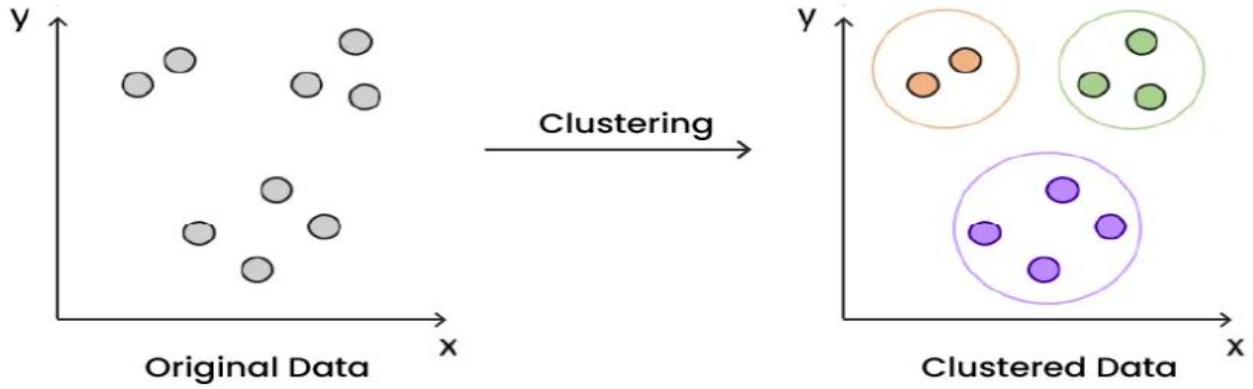
UNIT – V

Clustering: Introduction to Clustering, Partitioning of Data, Matrix Factorization, Clustering of Patterns, Divisive Clustering, Agglomerative Clustering, Partitional Clustering, K-Means Clustering, Soft Partitioning, Soft Clustering, Fuzzy C-Means Clustering, Rough Clustering, Rough K-Means Clustering Algorithm, Expectation Maximization-Based Clustering, Spectral Clustering.

Introduction to Clustering

Clustering is a fundamental technique in unsupervised machine learning that involves grouping similar data points together based on their characteristics. Unlike supervised learning

where we have labeled data, clustering algorithms discover natural groupings in data without any prior knowledge of categories



Clustering is the process of:

- Partitioning a dataset into groups (called clusters)
- Where data points in the same cluster are more similar to each other
- And data points in different clusters are more dissimilar

Popular Clustering Algorithms

1. **K-Means**: Partition-based, requires specifying number of clusters (k)
2. **Hierarchical**: Creates a tree of clusters (agglomerative or divisive)
3. **DBSCAN**: Density-based, good for irregular cluster shapes
4. **Gaussian Mixture Models**: Probabilistic approach using distributions
5. **Spectral Clustering**: Uses graph theory and eigenvalues

Common Applications

- Customer segmentation in marketing
- Image segmentation in computer vision
- Document clustering in text analysis
- Anomaly detection in cybersecurity
- Gene expression analysis in bioinformatics

Partitioning of Data

Partitioning of data is a clustering technique where a dataset is divided into a fixed number (**k**) of **non-overlapping groups (clusters)**.

Each data object:

- Belongs to **exactly one cluster**
- Cannot be shared between clusters

Definition

Partitioning methods organize **n data objects into k clusters** such that:

- Each cluster contains at least one object
- All objects are assigned to **one and only one cluster**

Objective

The main goal is to:

- **Minimize intra-cluster distance** (within cluster similarity is high)
- **Maximize inter-cluster distance** (clusters are well separated)

How It Works

1. Choose number of clusters (**k**)
2. Initialize cluster centers (randomly or intelligently)
3. Assign each data point to the nearest cluster
4. Update cluster centers
5. Repeat until clusters stabilize

Common Algorithms

1. K-Means

- Uses **centroids (mean values)**
- Fast and widely used

2. K-Medoids

- Uses **actual data points (medoids)**
- More robust to noise and outliers

Types of Partitioning Methods

1. Distance-Based Partitioning

These rely on distance measures like:

- Euclidean distance
- Manhattan distance
- Cosine similarity

Used in:

- **K-Means**
- **K-Medoids**

2. Density-Aware Partitioning (Hybrid Ideas)

Some advanced partitioning approaches incorporate density concepts:

- Try to avoid assigning sparse points incorrectly
- Example: **K-Means variants with density weighting**

3. Fuzzy Partitioning (Soft Clustering)

Fuzzy C-Means (FCM)

- Each data point belongs to **multiple clusters** with a degree of membership (0 to 1)
- Not strict partitioning (soft assignment)

Matrix Factorization

Matrix factorization is a mathematical technique that decomposes a large matrix into two lower-rank matrices whose product approximates the original, revealing latent patterns in the data. It's used in data science and unsupervised learning applications.

Matrix Factorization Techniques

1. LU Decomposition
2. QR Decomposition
3. Singular Value Decomposition (SVD)
4. Non-negative Matrix Factorization (NMF)

1. LU Decomposition

LU decomposition factorizes a matrix into two triangular matrices: (L), a lower triangular matrix, and (U), an upper triangular matrix, such that their product reconstructs the original matrix (A).

It decomposes a square matrix **A** into two matrices:

- **L** = Lower triangular matrix
- **U** = Upper triangular matrix

The decomposition is:

A=LU

Where

- **L** has values below the diagonal (and usually 1's on the diagonal)
- **U** has values above the diagonal

the **main rule in LU decomposition using Gaussian elimination**.

We select L_{21}, L_{31}, etc. using the **multiplier formula**.

Key Formula

$$L_{ij} = \frac{a_{ij}}{a_{jj}}$$

Where

- a_{ij} = element we want to eliminate
- a_{jj} = pivot element

Example:

Let

$$A = \begin{bmatrix} 2 & 3 & 1 \\ 4 & 7 & 7 \\ -2 & 4 & 5 \end{bmatrix}$$

Step 1: Eliminate a_{21}

$$R_2 = R_2 - 2R_1$$

$$(4, 7, 7) - 2(2, 3, 1)$$

$$(0, 1, 5)$$

Multiplier

$$L_{21} = 2$$

Step 2: Eliminate a_{31}

For Row 3

Row 3 first element = -2

Pivot = 2

The multiplier is

$$L_{31} = \frac{a_{31}}{a_{11}}$$

$$L_{31} = \frac{-2}{2}$$

$$L_{31} = -1$$

$$R_3 = R_3 + R_1$$

$$(-2, 4, 5) + (2, 3, 1)$$

$$(0, 7, 6)$$

Multiplier

$$L_{31} = -1$$

Step 3: Eliminate a_{32}

$$R_3 = R_3 - 7R_2$$

$$(0, 7, 6) - (0, 7, 35)$$

$$(0, 0, -29)$$

Multiplier

$$L_{32} = 7$$

Final Matrices

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 7 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} 2 & 3 & 1 \\ 0 & 1 & 5 \\ 0 & 0 & -29 \end{bmatrix}$$

Thus

A=LU

2. **QR Decomposition**

QR decomposition expresses a matrix (A) as the product of an orthogonal matrix (Q) and an upper triangular matrix (R), defined as (A = QR). It is applied in image processing, including image compression and encryption.

Example:

Let's understand the QR Decomposition process by

Suppose we are provided with the matrix A:

$$A = \begin{bmatrix} 1 & 2 & 4 \\ 0 & 0 & 5 \\ 0 & 3 & 6 \end{bmatrix}$$

As mentioned in the steps before, we will be using Gram-Schmidt Orthogonalization.

We will be finding orthogonal components q_1 , q_2 and q_3 :

First, perform normalization and we get the first normalized vector:

$$q1 = \frac{a1}{\|a1\|} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The norm of the first column is calculated as:

$$||a_1|| = \sqrt{1^2 + 0^2 + 0^2} = 1$$

The inner product of between a_2 and q_1 is $< a_2.q_1 > = q^T.a_2$ is considered and the projection of the second column on q_1 is multiplied with the inner product.

