What is Clustering?

- **Cluster**
  - A group of data objects
  - Similar (or related) to one another within the same group
  - Dissimilar (or unrelated) to the objects in different groups

- **Clustering (or Cluster Analysis)**
  - Finding clusters from data objects
  - **Unsupervised learning**: no pre-defined classes

- **Applications**
  - A stand-alone method for data analysis
  - A preprocessing step for other data analysis
Applications of Clustering

- **Business**
  - Grouping customers to promote sales
- **Economy**
  - Finding stocks with similar patterns for investment
- **IT**
  - Grouping web documents for information retrieval
- **Biology**
  - Grouping genes to predict their biological functions
- **Geography**
  - Finding areas with similar land use for city planning
- **Weather**
  - Finding similar climate patterns for weather forecast

Measuring Quality of Clustering

- **High Quality Clusters have**
  - High intra-class similarity: **cohesive** within clusters
  - Low inter-class similarity: **distinctive** between clusters
- **Quality of Clustering Depends on**
  - Clustering methods
    - Handling both cohesiveness and distinctiveness
    - Ability to discover hidden patterns
    - Defining “similar enough” – problem of determining a threshold
  - Data sets
    - Amount of data
    - Complexity of data type
    - High dimensionality
Similarity / Dissimilarity Functions (1)

**Numerical Attributes**
- Minkowski distance, \( d = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \)
- Euclidean distance, when \( p=2 \)
- Manhattan distance, when \( p=1 \)

**Binary Attributes**
- If a binary variable is symmetric,
  - Dissimilarity \( d = \frac{r + s}{q + r + s + t} \)
- If a binary variable is asymmetric,
  - Dissimilarity \( d = \frac{r + s}{q + r + s} \), similarity (Jaccard index) \( s = \frac{q}{q + r + s} \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 q</td>
<td>r</td>
<td>q+r</td>
<td></td>
</tr>
<tr>
<td>0 s</td>
<td>t</td>
<td>s+t</td>
<td></td>
</tr>
<tr>
<td>sum q+s</td>
<td>r+t</td>
<td>p</td>
<td></td>
</tr>
</tbody>
</table>

contingency table

Similarity / Dissimilarity Functions (2)

**Categorical Attributes**
- Similarity (Jaccard index), \( s(x, y) = \frac{|X \cap Y|}{|X \cup Y|} \)
  
  where \( X \): the set of variables for the object \( x \)
  
  \( Y \): the set of variables for the object \( y \)
- Similarity (Geometric index), \( s(x, y) = \frac{|X \cap Y|^2}{(|X| \cdot |Y|)} \)
- Similarity (Dice index), \( s(x, y) = \frac{2|X \cap Y|}{|X| + |Y|} \)

**Mixed Attributes**
- Weighted combination
Issues in Clustering

- Ability to deal with **different types of data**
- **Scalability** (handling a very large amount of data)
- **High dimensionality**
- Insensitivity to **order of input records**
- Ability to deal with **noise and outliers**
- Ability to handle **dynamically changing data**
- Incorporation of **user-specified constraints**
- **Interpretability and usability**
- Discovery of clusters with **arbitrary shapes**
- Requirements of domain knowledge to determine **input parameters**

Categories of Clustering Methods

- **Partition-based Methods**
  - Finding the best partitions, e.g., k-means, k-medoids, CLARANS
- **Hierarchical Methods**
  - Hierarchically merging or dividing data, e.g., Agnes, Diana, BIRCH
- **Density-based Methods**
  - Finding densely populated groups of data, e.g., DBSCAN, OPTICS
- **Grid-based Methods**
  - Grouping data based on multi-level granularity, e.g., STING, CLIQUE
- **Model-based Methods**
  - Finding the best fit to models, e.g., EM, COBWEB, SOM
- **Pattern-based Methods**
  - Grouping data with similar patterns, e.g., p-Cluster
- **Constraint-based Methods**
  - Considering user-specified or application-specific constraints
Chapter 10, Clustering

- **Partition-based Methods**
  - Hierarchical Methods
  - Density-based Methods
  - Grid-based Methods
  - Pattern-based Methods
  - Constraint-based Methods
  - Cluster Validation

**Partition-based Methods**

- **Main Idea**
  - Constructing a partition of the data with $n$ objects into $k$ clusters

- **Issue**
  - Finding a partition that optimize the partitioning criterion:
    - high intra-class similarity and low inter-class similarity

- **Methods**
  - Brute force algorithm: ?
  - Heuristic method: $k$-means, $k$-medoids (PAM), CLARANS
**k-Means**

- **Main Idea**
  - Heuristic clustering algorithm
  - Converges a local optimum quickly

- **Process**
  1. Partition objects randomly into $k$ clusters.
  2. Compute the mean point of the objects in each cluster as a centroid.
  3. Assign each object to the nearest centroid and generate $k$ new clusters.
  4. Repeat (2) and (3), until there is no change of the objects in each cluster.

