Clustering

Session 08

1 Introduction

Clustering is an *unsupervised machine learning* technique that groups similar data points together into clusters based on their characteristics, without using any labels. The objective is to ensure that data points within the same cluster are more similar to each other than to those in different clusters, enabling the discovery of natural groupings and hidden patterns.

2 K-Means

- Centroid-based clustering method that partitions data into k clusters.
- Objective: minimize *inertia* (the within-cluster sum of squared distances):

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

  where μ_i is the centroid (mean) of cluster C_i.
- The inertia is non-increasing during the algorithm (it decreases or stays the same at each iteration).
- Iterative algorithm with two alternating steps:
  - **Assignment step:** assign each point to the nearest centroid based on distance metric (usually Euclidean distance).
  - **Update step:** recompute each centroid as the mean of the assigned points.
- Initialization matters, and different runs can lead to different results.
- Works best for well-separated clusters; sensitive to outliers and feature scaling.

2.1 Steps

2.1.1 Centroid Initialization

1. **Random initialization:** choose k centroids by sampling k points at random from the dataset.

2. **k-means++ initialization:** choose the first centroid uniformly at random, then choose the next centroids with probability proportional to the squared distance from the closest already chosen centroid. This typically improves convergence and final quality.

2.1.2 Assigning points to centroids

1. For each point x_i, compute its distance to all centroids $\mu_1, \ldots, \mu_k$ and assign it to the nearest centroid:

$$c_i = \arg \min_{j \in \{1,\ldots,k\}} d(x_i, \mu_j)$$

2. **Distance metric:** most commonly Euclidean distance:

$$d(x_i, \mu_j) = \|x_i - \mu_j\|_2 = \sqrt{\sum_{t=1}^{d} (x_{it} - \mu_{jt})^2}$$

2.1.3 Recomputing the Centroids

After assigning points to clusters, each centroid is updated as the mean of the points in its cluster:

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i, \quad j = 1, \ldots, k$$

2.1.4 Stopping Conditions

- A maximum number of iterations is reached.
- Cluster assignments no longer change between iterations.
- Centroids no longer change between iterations.

2.1.5 Choosing k

In K-Means, k denotes the number of clusters. Because k is usually unknown, we run K-Means for multiple values of k and plot the inertia. The optimal k is often chosen using the *elbow method*, i.e., the value where the decrease in inertia starts to slow down significantly.

2.2 Pseudocode

Input: $x_1, \ldots, x_n \in \mathbb{R}^d$ and $K \in \mathbb{N}^*$.
Output: a K-partition of $\{x_1, \ldots, x_n\}$, i.e., a decomposition of this set into a collection of K disjoint sets (not necessarily nonempty).

2.3 Pseudocode

Algorithm 1 K-Means

Require: $x_1, \ldots, x_n \in \mathbb{R}^d$, $K \in \mathbb{N}^*$
Ensure: A K-partition $\{C_1, \ldots, C_K\}$ of $\{x_1, \ldots, x_n\}$

Choose initial centroids $\mu_1, \ldots, \mu_K$ (e.g., randomly from the data)
repeat ▷ Assignment step
  for $i = 1$ to n **do**
    $c_i \leftarrow \arg\min_{j \in \{1,\ldots,K\}} \|x_i - \mu_j\|^2$
  end for ▷ Update step
  for $j = 1$ to K **do**
    $C_j \leftarrow \{x_i \mid c_i = j\}$
    $\mu_j \leftarrow \dfrac{1}{|C_j|} \sum_{x_i \in C_j} x_i$
  end for
until termination condition
return $C_1, \ldots, C_K$

2.4 Example

Consider, in the two-dimensional Euclidean plane, a dataset consisting of the following points: $A(-1, 0)$, $B(1, 0)$, $C(0, 1)$, $D(3, 0)$, and $E(3, 1)$. Manually run 2-Means considering as initial centroids points $(-1, 0)$ and $(3, 1)$.

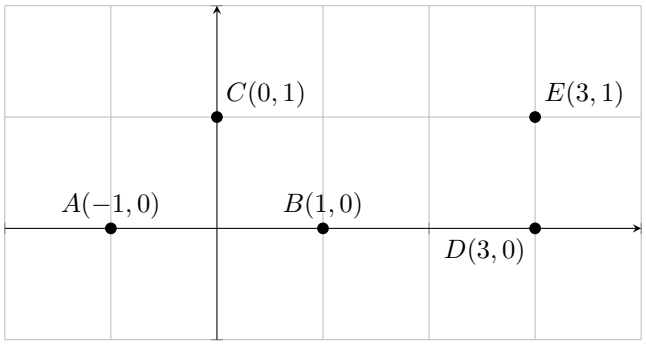

Figure 1: Dataset points.