$q_2^{'}$ is the residual of the projection:

$$q2' = a2 - < a2.q1 > q1$$

$$= \begin{bmatrix} 2 \\ 0 \\ 3 \end{bmatrix} - 2 * \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}$$

Now, we will normalize the residual. :

$$q3 = \frac{q3'}{\|q3'\|} = \frac{q3'}{5} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

We got Q matrix.

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Now, we will normalize the residual:

$$q2 = \frac{q2'}{\|q2'\|} = \frac{q2'}{3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Now, we will project a_3 on q_1 and q_2 :

$$q3' = a3 - < a3, q1 > q1 - < a3, q2 > q2$$

$$= \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} - 4 * \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - 6 * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$$

Compute $R = Q^T A$

$$Q^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 2 & 4 \\ 0 & 3 & 6 \\ 0 & 0 & 5 \end{bmatrix}$$

Final QR Factorization

$$A = QR$$

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 2 & 4 \\ 0 & 3 & 6 \\ 0 & 0 & 5 \end{bmatrix}$$

3. **Singular Value Decomposition (SVD)**

   Singular Value Decomposition (SVD) is a factorization method in linear algebra that decomposes a matrix into three other matrices, providing a way to represent data in terms of its singular values.

   SVD helps you split that table into three parts:

   - **U**: This part tells you about the people (like their general preferences).
   - **Σ**: This part shows how important each factor is (how much each rating matters).
   - **V^T**: This part tells you about the products (how similar they are to each other)

- Mathematically, the SVD of a matrix A (of size $m \times n$) is represented as: $A = U\Sigma V^T$

Here:

- U: An $m \times m$ orthogonal matrix whose columns are the left singular vectors of A.
- Σ: A diagonal $m \times n$ matrix containing the singular values of A in descending order.
- V^T: The transpose of an $n \times n$ orthogonal matrix, where the columns are the right singular vectors of A.

Example:

To perform Singular Value Decomposition (SVD) for the matrix $A = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}$, let's break it down step by step.

Step 1: Compute AA^T

First, we need to calculate the matrix AA^T (where A^T is the transpose of matrix A):

$$A = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{bmatrix}$$

Now, compute AA^T:

$$AA^T = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix} \cdot \begin{bmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{bmatrix} = \begin{bmatrix} 17 & 8 \\ 8 & 17 \end{bmatrix}$$

Step 2: Find the Eigenvalues of AA^T

To find the eigenvalues of AA^T, we solve the characteristic equation:

$$\det(AA^T - \lambda I) = 0$$
$$\det \begin{bmatrix} 17-\lambda & 8 \\ 8 & 17-\lambda \end{bmatrix} = 0$$
$$(\lambda - 25)(\lambda - 9) = 0$$

Thus, the eigenvalues are $\lambda_1 = 25$ and $\lambda_2 = 9$. These eigenvalues correspond to the singular values $\sigma_1 = 5$ and $\sigma_2 = 3$, since the singular values are the square roots of the eigenvalues.

Step 3: Find the Right Singular Vectors (Eigenvectors of $A^T A$)

Next, we find the eigenvectors of $A^T A$ for $\lambda = 25$ and $\lambda = 9$.

***For** $\lambda = 25$:*

Solve $(A^T A - 25I)v = 0$:

$$A^T A - 25I = \begin{bmatrix} -12 & 12 & 2 \\ 12 & -12 & -2 \\ 2 & -2 & -17 \end{bmatrix}$$

Row-reduce this matrix to:

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

The eigenvector corresponding to $\lambda = 25$ is:

$$v_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

***For** $\lambda = 9$:*

Solve $(A^T A - 9I)v = 0$:

The eigenvector corresponding to $\lambda = 9$ is:

$$v_2 = \begin{bmatrix} \frac{1}{\sqrt{18}} \\ \frac{-1}{\sqrt{18}} \\ \frac{4}{\sqrt{18}} \end{bmatrix}$$

For the third eigenvector v_3:

Since v_3 must be perpendicular to v_1 and v_2, we solve the system $v_1^T v_3 = 0$ and $v_2^T v_3 = 0$, leading to:

$$v_3 = \begin{bmatrix} \frac{2}{3} \\ \frac{-2}{3} \\ \frac{-1}{3} \end{bmatrix}$$

Step 4: Compute the Left Singular Vectors (Matrix U)

To compute the left singular vectors U, we use the formula $u_i = \frac{1}{\sigma_i} A v_i$. *This results in:*

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

Step 5: Final SVD Equation

Finally, the Singular Value Decomposition of matrix A *is:*

$$A = U \Sigma V^T$$

Where:

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{18}} & \frac{-1}{\sqrt{18}} & \frac{4}{\sqrt{18}} \\ \frac{2}{3} & \frac{-2}{3} & \frac{1}{3} \end{bmatrix}$$

Thus, the SVD of matrix A *is:*

$$A = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{18}} & \frac{-1}{\sqrt{18}} & \frac{4}{\sqrt{18}} \\ \frac{2}{3} & \frac{-2}{3} & \frac{1}{3} \end{bmatrix}$$

4. Non-negative Matrix Factorization (NMF)

This variation of matrix factorization adds the constraint that none of the matrix's R, P or Q can contain negative elements. NMF is especially useful in applications where non-negativity is inherent to the data, such as image processing, audio processing and spectral data analysis.

For a matrix A of dimensions *m*×*n* where each element is ≥0 NMF factorizes it into two matrices *W* and *H* with dimensions *m*×*k* and *k*×*n* respectively where both matrices contain only non-negative elements:

$$A_{m \times n} \approx W_{m \times k} H_{k \times n}$$

where:

- *A* : Original input matrix (a linear combination of W and H)
- *W*: Feature matrix (basis components)
- *H*: Coefficient matrix (weights associated with W)
- *k*: Rank (dimensionality of the reduced representation where $k \leq \min(m,n)$)

Working of NMF

NMF decomposes a data matrix A*A* into two smaller matrices *W* and *H* using an iterative optimization process that minimizes reconstruction error:

1. Initialization: Start with random non-negative values for *W* and *H*.

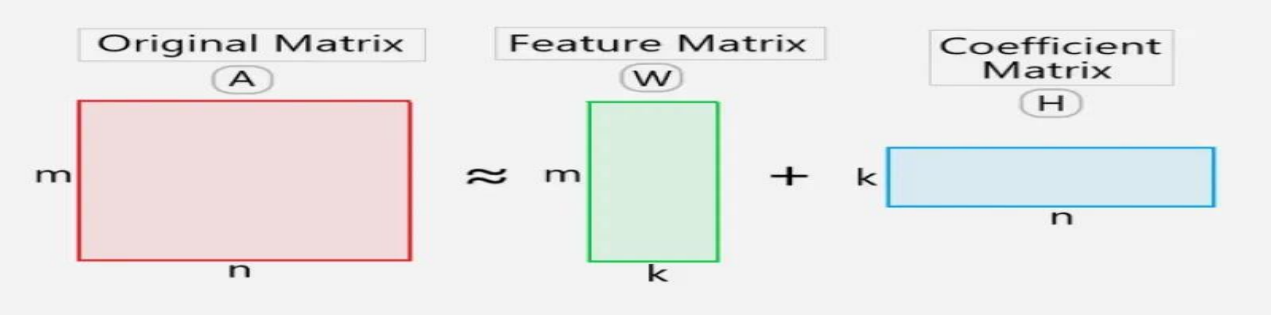
2. Iterative Update: Modify W*W*and *H* to minimize the difference between *A* and *W*×*H*.

3. Stopping Criteria: The process stops when:

- The reconstruction error stabilizes.
- A set number of iterations is reached.

Common optimization techniques for NMF include:

- **Multiplicative Update Rules**: Ensures non-negativity by iteratively adjusting *W* and *H*.
- **Alternating Least Squares (ALS)**: Solves for *W* while keeping *H* fixed and vice versa, in an alternating manner.



Example:

Step 1: Given Matrix

Let

$$V = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}$$

Size of V is 2 × 2.