---

**Example of k-Means**

1. Random partition:
   - Objects are randomly distributed.
2. Compute cluster means:
   - Centroids are calculated.
3. Re-assign objects:
   - Objects are moved to the nearest centroid.
4. Compute cluster means:
   - New centroids are calculated.
5. Re-assign objects:
   - Objects are moved again to the nearest centroid.
Summary of $k$-Means

**Strength**
- Relatively efficient
  - $O(?)$ where $n$ objects, $k$ clusters and $t$ iterations
  - Normally, $t, k \ll n$

**Weakness**
- Need to specify $k$, the number of clusters, in advance
- Sensitive to noise and outliers
- Applicable to only numeric data (when the mean is defined), not categorical data
- Not suitable to detect clusters with non-convex shapes
- Fall into local optimum, not identifying global optimum of clusters

$k$-Medoids

**Main Idea**
- The same process to $k$-means
- Instead of taking the mean of objects as a centroid for each cluster, use a medoid, the most centrally located object in a cluster

**Process of PAM (Partitioning around Medoids)**
1. Select $k$ medoids randomly
2. Compute the total cost as the sum of distance between each non-medoid and its nearest medoid
3. Replace one medoid with one non-medoid if the swap decreases the total cost
4. Repeat (3) until there is no change of the objects in each cluster
Example of \(k\)-Medoids

Random selection of \(k\) medoids & compute cost

Swapping & compute cost

Swapping & compute cost

Stop swapping if the lowest cost

Example of \(k\)-Medoids

- **Clustering**
  - All non-medoids are assigned to the closest medoid to form a set of clusters for each iteration
  - Cluster membership change example
    - \(O_i, O_j\) are original medoids
    - \(O_j\) is swapped with \(O_{random}\)

1. Reassigned to \(O_i\)
2. Reassigned to \(O_{random}\)
3. No change
4. Reassigned to \(O_{random}\)
Summary of $k$-Medoids

- **Strength**
  - Robust to noise and outliers

- **Weakness**
  - Not scalable to a large data set
  - $O(?)$ where $n$ objects and $k$ clusters
  - How to improve efficiency and scalability?

Improved $k$-Medoids

- **Main Idea**
  - combining $k$-means and $k$-medoids

- **Process**
  1. Initial $k$-medoids selection
     - ?
  2. Iterative update of $k$-medoids
     - ?

- **Reference**
CLARA (Clustering Large Applications)

- **Process**
  - Draw multiple samples of the data set
  - Apply PAM on each sample
  - Measure the quality (total cost) of clusters in the entire data set
  - Output the best clustering results

- **Strength**
  - Solve inefficiency of PAM in a large data set

- **Weakness**
  - Efficiency depends on the sample size.
  - The output does not represent the result from the whole data set if the sample is biased.

CLARANS

- **CLARANS**
  - Clustering Algorithm with Randomized Search

- **Main Idea**
  - Select a non-medoid to replace with a medoid in a sample (like CLARA)
  - Not restrict the medoid search in a particular sample (unlike CLARA)
  - Dynamically change the sample in every step of medoid search
  - Not confine the search in a localized area

- **Strength**
  - More efficient than PAM and more accurate than CLARA

- **Reference**
  - Ng, R. and Han, J., “CLARANS: A Method for Clustering Objects for Spatial Data Mining.” *IEEE Transactions on Knowledge and Data Engineering* (2002)
Chapter 10, Clustering

- Partition-based Methods
  - Hierarchical Methods
- Density-based Methods
- Grid-based Methods
- Pattern-based Methods
- Constraint-based Methods
- Cluster Validation

Hierarchical Methods

- **Main Idea**
  - Decomposing data objects into several levels of nested partitioning (tree of clusters)

  ![Hierarchical Clustering Diagram]

  - Agglomerative (AGNES): bottom-up approach
  - Divisive (DIANA): top-down approach
AGNES (Agglomerative Nesting)

- **Process**
  - Start with all single-node clusters
  - Iteratively merge the closest (the most similar) clusters
  - Eventually, all nodes belong to one cluster.

