Chapter 6: Data Mining Methods: Unsupervised Methods

Sunil Regmi, Lecturer

DoAI, Kathmandu University

Cluster Analysis: Introduction

- **Clustering:** Grouping objects so that:
  - Objects in the same cluster are similar.
  - Objects in different clusters are dissimilar.
- **Key Characteristics:**
  - No predefined class labels; used in unsupervised learning.
  - Discovers natural groupings in data.
  - Requires different techniques from classification/association.

Clustering vs. Classification

- **Clustering:**
  - No target variable to predict.
  - Finds hidden patterns or groupings.
  - Maximizes intra-group similarity; minimizes inter-group similarity.
  - A form of **unsupervised learning**.

Applications of Cluster Analysis

- **Business:** Customer segmentation for marketing.
- **Web Search:** Organize search results by intent/topic.
- **Climate Studies:** Identify patterns in pressure/ocean affecting land climate.
- **Biology:** Group genes by function, classify species.

Types of Clustering Algorithms (1/2)

- **1. Partitioning Methods:**
  - Divides n objects into k clusters ($k \leq n$).
  - Iteratively reassigns objects to improve clustering.
  - E.g., *k-means*, *k-medoids*.
- **2. Hierarchical Methods:**
  - Builds a tree (dendrogram) of nested clusters.
  - Types: Agglomerative (bottom-up), Divisive (top-down).
  - Pros: No need to specify k; Cons: Irreversible decisions.

Types of Clustering Algorithms (2/2)

- **3. Density-Based Methods:**
  - Clusters based on density of points (e.g., *DBSCAN*).
  - Can discover clusters of arbitrary shape.
- **4. Grid-Based Methods:**
  - Space is quantized into grids.
  - Fast processing, independent of number of objects.
- **5. Model-Based Methods:**
  - Assumes underlying distribution (e.g., Gaussian Mixture Models).
  - Automatically determines number of clusters.
  - Handles noise effectively.

Partitioning Algorithms

- Divides dataset into k disjoint clusters satisfying:
  - Each cluster has at least one object.
  - No overlap: one object = one cluster.
- Optimizes objective function (e.g., intra-cluster distance).
- Works well for spherical clusters in small/medium datasets.

K-means Clustering

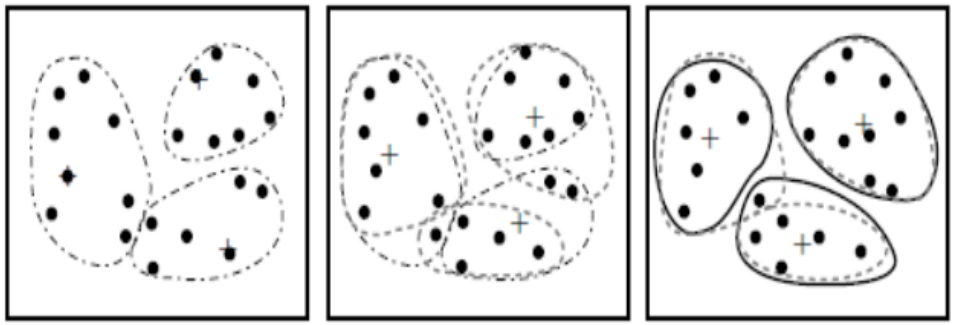
- Defines clusters using a **centroid** (mean of points in a group)
- Process:
  - Choose **k** initial centroids (user-specified)
  - Assign points to closest centroid
  - Update centroids based on assigned points
  - Repeat until centroids stabilize

K-means Algorithm

Input: k (number of clusters), D (data set with n objects)
Output: Set of k clusters
Method:

1. Arbitrarily choose k objects from D as initial cluster centers
2. Repeat:
   - Form k clusters by assigning each point to its closest centroid
   - Recompute the centroid of each cluster
3. Until centroids do not change

K-means Illustration



Clustering of a set of objects based on the *k*-means method

- Figure above: Shows K-means with $k = 3$
  - Centroids marked by '+'
  - Points assigned to clusters (dotted outlines)
- Steps:
  - (a) Initial random centroids, assign points
  - (b) Update centroids, reassign points
  - (c) Final clusters (no changes)

Example 1: K-means (Dataset)

- Data set: $D = \{2, 3, 4, 10, 11, 12, 20, 25, 30\}$
- Task: Divide into **three clusters** ($k = 3$)
- Initial centroids (chosen arbitrarily): 2, 10, 20
- Use Euclidean distance for assignment

