that combines the strength of both to handle both numerical and categorical data. The distance metric used by K-Prototypes is a combination of the Euclidean distance for numerical features and a matching dissimilarity measure for categorical features.
The objective function in K-Prototypes is —

$$\text{Cost function} = \sum_{x \in C_i} \sum_{\substack{\text{numerical} \\ \text{feats}}} \|x - \mu\|^2 + \sum_{\substack{\text{categorical} \\ \text{features}}} \delta(x, \nu)$$

where,

→ $\|x - \nu\|^2$ is the squared Euclidean distance for numerical feats

→ $\delta(x, \nu)$ is the dissimilarity feature for categorical feats, which counts the number of mismatches between the feature values of $x$ and the centroid $\nu$.

K-prototype effectively balances the influence of both numerical and categorical features during clustering.

Ques 2 How can we validate the number of clusters in K-Means?

Ans → Validating the number of clusters in K-Means is crucial for determining how well the clustering algorithm has performed. There are several methods for validating the number of clusters such as **Silhoutte Score, Dunn's index, Davies Bouldin Index, Calinski-Harabasz index.** These methods evaluate how well-separated and compact the clusters are, helping to determine optimal k.

(1) **Silhoutte Score:** It measures how similar is a point is to its own cluster (cohesion) compared to other clusters (separation). The silhoutte score ranges in between -1 to 1 where higher silhoutte score indicates better clustering, where points are well-matched to their own clusters and poorly matched to others.

For each point i, the Silhoutte score is calculated as—

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

→ a(i) is the average **intra-cluster distance**: The average distance between point i and all other points in the same cluster.

→ b(i) is the average **inter-cluster distance**: the smallest average distance of point i to all points in its nearest cluster (not including its own cluster).

The Silhoutte score for an entire dataset is the mean of the individual silhoutte scores across all points ranging from -1 to +1. A high average Silhoutte score close to 1 suggests that the clusters are well separated and well defined. A score around 0 indicates that the clusters may overlap or that the point is near the decision boundary between two clusters. A score closer to -1 indicates poor clustering, where points may have been misclassified into the wrong clusters.

(2) **Dunn's Index:** Dunn's index also evaluates clustering by comparing the inter-cluster distances with the intra-cluster distances. The aim is to maximize the inter-cluster distances (separation) while minimizing the intra-cluster distances (compactness).
The formula for Dunn's Index is given as —

$$D = \frac{\min(\delta(C_i, C_j))}{\max(\Delta(C_k))}$$

where, $\delta(C_i, C_j)$ is the distances between clusters $C_i$ & $C_j$ (inter-cluster distance)

→ $\Delta(C_k)$ is the diameter of cluster $C_k$, which is the maximum distance between any two points in the cluster $k$ (intra-cluster distance).
In simple terms: Numerator (Separation) Measures the smallest distance between two clusters and denominator (compactness) measures the largest intra-cluster distance. A higher Dunn's index indicates better clustering.

(3) **Davies Bouldin Index:** It measures the ratio of (avg similarity) each cluster with its most similar cluster, based on cluster diameter and centroid distance. It aims to minimize the ratio of within-cluster scatter to between-cluster separation. Low value of Davies Bouldin index indicates good clustering.

(4) **Calinski-Harabasz Index:** It measures the ratio of the sum of between-cluster dispersion and with-in cluster dispersion. This index evaluates how well the clusters are separated and how well compact the clusters are. Unlike Davies-Bouldin index, a higher Calinski-Harabasz index indicates better clustering.

(5) **Elbow Curve:** Discussed on page 3.

Ques.3 Why do we need to perform feature scaling in K Means?
Ans→ Feature scaling is essential in K-Means because the algorithm relies on the distance metric to assign points to

clusters. If the features in the dataset have different units or scales, it can cause **bias** in how distances are computed, leading to poor clustering results. If one feature has a much larger distance than others (e.g. income in thousands versus age in years), the feature will larger scale will dominate the distance calculation. This can distort the clustering process, as the algorithm will give more importance to features with larger numerical ranges.

Ques. 4. Why elbow curve is in shape of an elbow why not sine curve, exponential curve, etc.

Ans→ The elbow curve is shaped as it is because the WCSS initially drops significantly as K increases, but after a certain point further increasing K provides less benefit, forming an elbow shaped curve. The relationship between K and WCSS is monotonic in nature.