Unsupervised Learning with K-Means

Dr. EL BENANY Mohamed Mahmoud

January 13, 2025

Presentation plan

How does the Elbow Method work?

The Elbow Method is a simple but effective technique used to determine the optimal number of clusters (K) in a K-Means clustering algorithm. It examines the relationship between the number of clusters and the within-cluster sum of squares (WCSS), a measure of the variance within each cluster.
Identifying the ideal number of clusters is a crucial step in this algorithm. The Elbow Method is a widely used technique to determine the optimal value of K.

K Means Clustering Using the Elbow Method

Identifying the ideal number of clusters is a crucial step in this algorithm. The Elbow Method is a widely used technique to determine the optimal value of K.

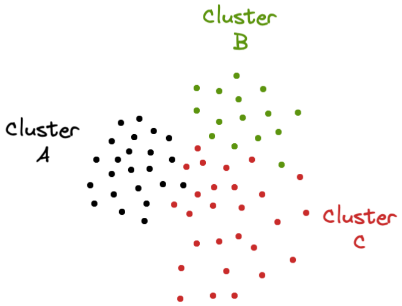
K Means Clustering Using the Elbow Method

In the Elbow Method, we systematically experiment with different numbers of clusters (K) ranging from 1 to 10. With each K value, we compute the Within-Cluster Sum of Squares (WCSS). When we plot WCSS against K, the resulting graph resembles an elbow. As we increase the number of clusters, the WCSS value begins to decrease. Notably, the WCSS is at its highest when K=1.

- Elbow Method.
- Optimization Problem
- Max inter-claster Distance
- Min intea-claster Distance
- Mutually exclusive
- Centroid-based
- Distance Function

INTRODUCTION - What is clustering?

- **Clustering** is the classification of objects into different groups.
- More precisely, it refers to the partitioning of a data set into subsets (**clusters**).
- Data in each subset (ideally) share some common trait, often according to a defined **distance measure**.



Types of clustering

- **Hierarchical algorithms**: These find successive clusters using previously established clusters.
  - **Agglomerative ("bottom-up")**: Begin with each element as a separate cluster and merge them into successively larger clusters.
  - **Divisive ("top-down")**: Begin with the whole set and proceed to divide it into successively smaller clusters.
- **Partitional clustering**: Partitional algorithms determine all clusters at once. They include:
  - **K-means and derivatives**
  - **Fuzzy c-means clustering**
  - **QT clustering algorithm**

Types of Distances

Definition: Distance measures determine how the similarity or dissimilarity between two elements is calculated. These measures influence the structure and shape of the resulting clusters in clustering algorithms.

Objective: The goal of distance measures is to:

- Quantify the closeness or separation between data points.
- Enable clustering algorithms to group similar data points together.
- Adapt to different types of data, including numerical, categorical, or mixed data.

Types of Distances

Definition: Distance measures determine how the similarity or dissimilarity between two elements is calculated. These measures influence the structure and shape of the resulting clusters in clustering algorithms.

Objective: The goal of distance measures is to:

- Quantify the closeness or separation between data points.
- Enable clustering algorithms to group similar data points together.
- Adapt to different types of data, including numerical, categorical, or mixed data.

Euclidean Distance

Formula:

$$d_{\text{Euclidean}}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

When Used:

- Used for numerical data in clustering algorithms like K-means.
- Suitable when the scale of features is uniform.

Example: If $x = (2, 3)$ and $y = (5, 7)$, the Euclidean distance is:

$$d_{\text{Euclidean}} = \sqrt{(5-2)^2 + (7-3)^2} = \sqrt{9+16} = 5$$

Manhattan Distance

Formula:

$$d_{\text{Manhattan}}(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

When Used:

- Suitable for grid-based systems, such as navigating a city where movement is restricted to orthogonal directions.

Example: If $x = (2, 3)$ and $y = (5, 7)$, the Manhattan distance is:

$$d_{\text{Manhattan}} = |5 - 2| + |7 - 3| = 3 + 4 = 7$$

Maximum Norm (Chebyshev Distance)

Formula:

$$d_{\text{Max}}(x, y) = \max_{i=1,\ldots,n} |x_i - y_i|$$

When Used:

- Useful in applications where the largest difference between dimensions dominates, such as board games like chess.