Example 1: Iteration 1 (K-means)

Assign Points to Closest Centroid:

Centroid	Points (Distances)
2	{2, 3, 4} (0, 1, 2)
10	{10, 11, 12} (0, 1, 2)
20	{20, 25, 30} (0, 5, 10)

Update Centroids:

- Cluster 1:

$$\text{mean}(\{2, 3, 4\}) = \frac{2+3+4}{3} = 3$$

- Cluster 2:

$$\text{mean}(\{10, 11, 12\}) = \frac{10+11+12}{3} = 11$$

- Cluster 3:

$$\text{mean}(\{20, 25, 30\}) = \frac{20+25+30}{3} = 25$$

New Centroids

{3, 11, 25}

Example 1: Iteration 2 and Convergence

Assign Points to New Centroids:

Centroid	Points
3	{2, 3, 4}
11	{10, 11, 12}
25	{20, 25, 30}

Convergence Details:

- Update centroids: Same (3, 11, 25).
- Convergence: No changes in assignments.

Final Clusters

Cluster 1: {2, 3, 4} Cluster 2: {10, 11, 12} Cluster 3: {20, 25, 30}

Example 2: K-means (Dataset)

Transactions:

TID	Items
T1	Bread, Jelly, Butter
T2	Bread, Butter
T3	Bread, Milk, Butter
T4	Coke, Bread
T5	Coke, Milk

Binary Vectors (Bread, Butter, Jelly, Milk, Coke):

TID	Vector
T1	(1, 1, 1, 0, 0)
T2	(1, 1, 0, 0, 0)
T3	(1, 1, 0, 1, 0)
T4	(1, 0, 0, 0, 1)
T5	(0, 0, 0, 1, 1)

Task

Cluster with $k = 2$, up to 2 iterations

Example 2: Initial Setup

- Initial centroids (arbitrarily chosen): T1 (1, 1, 1, 0, 0), T5 (0, 0, 0, 1, 1)
- **Assign points to closest centroid** (Euclidean distance):
  - T1 to T1: distance $= 0$
  - T2 to T1: distance $= \sqrt{(0-1)^2} = 1$; to T5: $\sqrt{(1-0)^2 + (1-0)^2 + (0-1)^2 + (0-1)^2} = 2$
  - T3 to T1: distance $= \sqrt{(0-1)^2} = 1$; to T5: $\sqrt{(1-0)^2 + (1-0)^2 + (1-1)^2} = \sqrt{2}$
  - T4 to T1: distance $= \sqrt{(0-1)^2 + (1-0)^2} = \sqrt{2}$; to T5: $\sqrt{(1-0)^2 + (0-1)^2} = \sqrt{2}$
  - T5 to T5: distance $= 0$
- Clusters:
  - Centroid T1: {T1, T2, T3, T4}
  - Centroid T5: {T5}

Example 2: Iteration 1

- **Update centroids**:
  - Cluster 1 ({T1, T2, T3, T4}): Mean of (1,1,1,0,0), (1,1,0,0,0), (1,1,0,1,0), (1,0,0,0,1) = $(1, \frac{3}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$
  - Cluster 2 ({T5}): (0, 0, 0, 1, 1)
- **Reassign points** (to centroids $(1, 0.75, 0.25, 0.25, 0.25)$, (0, 0, 0, 1, 1)):
  - T1: Closer to Cluster 1 (distance ≈ 0.79) than Cluster 2 (≈ 2.24)
  - T2: Closer to Cluster 1 (≈ 0.5) than Cluster 2 (≈ 2)
  - T3: Closer to Cluster 1 (≈ 0.87) than Cluster 2 (≈ 1.41)
  - T4: Closer to Cluster 1 (≈ 0.87) than Cluster 2 (≈ 1.41)
  - T5: Closer to Cluster 2 (distance $= 0$) than Cluster 1 (≈ 1.5)
- Clusters: {T1, T2, T3, T4}, {T5}

Example 2: Iteration 2

- **Update centroids**: Same as previous (no change in assignments)
- Clusters:
  - Cluster 1: {T1, T2, T3, T4} (Bread-heavy transactions)
  - Cluster 2: {T5} (Coke, Milk transaction)
- Stop after 2 iterations (as specified)

Strengths of K-means

- **Simple** and versatile for various data types
- Effective for **compact, well-separated** clusters
- **Scalable** and efficient for large data sets
- Often performs well with multiple runs

Weaknesses of K-means

- Struggles with **outliers** (outlier detection can help)
- Not suitable for **nonconvex** clusters or clusters of **varying sizes**
- Requires a notion of a **centroid**, limiting use with categorical data

