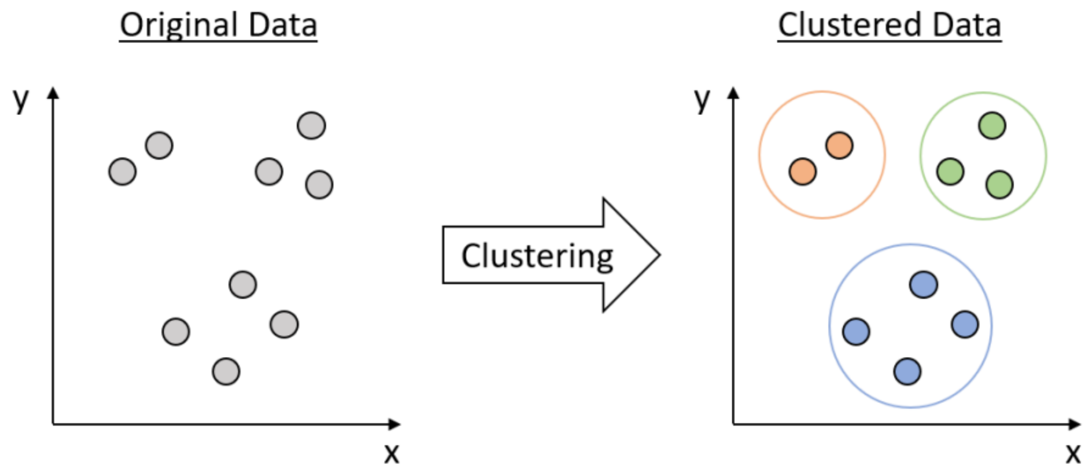
UNIT-V

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

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

What is Clustering?

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

Key Characteristics of Clustering

1. **Unsupervised Learning**: No predefined labels or categories
2. **Exploratory**: Helps discover hidden patterns in data
3. **Similarity-based**: Groups formed based on similarity measures

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**

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

Basic Clustering Process

1. **Feature Selection/Extraction**: Choose relevant features
2. **Similarity Measure**: Select distance metric (Euclidean, Manhattan, cosine, etc.)
3. **Algorithm Application**: Run clustering algorithm
4. **Evaluation**: Assess cluster quality
5. **Interpretation**: Analyze and interpret results

Clustering of Patterns:

Key Concepts in Clustering of Patterns

1. What is a Pattern?

A **pattern** is a data point represented as a feature vector in a multidimensional space. Examples include:

- **Customer data** (age, income, purchase history)
- **Images** (pixel values, color histograms)
- **Text documents** (TF-IDF vectors, word embeddings)
- **Biological data** (gene expressions, protein sequences)

2. Goal of Clustering Patterns

- Identify natural groupings in the data.
- Reduce complexity by summarizing similar patterns.
- Detect anomalies (patterns that don't belong to any cluster).
- Support decision-making (e.g., customer segmentation).

Types of Pattern Clustering Methods

Clustering Method	Description	Example Algorithms	Best Used When
Partition-based	Divides data into *k* non-overlapping clusters	K-Means, K-Medoids	Number of clusters (*k*) is known
Hierarchical	Builds a tree-like structure of clusters (nested clusters)	Agglomerative, Divisive	Data has hierarchical relationships

Clustering Method	Description	Example Algorithms	Best Used When
Density-based	Forms clusters based on dense regions of data points	DBSCAN, OPTICS	Clusters have arbitrary shapes
Model-based	Assumes data is generated from a probabilistic model	Gaussian Mixture Models (GMM)	Data fits a statistical distribution
Grid-based	Divides data space into grid cells and clusters them	STING, CLIQUE	Large datasets with uniform density
Spectral	Uses graph theory to find clusters based on connectivity	Spectral Clustering	Data has complex structures

Steps in Pattern Clustering

1. **Feature Extraction/Selection**
   - Choose relevant features (e.g., PCA for dimensionality reduction).
2. **Distance/Similarity Measure**
   - **Euclidean distance** (for numerical data).
   - **Cosine similarity** (for text/image data).
   - **Manhattan distance** (for high-dimensional data).
3. **Apply Clustering Algorithm**
   - Select an appropriate method (e.g., K-Means for spherical clusters, DBSCAN for noisy data).
4. **Evaluate Clusters**

    - **Internal validation** (Silhouette Score, Davies-Bouldin Index).
    - **External validation** (if ground truth exists, e.g., Adjusted Rand Index).
5. **Interpret & Use Results**
    - Assign labels (e.g., "High-value customers" vs. "Low-engagement users").
    - Use for recommendation systems, anomaly detection, etc.