Distance Measures between Clusters

- **Single-Link Distance:**
  \[ d(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y) \]

- **Complete-Link Distance:**
  \[ d(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y) \]

- **Average-Link Distance:**
  \[ d(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y) \]

- **Centroid Distance:**
  \[ d(C_i, C_j) = d(m_i, m_j) \]
  where \( m_i \) and \( m_j \) are means of \( C_i \) and \( C_j \)
Comparison of Distance Measures

- **Single-Link Distance**
  - **Strength**
    - Handles non-sphere shape clusters
  - **Limitations**
    - Sensitive to noise

- **Complete-Link Distance**
  - **Strength**
    - Less sensitive to noise
  - **Limitations**
    - Biased large-sized clusters (or uneven-sized clusters)
DIANA (Divisive Analysis)

- **Process**
  - Start with one single clusters with all nodes
  - Iteratively divide the farthest (the most dissimilar) clusters
  - Eventually, all clusters have a single node.

- **Strength**
  - Not require the number of clusters, \( k \), in advance
  - Reveal a hierarchical structure of clusters

- **Weakness**
  - Require the stopping condition
  - Sensitive to noise
  - Not able to undo what was done previously
  - Not scalable, at least \( O(n^2) \) where \( n \) objects
    → How to improve efficiency and scalability?

Summary of Hierarchical Methods
BIRCH

- **BIRCH**
  - Balanced Iterative Reducing and Clustering using Hierarchies

- **Main Idea**
  - Building CF tree, a hierarchical data structure for multi-phase clustering

- **Process**
  - Phase 1: Scan DB to build an initial in-memory CF tree
  - Phase 2: Incrementally build the higher-level of the CF tree by grouping nodes

---

Clustering Feature Vectors

- **Clustering Feature**
  - Three-dimensional vector summarizing information of data objects for a cluster
    - \( CF = \langle n, LS, SS \rangle \) where \( n \) is the number of data points
      - \( LS \) is the linear sum of \( n \) points, \( \sum_{i=1}^{n} x_i \)
      - \( SS \) is the square sum of \( n \) points, \( \sum_{i=1}^{n} x_i^2 \)

- **Example**
  - \( (3,4), (2,6), (4,5), (4,7), (3,8) \)
  - \( CF = \langle 5, (16,30), (54,190) \rangle \)
Clustering Feature Vectors – cont’

- **Similarity Computation**
  - Average-link distance between two clusters, \( C_i \) and \( C_j \),
  
  \[
  d(C_i, C_j) = \sqrt{\frac{\sum_{x \in C_i} \sum_{y \in C_j} (x - y)^2}{n_i n_j}}
  \]

  can be calculated using the components in their clustering feature vectors.

- **Merging**
  - \( CF_1 = \langle n_1, LS_1, SS_1 \rangle \)
  - \( CF_2 = \langle n_2, LS_2, SS_2 \rangle \)
  - \( CF_1 + CF_2 = \langle n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2 \rangle \)

**CF (Clustering Feature) Tree**

- **CF Tree**
  - Height-balanced tree structure that stores the clustering features
  - Non-leaf nodes store the sums of descendent clustering features
### Summary of BIRCH

- **Strength**
  - Linearly scalable, \( \sim O(n) \) where \( n \) is the number of objects
  - Efficient memory usage

- **Weakness**
  - Only find spherical clusters by the distance measure
  - Only applicable to numeric attributes
    → To apply to categorical attributes, use a new similarity measure, e.g., ROCK

- **Reference**

### ROCK

- **ROCK**
  - Robust Clustering using Links

- **Main Idea**
  - Clustering categorical data
    - Previous approaches
      - Jaccard index (ratio of common attributes)
      - Distance measurement between boolean vectors
      - But, not reflect distribution patterns of attributes in the datasets
    - Use **neighbors** and **links** to measure similarity or proximity

- **Definitions**
  - **Neighbors**: two objects, a and b, are neighbors if \( sim(a, b) > \theta \)
    E.g., the \( sim \) function can be the Jaccard index
  - **Link**: the number of common neighbors between two objects
Examples

Example 1
- DB: \{a,b,c\}, \{a,b,d\}, \{a,b,f\}, \{a,c,d\}, \{a,d,e\}, \{b,c,e\}, \{b,f,g\}, \{c,d,e\}
- \(\theta = 0.5\)
- Neighbors of \{a,b,c\}: \{a,b,d\}, \{a,b,f\}, \{a,c,d\}, \{b,c,e\}
- Neighbors of \{c,d,e\}: \{a,c,d\}, \{a,d,e\}, \{b,c,e\}
- \(\text{Link}(\{a,b,c\}, \{c,d,e\}) = 2 \quad (\text{sim}(\{a,b,c\}, \{c,d,e\}) = 0.2)\)

