UNSUPERVISED LEARNING

DIFFERENCE BETWEEN SUPERVISED AND UNSUPERVISED LEARNING

Aspect	Supervised Learning	Unsupervised Learning
Input Data	Labeled data (features + correct output/label)	Unlabeled data (features only)
Goal	Predict outcomes based on training data	Discover patterns, groupings, or structures in data
Output	Known labels or targets	No predefined labels
Common Tasks	Classification (e.g., spam detection), Regression (e.g., price prediction)	Clustering (e.g., customer segmentation), Dimensionality Reduction
Examples	Predicting house prices, email spam filtering	Customer segmentation, anomaly detection
Evaluation	Accuracy measured by comparison with known labels	Harder to evaluate; often subjective
Algorithms	Linear Regression, Logistic Regression, SVM, Decision Trees, Neural Networks	K-Means, DBSCAN, PCA, Hierarchical Clustering
Real-World Analogy	A teacher provides answers, and the student learns to predict correctly	A student finds patterns in a book without any answers

---

WHAT IS CLUSTER ANALYSIS?

Definition: Cluster analysis groups data objects based on information found in the data that describes the objects and their relationships.

Goal: Objects **within a group should be similar** (or related), and objects in **different groups should be dissimilar** (or unrelated). Better clustering achieves greater similarity within groups and differences between groups.

Challenges in Defining Clusters:

- The notion of a cluster is often **imprecise** and depends on:
  - The **nature of the data**.
  - The **desired results**.
- **Example**:
  - 20 points are divided into clusters in different ways:

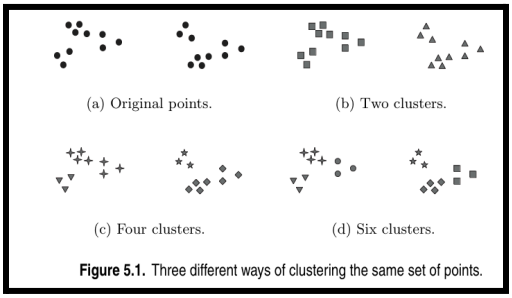


Figure 5.1. Three different ways of clustering the same set of points.

        - **Two clusters**.
        - **Four clusters**.
        - **Six clusters**.
    - The division into subclusters may be due to the **human visual system**, making it subjective.
    - Deciding the correct number of clusters depends on the **application and interpretation**.

Clustering vs. Classification:

- Clustering can be viewed as a form of classification where objects are labeled with **class (cluster) labels** derived only from the data.
- Classification is **supervised classification**, where:
    - A model is developed using objects with **known class labels**.
    - New, unlabeled objects are assigned class labels.
- For this reason, **clustering** is referred to as **unsupervised classification**.

Terminology:

- **Classification:** In data mining, generally refers to supervised classification unless specified.
- **Segmentation and Partitioning:**
    - Sometimes used synonymously with clustering but often refer to different techniques:
        - **Partitioning:** Divides graphs into subgraphs (e.g., graph partitioning).
        - **Segmentation:** Groups data based on simple techniques (e.g., segmenting images by pixel intensity or grouping people by income).
    - Some work in graph partitioning and image segmentation overlaps with cluster analysis.

What Is a Good Clustering?

Characteristics of High-Quality Clusters:

- **High intra-class similarity**: Objects within a cluster are highly similar.
- **Low inter-class similarity**: Objects in different clusters are highly dissimilar.

Factors Influencing Clustering Quality:

1. **Similarity Measure**:
    - The quality depends on the **similarity (or dissimilarity) measure** used and how it is implemented.
    - Similarity is typically expressed as a **distance function** (e.g., d(i,j)d(i, j)d(i,j)).
2. **Hidden Patterns**:
    - A good clustering method discovers some or all **hidden patterns** in the data.

Types of Data in Cluster Analysis

1. **Interval-Scaled Variables:**

- Data measured on a continuous scale with equal intervals.
- Examples: Temperature, height, weight, age.

2. **Binary Variables:**
   - Data with two possible states (e.g., 0 or 1, True or False).
3. **Nominal, Ordinal, and Ratio Variables:**
   - Nominal Variables: Categories without an order.
     - Example: Colors (red, blue, green), genders (male, female).
   - Ordinal Variables: Categories with a meaningful order but no fixed interval.
     - Example: Rankings (low, medium, high).
   - Ratio Variables: Continuous data with a meaningful zero point.
     - Example: Income, distance, time.
4. **Variables of Mixed Types:**
   - Data consisting of a combination of different types (e.g., interval, nominal, binary).

SIMILARITY AND DISSIMILARITY

1. **Importance:**
   - Used in clustering, some classification, and anomaly detection.
2. **Similarity:**
   - A numerical measure of how alike two data objects are.
   - Higher values indicate greater similarity.
   - Typically falls within the range [0, 1].
3. **Dissimilarity:**
   - A numerical measure of how different two data objects are.
   - Lower values indicate greater similarity.
   - Minimum dissimilarity is often 0.

I Interval-Scaled Variables:

Dissimilarities Between Data Objects with Multiple Numeric Attributes:

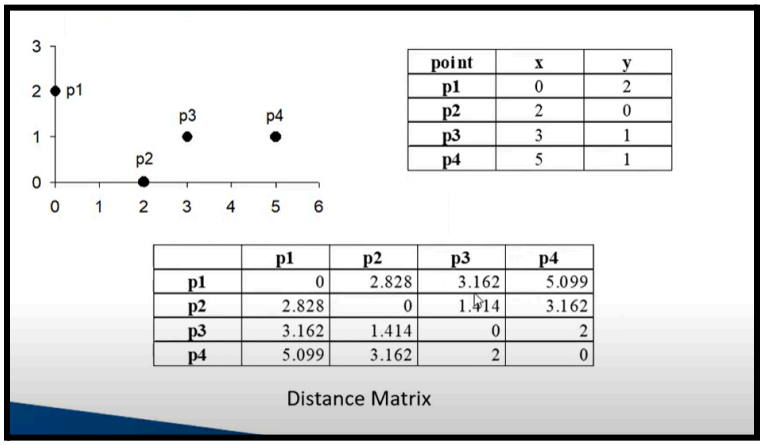
1. **Euclidean Distance:**
   - Formula:

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Where n is the number of dimensions and p_k and q_k are, respectively, the k^{th} attributes of data objects p and q.

Standardization is Necessary if the scales of the numeric attributes differ to ensure fair comparison.

Example:



2. Minkowski Distance

- Definition: A generalization of Euclidean Distance, used to calculate the distance between two objects p and q.
- Formula:

$$dist = \left(\sum_{k=1}^{n} | p_k - q_k |^r \right)^{\frac{1}{r}}$$

Where r is a parameter, n is the number of dimensions and p_k and q_k are, respectively, the kth attributes of data objects p and q.

- Special Cases:
  - When r=1: Manhattan Distance.
  - When r=2: Euclidean Distance

Example:

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Common Properties of a Distance

Definition: Distances, such as the Euclidean distance, have the following well-known properties:

1. **Positive Definiteness:**

$$d(p,q) \geq 0 \text{ for all } p \text{ and } q.$$
$$d(p,q) = 0 \text{ if and only if } p = q.$$

2. **Symmetry:**

$$d(p,q) = d(q,p) \text{ for all } p \text{ and } q.$$

3. **Triangle Inequality:**

$$d(p,r) \leq d(p,q) + d(q,r) \text{ for all points } p, q, \text{ and } r.$$

Where d(p,q)represents the distance (dissimilarity) between data points p and q.

A distance measure that satisfies all the above properties is called a metric.

II Binary Variables:

Similarity Between Binary Vectors

- **Common Scenario**:
  - Objects p and q have only **binary attributes** (0 or 1).
- **Quantities for Computing Similarities**:

$$M_{01}: \text{Number of attributes where } p = 0 \text{ and } q = 1.$$
$$M_{10}: \text{Number of attributes where } p = 1 \text{ and } q = 0.$$
$$M_{00}: \text{Number of attributes where } p = 0 \text{ and } q = 0.$$
$$M_{11}: \text{Number of attributes where } p = 1 \text{ and } q = 1.$$

- **Similarity Measures:**

1. **Simple Matching Coefficient (SMC)**:

Formula:

$$\text{SMC} = \frac{\text{Number of Matches}}{\text{Number of Attributes}} = \frac{M_{11} + M_{00}}{M_{01} + M_{10} + M_{11} + M_{00}}$$

2. **Jaccard Coefficient**:

Formula:

$$\text{J} = \frac{\text{Number of 11 Matches}}{\text{Number of Not-Both-Zero Attributes}} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

Example:

$p = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0$

$q = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1$

$M_{01} = 2$ (the number of attributes where p was 0 and q was 1)
$M_{10} = 1$ (the number of attributes where p was 1 and q was 0)
$M_{00} = 7$ (the number of attributes where p was 0 and q was 0)
$M_{11} = 0$ (the number of attributes where p was 1 and q was 1)

$SMC = (M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00}) = (0+7) / (2+1+0+7) = 0.7$

$J = (M_{11}) / (M_{01} + M_{10} + M_{11}) = 0 / (2 + 1 + 0) = 0$

$$\text{Rand statistic} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}} \quad (5.18)$$

$$\text{Jaccard coefficient} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} \quad (5.19)$$

Example 5.17 (Rand and Jaccard Measures)**.** Based on these formulas, we can readily compute the Rand statistic and Jaccard coefficient for the example based on Tables 5.10 and 5.11. Noting that $f_{00} = 4$, $f_{01} = 2$, $f_{10} = 2$, and $f_{11} = 2$, the Rand statistic $= (2 + 4)/10 = 0.6$ and the Jaccard coefficient $= 2/(2+2+2) = 0.33$. ■

We also note that the four quantities, f_{00}, f_{01}, f_{10}, and f_{11}, define a *contingency* table as shown in Table 5.12.

Table 5.12. Two-way contingency table for determining whether pairs of objects are in the same class and same cluster.

	Same Cluster	Different Cluster
Same Class	f_{11}	f_{10}
Different Class	f_{01}	f_{00}

III Variables of mixed type:

1. **Cosine Similarity**

- Definition:
    - Measures the cosine of the angle between two document vectors d1d_1d1 and d2d_2d2.

- Often used in text analysis and information retrieval to compare documents.

- Formula:

$$\cos(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\|\|d_2\|}$$

Where:

- $d_1 \cdot d_2$: Vector dot product of d_1 and d_2.
- $\|d_1\|$: Magnitude (length) of vector d_1.
- $\|d_2\|$: Magnitude (length) of vector d_2.

Example:

$d_1 = 3\ 2\ 0\ 5\ 0\ 0\ 0\ 2\ 0\ 0$
$d_2 = 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 2$

$d_1 \bullet d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5$
$||d_1|| = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$
$||d_2|| = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5} = (6)^{0.5} = 2.245$

$\cos(d_1, d_2) = .3150$
Cos(x,y) = 0 indicates both are dissimilar
Cos(x,y) = 1 indicates both are similar

2. **Extended Jaccard Coefficient (Tanimoto):**

- **Definition:**
  - A variation of the Jaccard Coefficient, designed for document data.
  - Reduces to the standard Jaccard Coefficient for binary attributes.
- **Formula:**

$$T(p, q) = \frac{p \cdot q}{\|p\|^2 + \|q\|^2 - p \cdot q}$$

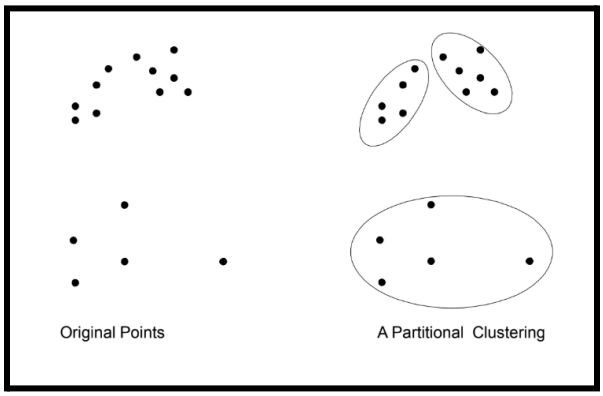
Where:

- $p \cdot q$: Dot product of vectors p and q.
- $\|p\|^2$: Squared magnitude of vector p.
- $\|q\|^2$: Squared magnitude of vector q.

DIFFERENT TYPES OF CLUSTERINGS:

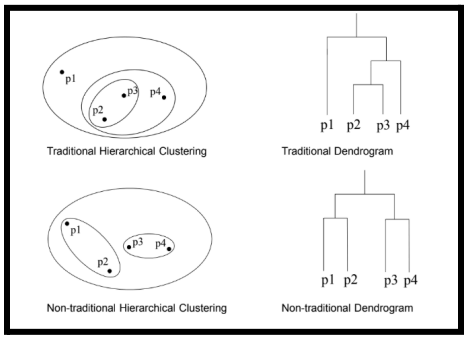
1. Partitioning Approach

- Method:
    - Construct various partitions of the data and evaluate them using a criterion (e.g., minimizing the sum of squared errors).
    - Subdivides data into subsets of k groups.
    - These k groups or clusters are to be pre defined.
- Typical Methods:
    - k-means, k-medoids, CLARANS.



2. Hierarchical Approach

- Method:
    - Create a hierarchical decomposition of the dataset based on a specific criterion.
    - Does not require pre definition of clusters.
- Typical Methods:
    - Diana, Agnes, BIRCH, CAMELEON.

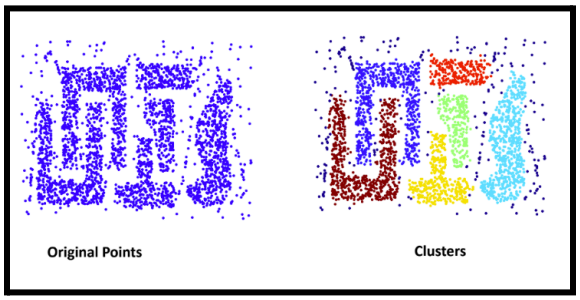


Aspect	Partitioning Clustering	Hierarchical Clustering
Definition	Divides data into **K fixed clusters.**	Creates a **tree-like structure** (dendrogram).
Input Requirement	Requires the number of clusters (K).	No need to predefine clusters.
Output	Produces non-overlapping clusters.	Produces nested clusters.
Cluster Shape	Assumes compact, spherical clusters.	Handles arbitrary shapes.
Computation	Faster, $O(n \cdot K \cdot I)$.	Slower, $O(n^2 \log n)$.
Scalability	Suitable for large datasets.	Better for small datasets.
Flexibility	Fixed clusters.	Flexible, explore clusters at any level.
Examples	K-means, K-medoids, CLARANS.	Agglomerative (single/complete-link), divisive.

Key Difference: Partitioning is efficient and good for large datasets with predefined clusters, while hierarchical is more flexible and better for exploring nested relationships.

3. Density-Based Approach

- Method:
  - Based on connectivity and density functions.
  - Uses 2 concepts, data reachability and dada connectivity.
- Typical Methods:
  - DBSCAN, OPTICS, DenClue.



4. Grid-Based Approach

- Method:
  - Uses a multi-level granularity structure to cluster data.
- Typical Methods:
  - STING, WaveCluster, CLIQUE.

Additional Clustering Approaches

5. Model-Based Approach

- Method:
  - Hypothesize a model for each cluster and find the best fit for the data.
- Typical Methods:
  - EM (Expectation-Maximization), SOM (Self-Organizing Map), COBWEB.

6. Frequent Pattern-Based Approach

- Method:
  - Cluster data by analyzing frequent patterns.
- Typical Methods:
  - p-Cluster.

7. User-Guided or Constraint-Based Approach

- Method:
  - Clustering based on user-specified constraints or application-specific requirements.
- Typical Methods:
  - COD (obstacle-based), Constrained Clustering.

8. Link-Based Clustering

- Method:
  - Use massive links between objects to group them.
- Typical Methods:
  - SimRank, LinkClus.

Hierarchical vs. Partitional

- **Hierarchical Clustering:**
  - Clusters are nested and organized as a tree structure.
  - Each cluster (except leaf nodes) is the union of its subclusters.
  - The root node contains all data objects, and leaf nodes may represent singleton clusters.
  - Example: Figure 5.1 (a–d) shows a hierarchical clustering with 1, 2, 4, and 6 clusters at different levels.
  - Can be viewed as a sequence of partitional clusterings by cutting the tree at specific levels.
- **Partitional Clustering:**
  - Divides data into non-overlapping subsets where each data object belongs to exactly one cluster.
  - Example: Figures 5.1 (b–d) show partitional clustering.

