✅

K-Means, Elbow and Silhouette Method

Unsupervised Learning Algorithms

Unsupervised learning algorithms attempt to **discover patterns** in unlabeled datasets by identifying similarities or regularities. Common unsupervised tasks include **clustering** and **association**. Clustering algorithms, such as **K-Means**, aim to uncover similarities within a dataset by **grouping objects** so that **those in the same cluster are more similar to each other than to objects in other clusters**. Clustering is performed using criteria such as **shortest distances**, **data point density**, **graphs**, or **various statistical distributions**.

- **K-Means** groups similar data points into clusters by minimizing the average distance between geometric points. It iteratively partitions a dataset into a fixed number (K) of non-overlapping subgroups (or clusters), where each data point belongs to the cluster with the nearest mean (centroid).

Why K-Means?

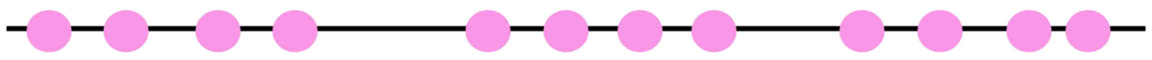
K-Means is widely used today in a variety of real-world applications, including:

- **Customer segmentation**: customers can be grouped to better tailor products and offers.
- **Text, document, or search result clustering**: grouping to identify topics in text.
- **Image clustering or image compression**: grouping similar features or colors in images.
- **Anomaly detection**: identifying what doesn’t fit — the outliers in clusters.
- **Semi-supervised learning**: clusters can be combined with a small set of labeled data and used in supervised learning to generate more valuable insights.

K-Means Clustering - Step by Step Explanation

The images and main idea were taken from StatQuest's youtube channel. Link: https://www.youtube.com/watch?v=4b5d3muPQmA&ab_channel=StatQuestwithJoshStarmer

Imagine you have some data that can be plotted along a line, and you know it needs to be grouped into three clusters. Maybe these measurements come from three different tumor types or cell categories.



In this case, the data forms **three obvious clusters**. But rather than relying on your eyes, let's see if we can get a computer to identify the same three clusters automatically using a technique called **K-means clustering**. We start with raw data that has not been clustered yet.

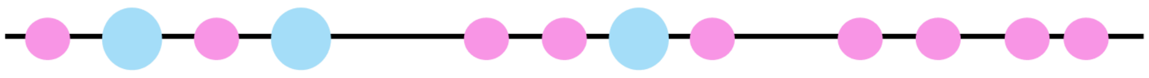
Step-by-Step Process

1. **Choose the Number of Clusters (K)**

   The first step is to decide how many clusters you want the algorithm to find in your data. This number is called **"K"** in K-means clustering. In this example, we set **K = 3**, meaning we want to group the data into **three clusters**.

2. **Randomly Select K Data Points as Initial Centroids**

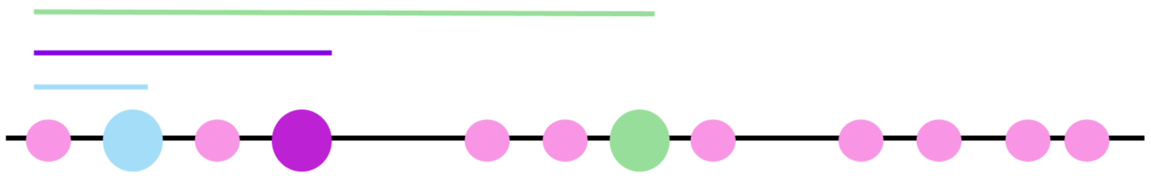
   These are just starting guesses for the center of each cluster. So, for K = 3, we randomly pick **three initial centroids** from the dataset.



Initial centroids.

3. **Measure the Distance Between Each Data Point and the Centroids**

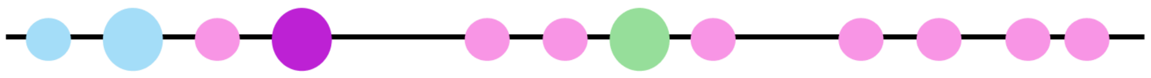
   For every data point in the dataset, calculate the distance to each of the **three centroids** (typically using **Euclidean distance**). For example, start measuring the distance between the 1st point to the 3 initial clusters.



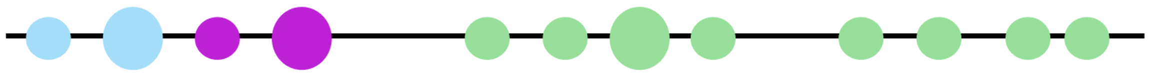
Distance between the first point to the 3 centroids.