We have the initial centroids:

$$\mu_1^{(0)} = (-1, 0), \qquad \mu_2^{(0)} = (3, 1).$$

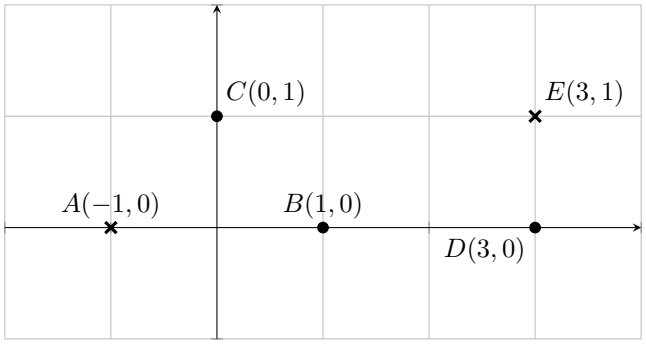

Figure 2: Initial centroids.

We compute the distances:

$$d\big((x, y), (a, b)\big) = \sqrt{(x - a)^2 + (y - b)^2}.$$

$$d(B, \mu_1^{(0)}) = d\big((1, 0), (-1, 0)\big) = \sqrt{(1 - (-1))^2 + (0 - 0)^2} = \sqrt{4 + 0} = 2$$

$$d(B, \mu_2^{(0)}) = d\big((1, 0), (3, 1)\big) = \sqrt{(1 - 3)^2 + (0 - 1)^2} = \sqrt{4 + 1} = \sqrt{5}$$

Since $d(B, \mu_1^{(0)}) < d(B, \mu_2^{(0)})$, point B is assigned to cluster $C_1^{(1)}$ (centroid $\mu_1^{(0)}$).

$$d(C, \mu_1^{(0)}) = d\big((0, 1), (-1, 0)\big) = \sqrt{(0 - (-1))^2 + (1 - 0)^2} = \sqrt{1 + 1} = \sqrt{2}$$

$$d(C, \mu_2^{(0)}) = d\big((0, 1), (3, 1)\big) = \sqrt{(0 - 3)^2 + (1 - 1)^2} = \sqrt{9 + 0} = 3$$

Since $d(C, \mu_1^{(0)}) < d(C, \mu_2^{(0)})$, point C is assigned to cluster $C_1^{(1)}$ (centroid $\mu_1^{(0)}$).

$$d(D, \mu_1^{(0)}) = d\big((3, 0), (-1, 0)\big) = \sqrt{(3 - (-1))^2 + (0 - 0)^2} = \sqrt{16 + 0} = 4$$

$$d(D, \mu_2^{(0)}) = d\big((3, 0), (3, 1)\big) = \sqrt{(3 - 3)^2 + (0 - 1)^2} = \sqrt{0 + 1} = 1$$

Since $d(D, \mu_2^{(0)}) < d(D, \mu_1^{(0)})$, point D is assigned to cluster $C_2^{(1)}$ (centroid $\mu_2^{(0)}$).

We have the clusters:

- $C_1^{(1)} = \{A, B, C\}$
- $C_2^{(1)} = \{D, E\}$

The new centroids are computed as the mean of the points in each cluster:

$$\mu_1^{(1)} = \frac{1}{|C_1^{(0)}|} \sum_{x \in C_1^{(0)}} x = \frac{1}{3}\Big((-1,0)+(1,0)+(0,1)\Big) = \left(\frac{-1+1+0}{3}, \frac{0+0+1}{3}\right) = \left(0, \frac{1}{3}\right).$$

$$\mu_2^{(1)} = \frac{1}{|C_2^{(0)}|} \sum_{x \in C_2^{(0)}} x = \frac{1}{2}\Big((3,0) + (3,1)\Big) = \left(\frac{3+3}{2}, \frac{0+1}{2}\right) = \left(3, \frac{1}{2}\right).$$

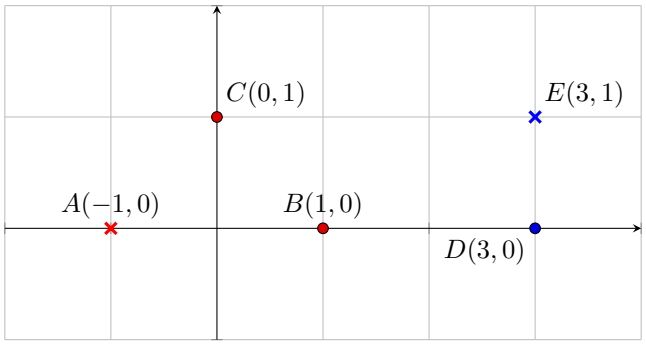

Figure 3: Initial clusters

Using the updated centroids

$$\mu_1^{(1)} = \left(0, \frac{1}{3}\right), \qquad \mu_2^{(1)} = \left(3, \frac{1}{2}\right),$$

we reassign each point to the nearest centroid (Euclidean distance).