Exclusive vs. Overlapping vs. Fuzzy

- **Exclusive Clustering:**
  - Each object is assigned to one cluster only.
  - Example: Figures 5.1 (b–d).
- **Overlapping (Non-Exclusive) Clustering:**
  - Objects can belong to more than one cluster.
  - Suitable for cases where:

        - An object logically belongs to multiple groups (e.g., a student employee at a university).
        - An object lies "between" clusters.
- **Fuzzy Clustering:**
    - Objects belong to all clusters with a membership weight between 0 (does not belong) and 1 (fully belongs).
    - Constraints:
        - The sum of membership weights for an object equals 1.
    - Probabilistic Clustering:
        - Similar to fuzzy clustering but uses probabilities for cluster membership.
    - Often converted to exclusive clustering by assigning each object to the cluster with the highest membership weight or probability.

Complete vs. Partial

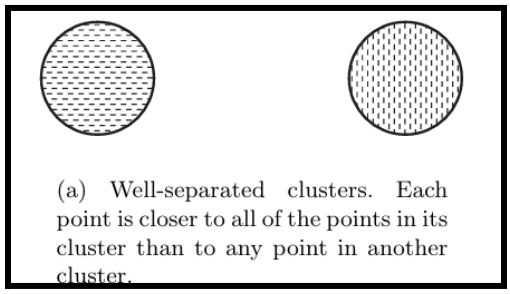
- **Complete Clustering:**
    - Assigns every object to a cluster.
    - Example: Clustering to organize documents for browsing requires all documents to be included.
- **Partial Clustering:**
    - Does not assign every object to a cluster.
    - Motivation:
        - Some objects (e.g., noise, outliers, or background data) may not belong to well-defined groups.
        - Example: Focusing on tightly related clusters in newspaper articles while excluding generic or unrelated stories.

DIFFERENT TYPES OF CLUSTERS

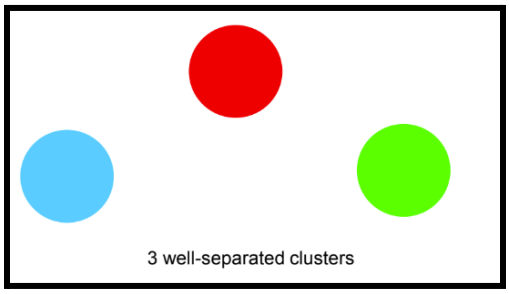
1. Well-Separated Clusters:

- **Definition**:
  A cluster is a set of objects where each object is closer (or more similar) to every other object in the cluster than to any object outside the cluster.
- **Key Characteristics**:
    - Often involves a threshold for closeness or similarity.
    - Clusters can be non-globular and take any shape.

- **Example**:

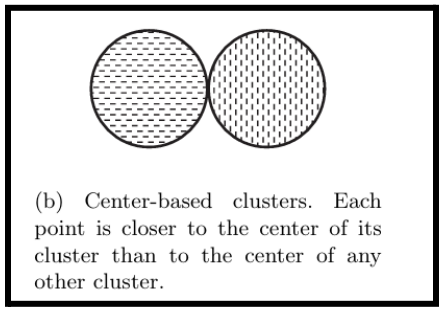


(a) Well-separated clusters. Each point is closer to all of the points in its cluster than to any point in another cluster.

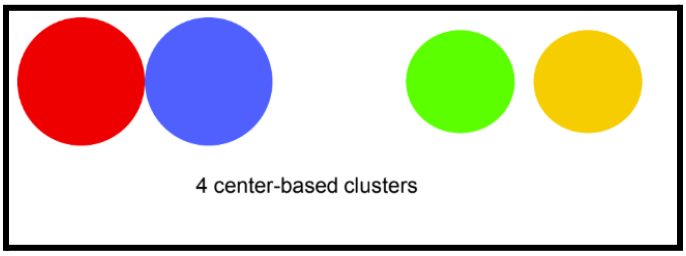


2. Prototype-Based / Center-Based Clusters:

- **Definition**:
  A cluster is a set of objects where each object is closer to the prototype of the cluster than to the prototype of any other cluster.
- **Prototypes**:
  - Centroid: Mean of all points in the cluster (used for continuous data).
  - Medoid: Most representative point in the cluster (used for categorical data).
- **Key Characteristics:**
  - Clusters are typically globular.
- **Example**:

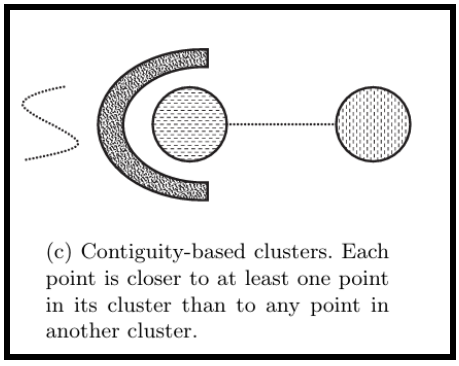


(b) Center-based clusters. Each point is closer to the center of its cluster than to the center of any other cluster.

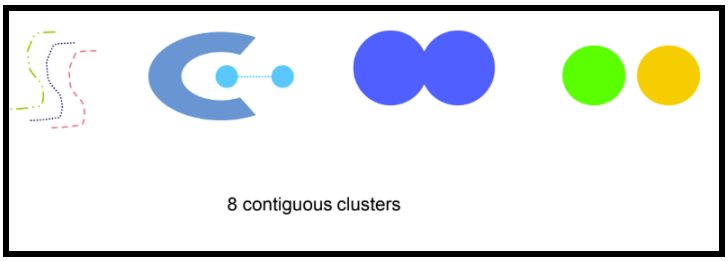


3. Graph-Based Clusters/ Contiguity Based Clusters

- **Definition**:
  A cluster is a connected component in a graph representation of data.
- **Types**:
  - Contiguity-Based Clusters: Objects are connected if they are within a specified distance.
    - Issue: Can merge distinct clusters due to noise (e.g., Figure 5.2(c)).
  - Clique-Based Clusters: Formed when a group of objects is completely connected.
- **Key Characteristics:**
  - Suitable for irregular or intertwined clusters.
  - Sensitive to noise.

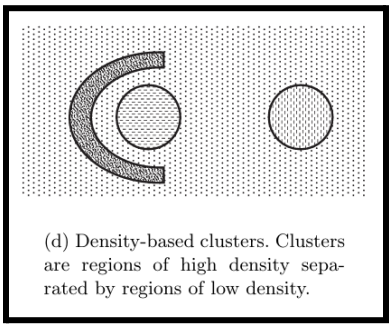


(c) Contiguity-based clusters. Each point is closer to at least one point in its cluster than to any point in another cluster.

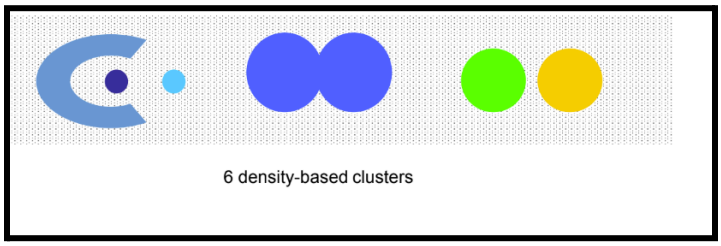


4. Density-Based Clusters

- **Definition**:
  A cluster is a dense region of objects surrounded by low-density regions.
- **Key Characteristics:**
  - Effective for irregular clusters and noisy data.
  - Avoids merging clusters via noise bridges (e.g., Figure 5.2(d)).
- **Example**:

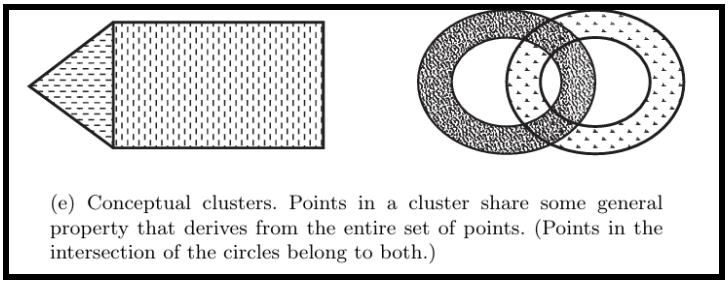


(d) Density-based clusters. Clusters are regions of high density separated by regions of low density.

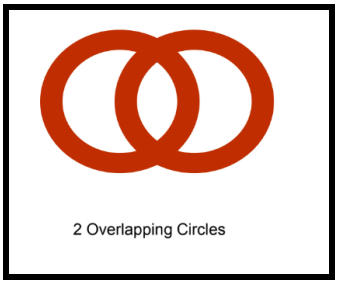


5. Shared-Property / Conceptual Clusters

- **Definition**:
  A cluster is a set of objects that share some common property.
- **Key Characteristics:**
  - Includes all previous notions of clusters (e.g., center-based, density-based).
  - Can detect complex cluster shapes (e.g., Figure 5.2(e): triangular, rectangular, intertwined circles).
  - Complex notions may overlap with pattern recognition.



(e) Conceptual clusters. Points in a cluster share some general property that derives from the entire set of points. (Points in the intersection of the circles belong to both.)



2 Overlapping Circles

Key Characteristics:

1. **Shared Properties:**
   - Objects in the cluster share a common characteristic or relationship.
   - Example: Objects closest to a centroid or medoid in a center-based cluster also share the property of proximity.
2. **Complex Structures:**
   - Conceptual clusters can include **non-standard shapes** or **complex relationships**.
   - Example (from Figure 5.2(e)):
     - A triangular area forms one cluster.
     - A rectangular area forms another cluster.
     - Two intertwined circles form additional clusters.
   - These require a clustering algorithm that can handle **specific concepts** or predefined structures.
3. **Flexibility:**
   - This approach can include other clustering types, such as center-based or density-based clusters, since these clusters also share certain properties (e.g., proximity to a centroid or high density).
4. **Conceptual Clustering Process:**
   - The process of finding conceptual clusters is called **conceptual clustering**.
   - The success of this process depends on the **definition of the concept** and the ability of the algorithm to detect it.

Role of Clustering

1. **Pattern Discovery**:
   - Identifies hidden patterns or structures in unlabelled data.
   - Example: Grouping customers with similar purchasing behaviors.
2. **Data Simplification**:
   - Reduces data complexity by summarizing large datasets into meaningful clusters.
   - Example: Representing millions of users with a few demographic groups.
3. **Outlier Detection**:
   - Helps identify anomalies or outliers that do not fit into any cluster.
   - Example: Fraud detection in financial transactions.
4. **Feature Understanding**:
   - Provides insights into the inherent properties of the data.

    - Example: Finding subcategories within a product catalog.
5. **Preprocessing for Supervised Learning**:
    - Creates groups that can act as features or labels for downstream machine learning tasks.
    - Example: Clustering unlabeled images before classification.
6. **Personalization**:
    - Enables customized recommendations or solutions based on cluster membership.
    - Example: Recommending movies based on a user's cluster.
7. **Dimensionality Reduction**:
    - Simplifies high-dimensional data by grouping similar data points.
    - Example: Visualizing data by clustering it into groups.
8. **Decision Making**:
    - Assists businesses in making informed decisions by identifying key groups in their data.
    - Example: Targeting marketing campaigns at specific customer clusters.

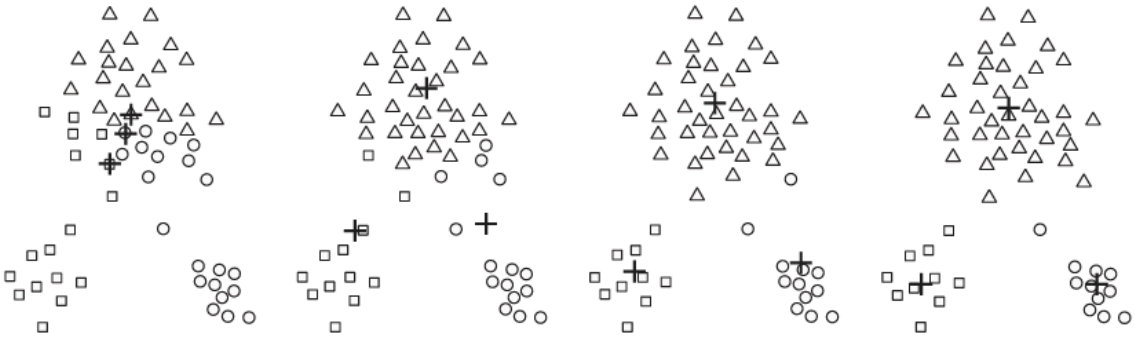
K-MEANS

Prototype-Based Clustering Technique- K Means Clustering:

- **Definition**:
  Prototype-based clustering techniques create a **one-level partitioning** of the data objects. • The clusters are formed to optimize an objective partitioning criterion • Objects within a cluster are similar • Objects of different clusters are dissimilar.
- **Prominent Techniques**:
    - **K-Means**:
        - Defines a **prototype** in terms of a **centroid**.
        - A **centroid** is the **mean** of a group of points.
        - Typically applied to objects in a **continuous n-dimensional space**.
    - **K-Medoids**:
        - Defines a **prototype** in terms of a **medoid**.
        - A **medoid** is the most **representative point** for a group of points.
        - Can be applied to a **wide range of data** as it requires only a **proximity measure** for a pair of objects.
- **Key Differences Between Centroid and Medoid**:
    - **Centroid**:
        - Almost never corresponds to an actual data point.
    - **Medoid**:
        - Must be an actual data point by definition.

THE BASIC K MEANS ALGORITHM:

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Figure 5.3. Using the K-means algorithm to find three clusters in sample data.

Basic Algorithm:

1. **Initialization**:
   - Choose **K initial centroids**, where K is the user-specified number of clusters.
2. **Assignment Step**:
   - Assign each data point to the **closest centroid**.
   - Each group of points assigned to a centroid forms a **cluster**.
3. **Update Step**:
   - Update the centroid of each cluster based on the points assigned to it (e.g., calculate the **mean** for each cluster).
4. **Repeat**:
   - Repeat the **assignment** and **update** steps until:
     - No point changes clusters, or
     - Centroids remain the same.

Detailed Steps:

- **Step 1 (Initialization)**:
    - Start with K centroids placed randomly or using another initialization method.
- **Step 2 (Assignment)**:
    - Assign points to the nearest centroid based on the proximity measure.
- **Step 3 (Update)**:
    - Recalculate the centroid based on the new cluster members.
- **Iteration**:
    - Repeat assignment and update until no further changes occur or a termination condition is met.

Key Characteristics of K-Means Clustering

- **Proximity Function:** used to measure the closeness between data points and centroids. It determines:

1. Cluster Assignment: Each data point is assigned to the cluster whose centroid is closest based on the chosen proximity function.
2. Centroid Update: The centroids are recalculated to minimize the overall dissimilarity (distance) between the points in a cluster and their centroid.

- **Initial Centroids**:
    - Chosen **randomly**, which can lead to different cluster outcomes for each run.
- **Cluster Properties**:
    - **Centroid**: Typically the **mean** of the points in the cluster.
    - **Closeness**: Measured by various similarity/distance metrics, such as:
        - **Euclidean distance**.
        - **Cosine similarity**.
        - **Correlation**
- **Convergence**:
    - K-means will converge for the common similarity measures mentioned above.
    - Most of the convergence occurs in the **first few iterations**.
    - Stopping condition is often modified to:
        - **"Until relatively few points change clusters"**, or
        - **"When a clustering measure doesn't change significantly."**

Complexity of K-Means

- **Time Complexity**:

$$O(n \cdot K \cdot I \cdot d)$$

Where:

- n: Number of points.
- K: Number of clusters.
- I: Number of iterations.
- d: Number of attributes.

Evaluating K-Means clusters:

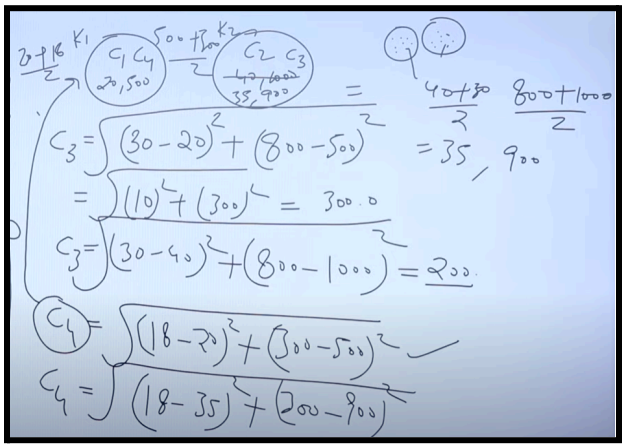
Formula:

Euclidean Distance Formula

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

Example 1:

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



Example 2

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).