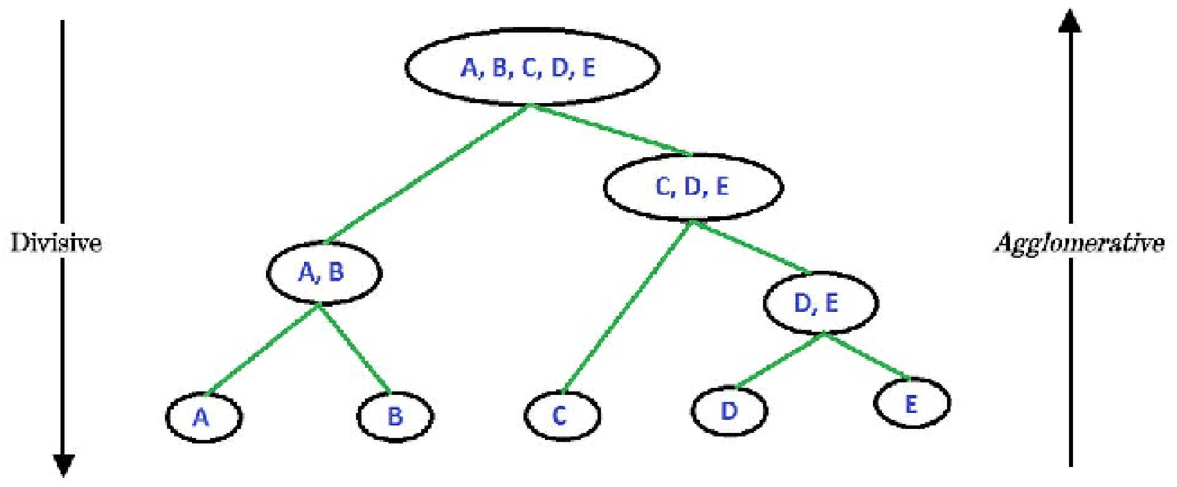
Applications of Pattern Clustering

- **Market Segmentation** → Group customers based on purchasing behavior.
- **Image Segmentation** → Cluster pixels to detect objects.
- **Document Clustering** → Organize news articles by topic.
- **Bioinformatics** → Group genes with similar expression patterns.
- **Fraud Detection** → Identify unusual transaction patterns.

Challenges in Pattern Clustering

- **Choosing the right number of clusters** (e.g., Elbow Method, Gap Statistic).
- **Handling high-dimensional data** (curse of dimensionality).
- **Dealing with noise and outliers** (DBSCAN is robust to this).
- **Scalability** (some algorithms struggle with big data).

Divisive Clustering:



Divisive clustering is a **hierarchical clustering** technique that follows a **top-down** strategy, starting with all data points in a single cluster and recursively splitting them into smaller clusters. Unlike **agglomerative clustering** (bottom-up), which merges clusters, divisive clustering partitions data by removing links between dissimilar points.

How Divisive Clustering Works

Algorithm Steps

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).

2. Common Splitting Methods

- **DIANA (Divisive Analysis)**:
  - Uses a distance matrix to iteratively separate the farthest points.
  - Works well for small datasets but is computationally expensive.
- **Bisecting K-Means**:
  - Splits clusters using K-Means (K=2) repeatedly.
  - Faster than DIANA and works well for large datasets.
- **Minimum Spanning Tree (MST)-based splitting**:
  - Removes the longest edge in a MST to divide clusters.

Advantages of Divisive Clustering

✓ **More interpretable than agglomerative clustering** (splits are easier to track).
✓ **Better for large datasets** when using efficient splitting methods (e.g., Bisecting K-Means).
✓ **Can produce more balanced clusters** compared to agglomerative approaches.

Disadvantages

Computationally expensive (especially DIANA, which is **O (n^2 log n)**).
Sensitive to initial splits (early mistakes propagate).
Requires a stopping criterion (number of clusters or threshold).

Applications of Divisive Clustering

- **Document organization** (splitting large text corpora into topics).
- **Market segmentation** (dividing customers into distinct groups).
- **Image segmentation** (hierarchically partitioning regions).
- **Biology** (taxonomy classification of species).

Agglomerative Clustering:

Agglomerative clustering is a hierarchical clustering method that follows a **bottom-up** strategy, where each data point starts as its own cluster and pairs of clusters are successively merged based on similarity. It produces a **dendrogram** that shows the hierarchical relationship between clusters at different levels of granularity.

How Agglomerative Clustering Works

1. Algorithm Steps

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).