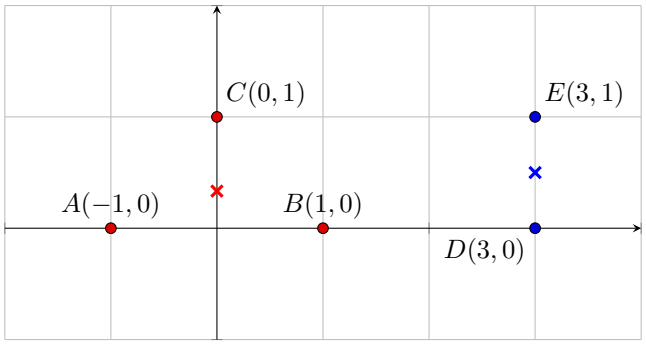

Figure 4: New centroids

We have finished the first iteration. We start iteration 2.

$$d\Big(A, \mu_1^{(1)}\Big) = \sqrt{(-1-0)^2 + \left(0 - \frac{1}{3}\right)^2} = \sqrt{1 + \frac{1}{9}} = \frac{\sqrt{10}}{3},$$

$$d\Big(A, \mu_2^{(1)}\Big) = \sqrt{(-1-3)^2 + \left(0 - \frac{1}{2}\right)^2} = \sqrt{16 + \frac{1}{4}} = \frac{\sqrt{65}}{2}.$$

Thus, A is assigned to $C_1^{(2)}$.

$$d\Big(B, \mu_1^{(1)}\Big) = \sqrt{(1-0)^2 + \left(0 - \frac{1}{3}\right)^2} = \sqrt{1 + \frac{1}{9}} = \frac{\sqrt{10}}{3},$$

$$d\Big(B, \mu_2^{(1)}\Big) = \sqrt{(1-3)^2 + \left(0 - \frac{1}{2}\right)^2} = \sqrt{4 + \frac{1}{4}} = \frac{\sqrt{17}}{2}.$$

Thus, B is assigned to $C_1^{(2)}$.

$$d\Big(C, \mu_1^{(1)}\Big) = \sqrt{(0-0)^2 + \left(1 - \frac{1}{3}\right)^2} = \sqrt{\left(\frac{2}{3}\right)^2} = \frac{2}{3},$$

$$d\Big(C, \mu_2^{(1)}\Big) = \sqrt{(0-3)^2 + \left(1 - \frac{1}{2}\right)^2} = \sqrt{9 + \frac{1}{4}} = \frac{\sqrt{37}}{2}.$$

Thus, C is assigned to $C_1^{(2)}$.

$$d\Big(D, \mu_1^{(1)}\Big) = \sqrt{(3-0)^2 + \left(0 - \frac{1}{3}\right)^2} = \sqrt{9 + \frac{1}{9}} = \frac{\sqrt{82}}{3},$$

$$d\Big(D, \mu_2^{(1)}\Big) = \sqrt{(3-3)^2 + \left(0 - \frac{1}{2}\right)^2} = \sqrt{\frac{1}{4}} = \frac{1}{2}.$$

Thus, D is assigned to $C_2^{(2)}$.

$$d\Big(E, \mu_1^{(1)}\Big) = \sqrt{(3-0)^2 + \left(1 - \frac{1}{3}\right)^2} = \sqrt{9 + \left(\frac{2}{3}\right)^2} = \sqrt{9 + \frac{4}{9}} = \frac{\sqrt{85}}{3},$$

$$d\Big(E, \mu_2^{(1)}\Big) = \sqrt{(3-3)^2 + \left(1 - \frac{1}{2}\right)^2} = \sqrt{\frac{1}{4}} = \frac{1}{2}.$$

Thus, E is assigned to $C_2^{(2)}$.

The new clusters:

$$C_1^{(2)} = \{A, B, C\}, \qquad C_2^{(2)} = \{D, E\}.$$

We observe that the cluster assignments did not change, so the centroids will remain the same. Therefore, the algorithm has converged.

3 Spectral Clustering

Because K-means assumes convex, centroid-based clusters, it often fails on non-convex structures; Spectral Clustering addresses this by using similarity-based connectivity.
Spectral Clustering is a clustering method that uses the **connectivity** (similarity) between data points to form clusters, instead of relying only on distances to centroids.