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

Data Points			Distance to 2	10	5	8	1	2	Cluster	New Cluster
A1	2	10								
A2	2	5								
A3	8	4								
B1	5	8								
B2	7	5								
B3	6	4								
C1	1	2								
C2	4	9								

Assume that the Initial centroids are A1,B1,C1 (center of each cluster).

We have to find the distance from the data points to the centroids. Using the above formula.

Data Points			Distance to 2	10	5	8	1	2	Cluster	New Cluster
A1	2	10								
A2	2	5								
A3	8	4								
B1	5	8								
B2	7	5								
B3	6	4								
C1	1	2								
C2	4	9								

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

Calculate the dist for A1 and Centroid using the formula.

Similarly for A2 and A3 etc.

Calculate for the other 2 centroids as well.

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

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

Now assign the Data point to one of the three clusters. How? Assign it to the smallest centroid value.

Note the cluster number it belongs to 1,2 or 3.

Data Points			Distance to						Cluster	New Cluster
			2	10	5	8	1	2		
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)

Now calculate the new centroid value. Take the average of the same cluster numbers. For example, the 1st data point only belongs to one cluster so no need to take avg. The 2nd cluster has five data points i.e A3, B1, B2, B3, C2, take avg of those. Similarly for the 3rd cluster. The ew centroid vals would be found.

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

Data Points			Distance to						Cluster	New Cluster
			2	10	6	6	1.5	1.5		
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

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

Take the new centroid vals as current centroids and calculate distances again. Assign the cluster vals similarly by taking the smallest val.

But C2 has moved from 2nd cluster to 1st. So, cal new centroid vals again. Repeat.

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

Data Points			Distance to						Cluster	New Cluster
			3	9.5	6.5	5.25	1.5	3.5		
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

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

Again, the data point has moved from one cluster to another.

Find new centroid vals by taking avg and repeat.

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

Data Points			Distance to 3.67, 9	Distance to 7, 4.33	Distance to 1.5, 3.5	Cluster	New Cluster
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

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

Finally we get the cluster the data points belong to.

Example 3 (1D):

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

The distance function used is Euclidean distance.

Initial cluster centroid are M1 = 4 and M2 = 11.

Initial Centroids:
M1: 4
M2: 11

Therefore
C1= {2, 4, 3}
C2= {10, 12, 20, 30, 11, 25}

New Centroids:
M1: 3
M2: 18

Data Points	Distance to M1	Distance to M2	Cluster	New Cluster
2	2	9	C1	
4	0	7	C1	
10	6	1	C2	
12	8	1	C2	
3	1	8	C1	
20	16	9	C2	
30	26	19	C2	
11	7	0	C2	
25	21	14	C2	

$$d(x_2, x_1) = \sqrt{(x_2 - x_1)^2}$$

Current Centroids:
M1: 3
M2: 18

Therefore
C1= {2, 4, 10, 3}
C2= {12, 20, 30, 11, 25}

New Centroids:
M1: 4.75
M2: 19.6

Data Points	Distance to M1	Distance to M2	Cluster	New Cluster
2	1	16	C1	C1
4	1	14	C1	C1
10	7	8	C2	C1
12	9	6	C2	C2
3	0	15	C1	C1
20	17	2	C2	C2
30	27	12	C2	C2
11	8	7	C2	C2
25	22	7	C2	C2

$$d(x_2, x_1) = \sqrt{(x_2 - x_1)^2}$$

Current Centroids:
M1: 4.75
M2: 19.6

Therefore
C1= {2, 4, 10, 11, 12, 3}
C2= {20, 30, 25}

New Centroids:
M1: 7
M2: 25

Data Points	Distance to M1	Distance to M2	Cluster	New Cluster
2	2.75	17.6	C1	C1
4	0.75	15.6	C1	C1
10	5.25	9.6	C1	C1
12	7.25	7.6	C2	C1
3	1.75	16.6	C1	C1
20	15.25	0.4	C2	C2
30	25.25	10.4	C2	C2
11	6.25	8.6	C2	C1
25	20.25	5.4	C2	C2

$$d(x_2, x_1) = \sqrt{(x_2 - x_1)^2}$$

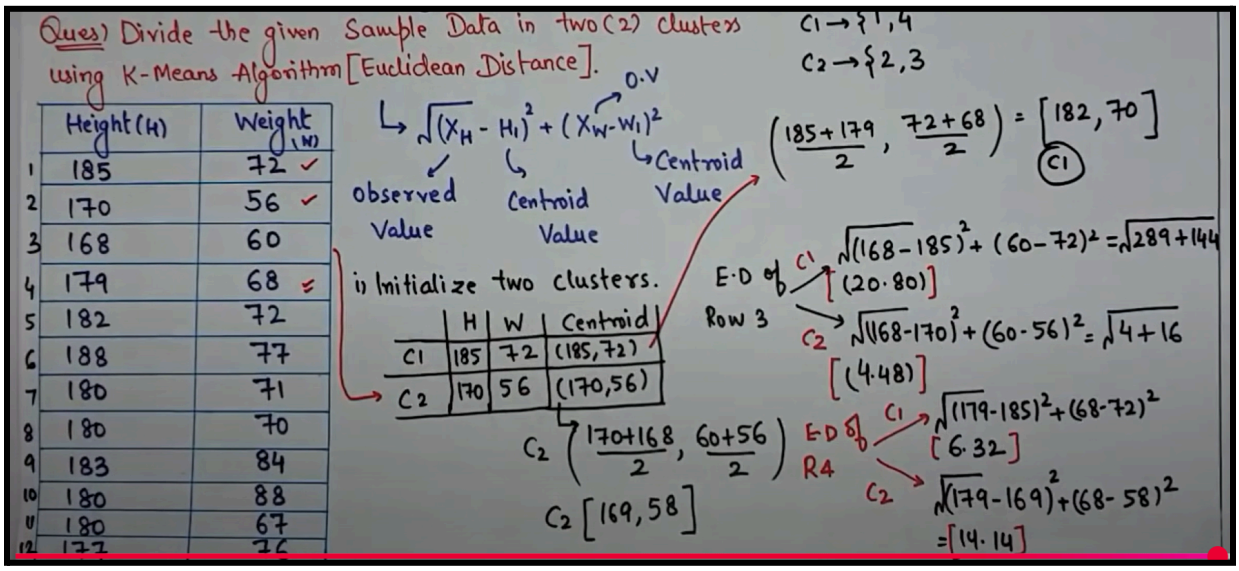
Current Centroids:
M1: 7
M2: 25

Final Cluster are:
C1= {2, 4, 10, 11, 12, 3}
C2= {20, 30, 25}

Data Points	Distance to M1	Distance to M2	Cluster	New Cluster
2	5	23	C1	C1
4	3	21	C1	C1
10	3	15	C1	C1
12	5	13	C1	C1
3	4	22	C1	C1
20	13	5	C2	C2
30	23	5	C2	C2
11	4	14	C1	C1
25	18	0	C2	C2

$$d(x_2, x_1) = \sqrt{(x_2 - x_1)^2}$$

Example 4



Sum of Squared Error (SSE) in K-Means:

1. **Definition**:
   - The **Sum of Squared Error (SSE)** is a common measure used to evaluate the quality of clustering.
   - It quantifies the error by measuring the **distance** of each point from its **nearest cluster centroid**.
2. **How SSE is Calculated**:
   - To compute **SSE**:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - m_i\|^2$$

Where:

- K: Number of clusters.
- x: Data point.
- m_i: Centroid (mean) of cluster C_i.

3. **Key Points**:
   - The **centroid** mi is the point that minimizes the squared error for a cluster (mean of all points in the cluster).
   - Clusters with **smaller SSE** indicate better compactness and better-defined clusters.
4. **Improving SSE**:
   - Increasing K (number of clusters) generally **reduces SSE**, as points are grouped into smaller, tighter clusters.
   - However:

        - **A good clustering with fewer clusters** can have a lower SSE than a poor clustering with more clusters.
        - Over-increasing K may lead to **overfitting**, as each data point could potentially become its own cluster.
5. **Cluster Selection**:
    - When comparing two clusters, choose the one with the **smaller SSE**.
    - The **elbow method** is often used to find the optimal K, balancing smaller SSE with a reasonable number of clusters.

Global Minimum vs. Local Minimum

In the context of K-Means clustering (or any optimization problem), the terms **global minimum** and **local minimum** refer to the outcomes of the optimization process.

Global Minimum

- **Definition**:
  The **global minimum** is the best possible solution to the optimization problem. For K-Means, it represents the clustering solution with the **lowest possible Sum of Squared Errors (SSE)** across all potential cluster configurations.
- **Characteristics**:
    - It is the **absolute lowest SSE** that can be achieved by any set of centroids.
    - Represents the **optimal clustering**, where points are grouped into their natural clusters.

Local Minimum

- **Definition**:
  A **local minimum** is a solution where the SSE is lower than for nearby configurations but is **not the absolute lowest SSE**. The algorithm "gets stuck" in a suboptimal state.
- **Characteristics**:
    - The solution is better than some alternatives but does not represent the **best overall solution**.
    - Can occur due to poor initialization of centroids.

Choosing Initial Centroids in K-Means

The Importance of Initial Centroids:

- The quality of clusters produced by K-Means heavily depends on the choice of **initial centroids**.
- **Random Initialization** is commonly used, but it often leads to suboptimal results:
    - Different runs of K-Means may produce different **total SSEs** (Sum of Squared Errors).
    - Some runs achieve the **global minimum SSE** (optimal clustering), while others get stuck in **local minima** (suboptimal clustering).

1. **Optimal vs. Suboptimal Clustering**:
    - **Figure 5.4(a)**:
        - Displays a clustering solution that achieves the **global minimum SSE** for three clusters.
        - Represents the **optimal clustering** where points are grouped into their natural clusters.
    - **Figure 5.4(b)**:
        - Illustrates a **suboptimal clustering** resulting in a **higher SSE**, which corresponds to a **local minimum**.
        - Indicates that the choice of initial centroids led to an imperfect clustering solution.

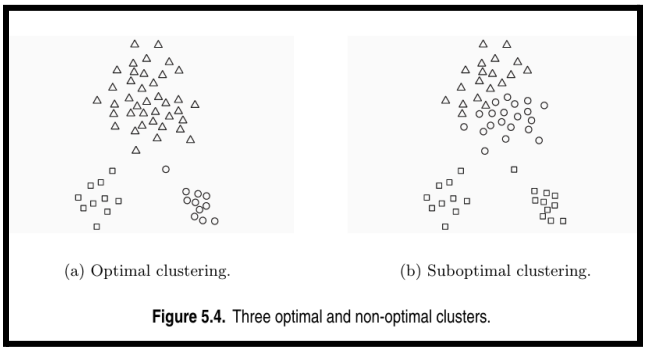
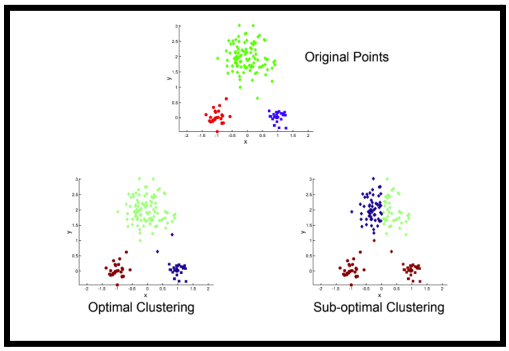


Figure 5.4. Three optimal and non-optimal clusters.



    - SSE will choose the optimal clustering as it has minimum clusters or k value.
2. **Poor Initial Centroids**:
    - All initial centroids are chosen from one natural cluster, yet the algorithm still finds the **minimum SSE clustering**. As we can see the SSE will be largest in the first iteration and reduced in the 2nd iteration so on… in the last iteration it will find the local/ global minima.

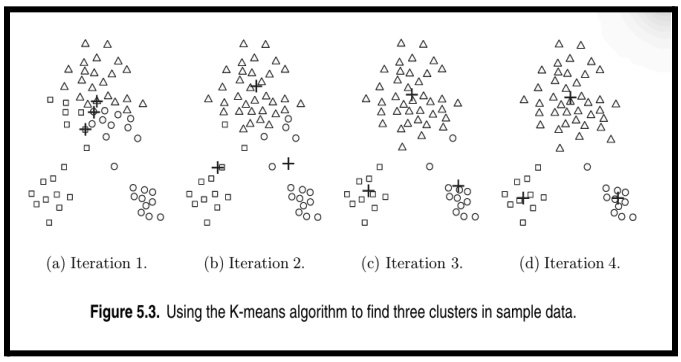


Figure 5.3. Using the K-means algorithm to find three clusters in sample data.

    - Initial centroids are better distributed, but the algorithm results in **suboptimal clustering** with **higher SSE**, demonstrating the sensitivity of K-Means to centroid initialization.

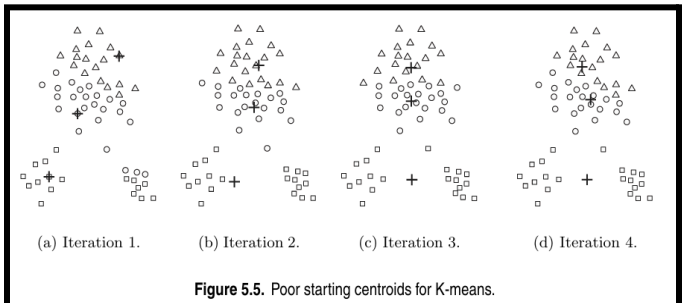


Figure 5.5. Poor starting centroids for K-means.

Limits of Random Initialization and Advanced Centroid Selection in K-Means

1. Limits of Random Initialization

- **Problem**:
  - When centroids are chosen randomly, K-Means often produces **suboptimal clustering**.
  - Multiple runs with random initializations may result in **different SSEs**, as random choices can miss the **global minimum**.
- Two pairs of clusters, where clusters in each pair (top-bottom) are **closer** to each other than to clusters in the other pair.

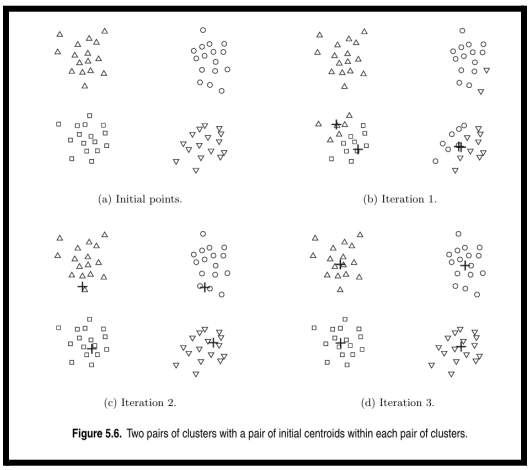


Figure 5.6. Two pairs of clusters with a pair of initial centroids within each pair of clusters.

Challenge: If initial centroids are both placed in a single cluster of a pair, the algorithm may fail to separate the two clusters in that pair.

Outcomes: If each pair starts with **two initial centroids**, the centroids redistribute themselves to correctly identify the clusters. If one pair starts with **one centroid** and the other pair starts with **three centroids**, two true clusters may combine, and one true cluster may split, resulting in a **suboptimal solution**.

- **As the Number of Clusters Increases**: The probability of a pair of clusters having only one initial centroid increases, leading to **local minima** instead of the global minimum.

2. Strategies to Improve Centroid Initialization

a. Multiple Runs:

- Perform multiple K-Means runs with different random initial centroids and choose the clustering with the **lowest SSE**.
- **Limitations**:
  - May still fail in datasets with complex structures.

b. Hierarchical Clustering for Initialization:

- **Process**:
  - Take a small sample of points.
  - Apply **hierarchical clustering** to this sample.
  - Extract K clusters and use their centroids as initial centroids for K-Means.
- **Advantages**:
  - Often produces better results, as hierarchical clustering captures **initial groupings**.
- **Limitations**:
  - Computationally expensive for large datasets.
  - Only practical if the sample size and K are relatively small.

c. Distance-Based Centroid Selection:

- **Process**:
  - Select the first centroid randomly.
  - For subsequent centroids, select the point that is **farthest** from any already chosen centroid.
  - Ensures centroids are well-separated initially.
- **Limitations**:
  - May select **outliers** as centroids, resulting in poor clustering.
  - Computationally expensive to find the farthest point.
- **Solution**:
  - Apply this approach to a **sample of points** to reduce computational cost and avoid outliers.

d. K-Means++ Initialization:

- **Process**:
  - Select centroids incrementally:
    1. Pick the first centroid randomly.
    2. For each subsequent centroid:
       - Compute the distance d(x)d(x)d(x) of each point to its closest centroid.
       - Assign each point a **probability proportional to d(x)2d(x)^2d(x)2**.
       - Choose the next centroid using these weighted probabilities.
  - Repeat until K centroids are chosen.
- **Advantages**:
  - Ensures centroids are well-separated and reduces the likelihood of poor clustering.
  - Produces clustering solutions with **lower SSE** compared to random initialization.
  - Efficient and widely used.