Assume r = 2 latent features

So

$$W_{2\times 2}, \quad H_{2\times 2}$$

Step 2: Initialize Non-Negative Matrices

Assume initial values:

$$W = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$$

(All values must be ≥ **0**)

Step 3: Compute Product WH

$$WH = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$$

First row:

$$(1 \times 1) + (2 \times 2) = 5$$

$$(1 \times 1) + (2 \times 1) = 3$$

Second row:

$$(1 \times 1) + (1 \times 2) = 3$$

$$(1 \times 1) + (1 \times 1) = 2$$

$$WH = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}$$

Step 4: Compare With Original Matrix

$$V = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}$$

$$WH = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}$$

Thus

$$V \approx WH$$

In **Non-Negative Matrix Factorization (NMF)**, usually

$$V \neq WH$$

because we are doing **approximation**, not exact factorization.

$V \approx WH$

So if WH is **not equal** to V, we **improve the matrices W and H** using **iterations.**

What To Do Next When $V \neq WH$

Step 1: Compute Reconstruction Error

We measure how far WH is from V.

$$Error = ||V - WH||^2$$

If the error is **large**, we update the matrices.

Step 2: Update H

Update rule:

$$H_{ij} = H_{ij} \times \frac{(W^T V)_{ij}}{(W^T W H)_{ij}}$$

This adjusts **coefficients**.

Step 3: Update W

Update rule:

$$W_{ij} = W_{ij} \times \frac{(V H^T)_{ij}}{(W H H^T)_{ij}}$$

This adjusts **basis vectors**.

Step 4: Recalculate

Compute again

$$WH$$

Compare with V.

Clustering of Patterns

Clustering of patterns is an **unsupervised learning technique** used in pattern recognition and machine learning to group similar patterns or data points into clusters.

Definition

Clustering is the process of **organizing objects into groups (clusters)** such that:

- Objects in the **same cluster are similar**
- Objects in **different clusters are dissimilar**

Key Concepts in Clustering of Patterns:

1.**Similarity Measure**: Clustering relies on a similarity metric (e.g., Euclidean distance, cosine similarity) to group similar patterns together.

2.**Unsupervised Learning**: Since clustering does not use labeled data, it falls under unsupervised learning techniques.

3.**Cluster Formation**: Clusters should maximize intra-cluster similarity (similar elements in the same cluster) and minimize inter-cluster similarity (different elements in separate clusters).

Types of Clustering

1. Hard Clustering

Hard clustering assigns each data point to exactly one cluster. A data point cannot belong to multiple clusters, making the grouping clear and easy to interpret.

- Each data point belongs to only one cluster
- No overlap between clusters
- Simple and easy to interpret

Common Uses

- **Market segmentation:** Businesses group customers with similar buying behaviour to design targeted marketing strategies.
- **Customer grouping:** Companies organize customers into clear categories for better service and analysis.
- **Document clustering:** Documents with similar topics or keywords are grouped together for easier organization.

2. Soft Clustering

Soft clustering allows a data point to belong to multiple clusters with different probabilities. Instead of assigning a strict cluster, it gives a degree of membership to each cluster.

Example

A data point may belong 70% to Cluster 1 and 30% to Cluster 2, indicating that it shares characteristics with both groups.

Use Cases

- **Overlapping class boundaries:** Useful when data points cannot be clearly separated into distinct groups.
- **Customer personas:** Helps represent customers who share traits with multiple behavioral groups.
- **Medical diagnosis:** Patients may show symptoms related to multiple conditions.

Clustering Methods

Clustering methods can be classified on the basis of how they form clusters

1. Centroid based Clustering

Centroid based clustering groups data points around central points called centroids. Each cluster is represented by the average of its points and data points are assigned to the nearest centroid.

Algorithms:

- **K-means:** Iteratively assigns points to nearest centroid and recalculates centroids to minimize intra cluster variance.
- **K-medoids:** Similar to K-means but uses actual data points (medoids) as centers, robust to outliers.

Advantages:

- Fast and scalable for large datasets.
- Simple to implement and interpret.

2. Density based Clustering

Density based clustering identifies clusters as regions where data points are densely packed together. Points in low density areas are treated as noise.

Algorithms:

- **DBSCAN:** Groups points with sufficient neighbors; labels sparse points as noise.
- **OPTICS:** Extends DBSCAN to handle varying densities.

Advantages:

- Handles clusters of varying shapes and sizes.
- Does not require cluster count upfront.
- Effective in noisy datasets.

3. Connectivity based Clustering

Connectivity based or Hierarchical clustering builds clusters by gradually merging or splitting groups of data points. It creates a tree like structure called a dendrogram that shows relationships between clusters.

Approaches:

- **Agglomerative:** Starts with each point as a cluster and merges them step by step.
- **Divisive:** Starts with one cluster and splits it into smaller clusters.

Advantages:

- Provides a full hierarchy, easy to visualize
- No need to specify number of clusters upfront

4. Distribution-based Clustering

Distribution based clustering assumes that data points come from a mixture of probability distributions. Each cluster is modelled as a statistical distribution.

Algorithm:

- **Gaussian Mixture Model (GMM):** Fits data as a weighted mixture of Gaussian distributions, assigns data points based on likelihood

Advantages:

- Flexible cluster shapes
- Provides probabilistic memberships
- Suitable for overlapping clusters

5. Fuzzy Clustering

Fuzzy clustering allows data points to belong to multiple clusters with different degrees of membership. It is useful when cluster boundaries are not clear.

Algorithm:

- **Fuzzy C-Means:** Similar to K-means but with fuzzy memberships updated iteratively

Advantages:

- Models data ambiguity explicitly
- Useful for complex or imprecise data

Applications

Clustering is widely used in data analysis and machine learning to identify patterns in unlabeled data.

- **Customer Segmentation:** Group customers based on behavior or demographics.
- **Anomaly Detection:** Detect unusual activities in finance, security or sensor data.
- **Image Segmentation:** Divide images into meaningful regions for computer vision tasks.
- **Recommendation Systems:** Group similar users or items for personalized suggestions.
- **Market Basket Analysis:** Identify products frequently purchased together.

Divisive Clustering

Divisive Clustering is a type of hierarchical clustering that follows a top-down approach. It starts by placing all data points into one large cluster and then recursively splits that cluster into smaller ones based on differences or distances between the points. This process continues until each cluster contains only similar data points or meets a stopping condition.

Algorithm:

1. **Start with one cluster** containing all data points.

2. **Choose a cluster to split** (usually the largest or least cohesive).

3. **Apply a splitting criterion** (e.g., distance-based, density-based, or using a flat clustering method like K-Means).

4. **Recursively split** until a stopping condition is met (e.g., desired number of clusters or maximum cluster separation).

Example:

Step 1: Initially $C_l = \{A, B, C, D, E\}$

Step 2: $C_i = C_l$ and $C_j = \varnothing$

Step 3: Initial iteration

Let us calculate the average dissimilarities of the objects in C_i with the other objects in C_i.

	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0

Average dissimilarity of A

$$= \frac{1}{4}\big(d(a,b) + d(a,c) + d(a,d) + d(a,e)\big) = \frac{1}{4}(1+2+2+3) = \frac{8}{4} = 2.00$$

For B

$$= \frac{1}{4}(d(b,a) + d(b,c) + d(b,d) + d(b,e)) = \frac{1}{4}(1+2+4+3) = \frac{10}{4} = 2.5$$

For C

$$= \frac{1}{4}(2+2+1+5) = \frac{10}{4} = 2.5$$

For D

$$= \frac{1}{4}(2+4+1+3) = \frac{10}{4} = 2.5$$

For E

$$= \frac{1}{4}(3+3+5+3) = \frac{14}{4} = 3.5$$

The highest average dissimilarity is 3.50, for object E.