Example: If $x = (2, 3)$ and $y = (5, 7)$, the maximum norm is:

$$d_{\text{Max}} = \max(|5 - 2|, |7 - 3|) = \max(3, 4) = 4$$

Mahalanobis Distance

Formula:

$$d_{\text{Mahalanobis}}(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$$

When Used:

- Used when data have different scales or are correlated.
- Common in anomaly detection and multivariate statistical analyses.

Example: If $x = (1, 2)$, $y = (3, 4)$, and $\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, then:

$$d_{\text{Mahalanobis}} = \sqrt{(x - y)^T \Sigma^{-1} (x - y)} = \sqrt{2}$$

Cosine Distance

Formula:

$$\text{Cosine Similarity: } \text{sim}(x, y) = \frac{x \cdot y}{\|x\|\|y\|}$$

$$\text{Cosine Distance: } d_{\text{Cosine}} = 1 - \text{sim}(x, y)$$

When Used:

- Common for high-dimensional data, such as text data or document similarity.

Example: If $x = (1, 0, 1)$ and $y = (0, 1, 1)$, the cosine similarity is:

$$\text{sim}(x, y) = \frac{1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1}{\sqrt{1^2 + 0^2 + 1^2} \cdot \sqrt{0^2 + 1^2 + 1^2}} = \frac{1}{\sqrt{2} \cdot \sqrt{2}} = 0.5$$

The cosine distance is $1 - 0.5 = 0.5$.

Hamming Distance

Formula:

$$d_{\text{Hamming}}(x, y) = \sum_{i=1}^{n} \mathbb{1}(x_i \neq y_i)$$

When Used:

- Used for categorical or binary data, such as comparing strings or DNA sequences.

Example: If $x = (1, 0, 1, 1)$ and $y = (1, 1, 0, 1)$, the Hamming distance is:

$$d_{\text{Hamming}} = \mathbb{1}(1 \neq 1) + \mathbb{1}(0 \neq 1) + \mathbb{1}(1 \neq 0) + \mathbb{1}(1 \neq 1) = 0+1+1+0 = 2$$

K-MEANS CLUSTERING

- The **k-means algorithm** is an algorithm to cluster n objects based on attributes into k partitions, where $k < n$.
- It is similar to the **expectation-maximization algorithm** for mixtures of Gaussians, as both aim to find the centers of natural clusters in the data.
- Assumes that the object attributes form a **vector space**.

Partitioning Algorithm for Clustering

- An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points.
- The goal is to minimize the **sum-of-squares criterion**:

$$\sum_{j=1}^{K} \sum_{x_n \in S_j} \|x_n - \mu_j\|^2$$

where:

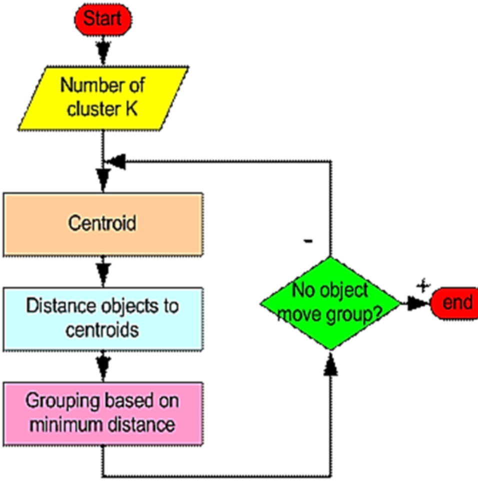
- x_n is a vector representing the n-th data point.
- μ_j is the **geometric centroid** of the data points in S_j.

Simply Speaking: K-Means Clustering

- Simply speaking, **k-means clustering** is an algorithm to classify or group the objects based on attributes/features into K number of groups.
- K is a positive integer number.
- The grouping is done by minimizing the sum of squares of distances between the data and the corresponding **cluster centroid**.

How the K-Means Clustering Algorithm Works?