K-medoids Clustering

- A **partitioning clustering** method
- Uses actual data points as **medoids** (representative objects) instead of centroids
- Goal: Minimize the sum of dissimilarities between points and their cluster medoid
- More **robust to outliers** compared to K-means

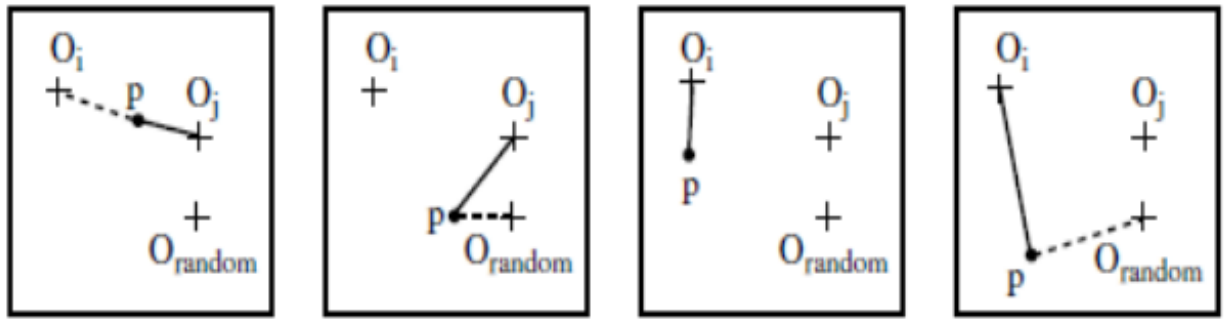
K-medoids Algorithm

Input: k (number of clusters), D (dataset with n objects)
Output: Set of k clusters
Method:

1. Arbitrarily select k objects from D as initial medoids
2. Repeat:
   - Assign each object to the cluster with the **nearest medoid**
   - Randomly select a **non-medoid** object O_{random}
   - Compute total cost (sum of dissimilarities) of swapping medoid O_j with O_{random}
   - If total cost decreases, perform the swap
3. Until no change in medoids

Example: K-medoids Setup

- Dataset: $D = \{2, 4, 6, 10, 12, 100\}$, $k = 2$
- Initial medoids: 2, 10
- **Initial assignment** (Manhattan distance):
  - Medoid 2: {2, 4, 6} (distances: 0, 2, 4)
  - Medoid 10: {10, 12, 100} (distances: 0, 2, 90)
- **Total cost**:
  $|4-2| + |6-2| + |12-10| + |100-10| = 2+4+2+90 = 98$



Four cases of the cost function for *k*-medoids clustering

Example: Case 1 (Swap Medoid 10 with 4)

- New medoids: 2, 4
- Assign points:
  - Medoid 2: $\{2, 10, 12, 100\}$ (distances: 0, 8, 10, 98)
  - Medoid 4: $\{4, 6\}$ (distances: 0, 2)
- **Total cost**:
  $|10-2|+|12-2|+|100-2|+|6-4| = 8+10+98+2 = 118$
- **Result**: Cost increases ($118 > 98$), do not swap

Example: Case 2 (Swap Medoid 10 with 6)

- New medoids: 2, 6
- Assign points:
  - Medoid 2: $\{2, 4, 10, 12, 100\}$ (distances: 0, 2, 8, 10, 98)
  - Medoid 6: $\{6\}$ (distance: 0)
- **Total cost**:
  $|4-2|+|10-2|+|12-2|+|100-2| = 2+8+10+98 = 118$
- **Result**: Cost increases ($118 > 98$), do not swap

Example: Case 3 (Swap Medoid 10 with 12)

- New medoids: 2, 12
- Assign points:
  - Medoid 2: $\{2, 4, 6\}$ (distances: 0, 2, 4)
  - Medoid 12: $\{10, 12, 100\}$ (distances: 2, 0, 88)
- **Total cost**:
  $|4-2| + |6-2| + |10-12| + |100-12| = 2+4+2+88 = 96$
- **Result**: Cost decreases ($96 < 98$), perform swap

Example: Case 4 (Swap Medoid 10 with 100)

- New medoids: 2, 100
- Assign points:
  - Medoid 2: $\{2, 4, 6, 10, 12\}$ (distances: 0, 2, 4, 8, 10)
  - Medoid 100: $\{100\}$ (distance: 0)
- **Total cost**:
  $|4-2| + |6-2| + |10-2| + |12-2| = 2 + 4 + 8 + 10 = 24$
- **Result**: Cost decreases ($24 < 98$), perform swap (better than Case 3)