So, we move E to the new cluster C_j

$$C_i = \{A, B, C, D\} \quad \text{and} \quad C_j = \{E\}$$

Step 3: Remaining iterations

$$C_i = \{A, B, C, D\} \quad \text{and} \quad C_j = \{E\}$$

We calculate distance for each object $x \in C_i$:

$$D_x = \text{avg dissimilarity of } x \text{ with objects in } C_i \; - \; \text{dissimilarity of } x \text{ with } C_j$$

(i) 2nd iteration

For A:

$$D_a = \frac{1}{3}\big(d(a,b) + d(a,c) + d(a,d)\big) - d(a,e)$$

$$= \frac{1}{3}(1+2+2) - 3 = \frac{5}{3} - 3 = -\frac{4}{3} \approx -1.33$$

For B:

$$D_b = \frac{1}{3}\big(d(b,a) + d(b,c) + d(b,d)\big) - d(b,e)$$

$$= \frac{1}{3}(1+2+4) - 3 = \frac{7}{3} - 3 = -\frac{2}{3} \approx -0.67$$

For C

$$D_c = \frac{1}{3}\big(d(c,a) + d(c,b) + d(c,d)\big) - d(c,e)$$

$$= \frac{1}{3}(2+2+1) - 5 = \frac{5}{3} - 5 = -\frac{10}{3} \approx -3.33$$

For D

$$D_d = \frac{1}{3}\big(d(d,a) + d(d,b) + d(d,c)\big) - d(d,e)$$

$$= \frac{1}{3}(2+4+1) - 3 = \frac{7}{3} - 3 = -\frac{2}{3} \approx -0.67$$

Calculated Differences (D_x):

- $D_a = -1.33$
- $D_b = -0.67$
- $D_c = -3.33$
- $D_d = -0.67$

Decision Rule: Since all values of $D_x < 0$, no object is moved.

Finalized Split:

$$C_i = \{A, B, C, D\} \text{ and } C_j = \{E\}$$

Initial Setup

- **Step 1:** All objects are initially in one large cluster, $C_l = \{A, B, C, D\}$.
- **Step 2:** Two working sets are initialized: C_i (the current cluster) and C_j (the "splinter group" which starts empty, $\emptyset$).
- **Step 3 (Initial Iteration):** The goal is to find the most "dissimilar" object in the current cluster to start a new group.

Calculation: Average Dissimilarity of A

$$= \frac{1}{3}(d(a,b) + d(a,c) + d(a,d))$$

$$= \frac{1}{3}(1 + 2 + 2) = \frac{5}{3} \approx 1.67$$

Average Dissimilarity of C

$$\text{Avg}(C) = \frac{1}{3}(2 + 2 + 1) = \frac{5}{3} \approx \mathbf{1.67}$$

Average Dissimilarity of B

$$\text{Avg}(B) = \frac{1}{3}(1 + 2 + 4) = \frac{7}{3} \approx \mathbf{2.33}$$

Average Dissimilarity of D

$$\text{Avg}(D) = \frac{1}{3}(2 + 4 + 1) = \frac{7}{3} \approx \mathbf{2.33}$$

Here is the cleaned and corrected text from the image:

- The highest average dissimilarity is **2.33**, and both **B** and **D** have this value.
- Let us arbitrarily choose **B** and put it into C_j.
- $C_i = \{A, C, D\}$ and $C_j = \{B\}$

Step 3: Remaining iterations

$C_i = \{A, C, D\}$ and $C_j = \{B\}$

• We calculate distance for each object $x \in C_i$:

$$D_x = \text{avg dissimilarity of } x \text{ with objects in } C_i \; - \; \text{dissimilarity of } x \text{ with } C_j$$

Calculations:

$$D_a = \frac{1}{2}(d(a,c) + d(a,d)) - \frac{1}{1}(d(a,b))$$

$$= \frac{1}{2}(2+2) - 1 = \frac{4}{2} - 1 = 2 - 1 = 1.0$$

$$D_c = \frac{1}{2}(d(c,a) + d(c,d)) - \frac{1}{1}(d(c,b))$$

$$= \frac{1}{2}(2+1) - 2 = \frac{3}{2} - 2 = 1.5 - 2 = -0.5$$

$$D_d = \frac{1}{2}(d(d,a) + d(d,c)) - \frac{1}{1}(d(d,b))$$

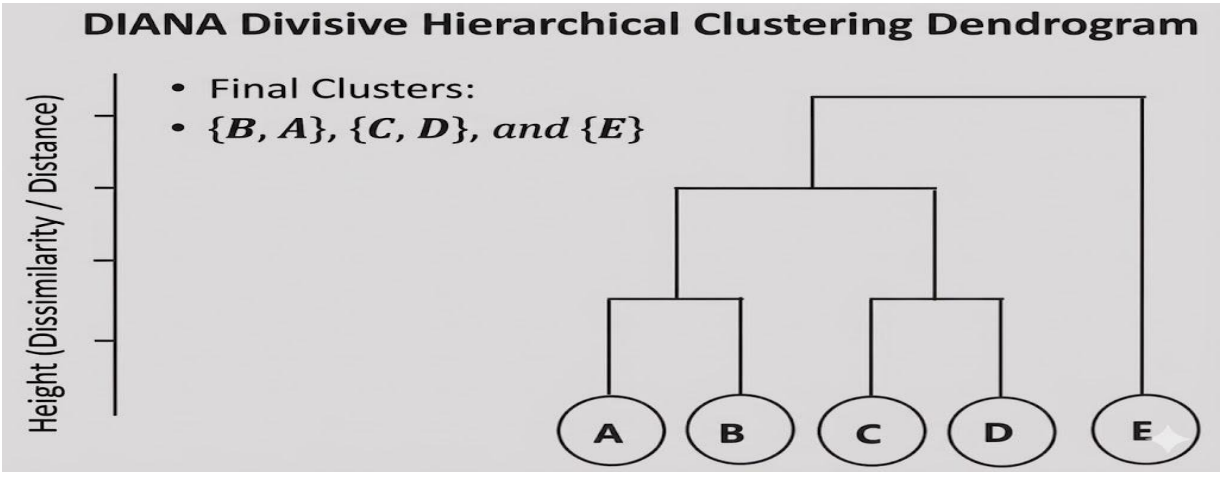
$$= \frac{1}{2}(2+1) - 4 = \frac{3}{2} - 4 = 1.5 - 4 = -2.5$$

- $D_a = 1$
- $D_c = -0.5$
- $D_d = -2.5$

Highest average dissimilarity is **1**, for object **A**.

So, we move **A** to the new cluster C_j.

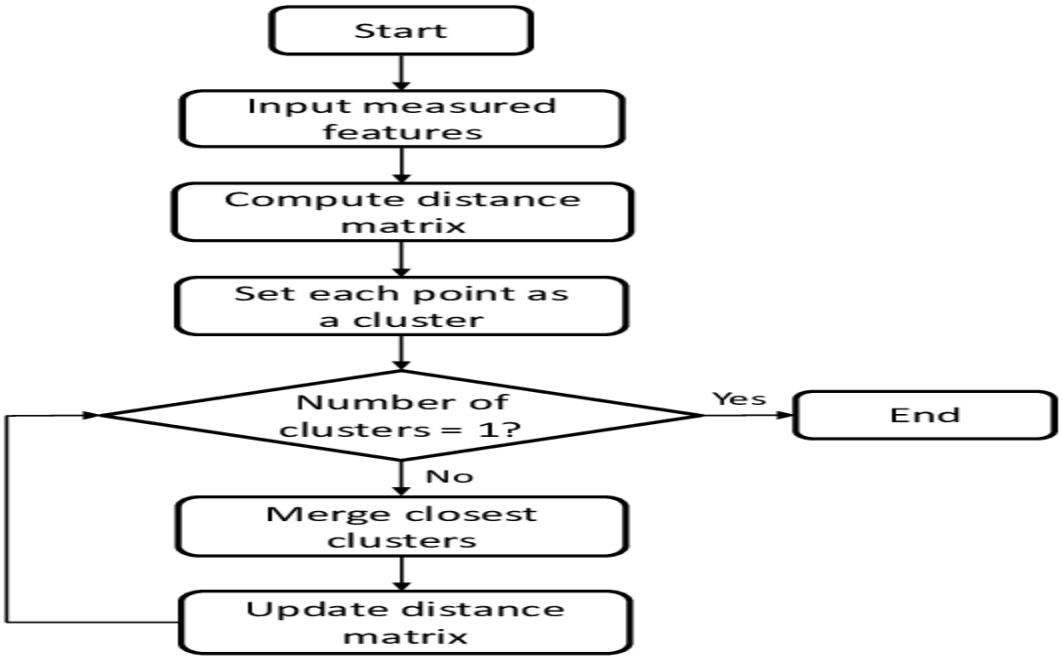
$$C_i = \{C, D\} \quad \text{and} \quad C_j = \{B, A\}$$



Agglomerative Clustering

Agglomerative Clustering is used which is a type of hierarchical clustering. It follows a bottom-up approach, where each data point starts as its own cluster and gradually merges with others based on similarity.

