K-Means Clustering – Complete Beginner-to-Advanced Revision Guide

---

1. Introduction

What is K-Means Clustering?

K-Means is an **unsupervised learning algorithm** used to partition a dataset into **K distinct, non-overlapping clusters** based on similarity.

Each cluster is represented by its **centroid**, and the algorithm aims to minimise **within-cluster variance**.

Core Intuition

> "Group points so that those within a cluster are as close as possible."

- Clusters are defined by proximity
- Each point belongs to exactly one cluster
- Centroids represent cluster centres

Real-World Applications

- Customer segmentation
- Image compression
- Market basket analysis
- Document clustering
- Anomaly detection (with care)

---

2. Mathematical Foundations

2.1 Objective Function (Inertia)

K-Means minimises the **within-cluster sum of squares (WCSS)**:

$$ J = \sum_{k=1}^{K} \sum_{x_i \in C_k} |x_i - \mu_k|^2 $$

Where: - C_k = cluster k - μ_k = centroid of cluster k

2.2 Distance Metric

Standard K-Means uses **Euclidean distance**:

$$ d(x, \mu) = \sqrt{\sum_{j=1}^{d} (x_j - \mu_j)^2} $$

2.3 Why Mean Minimises Squared Distance

For a cluster C, centroid is:

$$ \mu = \frac{1}{|C|} \sum_{x_i \in C} x_i $$

This minimises:

$$ \sum_{x_i \in C} |x_i - \mu|^2 $$

(by taking derivative and setting to zero)

---

3. K-Means Algorithm

Step-by-Step Procedure

1. Choose K
2. Initialise K centroids (random or k-means++)
3. Assign each point to nearest centroid
4. Update centroids
5. Repeat until convergence

Convergence Criteria

- No change in assignments
- Centroids stop moving
- Maximum iterations reached

---

4. Initialization Methods

Random Initialization

- Fast
- Can lead to poor local minima

K-Means++ (Recommended)

Selects spread-out initial centroids:

$$ P(x) = \frac{D(x)^2}{\sum D(x)^2} $$

Improves convergence and stability

---

5. Choosing the Number of Clusters (K)

Elbow Method

Plot WCSS vs K and look for elbow

Silhouette Score

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

- a(i): mean intra-cluster distance
- b(i): mean nearest-cluster distance

---

6. Model Evaluation

Internal Metrics

- Inertia (WCSS)
- Silhouette Score
- Davies–Bouldin Index

External Metrics (If labels available)

- Adjusted Rand Index (ARI)
- Normalised Mutual Information (NMI)

7. Interpretation of Clusters

- Centroid coordinates describe typical member
- Distance to centroid measures cluster fit
- Clusters are spherical in nature

8. Assumptions

- Clusters are spherical
- Equal cluster variance
- Similar cluster sizes
- Features are scaled

9. Common Pitfalls & Misconceptions

- ❌ Not scaling features
- ❌ Assuming K-Means finds global optimum
- ❌ Using K-Means for non-spherical clusters
- ❌ Interpreting clusters as ground truth

10. Python Implementation

10.1 From Scratch (NumPy)

```
import numpy as np

class KMeans:
    def __init__(self, k=3, max_iters=100):
        self.k = k
        self.max_iters = max_iters

    def fit(self, X):
        n, d = X.shape
        self.centroids = X[np.random.choice(n, self.k, replace=False)]

        for _ in range(self.max_iters):
            clusters = [[] for _ in range(self.k)]
```

```python
            for x in X:
                distances = [np.linalg.norm(x - c) for c in self.centroids]
                clusters[np.argmin(distances)].append(x)

            new_centroids = np.array([np.mean(c, axis=0) for c in clusters])
            if np.allclose(self.centroids, new_centroids):
                break
            self.centroids = new_centroids
```

10.2 Using scikit-learn

```python
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

X_scaled = StandardScaler().fit_transform(X)

model = KMeans(n_clusters=3, init='k-means++', n_init=10)
labels = model.fit_predict(X_scaled)

print("Silhouette Score:", silhouette_score(X_scaled, labels))
```

10.3 Visualization

```python
import matplotlib.pyplot as plt

plt.scatter(X_scaled[:,0], X_scaled[:,1], c=labels)
plt.scatter(model.cluster_centers_[:,0], model.cluster_centers_[:,1],
            c='red', marker='x')
plt.title("K-Means Clustering")
plt.show()
```

---

11. Advanced Topics

- Mini-batch K-Means
- K-Medoids
- Kernel K-Means

• Spectral clustering comparison

---

12. When NOT to Use K-Means

• Non-spherical clusters
• Heavy noise/outliers
• Different cluster densities

---

13. Best Practices

• Always scale features
• Use k-means++
• Run multiple initializations
• Validate with silhouette score

---

14. Summary

K-Means is: - Simple - Fast - Scalable - Geometry-based

But: - Sensitive to initialization - Assumes spherical clusters

---

End of Revision Guide