Example: Final Clusters

- After Case 4 (lowest cost), medoids: 2, 100
- Final clusters:
  - Cluster 1 (Medoid 2): {2, 4, 6, 10, 12}
  - Cluster 2 (Medoid 100): {100}
- Total cost: 24 (optimal for this iteration)
- Continue checking swaps until no further cost reduction

Advantages of K-medoids

- **Robust to outliers**: Uses actual objects, reducing impact of extreme values (e.g., 100)
- **Flexible distance measures**: Works with any dissimilarity metric
- **Interpretable**: Medoids are actual data points

Limitations of K-medoids

- **Computationally expensive**: Swap evaluations require multiple distance calculations
- **Sensitive to initial medoids**: Poor choices may lead to suboptimal clusters
- **Fixed** k: Number of clusters must be predefined

K-medoids vs. K-means

K-medoids:

- Uses medoids (actual points)
- Robust to outliers
- Higher computational cost

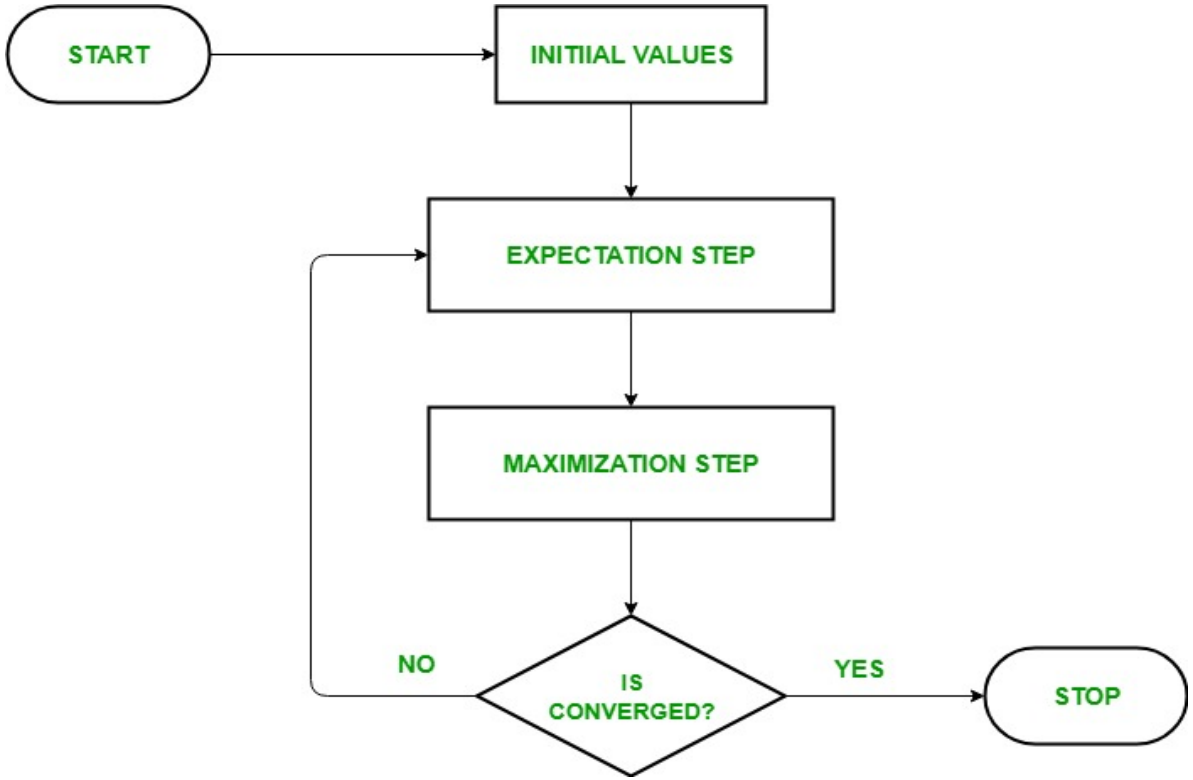
K-means:

- Uses centroids (mean)
- Sensitive to outliers
- More efficient

Introduction to EM Algorithm

- A **model-based clustering** method for finding maximum likelihood estimates
- Used when data has **latent variables** (unobserved variables)
- Iteratively refines parameters to fit a probabilistic model (e.g., Gaussian Mixture Models)
- Goal: Maximize the likelihood of observed data