- The merging continues until all points form a single cluster or a set number of clusters remain.
- It uses distance metrics like Euclidean or Manhattan distance to measure similarity.
- The process is often visualized using a dendrogram, which shows the hierarchy of cluster formation.
- Common linkage methods include single, complete, average and ward linkage.



Algorithm:

1. **Initialize**: Treat each data point as a single cluster.

2. **Compute proximity matrix**: Calculate distances between all clusters.

3. **Merge closest clusters**: Combine the two most similar clusters.

4. **Update proximity matrix**: Recompute distances between the new cluster and remaining clusters.

5. **Repeat**: Continue merging until only one cluster remains (or a stopping criterion is met).

Points	A	B
1	1	4
2	1	2
3	0	4
4	6	1
5	7	0

- Calculate the **pairwise distance** between data points
- **Euclidean distance formula**
- $P(x_1, y_1)$ and $Q(x_2, y_2)$
- $d(P,Q) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

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

Points	A	B
1	1	4
2	1	2
3	0	4
4	6	1
5	7	0

1	2	3	4	5
0	0	2.00	1.00	5.83
2	2.00	0	2.24	5.10
3	1.00	2.24	0	6.32
4	5.83	5.10	6.71	0
5	7.21	6.32	8.06	1.41
5	7.21	6.32	8.06	1.41

- **Initial Clusters**
- Each point is its own cluster:
- C1 = {1}
- C2 = {2}
- C3 = {3}
- C4 = {4}
- C5 = {5}

- Find Minimum distance:
- d(1,3) = 1.00
- Merge **C1 & C3**
- New cluster: C13 = {1,3}
- New clusters: C13, C2, C4, C5

	C13	C2	C4	C5
C13	0	2.24	6.71	8.06
C2	2.24	0	5.10	6.32
C4	6.71	5.10	0	1.41
C5	8.06	6.32	1.41	0

Complete linkage: max distance between points in clusters.

Distance Calculations

- $d(C13, C2)$:
  The distance is the maximum of the distance from point 1 to 2, and point 3 to 2.

$$\max(d(1,2), d(3,2)) = \max(2.00, 2.24) = \mathbf{2.24}$$

- $d(C13, C4)$:
  The distance is the maximum of the distance from point 1 to 4, and point 3 to 4.

$$\max(d(1,4), d(3,4)) = \max(5.83, 6.71) = \mathbf{6.71}$$

- $d(C13, C5)$:
  The distance is the maximum of the distance from point 1 to 5, and point 3 to 5.

$$\max(d(1,5), d(3,5)) = \max(7.21, 8.06) = \mathbf{8.06}$$

- Find Minimum distance:
- d(4,5) = 1.41
- Merge C4 & C5 → C45 = {4,5}
- New Clusters: C13, C2, C45

- Recompute Distances Using Complete Linkage
- d(C13, C2) = max(d(1,2), d(3,2)) = max(2.00, 2.24) = **2.24**
- d(C13, C45) = max(d(1,4), d(1,5), d(3,4), d(3,5)) = max(5.83, 7.21, 6.71, 8.06) = **8.06**
- d(C2, C45) = max(d(2,4), d(2,5)) = max(5.10, 6.32) = **6.32**

Complete linkage: **max distance between points in clusters.**

	C13	C2	C45
C13	**0**	**2.24**	**8.06**
C2	**2.24**	0	6.32
C45	**8.06**	6.32	0

New Clusters: C13, C2, C45

- **Find Minimum distance**
- $d(C13, C2) = 2.24$
- **Merge** $\rightarrow C132 = \{1, 2, 3\}$
- **Clusters:** $C132, C45$

	C13	C45
C132	0	8.06
C45	8.06	0

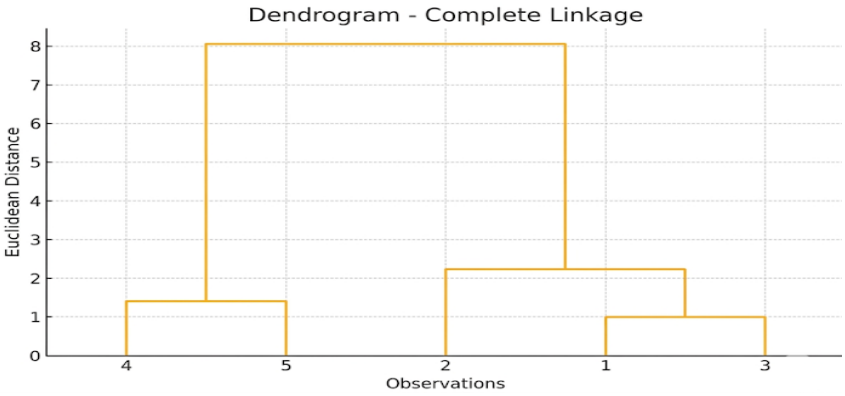
Recompute Distances Using Complete Linkage

Complete linkage measures the **max distance** between any two points in different clusters.

- $d(C132, C45) = \max(d(1,4), d(1,5), d(2,4), d(2,5), d(3,4), d(3,5))$
- $= \max(5.83, 7.21, 5.10, 6.32, 6.71, 8.06) = 8.06$

- **Now final merge: C132 & C45**
- **With distance 8.06**
- **Clusters: C13245**

- d(C1, C3) = 1
- d(C4, C5) = 1.41
- d(C13, C2) = 2.24
- d(C132, C45) = 8.06



Partitional Clustering

Partitional clustering is a family of clustering algorithms that divide a dataset into **non-overlapping, flat clusters** without any hierarchical structure. Unlike hierarchical methods (agglomerative/divisive), partitional clustering creates a single-level partitioning of data, making it faster and more scalable for large datasets.

Key Characteristics of Partitional Clustering

- **Flat structure**: No parent-child relationships between clusters.
- **Requires predefined number of clusters (k)**: Unlike hierarchical methods.
- **Optimizes a global objective function**: Minimizes within-cluster variance.
- **Deterministic or probabilistic**: Hard clustering (e.g., K-Means) vs. soft clustering (e.g., GMM).

Popular Partitional Clustering Algorithms

1. K-Means Clustering

How it works:

1. Randomly initialize *k* centroids.

2. Assign each point to the nearest centroid.

3. Recompute centroids as the mean of assigned points.

4. Repeat until convergence.

- **Pros**: Fast, scalable, works well with spherical clusters.
- **Cons**: Sensitive to initialization, struggles with non-spherical clusters.

2. K-Medoids (PAM - Partitioning Around Medoids)

- **How it differs from K-Means**: Uses actual data points (medoids) as cluster centers instead of means.
- **Pros**: More robust to outliers.
- **Cons**: Computationally expensive (O (k (n-k)^2 per iteration)).

3. Fuzzy C-Means (FCM)

- **Soft clustering**: Each point has a probability of belonging to each cluster.
- **Pros**: Handles overlapping clusters well.
- **Cons**: Slower than K-Means, sensitive to initialization.

4. Gaussian Mixture Models (GMM)

- **Probabilistic approach**: Models clusters as Gaussian distributions.
- **Pros**: Can fit ellipsoidal clusters, provides probability estimates.
- **Cons**: Requires tuning of covariance matrices.