Algorithm 5.2 K-means++ initialization algorithm.

1: For the first centroid, pick one of the points at random.
2: **for** $i = 1$ to *number of trials* **do**
3: Compute the distance, $d(x)$, of each point to its closest centroid.
4: Assign each point a probability proportional to each point's $d(x)^2$.
5: Pick new centroid from the remaining points using the weighted probabilities.
6: **end for**

3. Advanced Techniques

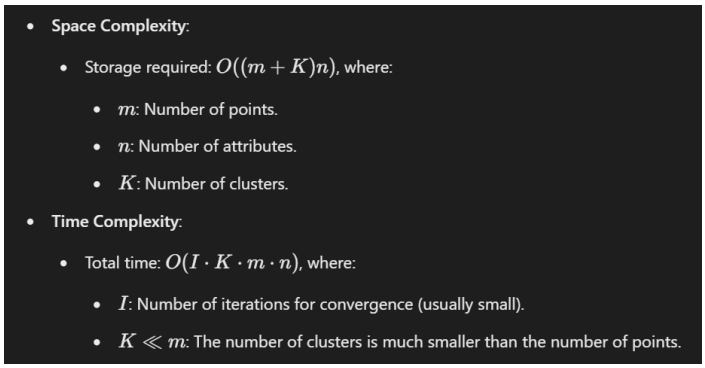
- **Bisecting K-Means**:
  - A variant of K-Means that splits clusters recursively, making it less sensitive to initialization.
- **Post-Processing**:
  - Adjusts the clusters produced by K-Means to improve quality (e.g., moving points between clusters to reduce SSE).
- **Combination**:

- K-Means++ can be combined with these approaches for better results.

Conclusion

- Random initialization in K-Means is simple but unreliable for complex datasets, often leading to local minima.
- Techniques like **hierarchical clustering**, **distance-based selection**, and **K-Means++** significantly improve the quality of initial centroids, resulting in better clustering outcomes with lower SSE.
- Among these, **K-Means++** is a practical and effective method widely used due to its balance of simplicity and efficiency.

Complexity of K-Means



ADDITIONAL PROBLEMS IN K-MEANS

1. Handling Empty Clusters

- **Problem**:
    1. Empty clusters occur if no points are assigned to a cluster during the assignment step. This results in undefined centroids and a higher SSE.
- **Solutions**:

Replacement Centroid Selection:

- Select the **point farthest away** from any current centroid.
- This reduces the contribution of the most problematic point to SSE.

2. **K-Means++ Approach**:
   - Select a new centroid using the K-Means++ strategy, ensuring better centroid placement.
3. **Random Replacement**:
   - Choose a replacement centroid **randomly** from the cluster with the highest SSE. This effectively splits that cluster and reduces overall SSE.
4. **Iterative Handling**:
   - Repeat this process if multiple clusters become empty.

2. Handling Outliers

- **Problem**:
  1. Outliers significantly influence centroids, reducing the representativeness of clusters and increasing SSE.
- **Solutions**:
  1. **Outlier Detection and Removal**:
     - Detects outliers **before clustering** using specialized techniques (e.g., statistical thresholds..
     - Remove points contributing excessively high SSE in a **post processing step**.
  2. **Eliminating Small Clusters**:
     - Small clusters often represent groups of outliers. Remove these clusters to improve overall clustering quality.
  3. **Applications to Consider**:
     - For **compression** applications, include all points.
     - For specific applications (e.g., financial analysis), consider retaining interesting outliers (e.g., highly profitable customers).

3. Reducing SSE with Post Processing

- **Problem**:
  1. K-Means often converges to a **local minimum**, resulting in higher SSE.
- **Strategies**:
  1. **Splitting Clusters**:
     - Split the cluster with the **largest SSE** or the largest standard deviation for a specific attribute.
     - Introduce a **new cluster centroid**, often the point farthest from any cluster center.
  2. **Merging Clusters**:
     - Merge clusters with the **closest centroids** or those that result in the **smallest increase in SSE**.
  3. **Alternating Splitting and Merging**:

        - Use iterative **splitting and merging phases** to escape local minima and improve clustering without increasing the number of clusters.
    4. **Fixed Number of Clusters**:
        - Focus on improving the quality of clustering (lower SSE) without increasing K.

4. Incremental Centroid Updates

- **Problem**:
    - Updating centroids after all points are assigned may lead to slower convergence or order dependency.
- **Solution**:
    - **Incremental Updates**:
        - Update centroids immediately after assigning a point to a cluster.
        - Guarantees no empty clusters since each cluster starts with at least one point.
    - **Dynamic Weight Adjustments**:
        - Adjust the weight of points being added to centroids over time, improving accuracy and convergence.
- **Advantages**:
    - Can accommodate **non-SSE objectives**, allowing flexibility in optimizing cluster quality.
    - Improves convergence speed in some cases.
- **Disadvantages**:
    - **Order Dependency**:
        - Cluster results may depend on the order in which points are processed.
        - This can be mitigated by **randomizing point order**.
    - **Higher Cost**:
        - Incremental updates require more computational effort.

5. Time and Space Complexity

- **Space Complexity**:
    - O((m+K)n)O((m + K)n)O((m+K)n):
        - mmm: Number of points.
        - KKK: Number of clusters.
        - nnn: Number of attributes.
- **Time Complexity**:
    - O(I·K·m·n)O(I \cdot K \cdot m \cdot n)O(I·K·m·n):
        - III: Number of iterations (usually small, as K-Means converges quickly).

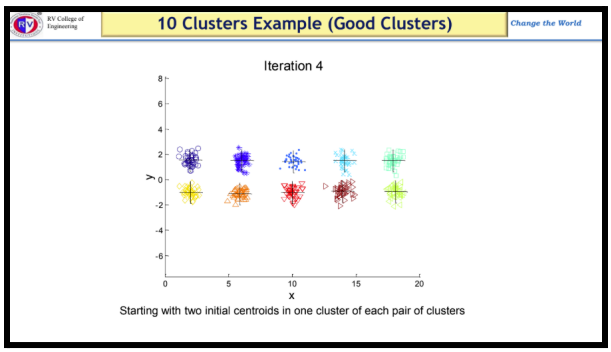
6. Alternative Initialization and Postprocessing

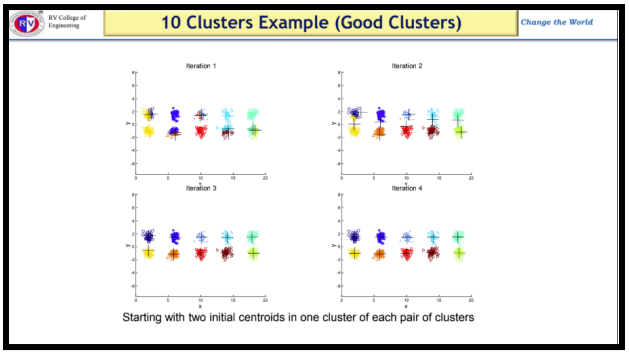
- **Advanced Initialization**:
    - Use methods like **K-Means++** for better initial centroid placement:
        - Randomly select the first centroid.
        - Choose subsequent centroids with probabilities proportional to the square of their distances to existing centroids.

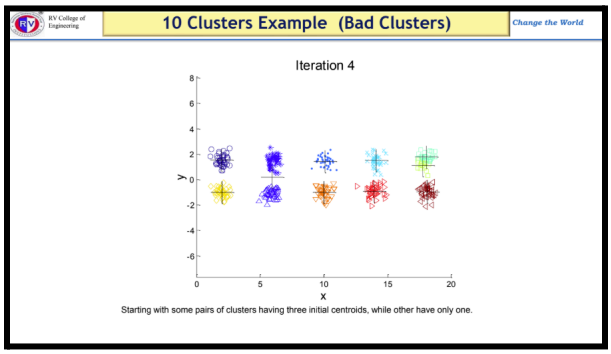
    - Reduces the likelihood of suboptimal clustering and improves SSE.
- **Post Processing**:
    - Apply **splitting, merging**, or **outlier handling** techniques to refine clusters after initial convergence.

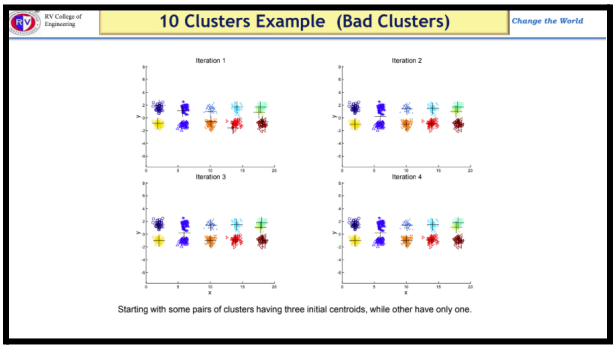
Conclusion

- K-Means is simple and efficient but faces challenges like **empty clusters**, **outliers**, and **local minima**. These issues can be mitigated through strategies like:
    - Improved centroid initialization (e.g., K-Means++).
    - Incremental centroid updates.
    - Postprocessing to refine clusters.
- By addressing these challenges, K-Means can produce more accurate and meaningful clustering results.









BISECTING K-MEANS ALGORITHM

The **Bisecting K-Means** algorithm is an extension of the basic K-Means algorithm, designed to address some of its limitations, especially with initialization. It systematically splits data into clusters in a stepwise manner, aiming for better clustering results.

Algorithm 5.3 Bisecting K-means algorithm.

1: Initialize the list of clusters to contain the cluster consisting of all points.
2: **repeat**
3: Remove a cluster from the list of clusters.
4: {Perform several "trial" bisections of the chosen cluster.}
5: **for** $i = 1$ to *number of trials* **do**
6: Bisect the selected cluster using basic K-means.
7: **end for**
8: Select the two clusters from the bisection with the lowest total SSE.
9: Add these two clusters to the list of clusters.
10: **until** The list of clusters contains K clusters.

Overview of the Algorithm

Idea:

To create K clusters:

1. Start with **all data points in one cluster**.
2. **Split one cluster into two** using K-Means (bisection).
3. Repeat this process, selecting a cluster to split, until K clusters are formed.

Steps:

1. **Initialize**:
    - Start with one cluster containing all data points.
2. **Iterative Splitting**:
    - Remove a cluster from the current list of clusters.
    - Perform several **trial bisections** of this cluster using the basic K-Means algorithm.
    - Select the two resulting clusters with the **lowest total SSE**.
    - Add these two clusters to the list of clusters.
3. **Repeat**:
    - Continue until the required number of clusters (K) is reached.

Cluster Selection Criteria for Splitting

At each step, the cluster to split can be chosen based on:

1. **Size**:
    - Select the largest cluster.
2. **SSE**:
    - Select the cluster with the largest Sum of Squared Errors (SSE).
3. **Combination**:
    - Consider both size and SSE to determine the cluster that would benefit most from splitting.

Advantages of Bisecting K-Means

1. **Less Susceptible to Initialization Problems**:
    - At each step, there are only **two centroids**, which simplifies the initialization process.
    - Multiple trial bisections are performed, and the one with the **lowest SSE** is chosen, reducing the chance of poor clustering due to random centroid initialization.
2. **Improved Clustering Quality**:
    - By focusing on splitting one cluster at a time, it produces better-defined clusters compared to directly initializing K centroids randomly in basic K-Means.
3. **Hierarchical Clustering**:
    - The sequence of cluster splits creates a **hierarchical structure** that can be used for further analysis.

Example:

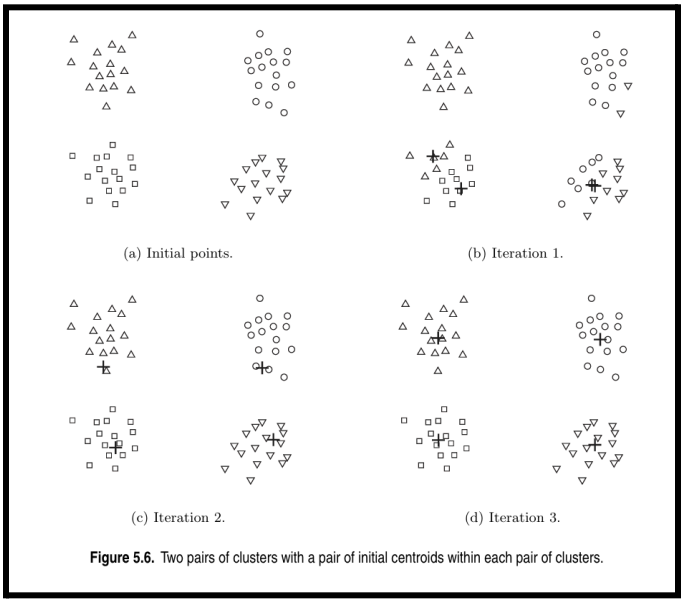


Figure 5.6. Two pairs of clusters with a pair of initial centroids within each pair of clusters.

Process:

- **Iteration 1**:
    - Splits the dataset into **two pairs of clusters**.
- **Iteration 2**:
    - Splits the **rightmost pair of clusters**.
- **Iteration 3**:
    - Splits the **leftmost pair of clusters**.

- **Result**:
    - Four clusters are identified, effectively overcoming initialization issues.

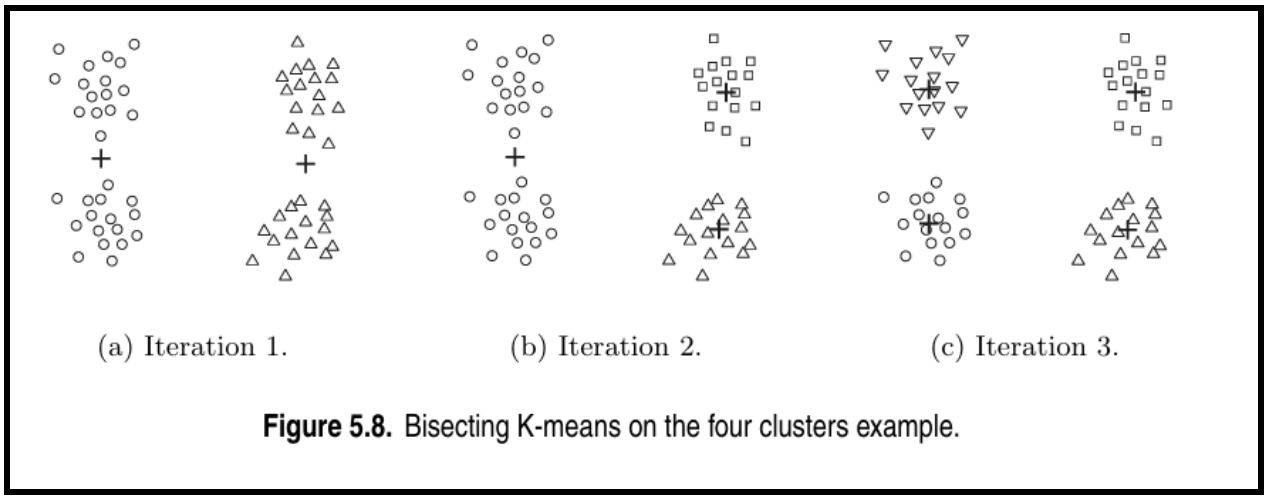


Figure 5.8. Bisecting K-means on the four clusters example.

Post Processing

- The clusters produced by Bisecting K-Means are **not necessarily optimal** with respect to the total SSE.
- To refine the clustering:
    - Use the cluster centroids from the bisecting process as **initial centroids** for a final run of the basic K-Means algorithm.

Hierarchical Clustering with Bisecting K-Means

- By recording the sequence of cluster splits:
    - Bisecting K-Means can create a **hierarchical clustering structure**.
    - This structure is useful for applications where clusters need to be organized in a nested way.

Conclusion

The Bisecting K-Means algorithm:

1. **Improves robustness** against random initialization problems.
2. **Produces well-defined clusters** through systematic splitting and trial bisections.
3. **Can be used for hierarchical clustering**, adding flexibility to the clustering process.

