K-Means

🧠 What is K-Means Clustering?

K-Means is an **unsupervised learning algorithm** that groups data points into **K clusters** based on similarity.

It tries to find natural groupings in your data **without any labels**.

🧩 The Goal

> Group points such that points within the same cluster are as similar as possible,
>
> and points **across different clusters** are as different as possible.

In simple words:

> "Find K points (called centroids) that best represent your data."

🎯 Why Do We Use K-Means?

- When you **don't have labels** but suspect that data has some structure.
- To **discover hidden patterns or groups** (like customer segments, topics, or behavior types).
- To **reduce dimensionality** or **preprocess** data for other models.

⚙️ The Core Intuition

Think of K-Means as:

1. Guessing K "centers" in the data.
2. Assigning each data point to the *nearest* center.

3. Moving the centers to the *average* position of their assigned points.
4. Repeating until nothing changes.

📐 The Math Behind K-Means

Let's define:

- $X = x_1, x_2, ..., x_n$: the data points
- k: number of clusters
- $\mu_1, \mu_2, ..., \mu_k$: centroids of each cluster
- C_i: the set of points assigned to cluster i

🔷 The Objective Function (Loss)

K-Means minimizes the **Sum of Squared Errors (SSE)** or **Within-Cluster Sum of Squares (WCSS)**:

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

Goal:

Find cluster centers μ_i and assignments C_i that minimize J.

🔷 Step-by-Step Algorithm

1. **Initialization:**

   Randomly choose k initial centroids (e.g., randomly pick k data points).

2. **Assignment step:**

   For each data point x_j, assign it to the cluster whose centroid is closest:

   $C_i = x_j : ||x_j - \mu_i||^2 \leq ||x_j - \mu_l||^2, \forall l$

3. **Update step:**

   Update each centroid to be the mean of its assigned points:

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

4. **Repeat steps 2 & 3** until:
    - Assignments stop changing, or
    - The centroids converge (movement is minimal).

🧮 Example Intuition

Imagine you have these points:

```
(1,1), (2,1), (4,3), (5,4)
```

and you set K = 2.

- Initially, pick (1,1) and (5,4) as centroids.
- Points near (1,1) go to Cluster 1; points near (5,4) go to Cluster 2.
- Update each centroid as the average of its cluster.
- Repeat until centroids stop moving.

Result:

You get 2 tight clusters, each represented by a mean point (centroid).

🔢 Choosing K

You can’t know K upfront — you must **estimate it**.

Common methods:

- **Elbow Method:**

  Plot K vs WCSS (SSE). Choose the “elbow” point where the reduction slows down.

- **Silhouette Score:**

  Measures how well each point fits within its cluster compared to others.

- **Gap Statistic:**

  Compares WCSS with expected values from random data.

⚖️ Pros and Cons

✅ Pros	❌ Cons
Simple and fast	Must predefine K
Works well on spherical clusters	Sensitive to initial centroids
Scales to large datasets	Fails for non-linear clusters
Easy to interpret	Sensitive to outliers

💡 Use Cases

- Customer segmentation (grouping customers by buying behavior)
- Image compression (each cluster = representative color)
- Document clustering (topic discovery)
- Anomaly detection (outliers far from all centroids)

🧠 Intuition Summary

Concept	Meaning
Clusters	Groups of similar points
Centroid	Mean point representing each group
Objective	Minimize within-cluster distances
Distance metric	Usually Euclidean
Learning type	Unsupervised

🌀 Convergence Property of K-Means

🔷 Claim:

> K-Means always converges to some local minimum of its objective function.

🔶 Meaning:

- The **objective function** in K-Means is the **total within-cluster variation**, given by:

  $$J = \sum_{k=1}^{K} \sum_{i \in \mathcal{L}_k} |\bar{x}_i - \bar{\mu}_k|^2$$

  where $\bar{\mu}_k$ is the centroid (mean) of cluster k.
- During training, K-Means **repeatedly alternates** between:
  1. **Assignment step:** Assign each data point to the nearest centroid.
  2. **Update step:** Recompute each centroid as the mean of its assigned points.
- Each step **reduces or keeps the same** value of J.