4. **Assign Each Data Point to the Nearest Cluster**

   Based on the calculated distances, each data point is assigned to the **nearest centroid**, forming **three groups (clusters)**. Following the example, we need to assign the first point to the nearest cluster that, in this case, is the blue cluster.



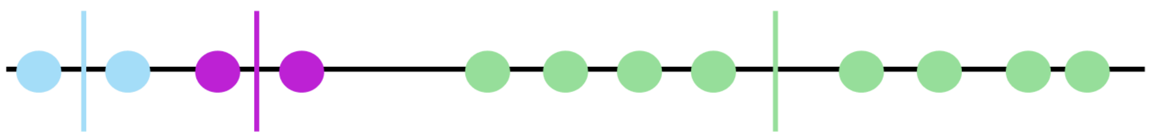
Assigning the first point to its nearest cluster.



All points assigned.

5. **Update the Centroids**

   For each cluster, calculate the **new centroid** by taking the **mean (average position)** of all data points assigned to that cluster. These new centroids now represent the **new center of each cluster**.



6. **Repeat Steps 3 to 5 Until Convergence**

   Recalculate distances from all data points to the new centroids and reassign the points if needed. This process repeats until the **cluster assignments stop changing**, meaning the algorithm has **converged (ended)**.

Evaluating Clustering Quality

Once convergence is reached, the **quality of clustering** can be assessed by calculating the **total variation (intra-cluster variance)**—that is, how close the data points are to their cluster centroids.

>> A **lower total variance** indicates a **better clustering result.**

Note:

K-means does not guarantee the best clustering on the first try. Since the result depends on the initial choice of centroids, it can sometimes get stuck in a **local minimum** (i.e., a suboptimal grouping). To solve this, K-means is typically **run multiple times with different random initializations**, and **the solution with the lowest total variance is chosen as the final result**.

How do you find the best value for “K”?

There is no single “correct” value for **K**, but we can try to estimate the best number of clusters by comparing how the clustering performance changes with different values of **K**.

Trying Different Numbers for K

The most common approach is to run the K-means algorithm **multiple times**, each time using a different value for **K**, and then **compare the total variation** (also called **Within-Cluster Sum of Squares – WCSS**) for each value. Using the last example:

- K = 1 is the worst scenario.
    - When **K = 1**, **all data points are grouped into a single cluster**, so the **total variation** (WCSS) corresponds to **all the variability present in the dataset**.

Mathematically

- The **maximum variation** is simply the **sum of squared distances between each data point and the overall centroid (the global mean of the dataset)**. This is called **Total Sum of Squares (TSS)**.
- **Total Sum of Squares** measures the **total variance in the dataset**, **without** considering clusters.
- It's calculated as:

$$TSS = \sum_{i=1}^{n} \|x_i - \bar{x}\|^2$$

  Where:

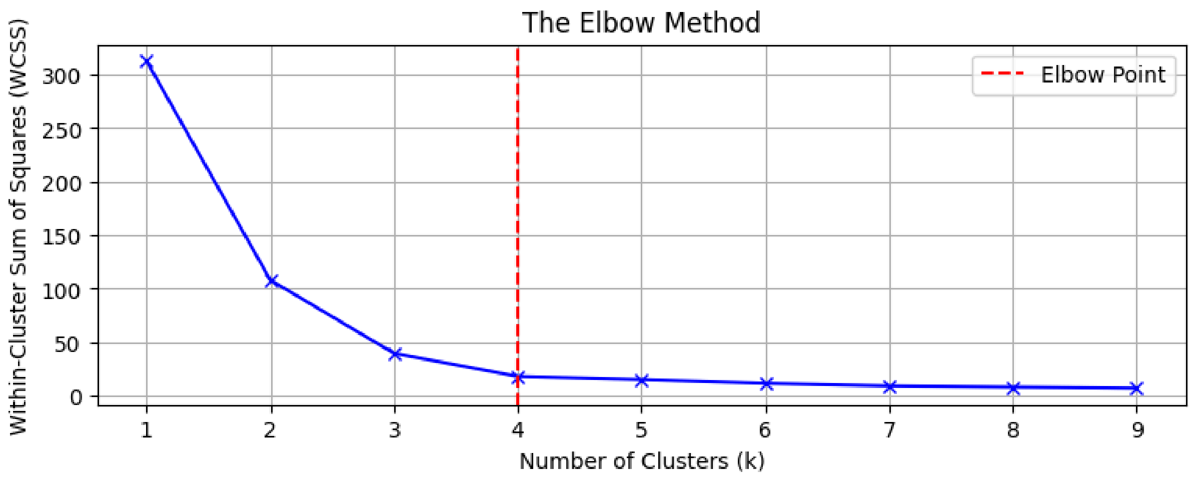
  - x_i is each data point.
  - x¯ is the overall mean (centroid),
  - || this part || is the **squared distance** between each point and the global centroid.