Example:

Bisecting K-Means Clustering – Solved Example

- Use **Bisecting K Means clustering** to cluster the following data into **three** groups.
- Data Points: {(2, 3), (3, 3), (6, 8), (8, 8), (7, 5), (9, 7)}

- **Step 3:**
- Use K-means with K=2 to split the cluster
- **Initial Centroids:**
- Centroid 1: (2, 3)
- Centroid 2: (8, 8)
- **Euclidian Distance**
- $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Data Points	Distance to (2,3)	Distance to (8, 8)	Cluster	New Cluster
(2, 3)	0	7.81	A	
(3, 3)	1	7.07	A	
(6, 8)	6.4	2	B	
(8, 8)	7.81	0	B	
(7, 5)	5.39	3.16	B	
(9, 7)	8.06	1.41	B	

- **Step 4:**
- Measure the distance for each intra cluster.
  - Sum of square Distance
  - $-\sum_{i=0}^{n}(x_i - \bar{x})^2 * (y_i - \bar{y})^2$
- Centroid 1: (2.5, 3)
- Centroid 2: (7.5, 7)

Data Points	Distance to (2.5,3)	Distance to (7.5, 7)	Cluster	New Cluster
(2, 3)	0.50	6.80	A	A
(3, 3)	0.50	6.02	A	A
(6, 8)	6.10	1.80	B	B
(8, 8)	7.43	1.12	B	B
(7, 5)	4.92	2.06	B	B
(9, 7)	7.63	1.50	B	B

- **Step 3:**
- Use K-means with K=2 to split the cluster
- **New Centroids:**
- Centroid 1: (2.5, 3)
- Centroid 2: (7.5, 7)
- **Euclidian Distance**
- $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Data Points	Distance to (2.5,3)	Distance to (7.5, 7)	Cluster	New Cluster
(2, 3)	0.50	6.80	A	A
(3, 3)	0.50	6.02	A	A
(6, 8)	6.10	1.80	B	B
(8, 8)	7.43	1.12	B	B
(7, 5)	4.92	2.06	B	B
(9, 7)	7.63	1.50	B	B

- **Step 4:**
  - Sum of square Distance
  - $-\sum_{i=0}^{n}(x_i - \bar{x})^2 * (y_i - \bar{y})^2$
- Centroid 1: (2.5, 3)
- Centroid 2: (7.5, 7)
- $A \rightarrow (2 - 2.5)^2 * (3 - 3)^2 + (3 - 2.5)^2 + (3 - 3)^2 = 0$
- $B \rightarrow 3.5$

Data Points	Distance to (2.5,3)	Distance to (7.5, 7)	Cluster	New Cluster
(2, 3)	0.50	6.80	A	A
(3, 3)	0.50	6.02	A	A
(6, 8)	6.10	1.80	B	B
(8, 8)	7.43	1.12	B	B
(7, 5)	4.92	2.06	B	B
(9, 7)	7.63	1.50	B	B

- **Step 5:**
- Select the cluster that have the largest distance and split to 2 cluster using K-means.
- $A \rightarrow 0$
- $B \rightarrow 3.5$

Data Points	Distance to (2.5,3)	Distance to (7.5, 7)	Cluster	New Cluster
(2, 3)	0.50	6.80	A	A
(3, 3)	0.50	6.02	A	A
(6, 8)	6.10	1.80	B	B
(8, 8)	7.43	1.12	B	B
(7, 5)	4.92	2.06	B	B
(9, 7)	7.63	1.50	B	B

- **Step 3:**
- Use K-means with K=2 to split the cluster
- **New Centroids:**
- Centroid 3: (6, 8)
- Centroid 4: (8, 6.66)
- **Euclidian Distance**
- $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Data Points	Distance to (6, 8)	Distance to (8,6.66)	Cluster	New Cluster
(6, 8)	1.00	2.41	C	C
(8, 8)	2.24	1.34	D	D
(7, 5)	2.24	1.94	D	D
(9, 7)	3.00	1.06	D	D

- **Step 6:**
- Repeat step 3–5 until the number of cluster = K.
- $A \rightarrow \{(2, 3), (3, 3)\}$
- $C \rightarrow \{(6, 8)\}$
- $D \rightarrow \{(8, 8), (7, 5), (9, 7)\}$

Data Points	Distance to (6, 8)	Distance to (8,6.66)	Cluster	New Cluster
(6, 8)	1.00	2.41	C	C
(8, 8)	2.24	1.34	D	D
(7, 5)	2.24	1.94	D	D
(9, 7)	3.00	1.06	D	D

K MEANS AND DIFFERENT TYPES OF CLUSTERS

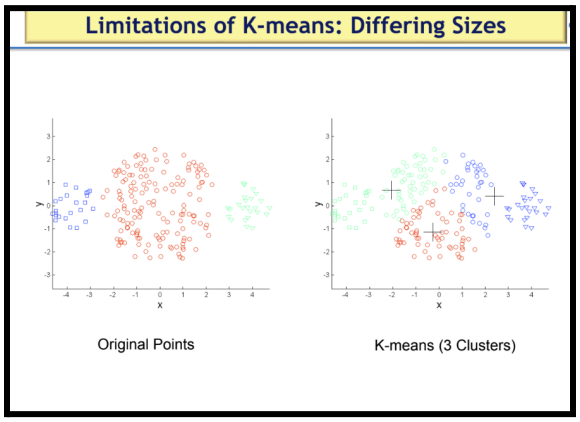
Limitations of K-Means with Non-Standard Cluster Shapes, Sizes, and Densities

K-Means' Assumptions

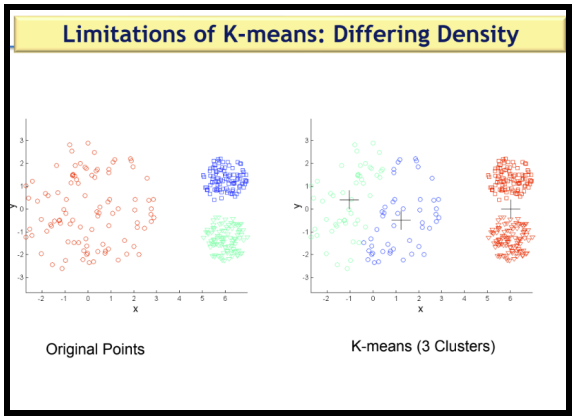
- K-Means works best when:
  - Clusters are **spherical** (globular).
  - Clusters have **similar sizes**.
  - Clusters have **similar densities**.
- These assumptions lead to challenges when clusters deviate from these characteristics.
- K-means has problems when clusters are of differing
- • Sizes
- • Densities
- • Non-globular shapes
- K-means has problems when the data contains outliers.

Challenges of K-Means

1. **Clusters of Different Sizes**:

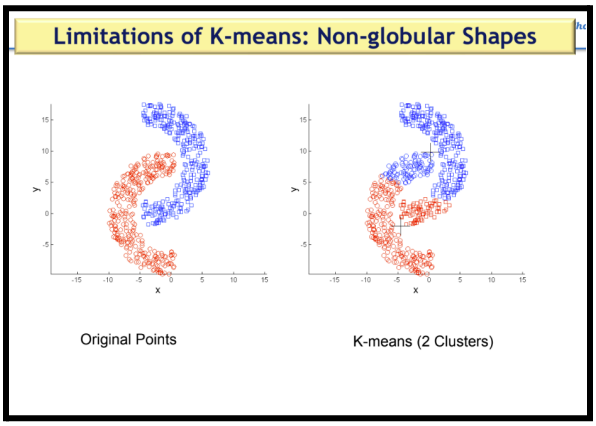


   - **Problem**:
     - When clusters differ in size, K-Means tends to split the larger cluster into multiple smaller clusters.
     - Smaller clusters may incorrectly merge with parts of the larger cluster.
   - **Example**:
     - In the figure, one large cluster is split, and a smaller cluster is combined with part of the larger cluster.
2. **Clusters of Different Densities**:



- **Problem**:
  - Dense clusters attract more centroids, causing sparse clusters to be inadequately represented.
  - This results in some dense clusters being over-split and sparse clusters being under-represented.
- **Example**:
  - In the figure, the two denser clusters are more accurately captured, while the sparse cluster is misrepresented.

3. **Non-Spherical Clusters**:



Problem:

- K-Means struggles with non-globular shapes because it minimizes the sum of squared Euclidean distances.
- Points from adjacent parts of two natural clusters may be assigned to the same cluster.

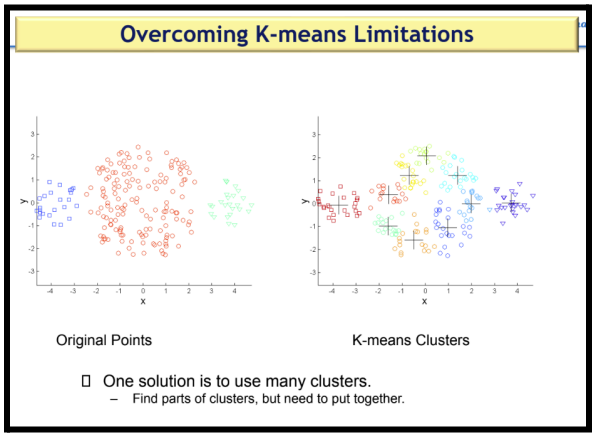
- **Example**:
  - In the figure, two clusters with irregular shapes are incorrectly mixed by K-Means.

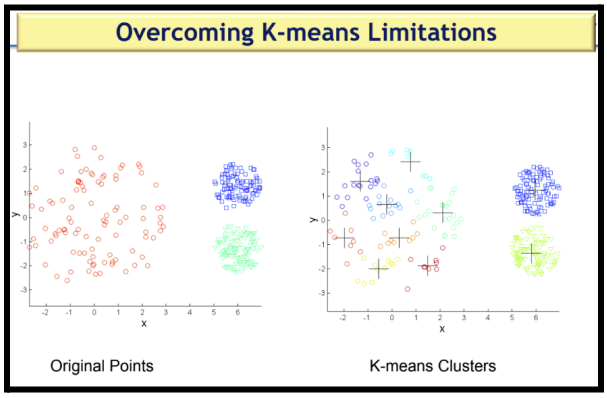
Reasons for These Challenges

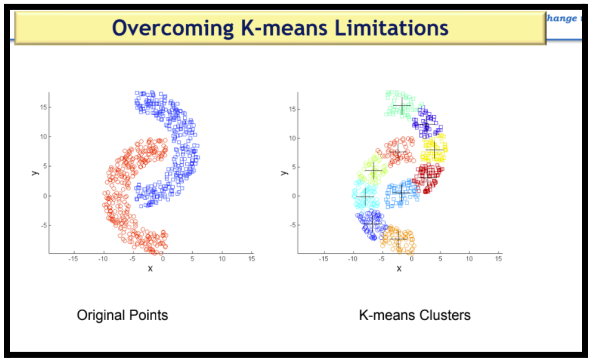
- K-Means minimizes the **Sum of Squared Errors (SSE)**, which assumes clusters are:
    - Compact and **globular**.
    - Of **similar size** and **density**.
- Clustering quality deteriorates when natural clusters violate these assumptions.

Overcoming K-Means Limitations

1. **Increasing the Number of Clusters (K)**:
    - Instead of trying to match natural clusters, specify a higher number of clusters.







Result:

- Each natural cluster is divided into smaller **pure subclusters**, improving the representation of each natural cluster.

    - **Example**:
        - using six clusters instead of two or three improves clustering.

2. **Using Variants of K-Means**:
    - **Bisecting K-Means**:
        - Splits clusters iteratively to handle initialization problems and improve quality.
    - **K-Medoids**:

        - Uses medoids (actual data points) instead of centroids, reducing sensitivity to outliers and non-spherical shapes.
    - **Density-Based Algorithms**:
        - Algorithms like **DBSCAN** handle non-globular clusters and varying densities better.
3. **Preprocessing or Post Processing**:
    - **Outlier Removal**:
        - Remove outliers before clustering to avoid distortion of centroids.
    - **Splitting and Merging**:
        - Use post processing techniques to refine clusters and reduce SSE.
4. **Alternative Objective Functions**:
    - Consider objective functions tailored for non-globular clusters or those that handle size and density variations better.

STRENGTHS AND WEAKNESSES OF K-MEANS

Strengths:

1. **Simplicity**:
    - Easy to implement and understand.
2. **Efficiency**:
    - Works well with large datasets due to linear time complexity.
3. **Flexibility**:
    - Can handle various data types with modifications.

Weaknesses:

1. **Assumptions**:
    - Requires clusters to be globular, similar in size and density.
2. **Sensitivity**:
    - Struggles with outliers and is heavily dependent on centroid initialization.
3. **Limitations**:
    - Poor performance on datasets with non-spherical clusters or widely varying sizes and densities.

K MEANS AS OPTIMIZATION PROBLEM

1. **Objective**:
    - Minimize the **Sum of Squared Errors (SSE)**:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} (x - c_i)^2$$

- C_i: Points in the i^{th} cluster.
- c_i: Centroid of the i^{th} cluster.

2. **Gradient Descent Approach**:
    - Start with initial centroids.
    - Iteratively update centroids to reduce SSE.
3. **Mathematical Derivation**:
    - The optimal centroid for minimizing SSE is the **mean** of the cluster points.
        - Solve for centroid ckk by setting the derivative of SSE to zero:

$$c_k = \frac{1}{m_k} \sum_{x \in C_k} x$$

Handling Different Objective Functions

1. **Sum of Absolute Errors (SAE)**:
    - Use **Manhattan Distance** instead of Euclidean Distance.

$$SAE = \sum_{i=1}^{K} \sum_{x \in C_i} |x - c_i|$$

    - Optimal centroid for minimizing SAE is the **median** of the cluster points.
2. **Other Functions**:
    - K-Means can be adapted to alternative proximity or similarity measures.

Aspect	K -Means Clustering	K-Medoids Clustering
Representation of Clusters	K-Means Clustering uses the mean of points (centroid) to represent a cluster.	It uses the most centrally located point (medoid) to represent a cluster.
Sensitivity to Outliers	Highly sensitive to outliers.	More robust to outliers.
Distance Metrics	K-Means primarily uses Euclidean distance.	Whereas it can use any distance metric.
Computational Efficiency	K-Means is generally faster and more efficient	It is slower due to the need to calculate all pairwise distances within clusters.
Cluster Shape Assumption	It assumes spherical clusters.	It does not make strong assumptions about cluster shapes.

CLUSTER EVALUATION

Why is Cluster Evaluation Necessary?

1. **Exploratory Nature of Clustering**:
    - Clustering is often used as part of exploratory data analysis (EDA), where the goal is to uncover hidden patterns or structures in the data.
    - Cluster evaluation might seem like an unnecessary complication for an informal process.
2. **Diverse Cluster Definitions**:
    - Different clustering algorithms produce different types of clusters, and evaluation depends on the **cluster type**:
        - **K-Means Clusters**: Evaluated using **Sum of Squared Errors (SSE)**.
        - **Density-Based Clusters**: May not be globular, so SSE is not suitable.
3. **Detecting Meaningful Clusters**:
    - Many clustering algorithms will **find clusters in any dataset**, even if no natural clusters exist.
    - Example: Randomly distributed points can be "forced" into clusters by algorithms, even though no meaningful clusters exist.

Illustrative Example: Clustering Random Points

- **Scenario**:
    - A dataset of 100 points is randomly distributed on a unit square (Figure 5.26(a)).
- **Results of Clustering**:
    - **DBSCAN** (Figure 5.26(b)):
        - Found three clusters, but they do not appear meaningful.

- **K-Means** (Figure 5.26(c)):
  - Forced the data into three clusters, even though there's no natural grouping.
- **Complete Linkage** (Figure 5.26(d)):
  - Produced clusters with no compelling structure.

- **Observation**:
  - Even with different algorithms, the clusters do not seem meaningful.
  - In higher dimensions, this issue becomes harder to detect visually.

Figure 5.26. Clustering of 100 uniformly distributed points.

Key Challenges in Cluster Evaluation

1. **No Ground Truth**:
   - Unlike supervised classification, clustering lacks labeled data, making evaluation more subjective.
2. **Algorithm-Specific Evaluations**:
   - The choice of evaluation metric depends on the algorithm and the type of clusters produced:
     - For **K-Means**: SSE is a natural metric.
     - For **DBSCAN**: Metrics like **density-based cohesion** or **separation** are more appropriate.
     - No universal metric works well for all clustering algorithms.

UNSUPERVISED MEASURES USING COHESION & SEPARATION

1. Cluster Evaluation Metrics: Cohesion and Separation

Cohesion:

- **Definition**: A measure of how closely the data points within a cluster are related to each other.
- **Graph-Based View**:
    - Cohesion is the **sum of the weights of the links** in the proximity graph that connect points **within a cluster**.

$$\text{Cohesion}(C_i) = \sum_{x \in C_i, y \in C_i} \text{proximity}(x, y)$$

        - Higher proximity (similarity) indicates higher cohesion.
- **Prototype-Based View**:
    - Cohesion is the **sum of proximities** of all points in a cluster to the cluster's prototype (centroid or medoid).

$$\text{Cohesion}(C_i) = \sum_{x \in C_i} \text{proximity}(x, c_i)$$

        - Where ci is the cluster centroid.
        - For Euclidean distance, this is equivalent to minimizing the cluster **SSE** (Sum of Squared Errors).