  Therefore, the algorithm will eventually stop when **no further improvement** is possible — i.e., it reaches a **local minimum**.

🔶 Key Takeaway:

- **Convergence is guaranteed**, but not necessarily to the **global minimum** (the best possible clustering).
- The result depends on the **initial positions** of the centroids.

🔶 Practical Tip:

To avoid bad local minima:

1. **Run K-Means multiple times** with different random initializations.
2. **Compare the final objective values** (total within-cluster variance).
3. **Pick the run with the lowest objective function value.**

🧩 Intuition:

Think of K-Means as a ball rolling downhill on a bumpy surface (objective landscape).

- It will always reach **a valley (local minimum)**.
- But if it started from different positions, it might end up in **different valleys**.
- So, running it multiple times increases the chance of finding the **lowest valley** (best clustering).

⚙️ Choosing the Number of Clusters (K)

🔷 Original K-Means Objective:

We try to minimize **within-cluster variation** given a fixed number of clusters K:

$$\min_{\mathcal{L}_1,\mathcal{L}_2,\ldots,\mathcal{L}K} \sum k = 1^K W(\mathcal{L}k)$$

$$\text{subject to}$$

$$\bigcup k = 1^K \mathcal{L}_k = 1, 2, \ldots, n, \quad \mathcal{L}_i \cap \mathcal{L}_j = \emptyset \text{ for } i \neq j$$

🔶 Idea: "Can we also optimize over K?"

One might think to include K itself as a variable:

$$\min_{K,\mathcal{L}_1,\ldots,\mathcal{L}_K} \sum_{k=1}^{K} W(\mathcal{L}_k)$$

But this is *not a good idea*.

🔷 Why It Fails:

If we allow K to vary freely, the optimal solution becomes trivial and meaningless:

- The best possible (smallest) within-cluster variation occurs when $K^* = n$ — meaning **each data point is its own cluster**.
- Then, $W(\mathcal{L}_k) = 0$ for all clusters (since each cluster has only one point).
- However, this provides **no useful structure** — it's pointless.

🔶 Better Alternative:

We can instead add a **penalty function** that discourages too many clusters:

$$\min_{K,\mathcal{L}_1,\ldots,\mathcal{L}_K} \left[\sum_{k=1}^{K} W(\mathcal{L}_k) + f(K) \right]$$

- $f(K)$: an **increasing function of K** (acts as a regularization term)
- This balances **fit** (lower $W(\mathcal{L}_k)$) and **simplicity** (fewer clusters).

🧠 Intuition:

- Small K: high within-cluster variation (underfitting).
- Large K: low within-cluster variation, but may overfit (too many small groups).
- The goal is to find an **optimal trade-off**.

⚠️ Practical Note:

In practice, instead of adding f(K) directly, we use **heuristics** such as:

- **Elbow Method**
- **Silhouette Score**
- **Gap Statistic**

These help identify a good K that balances compactness and separation.

⚠️ Limitations of K-Means Clustering

Limitation	Explanation	Example / Intuition
You must specify K (number of clusters) in advance	K-Means needs you to decide how many clusters exist before training — but in real life, you often don't know.	In customer segmentation, you might not know if there are 3, 5, or 10 groups. You have to guess or use methods like the **Elbow** or **Silhouette Score**, which are approximations.
Assumes spherical, equally sized clusters	K-Means works best when clusters are roughly **circular (in Euclidean distance)** and of similar size.	If your clusters are elongated or uneven (like ellipses), K-Means will misclassify boundary points.
Sensitive to initial centroids	Different random initializations can produce **different final clusters** (local minima problem).	Two different runs of K-Means on the same data may yield different results — unless you use **K-Means++ initialization**, which helps but doesn't guarantee global optimality.
Sensitive to outliers and noise	Outliers can pull centroids toward them, skewing clusters.	A single far-away point can shift a centroid, distorting cluster assignment for nearby data.
Not good for non-linear shapes	K-Means assumes convex clusters. It fails when clusters have complex boundaries.	For example, in a "moon-shaped" dataset (like two half-circles), K-Means will cut through the middle incorrectly.
Only works well with continuous numerical features	It relies on Euclidean distance, which doesn't make sense for categorical variables.	You can't directly cluster people by "gender" or "favorite color" using standard K-Means — it must be encoded carefully or you use alternatives like **K-Modes**.