Flowchart of How EM works



How EM Works

- Assumes data is generated by a mixture of underlying probability distributions
- Two main steps:
  - **Expectation (E-step)**: Estimate probabilities of latent variables given current parameters
  - **Maximization (M-step)**: Update parameters to maximize likelihood based on E-step
- Iterates until convergence (parameters stabilize)

EM Algorithm Steps

Input: Observed data, number of clusters k, initial parameters
Output: Optimized model parameters
Method:

1. Initialize parameters (e.g., means, variances for Gaussian Mixture Models)
2. **E-step**: Compute expected values of latent variables (posterior probabilities)
3. **M-step**: Update parameters to maximize expected log-likelihood
4. Repeat E-step and M-step until convergence

Example: Gaussian Mixture Model

- Data: Points assumed to come from k Gaussian distributions
- **Initialization**: Choose k means, variances, and mixing coefficients
- **E-step**: Calculate probability of each point belonging to each Gaussian
- **M-step**: Update means, variances, and coefficients based on probabilities
- Result: Clusters correspond to Gaussian components

Dataset

- Data: | sr_no | age | amount |
- C1: 20, 500
- C2: 40, 1000
- C3: 30, 800
- C4: 18, 300
- C5: 28, 1200
- C6: 35, 1400
- C7: 45, 1800

Assumption: Mixture of 2 Gaussians, $k = 2$

EM Algorithm Overview

- Purpose: Estimate parameters of a mixture model (e.g., GMM) with latent variables.
- Steps:
  1. Initialize parameters (means, variances, mixing coefficients).
  2. E-Step: Compute expected cluster assignments (responsibilities).
  3. M-Step: Maximize parameters based on responsibilities.
  4. Repeat until convergence.

Initialization

- $k = 2$ clusters
- Initial means:
  - $\mu_1 = (20, 500)$ (C1)
  - $\mu_2 = (45, 1800)$ (C7)
- Initial variances (diagonal): $\sigma_1^2 = (10^2, 100^2)$, $\sigma_2^2 = (10^2, 100^2)$
- Mixing coefficients: $\pi_1 = 0.5$, $\pi_2 = 0.5$

E-Step (Expectation)

- Compute responsibility $\gamma_{i,k}$:

$$\gamma_{i,k} = \frac{\pi_k \cdot N(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{k} \pi_j \cdot N(x_i|\mu_j, \Sigma_j)}$$

- Example for C2 (40, 1000):
  - Approximate $N(C2|\mu_1)$ and $N(C2|\mu_2)$
  - $\gamma_{C2,1} \approx 0.7$, $\gamma_{C2,2} \approx 0.3$ (based on proximity)
- Repeat for all points.

M-Step (Maximization)

- Update parameters:
  - $\mu_k = \frac{\sum_{i=1}^{n} \gamma_{i,k} x_i}{\sum_{i=1}^{n} \gamma_{i,k}}$ (weighted mean)
  - $\Sigma_k = \frac{\sum_{i=1}^{n} \gamma_{i,k} (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^{n} \gamma_{i,k}}$ (weighted covariance)
  - $\pi_k = \frac{\sum_{i=1}^{n} \gamma_{i,k}}{n}$ (weighted proportion)
- Example: Recalculate μ_1, μ_2 using all $\gamma_{i,k}$.

Iteration Example

- After E-Step and M-Step:
  - New μ_1 shifts toward points with high $\gamma_{i,1}$ (e.g., C1, C4).
  - New μ_2 shifts toward C5, C6, C7.
- Repeat E and M steps until $\mu_1, \mu_2, \Sigma_1, \Sigma_2$ stabilize.
- Final clusters approximate:
  - Cluster 1: {C1, C2, C3, C4}
  - Cluster 2: {C5, C6, C7}

Result

- EM converges to optimal Gaussian parameters.
- Clusters reflect data distribution (e.g., younger/lower amounts vs. older/higher amounts).
- Unlike K-Medoids, EM models probabilistic assignments, handling overlap naturally.