K-Means Clustering

K-Means Clustering is an **unsupervised machine learning algorithm** used to group data into **K distinct, non-overlapping clusters** based on feature similarity. It's widely used in data mining, pattern recognition, and image compression.

How K-Means Works:

1. **Initialize**: Choose K (number of clusters) and randomly initialize K cluster centroids.

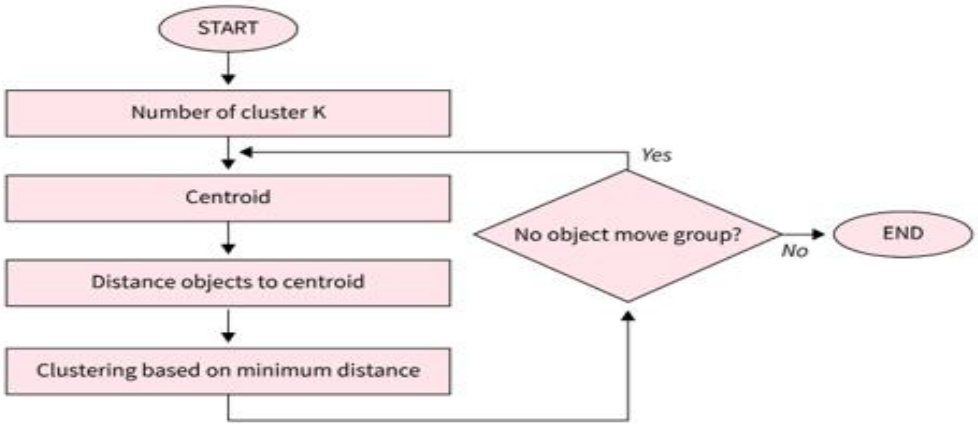
2. **Assign**: Assign each data point to the nearest centroid (based on distance—typically Euclidean).

3. **Update**: Recalculate the centroids as the mean of all points assigned to each cluster.

4. **Repeat**: Steps 2 and 3 until convergence (no change in assignments or centroids).

Key Concepts:

- **Centroid**: The center of a cluster (mean of all points).
- **Inertia**: The sum of squared distances between data points and their closest centroid (used to measure cluster tightness).
- **Elbow Method**: A technique for choosing the optimal K by plotting the inertia vs. K and looking for the "elbow" point.



Pros:

- Simple and fast.
- Scales to large datasets.
- Works well when clusters are spherical and evenly sized.

Cons:

- Requires specifying K in advance.
- Sensitive to initial placement of centroids.
- Struggles with non-spherical or overlapping clusters.

Example:

- Suppose that the data mining task is to cluster points into three clusters,
- where the points are
- A1(2, 10), A2(2, 5), A3(8, 4), B1(5, 8), B2(7, 5), B3(6, 4), C1(1, 2), C2(4, 9).

- The distance function is Euclidean distance.
- Suppose initially we assign A1, B1, and C1 as the center of each cluster, respectively.

- Initial Centroids:
- A1: (2, 10)
- B1: (5, 8)
- C1: (1, 2)

Data Points			Distance to 2, 10	Distance to 5, 8	Distance to 1, 2	Cluster	New Cluster
A1	2	10	0.00	3.61	8.06	1	
A2	2	5	5.00	4.24	3.16	3	
A3	8	4	8.49	5.00	7.28	2	
B1	5	8	3.61	0.00	7.21	2	
B2	7	5	7.07	3.61	6.71	2	
B3	6	4	7.21	4.12	5.39	2	
C1	1	2	8.06	7.21	0.00	3	
C2	4	9	2.24	1.41	7.62	2	

- New Centroids:
- A1: (2, 10)
- B1: (6, 6)
- C1: (1.5, 3.5)

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Current Centroids:
A1: (2, 10)
B1: (6, 6)
C1: (1.5, 3.5)

Data Points			Distance to 2, 10	Distance to 6, 6	Distance to 1.5, 1.5	Cluster	New Cluster
A1	2	10	0.00	5.66	6.52	1	1
A2	2	5	5.00	4.12	1.58	3	3
A3	8	4	8.49	2.83	6.52	2	2
B1	5	8	3.61	2.24	5.70	2	2
B2	7	5	7.07	1.41	5.70	2	2
B3	6	4	7.21	2.00	4.53	2	2
C1	1	2	8.06	6.40	1.58	3	3
C2	4	9	2.24	3.61	6.04	2	1

New Centroids:
A1: (3, 9.5)
B1: (6.5, 5.25)
C1: (1.5, 3.5)

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Current Centroids:
A1: (3, 9.5)
B1: (6.5, 5.25)
C1: (1.5, 3.5)

Data Points			Distance to 3, 9.5	Distance to 6.5, 5.25	Distance to 1.5, 3.5	Cluster	New Cluster
A1	2	10	1.12	6.54	6.52	1	1
A2	2	5	4.61	4.51	1.58	3	3
A3	8	4	7.43	1.95	6.52	2	2
B1	5	8	2.50	3.13	5.70	2	1
B2	7	5	6.02	0.56	5.70	2	2
B3	6	4	6.26	1.35	4.53	2	2
C1	1	2	7.76	6.39	1.58	3	3
C2	4	9	1.12	4.51	6.04	1	1

New Centroids:
A1: (3.67, 9)
B1: (7, 4.33)
C1: (1.5, 3.5)

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Current Centroids:
A1: (3.67, 9)
B1: (7, 4.33)
C1: (1.5, 3.5)

Data Points			Distance to						Cluster	New Cluster
			3.67	9	7	4.33	1.5	3.5		
A1	2	10	1.94		7.56		6.52		1	1
A2	2	5	4.33		5.04		1.58		3	3
A3	8	4	6.62		1.05		6.52		2	2
B1	5	8	1.67		4.18		5.70		1	1
B2	7	5	5.21		0.67		5.70		2	2
B3	6	4	5.52		1.05		4.53		2	2
C1	1	2	7.49		6.44		1.58		3	3
C2	4	9	0.33		5.55		6.04		1	1

$$d(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Soft Partitioning

Soft partitioning (or fuzzy clustering) assigns data points to **multiple clusters with varying degrees of membership** (typically [0, 1]), unlike hard clustering (e.g., K-Means) where each point belongs to exactly one cluster.

Key Characteristics

- **Probabilistic assignments**: Points have membership probabilities summing to 1
- **Overlapping clusters**: Accommodates ambiguous cases
- **Non-binary decisions**: Useful for boundary points

2. Core Algorithms

A. Fuzzy C-Means (FCM)

Objective Function (minimized):

The objective function is formally expressed as:

$$J_m = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^m \|x_j - v_i\|^2$$

Soft Clustering

Soft clustering (probabilistic clustering) assigns data points to **multiple clusters with membership probabilities** rather than forcing hard assignments. This approach better handles:

- Overlapping clusters
- Ambiguous/boundary points
- Real-world data uncertainty

Key Properties

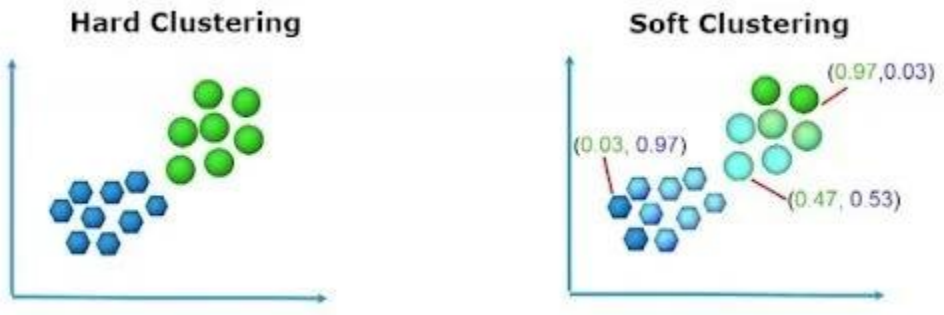
- Each point has a **membership vector** (sums to 1 across clusters)
- Models **cluster overlap** explicitly
- Provides **confidence measures** for assignments

How Soft Clustering Works

- 1. **Initialize Membership Values**: Randomly assign membership values to each data point for each cluster. These values indicate how strongly a data point belongs to a cluster (ranging between 0 and 1), and the sum of the memberships for each data point across all clusters must be **1**.
- 2. **Compute Cluster Centers**: Calculate the cluster centers (centroids) based on the weighted average of the data points, where the weights are the membership values. The weighted average takes into account the degree of membership, so the center moves towards the data points with higher membership.
- 3. **Update Membership Values**: After calculating the new cluster centers, recompute the membership values based on the distances between data points and the updated cluster centers. Points closer to a cluster center will have higher membership in that cluster.
- 4. **Repeat Until Convergence**: Continue iterating between updating the cluster centers and recalculating the membership values until the algorithm converges, i.e., the membership values and cluster centers no longer change significantly.