- The algorithm starts by selecting K initial centroids (either randomly or based on some heuristics).
- Each data point is assigned to the nearest centroid, forming K clusters.
- After all points have been assigned, the centroids are recalculated as the mean of all data points in each cluster.
- The assignment and recalculation steps are repeated iteratively until convergence (i.e., centroids do not change significantly).


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

Step 1: Begin with a Decision on k and Step 2: Initial Partition

- **Step 1**: Begin with a decision on the value of k, the number of clusters.
- **Step 2**: Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly, or systematically as follows:
  1. Take the first k training samples as single-element clusters.
  2. Assign each of the remaining $(N - k)$ training samples to the cluster with the nearest centroid.
  3. After each assignment, recompute the centroid of the gaining cluster.

Step 3 and Step 4: Update Clusters and Convergence

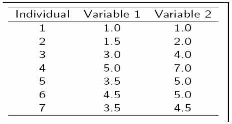
- **Step 3**: Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster. After this, update the centroid of the cluster gaining the new sample and the cluster losing the sample.
- **Step 4**: Repeat Step 3 until convergence is achieved, i.e., until a pass through the training samples causes no new assignments.

A Simple Example of K-Means Algorithm (using $K = 2$)

Step 1: Initialization

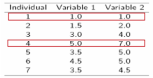
- Randomly, we choose the following two centroids for $k = 2$ clusters:
  1. $m_1 = (1.0, 1.0)$ (centroid of the first cluster)
  2. $m_2 = (5.0, 7.0)$ (centroid of the second cluster)

These centroids will be used to classify the data points into two clusters in the subsequent steps.

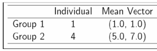


Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

centroid



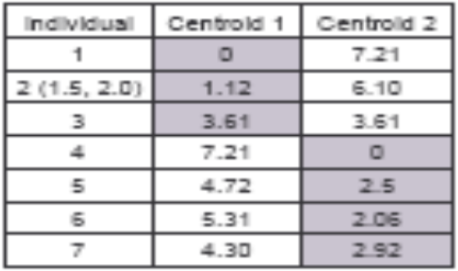
Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5



	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Step 2: Assigning Points to Clusters and Updating Centroids

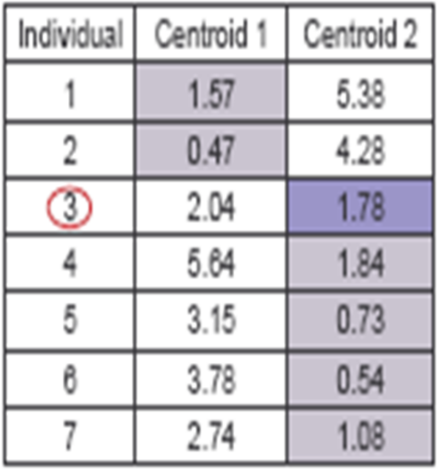
- After the initial assignment, we obtain two clusters containing:
  1. Cluster 1: $\{1, 2, 3\}$
  2. Cluster 2: $\{4, 5, 6, 7\}$
- The new centroids of these clusters are calculated as the mean of the data points in each cluster:
- New centroid of Cluster 1 (m_1): Average of points 1, 2, 3.
- New centroid of Cluster 2 (m_2): Average of points 4, 5, 6, 7.



Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

Step 3: Calculating Euclidean Distances and Updating Clusters

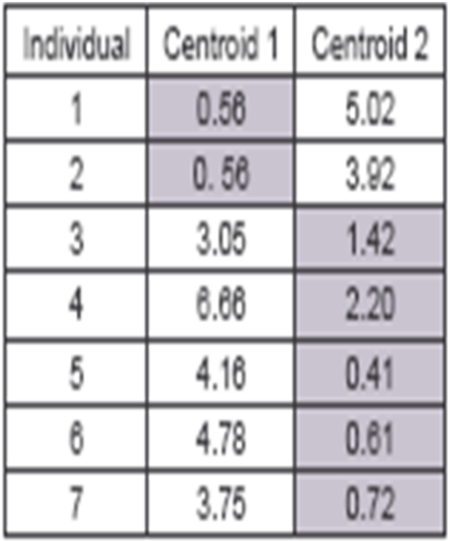
- Now, using these centroids, we compute the Euclidean distance of each object to the centroids, as shown in the table below.
- Based on the computed distances, the new clusters are:
  1. Cluster 1: {1, 2}
  2. Cluster 2: {3, 4, 5, 6, 7}
- The next centroids are updated as follows:
  - New centroid of Cluster 1 (m_1): $(1.25, 1.5)$
  - New centroid of Cluster 2 (m_2): $(3.9, 5.1)$



Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

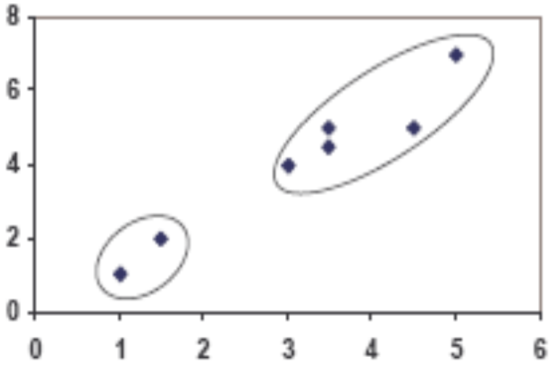
Step 4: Final Clusters and Stopping Condition

- The clusters obtained after the last update are:
  1. Cluster 1: $\{1, 2\}$
  2. Cluster 2: $\{3, 4, 5, 6, 7\}$
- Since there is no change in the clusters after this step, the algorithm comes to a halt here.
- The final result consists of 2 clusters:
  1. Cluster 1: $\{1, 2\}$
  2. Cluster 2: $\{3, 4, 5, 6, 7\}$



Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0. 56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72

PLOT



Weaknesses of K-Means Clustering

- When the number of data points is not large, the initial grouping can significantly determine the final clusters.
- The number of clusters, k, must be determined beforehand.
- The algorithm does not yield the same result with each run, as the resulting clusters depend on the initial random assignments.
- The true clusters are unknown because, when using the same data, the clustering results may vary if the data is inputted in a different order (especially when the dataset is small).
- It is sensitive to the initial conditions. Different initial conditions may produce different clustering results.
- The algorithm may be trapped in a local optimum.

Applications of K-Means Clustering

K-means clustering is relatively efficient and fast. It computes results at $O(tkn)$, where:

- n is the number of objects or points,
- k is the number of clusters,
- t is the number of iterations.

K-means clustering can be applied to various domains, such as:

- Machine learning or data mining
- Acoustic data in speech understanding to convert waveforms into one of k categories (known as Vector Quantization)
- Image Segmentation
- Choosing color palettes on old-fashioned graphical display devices
- Image Quantization

Conclusion

K-means algorithm is useful for undirected knowledge discovery and is relatively simple. K-means has found widespread usage in many fields, ranging from:

- Unsupervised learning of neural networks
- Pattern recognition
- Classification analysis
- Artificial intelligence
- Image processing
- Machine vision
- And many others

Real-Life Numerical Example of K-Means Clustering

We have 4 medicines as our training data points, each with 2 attributes representing coordinates: Weight Index (X) and pH (Y).

Object	Attribute 1 (X): Weight Index	Attribute 2 (Y): pH
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

The goal is to determine which medicines belong to cluster 1 and which belong to cluster 2.

Step 1: Initial Value of Centroids

Suppose we use Medicine A and Medicine B as the first centroids. Let the centroids be denoted as:

$$c_1 = (1,1) \quad \text{and} \quad c_2 = (2,1)$$

The distance matrix at iteration 0 is calculated as:

	Distance to c_1	Distance to c_2
Medicine A	$d(A, c_1)$	$d(A, c_2)$
Medicine B	$d(B, c_1)$	$d(B, c_2)$
Medicine C	$d(C, c_1)$	$d(C, c_2)$
Medicine D	$d(D, c_1)$	$d(D, c_2)$

Practice:

	Distance to c_1	Distance to c_2
Medicine A	0	1
Medicine B	1	0
Medicine C	3.162	2.236
Medicine D	5	3.162

Step 2: Objects Clustering

Based on the distance matrix, we assign each object to the closest centroid. The group matrix shows the assignments:

Medicine A	1	0
Medicine B	0	1
Medicine C	0	1
Medicine D	0	1

Medicine A is assigned to group 1, while Medicines B, C, and D are assigned to group 2.