Example 2
- DB: \{a,b,c\}, \{c,d,e\}, \{a,e,f\}, \{b,d,g\}, \{c,e,g\}
- \(\theta = 0.5\)
- Neighbors of \{a,b,c\}: None
- Neighbors of \{c,d,e\}: \{c,e,g\}
- \(\text{Link}(\{a,b,c\}, \{c,d,e\}) = 0 \quad (\text{sim}(\{a,b,c\}, \{c,d,e\}) = 0.2)\)

Summary of ROCK

Process
- Compute similarity matrix using link
- Run hierarchical (bottom-up) clustering

Strength
- Results depend on the other data objects
- Guarantee high intra-class similarity within a cluster

Weakness
- Not guarantee low inter-class similarity between clusters

Reference
Chapter 10, Clustering

- Partition-based Methods
- Hierarchical Methods
  - Density-based Methods
- Grid-based Methods
- Pattern-based Methods
- Constraint-based Methods
- Cluster Validation

Density-based Methods

- **Main Idea**
  - Clustering data objects located densely
  - Use density as the local clustering criterion

- **Issue**
  - Find clusters of arbitrary shapes
  - Handle noise
  - Determine density parameters as termination condition

- **Methods**
  - DBSCAN
  - OPTICS
DBSCAN

- **DBSCAN**
  - Density-based Spatial Clustering of Applications with Noise

- **ε-neighborhood**
  - \( \varepsilon \): minimum radius of neighborhood
  - \( \varepsilon \)-neighborhood of a data point \( p \), \( N_\varepsilon(p) = \{ q \in D \mid \text{dist}(p, q) \leq \varepsilon, p \neq q \} \)

- **Core / Border**
  - MinPts: minimum number of data points of an \( \varepsilon \)-neighborhood
  - Core: if \( |N_\varepsilon(p)| \geq \text{MinPts} \), \( p \) is a core
  - Border: if \( |N_\varepsilon(p)| < \text{MinPts} \), \( p \) is a border

Density-Reachability

- **Direct Density-Reachable**
  - An object \( q \) is directly density-reachable from the object \( p \), if \( p \) is a core and \( q \) is in \( \varepsilon \)-neighborhood of \( p \).
  - Symmetric or Asymmetric (?)

- **Density-Reachable**
  - An object \( q \) is (indirectly) density-reachable from the object \( p \), if there is a chain of points, \( p_1, p_2, \ldots p_n \), such that \( p_1 = p \), \( p_n = q \), and \( p_{i+1} \) is direct density-reachable from \( p_i \).
  - Symmetric or Asymmetric (?)
Density-Connectivity

- **Density-Connected**
  - An object $p$ is density-connected to an object $q$, if there is an object $o$, and both $p$ and $q$ are density-reachable from $o$.
  - Symmetric or Asymmetric (?)

- **Clusters**
  - A cluster is defined as a maximal set of density-connected points

---

DBSCAN Algorithm

- **Algorithm**
  1. Select an arbitrary data point $p$
  2. If $p$ is a core, retrieve all data points density-reachable from $p$ as a cluster
  3. Repeat (1)~(2) until there is no more data point to be selected

- **Results**

  ![DBSCAN Results](image)
Summary of DBSCAN

➢ **Strength**
  - Robust to noise
  - Find arbitrary shapes and sizes

➢ **Weakness**
  - Cannot handle varying densities
  - Sensitive to parameters, ε and MinPts
  - → Need a density-based algorithm without pre-setting parameters, e.g., OPTICS

➢ **Reference**

OPTICS

➢ **OPTICS**
  - Ordering Points to Identify Clustering Structures

➢ **Main Idea**
  - Clustering data points with varying density
  - Produce a specific order of data points
  - Provide a hierarchical structure of density-based clusters

➢ **Reachability-Distance**
  - Core-distance of p: distance between p and MinPts'th closest point
    - if \(|N_\varepsilon(p)| \geq \text{MinPts}\)
  - Reachability-distance of q from p: \(\max(\text{core-distance}(p), \text{distance}(p, q))\)
Summary of OPTICS

- **Strength**
  - Able to visualize graphically
  - Find density-based hierarchical clusters
  - Allow interactive clustering analysis