Fuzzy C-Means Clustering

- An extension of k-means. It is a Type of soft clustering
- Hierarchical, k-means generates partitions
    - each data point can only be assigned in one cluster called hard clustering.
- Fuzzy c-means allows data points to be assigned into more than one cluster
    - each data point has a degree of membership (or probability) of belonging to each cluster called Soft clustering.



Algorithm:

1.Choose the number of clusters (C).

2.Initialize membership matrix: Randomly initialize the degree of membership of each data point for each cluster.

3.Update cluster centers: Compute the cluster centers based on the weighted mean of the data points, where the weights are the membership values.

4.Update membership values: Recalculate the membership values based on the distance of the data points to the new cluster centers.

5.Repeat steps 3 and 4 until the membership matrix converges or stops changing significantly.

Example:

- **Step 1:** Given the data points based on the number of clusters required initialize the membership table with random values.
- Suppose the given data points are **{(1, 3), (2, 5), (6, 8), (7, 9)}**

Cluster	(1, 3)	(2, 5)	(4, 8)	(7, 9)
1	0.8	0.7	0.2	0.1
2	0.2	0.3	0.8	0.9

Step 2: Find out the centroid

The formula for finding the centroid V is:

$$V_{ij} = \frac{\sum_{k=1}^{n} \gamma_{ik}^{m} x_k}{\sum_{k=1}^{n} \gamma_{ik}^{m}}$$

Where:

- γ : Fuzzy membership value
- m : Fuzziness parameter (generally taken as **2**)
- x_k : Data point

$$V_{11} = \frac{0.8^2 \times 1 + 0.7^2 \times 2 + 0.2^2 \times 4 + 0.1^2 \times 7}{0.8^2 + 0.7^2 + 0.2^2 + 0.1^2} = 1.568$$

$$V_{12} = \frac{0.8^2 \times 3 + 0.7^2 \times 5 + 0.2^2 \times 8 + 0.1^2 \times 9}{0.8^2 + 0.7^2 + 0.2^2 + 0.1^2} = 4.051$$

Centroid of Cluster 1

$$V_1 = (1.568,\ 4.051)$$

$$V_{21} = \frac{0.2^2 \times 1 + 0.3^2 \times 2 + 0.8^2 \times 4 + 0.9^2 \times 7}{0.2^2 + 0.3^2 + 0.8^2 + 0.9^2} = 5.348$$

$$V_{22} = \frac{0.2^2 \times 3 + 0.3^2 \times 5 + 0.8^2 \times 8 + 0.9^2 \times 9}{0.2^2 + 0.3^2 + 0.8^2 + 0.9^2} = 8.215$$

Centroid of Cluster 2

$$V_2 = (5.348,\ 8.215)$$

Step 3: Find out the distance of each point from the centroid

Distance formula:

$$d_{ki} = \sqrt{(x_k - v_i)^2 + (y_k - v_i)^2}$$

Where
x_k = data point
v_i = centroid of cluster

Centroids:

$$V_1 = (1.568, 4.051)$$
$$V_2 = (5.348, 8.215)$$

$$d_{11} = \sqrt{(1 - 1.568)^2 + (3 - 4.051)^2} = 1.195$$
$$d_{12} = \sqrt{(1 - 5.348)^2 + (3 - 8.215)^2} = 6.788$$
$$d_{21} = \sqrt{(2 - 1.568)^2 + (5 - 4.051)^2} = 1.043$$
$$d_{22} = \sqrt{(2 - 5.348)^2 + (5 - 8.215)^2} = 4.645$$
$$d_{31} = \sqrt{(4 - 1.568)^2 + (8 - 4.051)^2} = 4.639$$
$$d_{32} = \sqrt{(4 - 5.348)^2 + (8 - 8.215)^2} = 1.366$$
$$d_{41} = \sqrt{(7 - 1.568)^2 + (9 - 4.051)^2} = 7.350$$
$$d_{42} = \sqrt{(7 - 5.348)^2 + (9 - 8.215)^2} = 1.830$$

- **Step 4: Updating membership values.**

- $\gamma_{ki} = \left(\sum_{j=1}^{n} \left\{ \frac{d_{ki}^2}{d_{kj}^2} \right\}^{\left(\frac{1}{(m-1)}\right)} \right)^{-1}$

$D_{11} = 1.2, \quad D_{12} = 6.79$
$D_{21} = 1.04, \quad D_{22} = 4.64$
$D_{31} = 4.63, \quad D_{32} = 1.36$
$D_{31} = 7.34, \quad D_{32} = 1.82$

Cluster	(1, 3)	(2, 5)	(4, 8)	(7, 9)
1	0.8	0.7	0.2	0.1
2	0.2	0.3	0.8	0.9

- For point 1 new membership values are:

- $\gamma_{11} = \left(\left\{ \frac{(1.2)^2}{(1.2)^2} + \frac{(1.2)^2}{(6.79)^2} \right\}^{\left(\frac{1}{(2-1)}\right)} \right)^{-1} = 0.97$
- $\gamma_{12} = \left(\left\{ \frac{(6.79)^2}{(1.2)^2} + \frac{(6.79)^2}{(6.79)^2} \right\}^{\left(\frac{1}{(2-1)}\right)} \right)^{-1} = 0.03$

- For point 2 new membership values are:

- $\gamma_{21} = \left(\left\{ \frac{(1.04)^2}{(1.04)^2} + \frac{(1.04)^2}{(4.64)^2} \right\}^{\left(\frac{1}{(2-1)}\right)} \right)^{-1} = 0.95$
- $\gamma_{22} = \left(\left\{ \frac{(4.64)^2}{(1.04)^2} + \frac{(4.64)^2}{(4.64)^2} \right\}^{\left(\frac{1}{(2-1)}\right)} \right)^{-1} = 0.05$

- For point 3 new membership values are:

- $\gamma_{31} = \left(\left\{ \frac{(4.63)^2}{(4.63)^2} + \frac{(4.63)^2}{(1.36)^2} \right\}^{\left(\frac{1}{(2-1)}\right)} \right)^{-1} = 0.08$
- $\gamma_{32} = \left(\left\{ \frac{(1.36)^2}{(4.63)^2} + \frac{(1.36)^2}{(1.36)^2} \right\}^{\left(\frac{1}{(2-1)}\right)} \right)^{-1} = 0.92$

- For point 4 new membership values are:

- $\gamma_{41} = \left(\left\{ \frac{(7.34)^2}{(7.34)^2} + \frac{(7.34)^2}{(1.82)^2} \right\}^{\left(\frac{1}{(2-1)}\right)} \right)^{-1} = 0.06$
- $\gamma_{42} = \left(\left\{ \frac{(1.82)^2}{(7.34)^2} + \frac{(1.82)^2}{(1.82)^2} \right\}^{\left(\frac{1}{(2-1)}\right)} \right)^{-1} = 0.94$

Cluster	(1, 3)	(2, 5)	(4, 8)	(7, 9)
1	0.97	0.95	0.08	0.06
2	0.03	0.05	0.92	0.94

Step 5: Repeat the steps (2-4) until the constant values are obtained for the membership values or the difference is less than the tolerance value

Rough Clustering

Rough Clustering is a concept that seeks to handle imprecise or uncertain data by allowing the formation of clusters that are not sharply defined, similar to **fuzzy clustering** but with a more flexible and less computationally intensive approach. It involves the use of **lower and upper approximation** to group data points, which means data points are assigned to a cluster with some level of **certainty** or **uncertainty**.