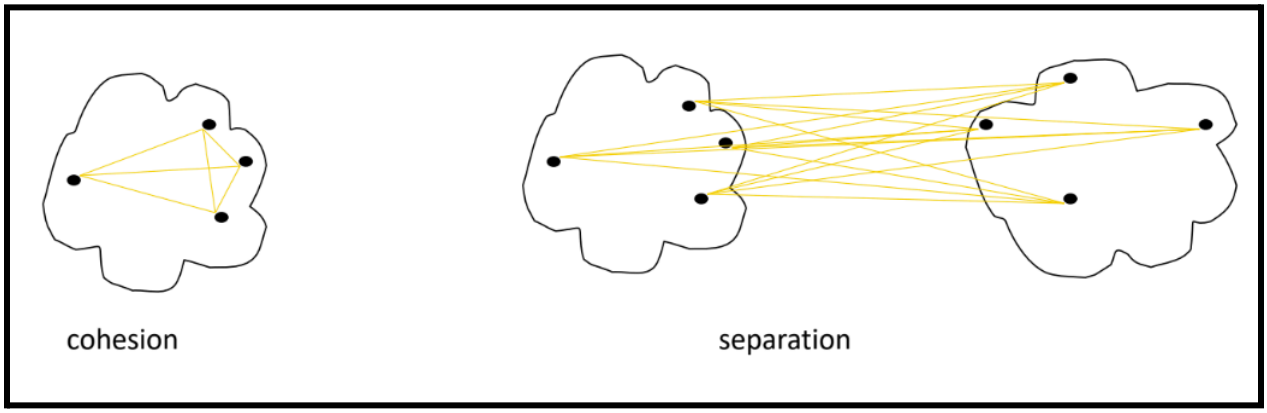
Separation:

- **Definition**: A measure of how distinct or well-separated different clusters are from each other.
- **Graph-Based View**:
    - Separation is the **sum of the weights of the links** connecting points in one cluster to points in another cluster.

$$\text{Separation}(C_i, C_j) = \sum_{x \in C_i, y \in C_j} \text{proximity}(x, y)$$

        - For similarity measures: Lower values are better.
        - For dissimilarity measures: Higher values are better.

- **Prototype-Based View**:
  - Separation is measured by the distance between cluster prototypes (centroids or medoids).
  - Formula: $$\text{Separation}(C_i, C_j) = \text{proximity}(c_i, c_j)$$
    - Alternatively, separation can be computed relative to an **overall centroid**: $$\text{Separation}(C_i) = \text{proximity}(c_i, c)$$
      - Where ccc is the overall centroid of all data points.



2. Combining Cohesion and Separation

- To evaluate overall cluster validity:

$$\text{Overall Validity} = \sum_{i=1}^{K} w_i \cdot \text{Validity}(C_i)$$

Relationship Between Cohesion and Separation:

- **Sum of Squares Relationship**:
  - The **Total Sum of Squares (TSS)** is the sum of:
    - **SSE (Cohesion)**: Measures within-cluster compactness.
    - **SSB (Separation)**: Measures between-cluster distinctness.
  - Formula:

$$TSS = SSE + SSB$$

  - Minimizing **SSE** (high cohesion) is equivalent to maximizing **SSB** (high separation).

3. Silhouette Coefficient

Definition:

- The **Silhouette Coefficient** combines both cohesion and separation into a single metric for evaluating clustering quality.

Steps to Compute Silhouette Coefficient:

1. **Cohesion (ai)**:
    - For point i, calculate the **average distance** to all other points in the same cluster.
    - Denote this as ai.
2. **Separation (bi)**:
    - For point i, calculate the **average distance** to all points in the nearest cluster not containing i.
    - Denote this as bi.
3. **Silhouette Coefficient (si)**:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

s_i ranges from -1 to 1:

- $s_i = 1$: Perfectly clustered (high cohesion, high separation).
- $s_i = 0$: Indistinct clustering (poor separation).
- $s_i < 0$: Misclustered (closer to points in a different cluster).

4. **Cluster and Overall Silhouette Coefficient**:
    - Compute the **average silhouette coefficient** for all points in a cluster or for the entire dataset to evaluate clustering quality.

Example: Silhouette Coefficient

- **Plot Description**:
    - Figure 5.29 shows silhouette coefficients for points in 10 clusters.
    - Darker shades correspond to **lower silhouette coefficients** (poorer clustering).

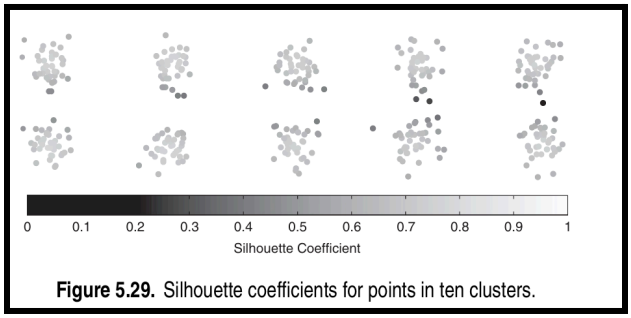
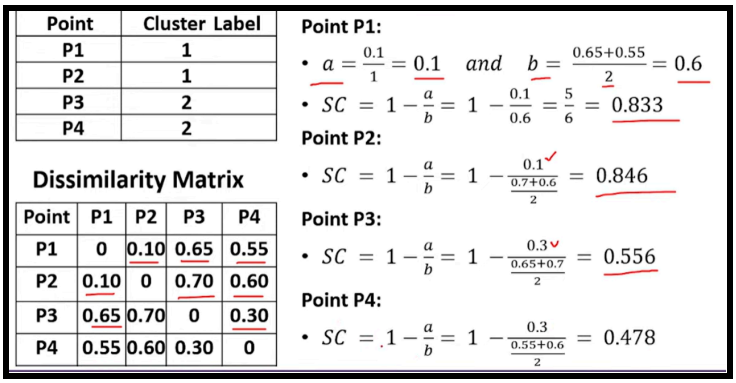


Figure 5.29. Silhouette coefficients for points in ten clusters.

Example



Point	Cluster Label
P1	1
P2	1
P3	2
P4	2

Dissimilarity Matrix

Point	P1	P2	P3	P4
P1	0	0.10	0.65	0.55
P2	0.10	0	0.70	0.60
P3	0.65	0.70	0	0.30
P4	0.55	0.60	0.30	0

Point P1:

- $a = \frac{0.1}{1} = 0.1$ and $b = \frac{0.65+0.55}{2} = 0.6$
- $SC = 1 - \frac{a}{b} = 1 - \frac{0.1}{0.6} = \frac{5}{6} = 0.833$

Point P2:

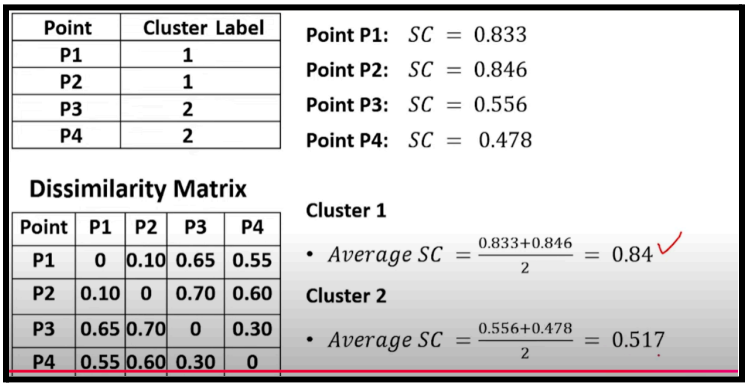
- $SC = 1 - \frac{a}{b} = 1 - \frac{0.1}{\frac{0.7+0.6}{2}} = 0.846$

Point P3:

- $SC = 1 - \frac{a}{b} = 1 - \frac{0.3}{\frac{0.65+0.7}{2}} = 0.556$

Point P4:

- $SC = 1 - \frac{a}{b} = 1 - \frac{0.3}{\frac{0.55+0.6}{2}} = 0.478$



Point	Cluster Label
P1	1
P2	1
P3	2
P4	2

Dissimilarity Matrix

Point	P1	P2	P3	P4
P1	0	0.10	0.65	0.55
P2	0.10	0	0.70	0.60
P3	0.65	0.70	0	0.30
P4	0.55	0.60	0.30	0

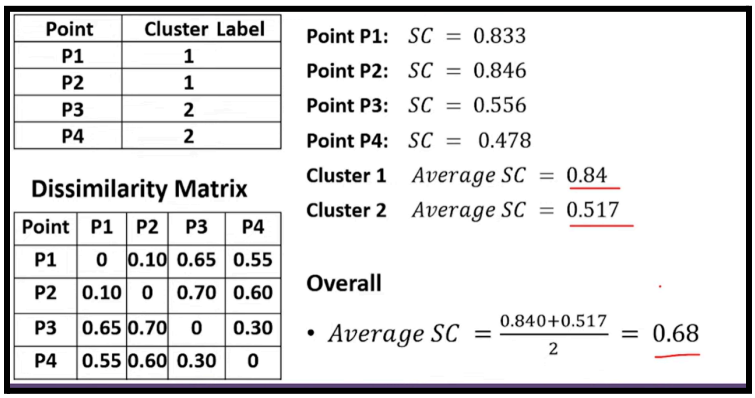
Point P1: $SC = 0.833$
Point P2: $SC = 0.846$
Point P3: $SC = 0.556$
Point P4: $SC = 0.478$

Cluster 1

- $Average\ SC = \frac{0.833+0.846}{2} = 0.84$

Cluster 2

- $Average\ SC = \frac{0.556+0.478}{2} = 0.517$



Point	Cluster Label
P1	1
P2	1
P3	2
P4	2

Dissimilarity Matrix

Point	P1	P2	P3	P4
P1	0	0.10	0.65	0.55
P2	0.10	0	0.70	0.60
P3	0.65	0.70	0	0.30
P4	0.55	0.60	0.30	0

Point P1: $SC = 0.833$
Point P2: $SC = 0.846$
Point P3: $SC = 0.556$
Point P4: $SC = 0.478$
Cluster 1 $Average\ SC = 0.84$
Cluster 2 $Average\ SC = 0.517$

Overall

- $Average\ SC = \frac{0.840+0.517}{2} = 0.68$

Applications of Cohesion, Separation, and Silhouette

- **Evaluating Individual Clusters**:
  - Rank clusters by cohesion and separation to identify poorly defined clusters:
    - Split clusters with low cohesion.
    - Merge clusters with low separation.
- **Evaluating Individual Points**:
  - Identify points near cluster edges (low cohesion) or that are likely misclustered.

UNSUPERVISED CLUSTER EVALUATION USING PROXIMITY MATRICES

Proximity Matrices:

- A **proximity matrix** is a square matrix where each entry represents the similarity (or distance) between two data points.
- These matrices can be used to assess clustering quality by comparing the **actual proximity matrix** to an **ideal proximity matrix**.

Methods:

1. **Correlation Between Actual and Ideal Matrices**:
    - **Ideal Proximity Matrix**:
        - Clusters are perfectly separated:
            - Similarity = 1 for points in the same cluster.
            - Similarity = 0 for points in different clusters.
    - **Correlation**:
        - Compute the correlation between the actual proximity matrix and the ideal proximity matrix.
        - **High Correlation**:
            - Indicates that points within the same cluster are close, and points from different clusters are well-separated.
        - **Example**:
            - For random data: Correlation is low (e.g., 0.58).
            - For well-separated clusters: Correlation is high (e.g., 0.92).
2. **Visualizing Proximity Matrices**:
    - **Reorder rows and columns** of the proximity matrix based on cluster labels.
    - **Block-Diagonal Patterns**:
        - Well-separated clusters form distinct blocks along the diagonal of the matrix.
        - Weak or noisy clustering produces less distinct block patterns.
    - **Example**:
        - Data with three well-separated clusters shows clear block-diagonal patterns.
        - Random data produces weak, unclear patterns.

Challenges:

- **High Computational Cost**:
    - Computing proximity matrices requires O(m^2) time for mmm data points.
- **Sampling for Large Datasets**:
    - Use a sample of data points from each cluster to compute proximity matrices efficiently.

Strengths:

- **Silhouette Coefficient**:
    - Provides a single interpretable value for evaluating clustering quality.
    - Suitable for partitional, center-based clusters like K-means.
- **Proximity Matrices**:
    - Allow detailed qualitative analysis of cluster structure.
    - Useful for comparing actual clustering results with idealized expectations.

Limitations:

- **Cluster Types**:
    - Many measures, including the silhouette coefficient, perform poorly on **density- or continuity-based clusters** (e.g., DBSCAN).
- **Over-Fitting with Clusters**:
    - Measures like the silhouette coefficient may overestimate quality when clusters are split too finely.
- **Mismatch with Algorithms**:
    - Evaluation measures may not align with clustering algorithms' objectives (e.g., SSE for K-means vs. silhouette coefficient).

Practical Applications of Proximity Matrices

1. **Judging Clustering Quality**:
    - Use proximity matrices to identify whether clusters are well-separated or noisy.
    - Block-diagonal patterns suggest high-quality clustering.
2. **Diagnosing Poor Clustering**:
    - Weak patterns in proximity matrices indicate overlapping or poorly separated clusters.
    - Useful for refining algorithms or selecting better parameters.
3. **Efficient Sampling**:
    - For large datasets, use **oversampling** for smaller clusters to ensure adequate representation in proximity matrix analysis.

DETERMINING THE CORRECT NUMBER OF CLUSTERS

Overview

Unsupervised cluster evaluation measures can help identify the **natural number of clusters** in a dataset. Two commonly used metrics for this purpose are:

1. **Sum of Squared Errors (SSE)**: Measures the compactness of clusters.
2. **Silhouette Coefficient**: Balances cohesion (compactness) and separation (distinctness).

Both metrics are plotted against the number of clusters to reveal patterns that suggest the optimal cluster count.

Example: Data with 10 Natural Clusters

Dataset Overview:

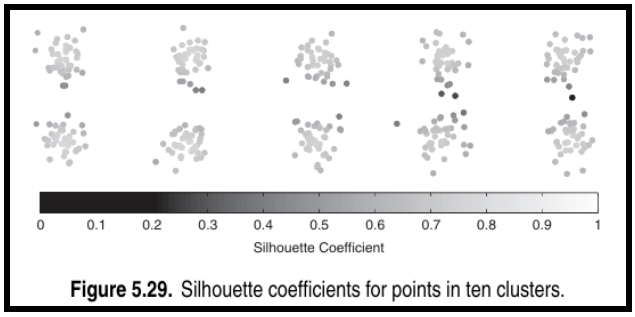


Figure 5.29. Silhouette coefficients for points in ten clusters.

SSE Plot :

- **SSE (Sum of Squared Errors)** decreases as the number of clusters increases.
    - This happens because more clusters reduce intra-cluster variance, making clusters smaller and more compact.
- A **distinct knee** (or "elbow") appears at **10 clusters**, indicating the natural number of clusters.
    - Adding more clusters beyond this point does not significantly reduce SSE, as clusters are already compact.

Silhouette Coefficient Plot (Figure 5.33):

- The **average silhouette coefficient** measures the balance between cohesion and separation.
- A **distinct peak** occurs at **10 clusters**, indicating the optimal cluster count.
    - Beyond this point, clusters become too finely split, reducing separation.

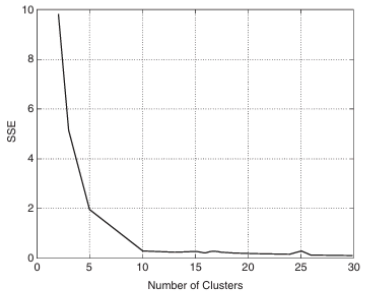


Figure 5.32. SSE versus number of clusters for the data of Figure 5.29 on page 364.

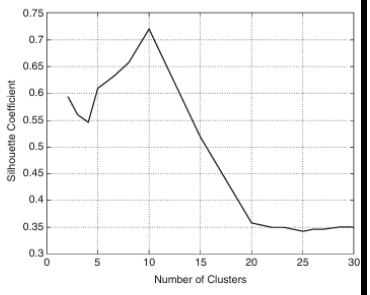


Figure 5.33. Average silhouette coefficient versus number of clusters for the data of Figure 5.29.