Iteration 1: Objects-Centroids Distances

We calculate the distances of all objects to the new centroids. The new centroids are:

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1) \quad \text{and} \quad c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4.5, 3.5$$

new distance matrix at iteration 1 is:

	Distance to new c_1	Distance to new c_2
Medicine A	$d(A, c_1)$	$d(A, c_2)$
Medicine B	$d(B, c_1)$	$d(B, c_2)$
Medicine C	$d(C, c_1)$	$d(C, c_2)$
Medicine D	$d(D, c_1)$	$d(D, c_2)$

Practice:

	Distance to new c_1	Distance to new c_2
Medicine A	0.5	3.5
Medicine B	0.5	3.201
Medicine C	2.121	1.118
Medicine D	3.535	0.707

Iteration 1: Objects Clustering

Based on the new distance matrix, the new clustering is:

Medicine A	1	0
Medicine B	1	0
Medicine C	0	1
Medicine D	0	1

Medicine B is moved to Group 1 while all the other objects remain the same.

Iteration 2: Determine New Centroids

After iteration 1, the centroids are recalculated as:

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5, 1), \quad c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = (4.5, 3.5)$$

These centroids do not change from the previous iteration.

Iteration 2: Objects-Centroids Distances

We repeat the distance calculation for the new centroids:

Distance to new c_1	Distance to new c_2	
Medicine A	$d(A, c_1)$	$d(A, c_2)$
Medicine B	$d(B, c_1)$	$d(B, c_2)$
Medicine C	$d(C, c_1)$	$d(C, c_2)$
Medicine D	$d(D, c_1)$	$d(D, c_2)$

The distance matrix at iteration 2 remains the same as in iteration 1:

Distance to new c_1	Distance to new c_2	
Medicine A	0.5	3.5
Medicine B	0.5	3.201
Medicine C	2.121	1.118
Medicine D	3.535	0.707

Iteration 2: Objects Clustering

After the second iteration, the objects do not move between groups anymore, indicating convergence. The final result consists of:

$$\text{Group 1: } \{A, B\}, \quad \text{Group 2: } \{C, D\}$$

Exemple2:Data Overview

- We have 8 objects (A, B, C, D, E, F, G, H).
- Each object has two attributes:
  - Attribute 1 (**X**): Weight Index
  - Attribute 2 (**Y**): pH
- Data table:

Object	**X (Weight Index)**	**Y (pH)**
A	1.0	1.0
B	2.0	1.0
C	4.0	3.0
D	5.0	4.0
E	1.5	1.5
F	3.0	3.0
G	4.5	3.5
H	5.5	4.5

Step 1: Initialization

- Randomly select initial centroids for $k = 2$ clusters.
- Let $c_1 = (1.0, 1.0)$ and $c_2 = (5.0, 4.0)$.

Step 2: Distance Matrix at Iteration 0

- Compute Euclidean distances of all objects from the centroids:

$$\text{Distance} = \sqrt{(X_{object} - X_{centroid})^2 + (Y_{object} - Y_{centroid})^2}$$

Object	**Distance to c_1**	**Distance to c_2**
A	0.0	5.0
B	1.0	4.2
C	3.6	1.4
D	5.0	0.0
E	0.7	4.3
F	2.8	1.0
G	4.3	0.5
H	5.7	1.1

Step 3: Object Assignment

- Assign each object to the nearest centroid:
  - Group 1: A, B, E
  - Group 2: C, D, F, G, H
- Recalculate centroids:

$$c_1 = \left(\frac{1.0 + 2.0 + 1.5}{3}, \frac{1.0 + 1.0 + 1.5}{3}\right) = (1.5, 1.17)$$

$$c_2 = \left(\frac{4.0 + 5.0 + 3.0 + 4.5 + 5.5}{5}, \frac{3.0 + 4.0 + 3.0 + 3.5 + 4.5}{5}\right) = ($$

Step 4: Iteration 1

- Compute distances to updated centroids.
- Reassign objects based on new distances:
  - Group 1: A, B, E
  - Group 2: C, D, F, G, H
- New centroids remain the same:

$$c_1 = (1.5, 1.17), \quad c_2 = (4.4, 3.6)$$

- Algorithm converges.