Applications of EM Algorithm

- **Clustering**: Gaussian Mixture Models for data segmentation
- **Missing Data**: Impute missing values in datasets
- **Image Processing**: Image segmentation and reconstruction
- **Natural Language Processing**: Parameter estimation in hidden Markov models

Advantages of EM Algorithm

- **Flexible**: Handles latent variables and incomplete data
- **Probabilistic**: Provides soft clustering (probability-based assignments)
- **Robust**: Converges to local maximum of likelihood function
- **Versatile**: Applicable to various probabilistic models

Disadvantages of EM Algorithm

- **Sensitive to Initialization**: May converge to local optima
- **Computationally Intensive**: Requires multiple iterations over data
- **Assumes Model**: Performance depends on correct model specification
- **Convergence Speed**: Can be slow for large datasets

EM vs. K-means

EM Algorithm:

- Probabilistic (soft clustering)
- Handles latent variables
- Model-based (e.g., Gaussian)

K-means:

- Deterministic (hard clustering)
- Uses centroids
- Distance-based

Introduction to Density-Based Clustering

- Discovers clusters of **arbitrary shape** in data with noise
- Clusters: Dense regions of objects separated by low-density regions
- **DBSCAN**: Density-Based Spatial Clustering of Applications with Noise
  - Grows clusters based on density connectivity
  - Effective and simple algorithm

DBSCAN: Key Concepts

- Defines clusters as maximal sets of **density-connected points**
- **Parameters**:
  - ε: Radius of neighborhood
  - MinPts: Minimum number of points in ε-neighborhood
- Density estimation: Count points within ε-radius of a point

Point Types in DBSCAN

- **Core Point**: Has at least MinPts points (including itself) within ε-neighborhood
  - Example (Fig. 3, MinPts $= 7$): Point A is core (7 points in neighborhood)
- **Border Point**: Not core, but within ε-neighborhood of a core point
  - Example: Point B is border (near core point A)
- **Noise Point**: Neither core nor border
  - Example: Point C is noise

Density Relationships

- **Directly Density-Reachable**: Point p is within ε-neighborhood of core point q
- **Density-Reachable**: Chain of directly density-reachable points from q to p
  - Example (Fig. 4, MinPts $= 3$): q is density-reachable from p via m
- **Density-Connected**: Points p and q are both density-reachable from a common point
  - Example: o, r, and s are density-connected

DBSCAN Algorithm

Input: Dataset D, ε, MinPts
Output: Set of clusters
Method:

1. Label all points as **core**, **border**, or **noise**
2. Eliminate noise points
3. Connect core points within ε of each other with edges
4. Form clusters from connected core points
5. Assign border points to associated core point clusters

Example: Density Connectivity (Fig. 4)

- Parameters: MinPts $= 3$, ε (circle radius)
- Core points: m, p, o, r (each has ≥ 3 points in neighborhood)
- Relationships:
  - q is directly density-reachable from m
  - q is density-reachable from p (via m)
  - o, r, s are density-connected

Strengths of DBSCAN

- **Handles arbitrary shapes**: Finds clusters K-means cannot
- **Noise-resistant**: Identifies and excludes outliers
- **Flexible**: No need to specify number of clusters

Weaknesses of DBSCAN

- **Varying densities**: Struggles with clusters of different densities
- **High-dimensional data**: Density estimation becomes challenging
- **Parameter sensitivity**: Results depend on ε and MinPts

DBSCAN vs. K-means

DBSCAN:

- Density-based
- Arbitrary shapes
- Noise-resistant

K-means:

- Centroid-based
- Spherical clusters
- Sensitive to outliers

Introduction to Hierarchical Clustering

- Second important category of clustering methods, alongside K-means.
- Relatively old but still widely used.
- Groups data objects into a tree of clusters.
- Classified as:
  - **Agglomerative** (bottom-up, merging)
  - **Divisive** (top-down, splitting)
- Limitation: No adjustment possible after merge or split decisions.

Types of Hierarchical Clustering

- **Agglomerative**:
  - Starts with points as individual clusters.
  - Merges the closest pair of clusters at each step.
  - Requires a definition of cluster proximity.
- **Divisive**:
  - Starts with one all-inclusive cluster.
  - Splits a cluster at each step until only singleton clusters remain.
  - Requires deciding which cluster to split and how.

Dendrogram Representation

- Hierarchical clustering displayed as a tree-like diagram called a dendrogram.
- Shows:
  - Cluster-subcluster relationships.
  - Order of merges (agglomerative) or splits (divisive).
- Useful for visualizing the hierarchical structure of the data.