How to Determine the Number of Clusters

1. **SSE Curve**:
    - Plot SSE against the number of clusters.
    - Look for the **knee** or **elbow**:

        - This is the point where adding more clusters results in diminishing returns in SSE reduction.
    - Example: The knee at 10 clusters in Figure 5.32.
2. **Silhouette Coefficient Curve**:
    - Plot the silhouette coefficient against the number of clusters.
    - Look for the **peak**:
        - The highest silhouette coefficient indicates the best balance between cohesion and separation.
    - Example: The peak at 10 clusters in Figure 5.33.
3. **Combined Insights**:
    - Use both curves to confirm the cluster count.
    - The knee in the SSE curve and the peak in the silhouette coefficient curve should align to suggest the natural number of clusters.

Challenges and Limitations

1. **Overlapping or Intertwined Clusters**:
    - When clusters are not well-separated or overlap significantly, both SSE and silhouette coefficient may fail to identify a clear optimal number of clusters.
2. **Nested Clusters**:
    - In datasets with **nested clusters** (e.g., clusters within clusters), identifying a single "natural" number of clusters may be ambiguous.
    - Example: In Figure 5.29, the dataset has **10 clusters**, but there are also **5 pairs of clusters** (top-to-bottom nested clusters).
        - The SSE curve shows a smaller knee at **5 clusters**, indicating the nested structure.
3. **Data Characteristics**:
    - The effectiveness of these methods depends on the dataset's shape, density, and cluster separation.

SUPERVISED MEASURES OF CLUSTER VALIDITY

Why Use Class Labels in Clustering?

Clustering is typically unsupervised, but class labels can sometimes be available or derived externally. In such cases, evaluating clustering performance against these labels is useful for:

1. **Comparing clustering algorithms** with a "ground truth."
2. **Assessing the capability** of clustering to replicate manual classification (e.g., news categorization).
3. **Supporting semi-supervised learning**, where clusters should align with class labels.

Two broad approaches to supervised cluster evaluation:

1. **Classification-Oriented Measures**: Evaluate the overlap between clusters and classes (e.g., entropy, purity, F-measure).
2. **Similarity-Oriented Measures**: Assess the agreement between clustering and class labels using binary similarity measures.

Classification-Oriented Measures

1. Entropy

- **Definition**: Measures the degree to which a cluster contains objects from a single class.
- For cluster i,

$$e_i = -\sum_{j=1}^{L} p_{ij} \log_2(p_{ij})$$

- $p_{ij} = \frac{m_{ij}}{m_i}$: Proportion of objects in cluster i belonging to class j.
- m_{ij}: Number of objects in cluster i from class j.
- m_i: Total objects in cluster i.

- **Total entropy**:

$$e = \sum_{i=1}^{K} \frac{m_i}{m} e_i$$

  - Weighted sum of cluster entropies (mmm = total objects, K = number of clusters).

2. Purity

- **Definition**: Measures the extent to which a cluster contains objects from a single class.
- For cluster i,

$$\text{purity}(i) = \max_j(p_{ij})$$

Overall purity:

$$\text{purity} = \sum_{i=1}^{K} \frac{m_i}{m} \text{purity}(i)$$

3. Precision and Recall

- **Precision**: Fraction of a cluster that belongs to a specific class.

$$\text{precision}(i,j) = p_{ij}$$

- **Recall**: Fraction of a class that is captured by a specific cluster.

$$\text{recall}(i,j) = \frac{m_{ij}}{m_j}$$

4. F-Measure

- **Definition**: Combines precision and recall to evaluate cluster-class alignment.
- For cluster i and class j:

$$F(i,j) = \frac{2 \cdot \text{precision}(i,j) \cdot \text{recall}(i,j)}{\text{precision}(i,j) + \text{recall}(i,j)}$$

- **Overall F-measure**:
  - Weighted average of F(i,j)values across clusters and classes.

Example to cal entropy and purity:

How to Calculate Entropy and Purity Confusion Matrix

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Total	Entropy	Purity
#1	1	1	0	11	4	676	693		
#2	27	89	333	827	253	33	1562		
#3	326	465	8	105	16	29	949		
Total	354	555	341	943	273	738	3204		

Entropy of cluster i:

$$e_i = -\sum_{j=1}^{L} \frac{m_{ij}}{m_i} \log_2 \frac{m_{ij}}{m_i}$$

- m_{ij} is the number of objects of *class j* in *cluster i*
- m_i the number of objects in *cluster i*

How to Calculate Entropy and Purity Confusion Matrix

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Total	Entropy	Purity
#1	1	1	0	11	4	676	693	0.202	
#2	27✓	89	333	827	253	33	1562		
#3	326	465	8	105	16	29	949		
Total	354	555	341	943	273	738	3204		

$$e_1 = -\frac{1}{693}\log_2\frac{1}{693} - \frac{1}{693}\log_2\frac{1}{693} - \frac{0}{693}\log_2\frac{0}{693} - \frac{11}{693}\log_2\frac{11}{693} - \frac{4}{693}\log_2\frac{4}{693} - \frac{676}{693}\log_2\frac{676}{693}$$

$$e_1 = 0.0136 + 0.0136 + 0 + 0.095 + 0.043 + 0.035 = 0.202$$

$$e_2 = -\frac{27}{1562}\log_2\frac{27}{1562} - \frac{89}{1562}\log_2\frac{89}{1562} - \frac{333}{1562}\log_2\frac{333}{1562} - \frac{827}{1562}\log_2\frac{827}{1562} - \frac{253}{1562}\log_2\frac{253}{1562} - \frac{33}{1562}\log_2\frac{33}{1562}$$

How to Calculate Entropy and Purity Confusion Matrix

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Total	Entropy	Purity
#1	1	1	0	11	4	676	693	0.202	
#2	27	89	333	827	253	33	1562	1.839	
#3	326	465	8	105	16	29	949	1.70	
Total	354	555	341	943	273	738	3204		

$$e_3 = -\frac{326}{949}\log_2\frac{326}{949} - \frac{465}{949}\log_2\frac{465}{949} - \frac{8}{949}\log_2\frac{8}{949} - \frac{105}{949}\log_2\frac{105}{949} - \frac{16}{949}\log_2\frac{16}{949} - \frac{29}{949}\log_2\frac{29}{949} = 1.70$$

$$e_{Total} = -\frac{354}{3204}\log_2\frac{354}{3204} - \frac{555}{3204}\log_2\frac{555}{3204} - \frac{341}{3204}\log_2\frac{341}{3204} - \frac{943}{3204}\log_2\frac{943}{3204} - \frac{273}{3204}\log_2\frac{273}{3204} - \frac{738}{3204}\log_2\frac{738}{3204}$$
$$= 1.44$$

How to Calculate Entropy and Purity Confusion Matrix

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Total	Entropy	Purity
#1	1	1	0	11	4	676	693	0.202	
#2	27	89	333	827	253	33	1562	1.839	
#3	326	465	8	105	16	29	949	1.70	
Total	354	555	341	943	273	738	3204	1.44	

Purity of cluster i:

- m_{ij} is the number of objects of *class j* in *cluster i*

$$p_i = \max_i\left(\frac{m_{ij}}{m_i}\right)$$

How to Calculate Entropy and Purity Confusion Matrix

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Total	Entropy	Purity
#1	1	1	0	11	4	676	693	0.202	
#2	27	89	333	827	253	33	1562	1.839	
#3	326	465	8	105	16	29	949	1.70	
Total	354	555	341	943	273	738	3204	1.44	

$$p_1 = \max_1\left(\frac{m_{1j}}{m_1}\right) = \max_1\left(\frac{1}{693}, \frac{1}{693}, \frac{0}{693}, \frac{11}{693}, \frac{4}{693}, \frac{676}{693}\right) = \frac{676}{693} = 0.975$$

$$p_2 = \max_2\left(\frac{m_{2j}}{m_2}\right) = \max_2\left(\frac{27}{1562}, \frac{89}{1562}, \frac{333}{1562}, \frac{827}{1562}, \frac{253}{1562}, \frac{33}{1562}\right) = \frac{827}{1562} = 0.529$$

$$p_3 = \max_3\left(\frac{m_{3j}}{m_3}\right) = \max_3\left(\frac{326}{949}, \frac{465}{949}, \frac{8}{949}, \frac{105}{949}, \frac{16}{949}, \frac{29}{949}\right) = \frac{465}{949} = 0.49$$

How to Calculate Entropy and Purity Confusion Matrix

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Total	Entropy	Purity
#1	1	1	0	11	4	676	693	0.202	0.975
#2	27	89	333	827	253	33	1562	1.839	0.529
#3	326	465	8	105	16	29	949	1.70	0.49
Total	354	555	341	943	273	738	3204	1.44	

$$Overall\ Purity = \sum_{i=1}^{k}\frac{m_i}{m}p_i$$

$$Overall\ Purity = \frac{693}{3204} * 0.975 + \frac{1562}{3204} * 0.529 + \frac{949}{3204} * 0.49 = 0.614$$

Similarity-Oriented Measures

1. **Matrix-Based Comparison**

- Compare two matrices:
    1. **Ideal Cluster Similarity Matrix**:
        - 1 if two objects are in the same cluster, 0 otherwise.
    2. **Class Similarity Matrix**:
        - 1 if two objects belong to the same class, 0 otherwise.
- **Hubert's Γ\GammaΓ Statistic**: Measures correlation between these matrices.

2. Binary Similarity Measures

- For all object pairs, compute:

- f_{00}: Pairs with different class and different cluster.
- f_{01}: Pairs with different class but same cluster.
- f_{10}: Pairs with same class but different cluster.
- f_{11}: Pairs with same class and same cluster.

- **Rand Statistic**:

$$\text{Rand} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

- **Jaccard Coefficient:**

$$\text{Jaccard} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

Example to cal correlation:

Compute Correlation Similarity and Ideal Similarity Matrix

Cluster Label

Point	Cluster Label
P1	1
P2	1
P3	2
P4	2

Similarity Matrix

Point	P1	P2	P3	P4
P1	1	0.8	0.65	0.55
P2	0.8	1	0.7	0.6
P3	0.65	0.7	1	0.3
P4	0.55	0.6	0.3	1

- Compute the correlation between the similarity matrix and the ideal similarity matrix.
- In ideal similarity matrix the ij^{th} entry is 1 if two objects belong to the same cluster, and 0 otherwise.

Compute Correlation Similarity and Ideal Similarity Matrix

Cluster Label

Point	Cluster Label
P1	1
P2	1
P3	2
P4	2

Similarity Matrix

Point	P1	P2	P3	P4
P1	1	0.8	0.65	0.55
P2	0.8	1	0.7	0.6
P3	0.65	0.7	1	0.3
P4	0.55	0.6	0.3	1

- Compute the correlation between the similarity matrix and the ideal similarity matrix.
- In ideal similarity matrix the ij^{th} entry is 1 if two objects belong to the same cluster, and 0 otherwise.

Compute Correlation Similarity and Ideal Similarity Matrix

- **x** =< 0.8, 0.65, 0.55, 0.7, 0.6, 0.3 > and **y** = < 1, 0, 0, 0, 0, 1 >,
- $\sigma = \sqrt{\frac{\sum(x_i-\mu)^2}{n-1}}$
- $\mu_x = \frac{0.8+0.65+0.55+0.7+0.6+0.3}{6} = 0.6$
- $\sigma_x = \sqrt{\frac{(0.8-0.6)^2+(0.65-0.6)^2+(0.55-0.6)^2+(0.7-0.6)^2+(0.6-0.6)^2+(0.3-0.6)^2}{6-1}}$
- $\sigma_x = 0.1703$

Compute Correlation Similarity and Ideal Similarity Matrix

- **x** =< 0.8, 0.65, 0.55, 0.7, 0.6, 0.3 > and **y** = < 1, 0, 0, 0, 0, 1 >,
- $\sigma = \sqrt{\frac{\sum(x_i-\mu)^2}{n-1}}$
- $\mu_y = \frac{1+0+0+0+0+1}{6} = 0.33$
- $\sigma_y = \sqrt{\frac{(1-0.33)^2+(0-0.33)^2+(0-0.33)^2+(0-0.33)^2+(0-0.33)^2+(1-0.33)^2}{6-1}}$
- $\sigma_y = 0.5164$

Compute Correlation Similarity and Ideal Similarity Matrix

- **x** =< 0.8, 0.65, 0.55, 0.7, 0.6, 0.3 > and **y** = < 1, 0, 0, 0, 0, 1 >,
- Covariance of **x** and **y**:
- $cov(\boldsymbol{x}, \boldsymbol{y}) = \frac{\Sigma(x_i - \mu_x) * (y_i - \mu_y)}{n-1}$
- $cov(\boldsymbol{x}, \boldsymbol{y}) = \frac{(0.8-0.6)*(1-0.33)+(0.65-0.6)*(0-0.33)+(0.55-0.6)*(0-0.33)+(0.7-0.6)*(0-0.33)+(0.6-0.6)*(0-0.33)+(0.3-0.6)*(1-0.33)}{6-1}$
- $cov(\boldsymbol{x}, \boldsymbol{y}) = -0.2$

Compute Correlation Similarity and Ideal Similarity Matrix

- **x** =< 0.8, 0.65, 0.55, 0.7, 0.6, 0.3 > and **y** = < 1, 0, 0, 0, 0, 1 >,
- Therefore,
- $corr(\boldsymbol{x}, \boldsymbol{y}) = \frac{cov(\boldsymbol{x}, \boldsymbol{y})}{\sigma_x \sigma_y} = \frac{-0.2}{0.1703 * 0.5164}$
- $corr(\boldsymbol{x}, \boldsymbol{y}) = -0.227$

Example Jaccard and Rand:

$$\text{Rand statistic} = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}} \quad (5.18)$$

$$\text{Jaccard coefficient} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} \quad (5.19)$$

Example 5.17 (Rand and Jaccard Measures)**.** Based on these formulas, we can readily compute the Rand statistic and Jaccard coefficient for the example based on Tables 5.10 and 5.11. Noting that $f_{00} = 4$, $f_{01} = 2$, $f_{10} = 2$, and $f_{11} = 2$, the Rand statistic $= (2 + 4)/10 = 0.6$ and the Jaccard coefficient $= 2/(2+2+2) = 0.33$. ■

We also note that the four quantities, f_{00}, f_{01}, f_{10}, and f_{11}, define a *contingency* table as shown in Table 5.12.

Table 5.12. Two-way contingency table for determining whether pairs of objects are in the same class and same cluster.

	Same Cluster	Different Cluster
Same Class	f_{11}	f_{10}
Different Class	f_{01}	f_{00}

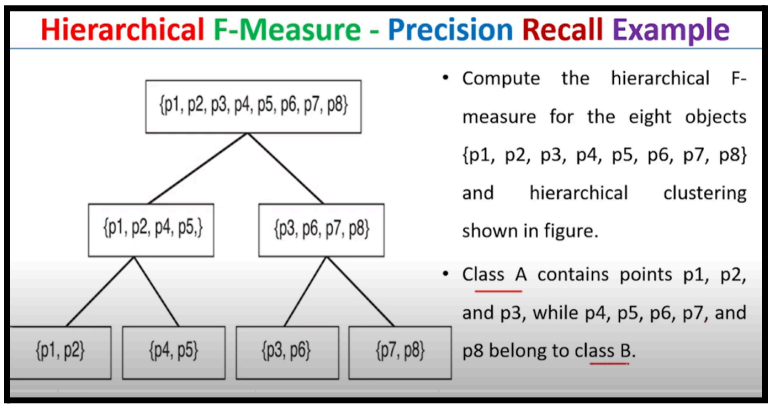
3. Hierarchical Clustering Validation

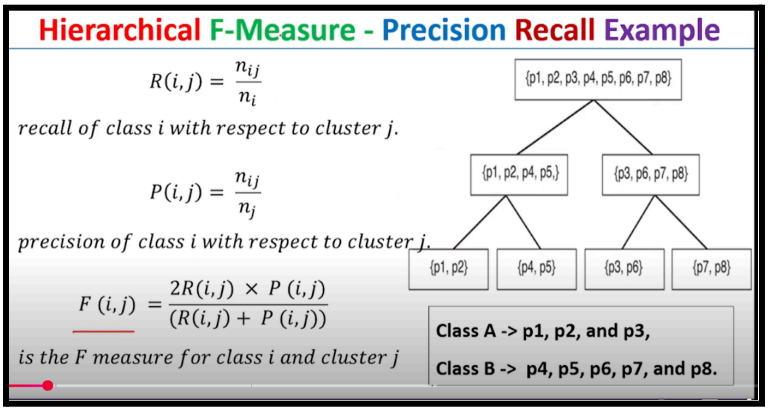
- Evaluate if each class is captured well by at least one cluster in the hierarchy.
- **Hierarchical F-Measure**:

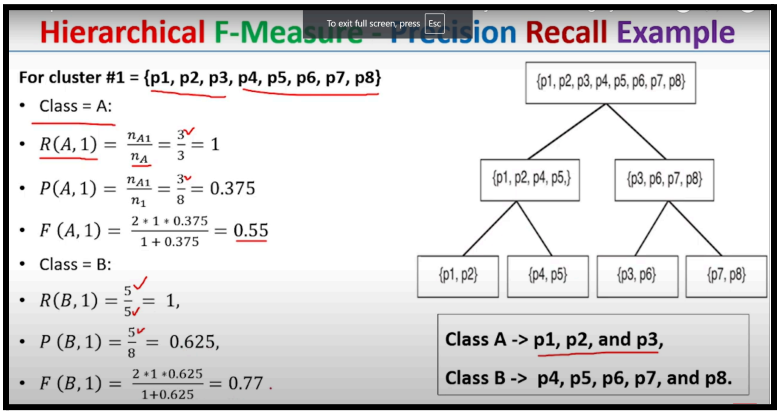
$$F = \sum_j \frac{m_j}{m} \max_i F(i, j)$$

- Combines the best F(i,j) scores for each class.