Final Clusters

- Final grouping:
  - Group 1: A, B, E
  - Group 2: C, D, F, G, H
- Centroids:

$$c_1 = (1.5, 1.17), \quad c_2 = (4.4, 3.6)$$

Real-Life Example 3

Dataset (Attributes: Age and Income in \$1000)
We will use 8 customers, each represented by two attributes: **Age** and **Income**.

Customer	Age (X)	Income (Y)
C1	25	35
C2	30	45
C3	22	40
C4	35	65
C5	40	70
C6	50	90
C7	55	95
C8	60	100

Frame 2: Step 1: Initial Centroids

Initial Centroids:

Let Customer C_1 and Customer C_6 serve as the centroids:

- $c_1 = (25, 35)$
- $c_2 = (50, 90)$

Frame 3: Step 2: Calculate Distances

Using the Euclidean distance formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Distance Matrix (Iteration 0):

Each row represents distances to c_1 and c_2.

Customer	Distance to c_1	Distance to c_2
C1	0	64.03
C2	11.18	53.15
C3	5	63.95
C4	31.62	36.06
C5	44.72	22.36
C6	64.03	0
C7	74.33	11.18
C8	84.85	22.36

Frame 4: Step 3: Assign Groups

Assign each customer to the cluster with the smallest distance.
The grouping is:

Customer	Group
C1	1
C2	1
C3	1
C4	2
C5	2
C6	2
C7	2
C8	2

Frame 5: Step 4: Update Centroids

Calculate the new centroids:

- **Cluster 1:**

$$c_1 = \left(\frac{25 + 30 + 22}{3}, \frac{35 + 45 + 40}{3} \right) = (25.67, 40)$$

- **Cluster 2:**

$$c_2 = \left(\frac{35 + 40 + 50 + 55 + 60}{5}, \frac{65 + 70 + 90 + 95 + 100}{5} \right) = (48, 8$$

Frame 6: Step 5: Recalculate Distances

Using the updated centroids, calculate distances again:

Customer	Distance to c_1	Distance to c_2
C1	5.82	59.73
C2	5.15	49.77
C3	3.33	59.73
C4	22.64	24.74
C5	35.35	13.34
C6	55.97	5.66
C7	66.33	11.18
C8	76.69	22.36

Frame 7: Step 6: Reassign Groups

Group assignments after iteration:

Customer	Group
C1	1
C2	1
C3	1
C4	2
C5	2
C6	2
C7	2
C8	2

Frame 8: Final Clusters

The groups remain unchanged, indicating **convergence**. Final clusters:

- **Cluster 1:** C1, C2, C3
- **Cluster 2:** C4, C5, C6, C7, C8

Exercise 1

We want to classify the following points into three classes: $A1(2,10)$, $A2(2,5)$, $A3(8,4)$, $B1(5,8)$, $B2(7,5)$, $B3(6,4)$, $C1(1,2)$, $C2(4,9)$.
We initially assume that the points $A1$, $B1$, and $C1$ are chosen as centers. Use the K-means algorithm to determine:

(a) The three centers calculated after the first iteration.

(b) The three final classes resulting from the application of the algorithm.

Exercise 2

Consider 10 points $a, b, c, d, e, f, g, h, i, j$ in the plane with the coordinates given in the following table:

a	b	c	d	e	f	g	h	i	j
1	2	3	5	4	6	8	10	12	10
2	5	2	4	7	7	3	4	1	0

Give the stabilized partitions $ps1$ and $ps2$ corresponding to:

- for $ps1$, initial centers $\{a, b, i\}$
- for $ps2$, initial centers $\{a, d, i\}$

Run the algorithm directly on the provided graph. What can we conclude?

Exercise 3

Consider the 6 points: $M1 = (-2, 3), M2 = (-2, 1), M3 = (-2, -1), M4 = (2, -1), M5 = (2, 1)$, and $M6 = (1, 0)$. Assuming that the first two points, $M1$ and $M2$, are the initial centers, describe the K-medoids algorithm for the distribution of these points.