In **Rough Clustering**, we divide data into two types of sets:

- **Lower approximation**: The set of data points that surely belong to a cluster.
- **Upper approximation**: The set of data points that may belong to a cluster, but are not certain.

Rough clustering is related to **rough sets theory**, a mathematical approach for dealing with vagueness and uncertainty in data. The concept is useful when there is incomplete or ambiguous information, making it difficult to form clear-cut clusters.

How Rough Clustering Works

1. **Lower Approximation**: This set consists of the data points that are definitely part of a cluster. They are certain to belong to that cluster.

2. **Upper Approximation**: This set consists of the data points that might belong to a cluster, but the membership is uncertain.

3. **Boundary Region**: The boundary region contains points that are neither definitely in the cluster nor definitely outside of it. These points are part of the upper approximation but not part of the lower approximation.

4. **Partitioning**: Similar to other clustering techniques, rough clustering partitions the data into disjoint clusters, but with an emphasis on handling uncertainty in the data.

Advantages of Rough Clustering

- **Handles Uncertainty:** It's effective when dealing with data that has vagueness or imprecision.
- **Flexibility:** Provides a way to partition data into clusters while considering uncertainty and ambiguity.
- **Robust:** Suitable for complex datasets where strict boundaries between clusters do not exist.

Disadvantages

- **Complexity:** It can be more difficult to implement compared to standard clustering methods (e.g., K-Means, DBSCAN).
- **Interpretability:** The concept of lower and upper approximations may not always be intuitive for all users.
- **Requires Parameter Tuning:** Setting the threshold for determining lower and upper approximations can be subjective.

Rough K-Means Clustering Algorithm

The **Rough K-Means** algorithm is an extension of the traditional **K-Means clustering** algorithm, incorporating the concept of **rough sets**. It aims to handle the uncertainty in cluster membership by dividing data points into **lower** and **upper approximations**, similar to rough clustering.

In **Rough K-Means**:

- **Lower approximation** refers to the data points that are definitely part of a cluster.
- **Upper approximation** refers to the data points that could belong to a cluster, but their membership is uncertain.
- The algorithm assigns each data point to a cluster with varying degrees of certainty, rather than a definite assignment like traditional K-Means.

Steps in Rough K-Means Clustering Algorithm

1. **Initialize K centroids**: Randomly choose K data points as initial centroids.
2. **Calculate distances**: For each data point, calculate the distance to each centroid.
3. **Determine membership**:
   - Assign each point to the **lower approximation** of the closest centroid if the distance is sufficiently small.

- Assign each point to the **upper approximation** of the closest centroid if the distance is slightly larger, indicating uncertainty.

4. **Update centroids**: Update the centroids by calculating the weighted average of the data points, where points in the lower approximation have a higher weight.
5. **Repeat**: Continue the process of calculating distances, updating membership, and recalculating centroids until convergence (i.e., the centroids stop changing).

Expectation Maximization-Based Clustering

The **Expectation Maximization (EM)** algorithm is a statistical technique for clustering that models data using a **probabilistic** approach. The EM algorithm is particularly useful for clustering when the data can be modeled as being generated from a mixture of different distributions (usually Gaussian distributions), such as in **Gaussian Mixture Models (GMM)**.

The key idea behind **EM-based clustering** is to assign data points to clusters with probabilities, rather than deterministic assignments like K-Means. This is useful when data has overlapping clusters or uncertainties about the assignments.

How Expectation Maximization (EM) Works:

The EM algorithm works in two main steps:

1. **Expectation Step (E-step)**: In this step, we estimate the probability that each data point belongs to each cluster (the **responsibility**). The algorithm calculates the **posterior probabilities** based on the current parameter estimates.

2. **Maximization Step (M-step)**: In this step, we update the parameters (such as means, covariances, and weights) of the Gaussian distributions based on the probabilities (responsibilities) calculated in the E-step. This maximizes the likelihood of the data given the current cluster assignments.

The **EM algorithm** iterates between these two steps until the model parameters converge, i.e., when the assignments or the likelihood do not change significantly between iterations.

Gaussian Mixture Model (GMM):

In the context of clustering, **Expectation Maximization** is often used to fit a **Gaussian Mixture Model (GMM)**, which is a probabilistic model that assumes that the data points are generated from a mixture of several Gaussian distributions.

The **parameters** for a Gaussian mixture model are:

- **Means**: The centers of the Gaussians (clusters).

- **Covariances**: The spread (shape) of the Gaussian distributions.
- **Weights**: The proportion of each Gaussian in the mixture.

Steps in EM for Clustering (GMM):

1. **Initialize parameters** (means, covariances, and weights) randomly or by using some heuristic (e.g., K-Means).
2. **E-step**: Calculate the responsibility of each Gaussian for each data point. This gives the probability that each data point belongs to each Gaussian.
3. **M-step**: Update the parameters of the Gaussian distributions based on the responsibilities.
4. **Repeat** the E-step and M-step until convergence (when the likelihood or parameters stop changing).

Advantages of Expectation Maximization-Based Clustering:

- **Probabilistic Clustering**: EM assigns probabilities to each data point's membership in a cluster, which is more flexible than hard clustering.
- **Handles Overlapping Clusters**: EM can model clusters that overlap, unlike K-Means which assumes hard boundaries between clusters.
- **Robust to Noise**: EM can handle noise and outliers by incorporating probabilistic assignments.

Disadvantages:

- **Sensitive to Initialization**: The results of the EM algorithm can depend on the initial parameters (means, covariances, and weights).
- **Complexity**: EM can be computationally expensive, especially with large datasets and many components.
- **Convergence to Local Optima**: Like K-Means, EM can get stuck in local optima, so it's sensitive to initialization.

Spectral Clustering.

Spectral Clustering is a technique that uses the eigenvalues and eigenvectors of a similarity matrix to reduce the dimensionality of the data, making it easier to perform clustering. It is based on the idea that the structure of a graph (created from data points) can reveal the underlying clusters.

In **spectral clustering**, data points are treated as nodes in a graph, and the edges represent similarities between the nodes. The algorithm uses the **Laplacian matrix** of the graph to find clusters.

Advantages of Spectral Clustering:

- **Nonlinear Clusters**: Spectral clustering can handle complex, nonlinear relationships between data points, unlike K-means which assumes spherical clusters.
- **Versatile Similarity Measures**: It can use a variety of similarity measures, such as nearest neighbors or Gaussian kernels.

- **Effective for Arbitrarily Shaped Clusters**: It's particularly effective for datasets that contain clusters of arbitrary shapes, like the "moons" dataset.

Disadvantages of Spectral Clustering:

- **Computational Complexity**: Spectral clustering can be computationally expensive for large datasets, especially when computing eigenvectors.
- **Choosing Parameters**: The quality of clustering is sensitive to the choice of parameters such as the number of neighbors or the scale of the Gaussian kernel.

Key Steps in Spectral Clustering

1. Compute the Similarity Matrix

A similarity (affinity) matrix is constructed, where each element represents similarity between data points.

$$S(i,j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$$

Where:

- σ is a scaling parameter

2. Compute the Degree Matrix

The degree matrix D is a diagonal matrix:

$$D_{ii} = \sum_j S(i,j)$$

3. Compute the Laplacian Matrix

$$L = D - S$$

Where:

- D = Degree matrix
- S = Similarity matrix

4. Compute the Eigenvectors

Compute eigenvectors of the Laplacian matrix.

The eigenvectors corresponding to the **smallest eigenvalues** capture important structure of the data.

5. Form Feature Matrix

Form matrix U using selected eigenvectors as columns.

6. Clustering

Treat rows of matrix U as new data points and apply **K-means Clustering**.

7. Assign Labels

Assign cluster labels obtained from K-means to the original data points.