Example Hierarchical F measure:

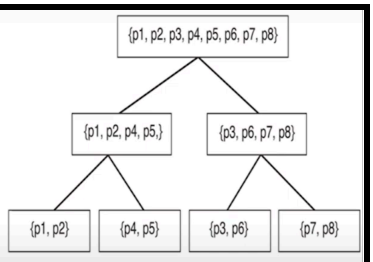






For cluster #2= {p1,p2,p4,p5}

- **Class = A:**
- $R(A,2) = \frac{2}{3} = 0.667,$
- $P\ (A,2) = \frac{2}{4} = 0.5,$
- $F\ (A,2)\ =\ 0.57$
- **Class = B:**
- $R(B,2) = \frac{2}{5} = 0.4,$
- $P\ (B,2) = \frac{2}{4} 0.5,$
- $F\ (B,2)\ =\ 0.44$



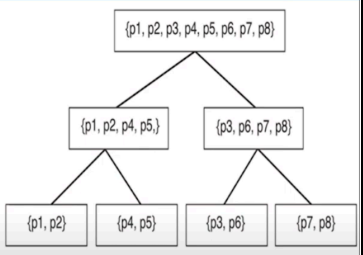
Class A -> p1, p2, and p3,

Class B -> p4, p5, p6, p7, and p8.

Hierarchical F-Measure - Precision Recall Example

For cluster #3= {p3, p6, p7, p8}

- Class = A:
- $R(A,3) = \frac{1}{3} 0.333,$
- $P\ (A,3) = \frac{1}{4} = 0.25,$
- $F\ (A,3)\ =\ 0.29$
- Class =B:
- $R(B,3) = \frac{3}{5} = 0.6,$
- $P\ (B,3) = \frac{3}{4} = 0.75,$
- $F\ (B,3)\ =\ 0.67$



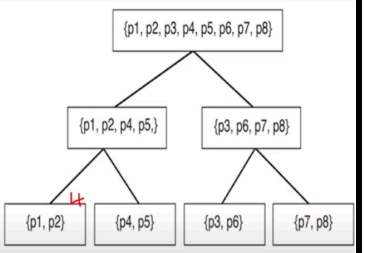
Class A -> p1, p2, and p3,

Class B -> p4, p5, p6, p7, and p8.

Hierarchical F-Measure - Precision Recall Example

For cluster #5 = {p4, p5}

- **Class = A:**
- $R(A,5)\ =\ 0,$
- $P\ (A,5)\ =\ 0,$
- $F\ (A,5)\ =\ 0$
- **Class =B:**
- $R(B,5) = \frac{2}{5} = 0.4,$
- $P\ (B,5) = \frac{2}{2} = 1,$
- $F\ (B,5)\ =\ 0.57$



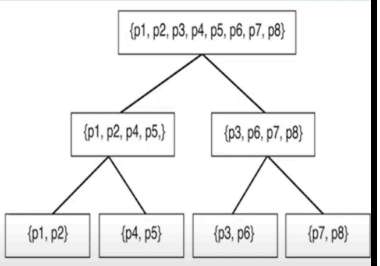
Class A -> p1, p2, and p3,

Class B -> p4, p5, p6, p7, and p8.

Hierarchical F-Measure - Precision Recall Example

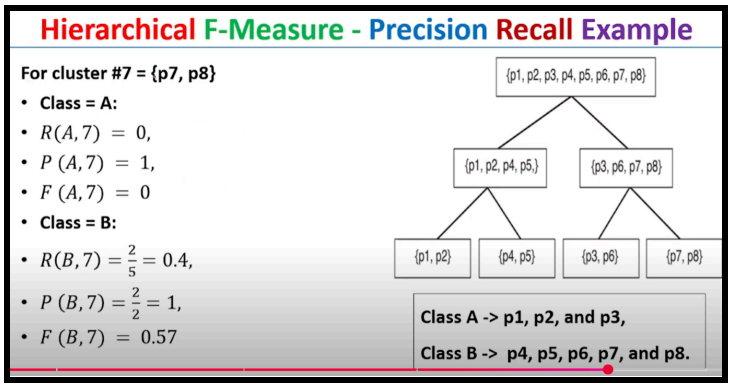
For cluster #6 = {p3, p6}

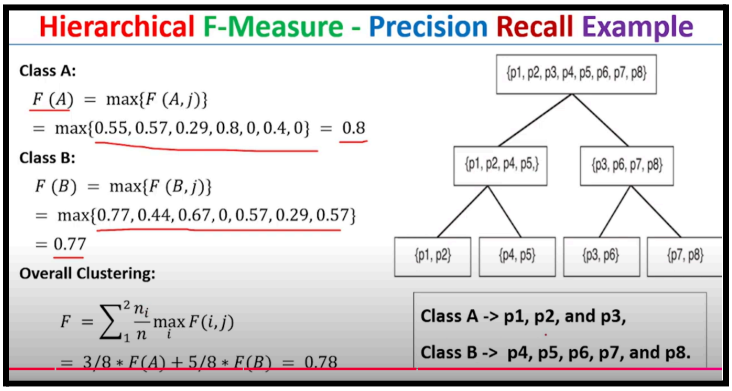
- **Class = A:**
- $R(A,6) = \frac{1}{3} = 0.33,$
- $P\ (A,6) = \frac{1}{2} = 0.5,$
- $F\ (A,6)\ =\ 0.4$
- **Class =B:**
- $R(B,6) = \frac{1}{5} = 0.2,$
- $P\ (B,6) = \frac{1}{2} = 0.5,$
- $F\ (B,6)\ =\ 0.29$



Class A -> p1, p2, and p3,

Class B -> p4, p5, p6, p7, and p8.





Key Takeaways

1. **Classification-Oriented Measures**:
   - Evaluate alignment between clusters and classes.
   - Include metrics like entropy, purity, precision, recall, and F-measure.
2. **Similarity-Oriented Measures**:
   - Compare cluster similarity to class similarity using statistics like Rand and Jaccard.
3. **Application**:
   - Useful for validating clustering quality and assessing how well clusters align with known classes.
4. **Example**:
   - K-means on newspaper articles demonstrated the utility of these measures to evaluate clustering performance.

ACCESSING THE SIGNIFICANCE OF CLUSTER VALIDITY MEASURES

Cluster validity measures are tools used to assess the "goodness" of the clusters obtained through clustering algorithms. These measures typically provide a single numerical value indicating the quality of clustering based on specific criteria (e.g., compactness, separation, or alignment with external labels).

- **Examples**:
    - **Purity**: Measures how well clusters align with external class labels (0 is bad, 1 is perfect alignment).
    - **Entropy**: Indicates how mixed clusters are in terms of class composition (lower is better; 0 is ideal).
    - **SSE (Sum of Squared Errors)**: Reflects the compactness of clusters (lower values are better).

Challenges in Interpreting Validity Measures

While the numerical results from these measures can provide some guidance, interpreting their significance can be challenging:

1. **Known Minimum/Maximum Values**:
    - For some measures (e.g., purity, entropy), the range of values is known:
        - **Purity**: Ranges from 0 (poor alignment) to 1 (perfect alignment).
        - **Entropy**: Lower values (closer to 0) indicate better clustering.
    - These benchmarks provide a straightforward way to interpret the results.
2. **No Fixed Range**:
    - Some measures, like SSE, lack a fixed range, and their interpretation depends on:
        - The scale of the data.
        - The clustering algorithm used.
        - The number of clusters specified.
3. **Intermediate Values**:
    - Intermediate values (e.g., a purity of 0.6 or entropy of 0.5) require additional context or comparative benchmarks to interpret meaningfully.

Statistical Interpretation of Validity Measures

When direct interpretation is difficult, statistical methods can help evaluate the **significance** of the results:

1. **Randomized Comparison**:
    - The clustering result is compared to the distribution of results obtained from random data.
    - **Example**:
        - A dataset with three well-separated clusters is analyzed using K-means.
        - Random data (e.g., 100 points uniformly distributed) is clustered repeatedly.
        - A histogram of SSE values is generated from 500 random runs (Figure 5.34).
        - If the SSE of the actual clustering (e.g., 0.0050) is significantly lower than the lowest SSE from random runs (e.g., 0.0173), it suggests the clustering reflects meaningful structure.
2. **Known Distributions**:
    - For measures like **Hubert's Γ statistic**, the statistical distribution is known.
    - These distributions can be used directly to compute **p-values** or normalize the measures for better interpretability.
3. **Normalized Measures**:

- By subtracting the mean and dividing by the standard deviation of the distribution, a normalized score can indicate how unusual the observed value is relative to random expectations.

Relative vs. Absolute Evaluation

Cluster validity measures can be used in two ways:

1. **Absolute Evaluation**:
   - Evaluate a clustering result in isolation.
   - Requires statistical analysis to determine if the measure is "good enough" (e.g., if the observed SSE is unlikely to occur by chance).
2. **Relative Evaluation**:
   - Compare multiple clustering results (e.g., from different algorithms or parameter settings).
   - Challenges:
     - Statistical significance: Are the differences between measures reproducible and not due to randomness?
     - Practical significance: Are the differences meaningful for the application? For instance, a 0.1% improvement in purity may not justify additional complexity.

Evaluating SSE with Randomization

Objective: Determine if the observed clustering reflects real structure or random noise.

- **Steps**:
  1. Cluster the actual dataset (e.g., three well-separated clusters in Figure 5.30).
  2. Generate random datasets with the same range and size as the actual dataset.
  3. Cluster the random datasets using the same algorithm (e.g., K-means) and record the SSE for each run.
  4. Compare the observed SSE with the distribution of SSE values from random runs.
     - **Observed SSE**: 0.0050 (from actual clustering).
     - **Random SSE**: Range is 0.0173 to higher values (from random data).
  5. Interpretation:
     - The observed SSE is significantly lower than the random SSE values.
     - Conclusion: The clustering is statistically significant and reflects meaningful structure..

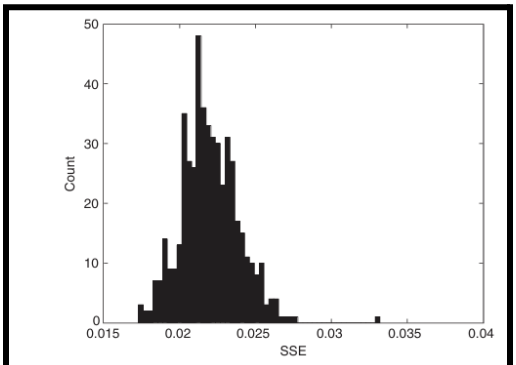


Figure 5.34. Histogram of SSE for 500 random data sets.

CHOOSING A CLUSTER VALIDITY MEASURE

Cluster validity measures are critical in assessing the quality of clustering results. However, selecting the appropriate measure depends on the **goal of clustering** and the **nature of the data**.

1. Understanding the Goal of Clustering

A. Clustering for Summarization

- **Purpose**: Summarize data by reducing complexity and compressing information.
- **Approach**:
    - Focus on minimizing **representation error**.
    - Use measures that prioritize compact clusters with minimal distances between points and their centroids.
- **Recommended Measures**:
    - **SSE (Sum of Squared Errors)**: Measures the total variance within clusters.
    - Suitable for clustering aimed at data compression (e.g., K-means or similar algorithms).

B. Clustering for Understanding

- **Purpose**: Discover meaningful insights or structure in the data.
- **Approach**:
    - Maximize **cohesion** (compactness of clusters) and **separation** (distance between clusters).
    - Use measures tailored to the specific clustering algorithm and data type.
- **Challenges**:
    - Some measures assume a single "correct" number of clusters, which may not capture subclusters or complex structures.
    - Not all measures are suitable for clusters with irregular shapes or density-based clusters.
- **Recommended Approach**:
    - Examine **cohesion** and **separation** manually for smaller numbers of clusters.
    - Use specialized measures for irregular or density-based clusters.

2. Measures for Different Scenarios

A. Unsupervised Clustering

- **Key Objective**: Evaluate clusters without external labels.
- **Common Measures**:
    1. **Cohesion and Separation**:
        - Cohesion: Compactness within clusters.
        - Separation: Distance between clusters.
    2. **Silhouette Coefficient**:

        - Combines cohesion and separation.
        - Indicates how well each point fits into its cluster versus the next closest cluster.
    3. **Graph-Based Measures**:
        - For irregular or intertwined clusters.
        - Example: Contiguity or density-based clustering validity.

B. Supervised Clustering

- **Key Objective**: Align clusters with externally provided class labels.
- **Common Measures**:
    1. **F-Measure and Hierarchical F-Measure**:
        - Evaluate how well clusters match the underlying class structure.
    2. **Entropy and Purity**:
        - Entropy: Measures cluster homogeneity.
        - Purity: Measures how dominated a cluster is by a single class.
    3. **Confusion Matrix**:
        - Shows relationships between clusters and class labels.
        - Highlights overlapping or mixed clusters.

3. Specific Challenges in Choosing Validity Measures

A. Irregular and Intertwined Clusters

- Many measures are unsuitable for **density-based** or **contiguity-based** clusters with non-globular shapes.
- Use specialized algorithms and tailored validity measures (e.g., for DBSCAN or OPTICS).

B. Choosing the Number of Clusters

- Some measures (e.g., SSE) tend to improve with an increasing number of clusters, up to one cluster per point.
- Look for "knees" or "peaks" in evaluation plots (e.g., SSE or silhouette coefficient vs. number of clusters) to determine the optimal number of clusters.

C. Confusion Matrix for Supervised Clustering

- Provides detailed insights into the alignment between clusters and classes.
- Highlights relationships that single measures (like F-measure) might not capture.

4. Clustering as an Exploratory Tool

- Clustering is often used to **explore data** rather than to arrive at definitive answers.
- Cluster validity measures are tools for gaining insights, not necessarily for producing a "correct" or "optimal" clustering.
- Choose measures based on their ability to enhance understanding of the data's underlying structure.

CLUSTER TENDENCY

The **Hopkins Statistic** is a statistical test used to determine if a dataset has a **non-random structure** or if it is **uniformly distributed** in the data space. It helps evaluate if clusters exist in the data.

Steps to Calculate Hopkins Statistic

1. **Data Sampling**:
    - Randomly sample n points (p1,p2,...,pn) **uniformly** from the dataset D.
    - For each sampled point pi, find its nearest neighbor xi in D, and calculate the distance:

$$x_i = \min\{\text{dist}(p_i, v)\} \quad \text{where } v \in D.$$

2. **Artificial Points Sampling**:
    - Randomly sample n artificial points (q1,q2,...qn) **uniformly** from the data space (not necessarily in D).
    - For each artificial point qi, find its nearest neighbor yi in D, excluding qiq_iqi, and calculate the distance:

$$y_i = \min\{\text{dist}(q_i, v)\} \quad \text{where } v \in D \text{ and } v \neq q_i.$$

3. **Hopkins Statistic Calculation**:
    - Use the following formula to calculate the **Hopkins Statistic H**:

$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} (x_i + y_i)}.$$

Interpreting Hopkins Statistic

- H≈0.5: The data is **uniformly distributed** (random), meaning there is no appear not clustering or structure.
- H≈0: The data is **highly skewed**, indicating the presence of **non-random structure** or clustering.
- The closer H is to **0**, the stronger the clustering tendency in the dataset.

Why Hopkins Statistic is Useful

- It tests for **spatial randomness** by comparing distances of actual data points with those of artificial random points.

- If the dataset has a strong cluster structure, the artificial points will tend to have larger nearest-neighbor distances than the actual data points.
- Helps determine if applying clustering techniques (like K-means or DBSCAN) is meaningful.