Limitation	Explanation	Example / Intuition
Scale-dependent	Features with larger scales dominate the distance calculation.	If one feature (like "income") ranges from 0–1,000,000 and another (like "age") ranges 0–100, the model will prioritize "income" differences unless you scale your data first.
Can get stuck in local minima	The optimization process doesn't guarantee finding the best clustering globally.	Especially true with poor initialization — the algorithm may settle for a suboptimal configuration.
Hard to evaluate objectively	Clustering is unsupervised, so there's no "ground truth" label to measure accuracy.	You must rely on indirect metrics (Silhouette score, inertia), which can be subjective.

Summary Limitations

Limitation Type	Effect on K-Means	Possible Fix
Choosing K	May underfit or overfit clusters	Use Elbow or Silhouette methods
Shape of clusters	Poor separation on non-spherical data	Try DBSCAN or Gaussian Mixture Models
Outliers	Skew centroids	Remove outliers or use K-Medoids
Initialization	Different results per run	Use K-Means++
Feature scaling	One feature dominates	Apply normalization/standardization
Non-numerical data	Poor clustering	Use K-Modes or K-Prototypes

🧩 Why Does K-Means Converge?

🔷 Informal Proof Outline

The K-Means objective function measures the **within-cluster variation**:

$$W(\mathcal{L}_k) = \frac{1}{|\mathcal{L}_k|} \sum_{i,i' \in \mathcal{L}_k} \|\bar{x}^{(i)} - \bar{x}^{(i')}\|^2$$

This formula takes all **pairs of points** within a cluster and computes the squared Euclidean distance between them.

🔶 Lemma 1 — Relating Pairwise Distances to Centroids

We can show that:

$$\frac{1}{|\mathcal{L}_k|} \sum_{i,i' \in \mathcal{L}_k} \|\bar{x}^{(i)} - \bar{x}^{(i')}\|^2 = 2 \sum_{i \in \mathcal{L}_k} \|\bar{x}^{(i)} - \bar{\mu}_k\|^2$$

where

$$\bar{\mu}_k = \frac{1}{|\mathcal{L}_k|} \sum_{i \in \mathcal{L}_k} \bar{x}^{(i)}$$

is the **centroid** of cluster k.

🔶 Intuition:

- The left side measures **pairwise distances** between all points in a cluster.
- The right side measures **distance of each point from the cluster mean** (centroid).
- These are proportional — meaning minimizing distances to the centroid also minimizes pairwise distances between points.

🔷 Lemma 2 — Centroid Minimizes the Sum of Squared Distances

For any point $\bar{y}$,

$$\sum_{i \in \mathcal{L}_k} \|\bar{x}^{(i)} - \bar{\mu}_k\|^2 \le \sum_{i \in \mathcal{L}_k} \|\bar{x}^{(i)} - \bar{y}\|^2$$

That is, the **centroid** $\bar{\mu}_k$ is the point that **minimizes** the total squared distance from all points in the cluster.

🧠 Intuition:

- The centroid acts as the "center of gravity" of the data points.
- Moving the centroid anywhere else **increases** the sum of squared distances.
- Hence, the **mean** is the best representative of its cluster.

🔶 Putting It Together — Why K-Means Converges

1. **Assignment Step:**

   Assign each data point to the nearest centroid.

   → This **decreases** or keeps the same total within-cluster variation W.

2. **Update Step:**

   Recompute the centroid as the mean of assigned points.

   → This also **decreases** or keeps the same W (by Lemma 2).

3. Since W is **non-negative** and decreases every iteration,

   → The algorithm must eventually **converge** to a fixed configuration.

🌀 Visual Intuition

- Lemma 1 (triangle diagram): shows pairwise distances between all points reduce to distances from the centroid.
- Lemma 2 (arrow diagram): shows the centroid is the unique point minimizing total squared distances (illustrated with arrows pointing toward the red centroid).

✅ Conclusion

- Each iteration of K-Means **reduces** the objective function.
- The objective is **bounded below** (cannot be negative).

- Therefore, **K-Means must converge —** typically to a **local minimum**.