2. Linkage Criteria (How to Measure Similarity Between Clusters)

Linkage Method	**Description**	**When to Use**
Single Linkage	Minimum distance between any two points in clusters	Tends to produce "chain-like" clusters
Complete Linkage	Maximum distance between any two points in clusters	Produces compact, well-separated clusters

Linkage Method	Description	When to Use
Average Linkage	Average distance between all pairs of points in clusters	Balanced approach, less sensitive to outliers
Ward's Method	Minimizes variance when merging clusters	Produces similarly sized clusters

Advantages of Agglomerative Clustering

✓ **No need to pre-specify the number of clusters** (can decide later using the dendrogram).
✓ **Produces a hierarchical structure** (useful for taxonomy and nested relationships).
✓ **Works well with small to medium-sized datasets**.

Disadvantages

Computationally expensive ($O(n^3)$ time complexity, not ideal for big data).
Sensitive to noise and outliers (especially with single linkage).
Once clusters are merged, they cannot be split again.

Applications

- **Biology**: Phylogenetic tree construction (evolutionary relationships).
- **Document clustering**: Grouping similar articles hierarchically.
- **Image segmentation**: Merging regions based on pixel similarity.
- **Social network analysis**: Detecting communities in networks.

Difference Agglomerative vs. Divisive Clustering

Agglomerative vs. Divisive Clustering

Feature	Agglomerative Clustering	Divisive Clustering
Approach	Bottom-up (merges clusters)	Top-down (splits clusters)
Speed	Slower ($O(n^3)$)	Faster with methods like Bisecting K-Means
Flexibility	Less flexible (no backtracking)	More flexible (can refine splits)
Use Cases	Better for small datasets with clear hierarchy	Better for large datasets needing coarse splits

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.

KMeans 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.

Mathematical Formulation

Minimizes Within-Cluster Sum of Squares (WCSS):

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

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):

$$J = \sum_{i} \sum_{j} (u_{ij})^m \|x_j - v_i\|^2$$

Soft Clustering:

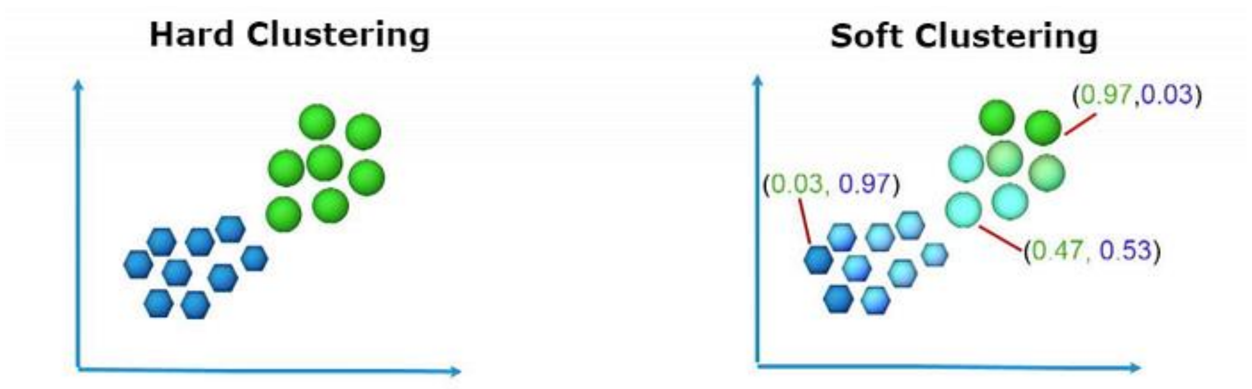
Soft clustering (probabilistic clustering) assigns data points to **multiple clusters with membership probabilities** rather than forcing hard assignments. This approach better handles:

- 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

Fuzzy C-Means Clustering:



Fuzzy C-Means (FCM) clustering is an **unsupervised soft clustering algorithm**, similar to K-Means, but with a key difference: **each data point can belong to multiple clusters with varying degrees of membership**, rather than being assigned to just one cluster.

Key Differences from K-Means:

- **K-Means**: Hard clustering — each point belongs to exactly one cluster.
- **Fuzzy C-Means**: Soft clustering — each point has a **membership score** (between 0 and 1) for each cluster.

How Fuzzy C-Means Works:

1. **Choose**:
    - Number of clusters C
    - Fuzziness parameter m>1 (controls the degree of fuzziness; typically m=2)
    - Convergence criteria (tolerance)
    -
2. **Initialize**:
    - Randomly initialize the membership matrix UUU, where each element u_{ij} indicates the membership of point x_j in cluster i
    -
3. **Repeat Until Convergence**:
    - **Update cluster centroids** c_i:

      $c_i=\sum_{j=1}^{N}u_{ij}^{m}\cdot x_{j/}\sum_{j=1}^{N} u_{ij}^{m}$

<u>Rough Clustering:</u>

Rough Clustering is a clustering technique based on **Rough Set Theory** (proposed by Zdzisław Pawlak). Unlike K-Means or Fuzzy C-Means, which use distance or membership degrees, rough clustering handles **imprecise, vague, or incomplete data** by creating **lower and upper approximations** of clusters.