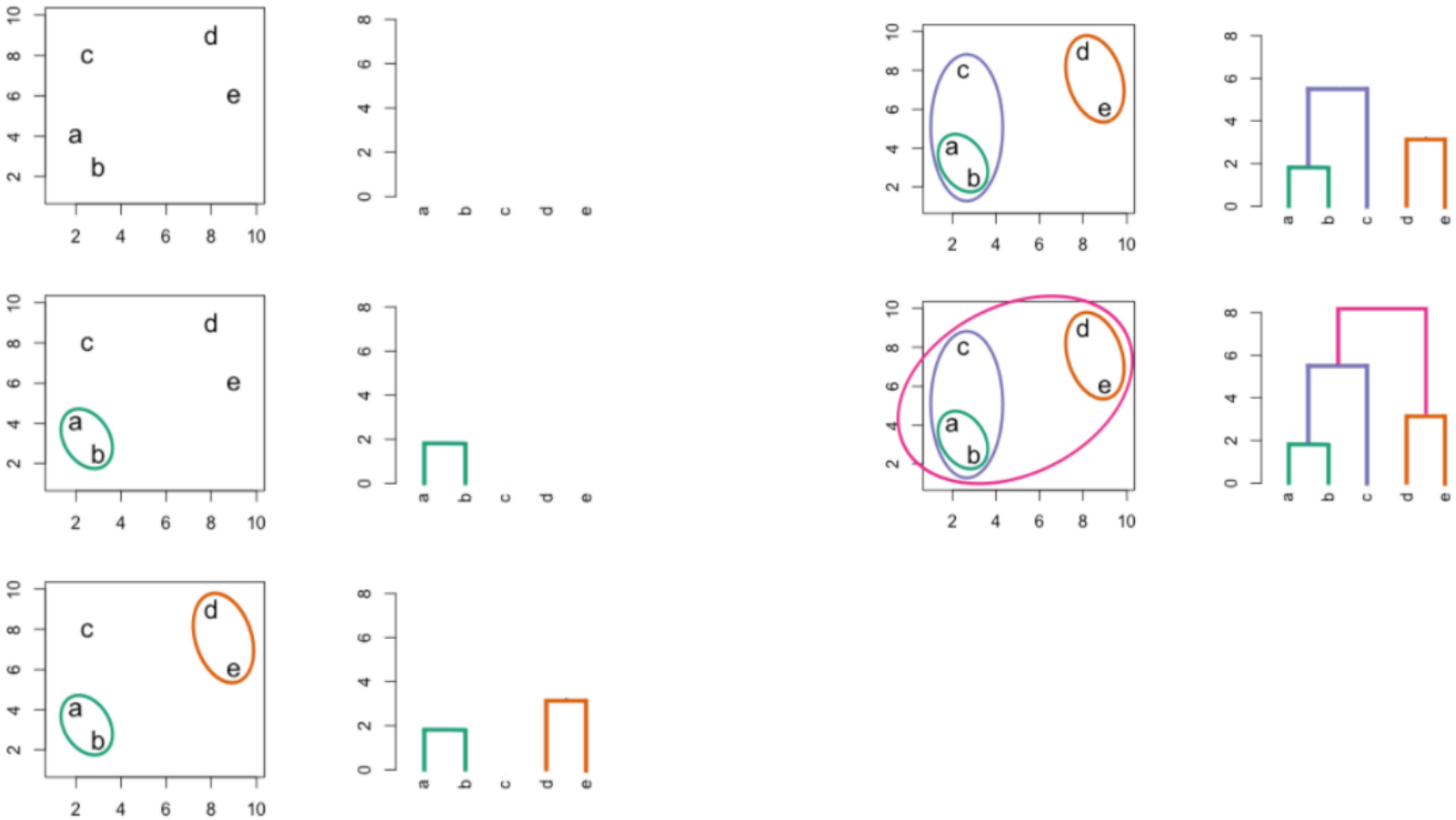
Agglomerative Clustering

- More frequently used in real-world applications than divisive methods.
- Starts with individual points as clusters.
- Successively merges the two closest clusters until one cluster remains.

Agglomerative Clustering Algorithm

1. Compute the proximity matrix, if necessary.
2. Repeat:
   1. Merge the closest two clusters.
   2. Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
3. Until only one cluster remains.

Example of Hierarchical Clustering in Action



Interpreting a Dendrogram

- At the bottom, each leaf represents one observation; leaves fuse into branches as we move up.
- Fusions indicate similarity: earlier fusions mean more similar groups.
- **Caveat:** Proximity on the horizontal axis does *not* imply similarity.
- Similarity is judged by the vertical axis height where branches first fuse.
- Final clusters are identified by a horizontal cut across the dendrogram; observations below the cut form clusters.
- Advantage: A single dendrogram offers multiple clustering options based on cut height.
- Challenge: Choosing the cut height is often unclear; typically based on fusion heights and desired cluster number.
- "Hierarchical" implies nested clusters; cuts at greater heights contain those at lower heights.