3.1 Main Idea

Build a **similarity graph** of the data, use the graph's **Laplacian matrix** to obtain a lower-dimensional representation (via eigenvectors), then cluster in that new space (often with K-Means).

3.2 Steps

1. **Build a similarity graph.** Represent each data point as a node and connect nodes that are similar, obtaining the similarity (adjacency) matrix $W = (w_{ij})$. Two common ways to build this graph are:

   - **ε-distance (radius) graph:** connect x_i and x_j if their distance is at most ε:
   $$w_{ij} = \begin{cases} 1, & \text{if } \|x_i - x_j\| \le \varepsilon, \\ 0, & \text{otherwise.} \end{cases}$$
   A weighted version can also be used, for example the RBF kernel:
   $$w_{ij} = \exp\left(-\gamma \|x_i - x_j\|^2\right),$$
   where $\gamma > 0$ controls how fast the similarity decreases with distance.
   - **k-nearest neighbors (k-NN) graph:** connect each point x_i to its k nearest neighbors. Then
   $$w_{ij} = \begin{cases} 1, & \text{if } x_j \in \text{kNN}(x_i) \text{ (or } x_i \in \text{kNN}(x_j)), \\ 0, & \text{otherwise.} \end{cases}$$
   Again, a weighted version can be used (e.g., using the same RBF kernel on the selected edges).

2. **Compute the degree matrix.** The degree of node i is the sum of its edge weights:
$$d_i = \sum_{j=1}^{n} w_{ij}, \qquad D = \text{diag}(d_1, \ldots, d_n).$$

3. **Compute the graph Laplacian.** The (unnormalized) graph Laplacian is:

$$L = D - W.$$

In practice, a normalized version is also common, for example the symmetric normalized Laplacian:

$$L_{\text{sym}} = D^{-1/2} L D^{-1/2}.$$

Since $D = \text{diag}(d_1, \ldots, d_n)$ is diagonal, its inverse square root is

$$D^{-1/2} = \text{diag}\left(\frac{1}{\sqrt{d_1}}, \ldots, \frac{1}{\sqrt{d_n}}\right).$$

It is called *normalized* because it rescales similarities by node degrees:

$$\left(D^{-1/2} L D^{-1/2}\right)_{ij} = \frac{l_{ij}}{\sqrt{d_i\, d_j}},$$

so highly connected nodes do not dominate the clustering.

4. **Eigenvector embedding (dimensionality reduction).**

Compute eigenvalues/eigenvectors of the Laplacian and take the k eigenvectors corresponding to the **smallest** eigenvalues (where k is the number of clusters). Stack them into a matrix

$$U \in \mathbb{R}^{n \times k},$$

where row i is a new k-dimensional representation of point x_i.

Eigenvalues and eigenvectors. Let $A \in \mathbb{R}^{n \times n}$. A nonzero vector $v \in \mathbb{R}^n$ is an eigenvector of A if there exists a scalar $\lambda \in \mathbb{R}$ such that

$$Av = \lambda v.$$

The scalar λ is the corresponding eigenvalue. (Eigenvalues satisfy $\det(A - \lambda I_n) = 0$.)

Example (computing eigenvalues and eigenvectors).

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

Eigenvalues satisfy the characteristic equation

$$Av = \lambda v \implies (A - \lambda I_2)v = 0$$

$$\det(A - \lambda I_2) = 0.$$

Compute:

$$A - \lambda I_2 = \begin{pmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{pmatrix}, \qquad \det(A - \lambda I_2) = (2-\lambda)^2 - 1.$$

Set to zero:

$$(2-\lambda)^2 - 1 = 0 \iff (2-\lambda)^2 = 1 \iff 2 - \lambda = \pm 1.$$

Thus the eigenvalues are:

$$\lambda_1 = 1, \qquad \lambda_2 = 3.$$

Eigenvector for $\lambda_1 = 1$:

$$(A - I_2)v = 0 \implies \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \implies \begin{pmatrix} x+y \\ y+y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \implies x = -y.$$

Let $x = a$ (with $a \neq 0$), then $y = -a$, so an eigenvector is

$$v_1 = (a, -a)^\top.$$

$a = 1$: $v_1 = (1, -1)^\top$.

Eigenvector for $\lambda_2 = 3$:

$$(A - 3I_2)v = 0 \implies \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \implies \begin{pmatrix} -x+y \\ x-y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \implies y = x.$$

Let $x = b$ (with $b \neq 0$), then $y = b$, so an eigenvector is

$$v_1 = (b, b)^\top.$$

For $b = 1$: $v_2 = (1, 1)^\top$.

5. **Cluster in the new space.** Run a standard clustering algorithm (typically K-Means) on the rows of U to obtain the final clusters.