Key Concepts:

- **Rough Set Theory**: A mathematical approach to deal with vagueness and uncertainty, using equivalence relations.
- **Lower Approximation**: The set of data points that **definitely** belong to a cluster.
- **Upper Approximation**: The set of data points that **possibly** belong to a cluster.
- **Boundary Region**: The difference between the upper and lower approximation — **uncertain membership**.

<u>Rough K-Means Clustering Algorithm:</u>

Algorithm Steps:

1. **Initialize**:
    - Choose K: number of clusters.
    - Randomly assign initial lower and upper approximations for each cluster.
2. **Centroid Computation**:
    - For each cluster, compute the centroid using:
        - Only lower approximation data (or)
        - A weighted sum of data from lower and boundary regions:

$$\text{Centroid}_i = w_{lower} \cdot \sum_{x \in \text{Lower}_i} x + w_{boundary} \cdot \sum_{x \in \text{Boundary}_i} x /$$

$w_{lower} \cdot |Lower_i| + w_{boundary} \cdot |Boundary_i$

where $w_{lower} > w_{boundary}$

3. **Assign Points**:
   - For each data point:
     - Assign to **lower approximation** of nearest cluster if distance is significantly smaller than to other centroids.
     - Otherwise, assign to the **upper approximations** of multiple clusters.
4. **Update Approximations**:
   - Recompute lower and upper approximations based on updated centroids.
5. **Repeat**:
   - Iterate steps 2–4 until convergence (e.g., no change in assignments or centroids).

Expectation Maximization-Based Clustering:

Expectation Maximization (EM)-Based Clustering **is a** probabilistic clustering method **that uses the** Expectation-Maximization algorithm **to estimate the parameters of a** mixture of probability distributions**, typically** Gaussian Mixture Models (GMMs)**.**

Core Concept:

Each data point is assumed to be generated from one of several underlying probability distributions (e.g., Gaussians), and the EM algorithm estimates the most likely parameters (means, variances, and mixing weights) of these distributions.

How EM-Based Clustering Works:

Step 1: Assume a Mixture Model

- Let the data be generated by a mixture of KKK components (e.g., Gaussians), each with its own parameters (mean μk\mu_kμk, covariance Σk\Sigma_kΣk, and mixing coefficient πk\pi_kπk).

Step 2: Initialize Parameters

- Randomly initialize the means, covariances, and mixing coefficients.

Step 3: Iterate Between Two Steps Until Convergence:

- **E-step (Expectation)**:
    - Compute the **posterior probability** (responsibility) that each data point x_i belongs to each cluster k:

      $\gamma_{ik}=\pi_k \cdot N(x_i | \mu_k, \Sigma_k) / \sum_{j=1}^{K} \pi_j \cdot N(x_i | \mu_j, \Sigma_j)$

  M-step (Maximization):

    - Update parameters using the responsibilities:
        - Mean:

          $\mu_k=\sum_{i=1}^{N} \gamma_{ik} x_i / \sum_{i=1}^{N} \gamma_{ik}$

        - Covariance:

          $\Sigma_k=\sum_{i=1}^{N} \gamma_{ik}(x_i-\mu_k)(x_i-\mu_k)^T / \sum_{i=1}^{N} \gamma_{ik}$

Spectral Clustering:

Spectral Clustering is a powerful unsupervised learning algorithm that uses the **spectrum (eigenvalues)** of a similarity matrix to perform

dimensionality reduction before clustering in fewer dimensions. It's especially effective for detecting **non-convex clusters**, where traditional methods like K-Means fail.

Rather than clustering data points directly, **Spectral Clustering**:

1. Builds a **similarity graph** of the data.
2. Computes the **Laplacian matrix** of the graph.
3. Uses **eigenvectors** of the Laplacian to embed the data in a low-dimensional space.
4. Applies a clustering algorithm (e.g., K-Means) on this embedding.

Step-by-Step Algorithm:

1. **Construct Similarity Graph**:
    - Compute a similarity matrix SSS, e.g., using a Gaussian (RBF) kernel:

      $S_{ij}=\exp(-\|x_i-x_j\|^2/2\sigma^2)$

      Or use a k-nearest neighbors (k-NN) graph.