Interpreting a Dendrogram

- "Hierarchical" implies nested clusters; cuts at greater heights contain those at lower heights.
- Limitation: Hierarchical structure may not suit all datasets (e.g., non-nested clusters like gender vs. nationality).
- Example: Males/females from America, Japan, France—gender (2 clusters) and nationality (3 clusters) are not nested; K-means may be better.

(a) Single-linkage: maximum similarity/ minimum distance

(b) Complete-linkage: minimum similarity/ maximum distance

(c) Average-linkage: average inter-similarity/ average inter-distance

Linkage Methods

- **Complete Linkage**
- Computes all pairwise dissimilarities between observations in cluster A and cluster B.
- Records the **largest** dissimilarity.
- Commonly used in practice due to its focus on maximum separation.

- **Single Linkage**
- Opposite of complete linkage; records the **smallest** dissimilarity.
- Can result in extended, trailing clusters where observations fuse one-at-a-time.
- May lead to "chaining" effect in dendrograms.

Linkage Methods (cont.)

- **Average Linkage**
- Computes all pairwise dissimilarities between observations in cluster A and cluster B.
- Records the **average** of the dissimilarities.
- Commonly used in practice for a balanced approach.

- **Centroid Linkage**
- Determines dissimilarity between the centroid of cluster A and the centroid of cluster B.
- Often used in genomics.
- Drawback: Potential **inversion**, where fusion height is below individual cluster heights in the dendrogram.

Choice of Initial Dissimilarity Measure

- First step in hierarchical clustering: Define pairwise dissimilarity for each observation pair.
- Options:
  - **Euclidean Distance**: Straight-line distance between two observations.
  - **Correlation-based Distance**: Similarity based on feature correlation, ignoring magnitude.

Impact of Dissimilarity Measure

- Choice strongly affects the final dendrogram.
- Example: Clustering shoppers by item purchases.
  - **Euclidean Distance**: Groups shoppers with few purchases together.
  - **Correlation-based Distance**: Groups shoppers with similar preferences, regardless of volume.
- Preferred choice: Correlation-based for preference-based clustering.

Scaling Variables

- Good practice: Scale all variables to have a standard deviation of one.
- Ensures equal importance of each variable in clustering.
- Exception: May not be desired if certain variables should dominate.

Introduction to Cluster Validation

- Cluster validation assesses the quality and reliability of clustering results.
- Essential for determining if the identified clusters are meaningful.
- Two main approaches:
  - **Intrinsic Methods**: Evaluate clusters based on internal properties.
  - **Extrinsic Methods**: Compare clusters to external ground truth.

Intrinsic Methods

- Assess clustering quality using only the data and clustering structure.
- No external information required.
- Common measures:
  - **Silhouette Coefficient**: Measures how similar an object is to its own cluster vs. others (range: -1 to 1, higher is better).
  - **Dunn Index**: Ratio of the smallest inter-cluster distance to the largest intra-cluster distance (higher is better).
  - **Davies-Bouldin Index**: Average similarity ratio of each cluster with its most similar cluster (lower is better).
- Useful for unsupervised learning when ground truth is unavailable.

Extrinsic Methods

- Evaluate clustering by comparing to known external labels or ground truth.
- Requires labeled data for validation.
- Common measures:
  - **Rand Index**: Measures similarity between predicted and true clusters (range: 0 to 1, higher is better).
  - **Adjusted Rand Index (ARI)**: Corrects Rand Index for chance (range: -1 to 1, higher is better).
  - **Normalized Mutual Information (NMI)**: Quantifies shared information between predicted and true clusters (range: 0 to 1, higher is better).
- Ideal for supervised or semi-supervised settings.

Comparison of Intrinsic vs. Extrinsic Methods

- **Intrinsic**:
  - Pros: No need for labeled data, fully data-driven.
  - Cons: May not reflect real-world relevance without context.
- **Extrinsic**:
  - Pros: Directly validates against known outcomes, more interpretable.
  - Cons: Requires labeled data, which may not always be available.
- Choice depends on data availability and problem context.

Practical Considerations

- Combine intrinsic and extrinsic methods for robust validation when possible.
- Preprocess data (e.g., scaling) to ensure fair distance calculations.
- Interpret results cautiously: High scores don't guarantee practical utility.
- Example: For your dataset (| sr_no | age | amount |), use Silhouette for intrinsic validation or ARI if age groups are labeled.