This **TSS (when K = 1)** represents the **maximum variation** in the dataset — it's the **worst-case scenario for K-means**, because **no structure has been discovered in the data yet**.

- As you **increase the value of K**, K-means tries to **split this total variation into smaller portions**, minimizing the **within-cluster variation** (WCSS). In other words, each time you add a cluster, the goal is to make clusters more compact and reduce how far each point is from its cluster center. Eventually, if **K = N (number of data points)**, then each cluster has only one point and **WCSS = 0**, but that's not useful in practice — the goal is to find the best balance using a method like the **Elbow Method** to determine the optimal number of clusters.

Elbow Method

The **Elbow Method** is a technique used to determine the optimal number of clusters (**K**) in K-means clustering. It works by **calculating the Within-Cluster Sum of Squares (WCSS)** for different values of **K**, and then plotting these values on a graph.



- Initially, as **K** increases, the variation within clusters decreases rapidly, because more clusters allow better grouping of similar data points. However, at a certain point, the rate of improvement slows down — this creates an **"elbow" shape** in the curve. The point at which this **elbow appears** is considered the best choice for **K**, because it represents a **good trade-off between cluster compactness and model simplicity**. In other words:
    - For each data point, we measure the euclidean distance from that data point to its cluster center. The elbow point shows us that by increasing the number of clusters more, we're getting a less and less ideal decreasing of the sum of squared error. On one hand, you don't want to have too many clusters, but neither too few.

Mathematically

- **SSE (Sum of Squared Errors or WCSS)** measures the **total variation within each cluster**, i.e., how far the data points are from the centroid of their respective cluster.

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

Where:

- k is the number of clusters.
- C_i is the set of points in cluster i.
- μ_i is the **centroid** (mean) of cluster i.
- x $\in C_i$: each individual data point x that belongs to cluster i.
- || this part || is the squared Euclidean distance between the point and its cluster centroid.

In simpler terms:

> For each cluster, calculate the distance between every point and the cluster center (centroid), square that distance, and sum all those squared distances. Then, add up the totals for all clusters — this gives you the SSE.

A lower SSE means the points are closer to their centroids → more compact clusters.

Silhouette Method

The **Silhouette Method** is a technique used to **measure the quality of clustering**. It works by comparing **how close each point is to its own cluster (cohesion)** versus **how far it is from the other clusters (separation)**.

- It only works when the number of clusters K ≥ 2.
- The higher the value of the silhouette score, the better it divides the original dataset into different clusters (the clusters are well-separated and compact).

Steps

1. **For Each Data Point i**

   Calculate a(i)

   - The **average distance** $d(i, j)$ between point i and all other points **in the same cluster A**.
   - This measures **intra-cluster cohesion** (how compact the cluster is).

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

   Calculate b(i)

   - The **minimum average distance** between point i and **all points in the nearest neighboring cluster (**i.e., the next closest cluster).
   - This measures **inter-cluster separation** (how far it is from the other clusters).

   $$b(i) = \min_{C_k \neq C_i} \left(\frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \right)$$

   Compute the Silhouette Coefficient s(i)

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

s(i) is the silhouette coefficient of the data point **o**

- ***a(i)*** is the *average distance* between i and all the other data points in the cluster to which **o** belongs.
- ***b(i)*** is the *minimum average distance* from **o** to all clusters to which **o** does not belong.

Silhouette Score

The value of the silhouette coefficient is between **[-1, 1]**.

- A score of 1 shows us that the data point i is very compact within the cluster to which it belongs and far away from the other clusters**.**
- The worst value is -1.
- Values near 0 denote overlapping clusters.

If most objects have high quality then the clustering configuration is appropriate. If many points have a low or negative value then the configuration is bad.

- The **best K** is typically the one that **maximizes the average silhouette score**.

Graph Analysis - Note

Check the Jupyter Notebook in this folder to see the graphs.

- Each colourful block is a cluster.
- The horizontal width shows the coefficient value for each point inside that cluster.
- Red line: general average silhouette. It shows the average silhouette of all the points.
    - More to the right → better the separation between the clusters.