- **Weakness**
  - May not cover all data points

- **Reference**
Chapter 10, Clustering

- Partition-based Methods
- Hierarchical Methods
- Density-based Methods
  - Grid-based Methods
- Pattern-based Methods
- Constraint-based Methods
- Cluster Validation

Grid-based Methods

- Main Idea
  - Use multi-resolution grid data structure
  - Fast processing time, independent of the number of data objects

- Methods
  - STING
  - CLIQUE
STING

- **STING**
  - Statistical Information Grid approach

- **Main Idea**
  - Grid-based, density-based, and hierarchical clustering (using a multi-layer grid structure)
  - Top-down approach

- **Process**
  1. Initially compute statistical parameters for each cell in the bottom level
  2. Go to the top level
  3. Remove irrelevant cells, and go to the next level
  4. Repeat (3) until the bottom level is reached

Grid Structure

- **Multi-Layer Grids**
  - The spatial area of data points is divided into rectangular cells
  - There are several different levels of the cells
  - Each cell is partitioned into some smaller cells in the next lower level
  - Statistical parameters, such as count, mean, stdev, min, max, are pre-computed and stored in each cell in the lowest level
  - Relevance of the cells in higher levels is determined using the statistical parameter values
Summary of STING

_strength_
- Efficient – use only the statistical information in the bottom-level cells after the first scan of DB
- Able to parallelize

_Weakness_
- Inaccurate
- Cluster boundaries are always horizontal and vertical
- Only applicable to numeric attributes

_Reference_

CLIQUE

_CLIQUE_
- Clustering in Quest

_Main Idea_
- Grid-based and density-based clustering
- Applicable to high-dimensional data such that each dimension has a different data type
- Partition each dimension into cells (units)
- Find sub-dimensional spaces with high-density units
- A cluster represents a maximal set of connected dense units within a sub-dimensional space → **Subspace clustering**
CLIQUE Algorithm

- **Process**
  1. Partition each dimension into the same number of equal-length units
  2. Identify subspaces that contain dense regions using the Apriori-like algorithm
     → Iterative increment of sub-dimensional spaces with high density
  3. Determine dense regions on maximal dimension spaces
  4. Combine connecting regions

- **Example**

Summary of CLIQUE

- **Strength**
  - Discovery of informative subspaces in high dimensionality
  - Scalable and efficient in high dimensional data space

- **Weakness**
  - Accuracy depends on the grid size and density threshold

- **Reference**
Chapter 10, Clustering

- Partition-based Methods
- Hierarchical Methods
- Density-based Methods
- Grid-based Methods
- Pattern-based Methods
- Constraint-based Methods
- Cluster Validation

Pattern-based Methods

- **Main Idea**
  - Clustering data objects having similar patterns across dimensions

- **Issue**
  - Find clusters in high-dimensional space
  - Subspace clustering
  - Find hidden patterns on a sub-dimensional space

- **Methods**
  - p-Clustering
  - OP-Clustering
  - MAPLE
p-Clustering

- p-Clustering
  - Pairwise Clustering

- Main Idea
  - Devised to apply for gene expression data clustering
    - A thousands of genes (dimensions)
  - Bi-clustering
    - Extension of subspace clustering
  - Finding similar data patterns (not similar data values)
    - e.g., shifting and scaling patterns
    - Can we use the typical Euclidean distance to find similar patterns?

Pattern Examples

- Examples

**Shifting pattern**

**Scaling pattern**
Pattern Discovery

- **Pattern Detection Model**
  - To detect shifting patterns,
    \[
    p\text{Score} \left( \begin{bmatrix} d_{xa} & d_{xb} \\ d_{ya} & d_{yb} \end{bmatrix} \right) = \left| (d_{xa} - d_{xb}) - (d_{ya} - d_{yb}) \right| \leq \delta
    \]
  - To detect scaling patterns,
    \[
    p\text{Score} \left( \begin{bmatrix} d_{xa} & d_{xa} \\ d_{ya} & d_{ya} \end{bmatrix} \right) = \frac{d_{xa}}{d_{ya}} \leq \delta
    \]
  - \( \delta \) is a user-specified parameter

Algorithm of p-Clustering

- **Process**
  - Iteration of pairwise clustering
  - For each pair of objects,
    1. Sort dimensions in an ascending order of \((d_x - d_y)\)
    2. Detect maximal size patterns in a maximal dimension space

- **Reference**
Chapter 10, Clustering

- Partition-based Methods
- Hierarchical Methods
- Density-based Methods
- Grid-based Methods
- Pattern-based Methods
  - Constraint-based Methods
- Cluster Validation

Constraint-based Clustering

- **Motivation**
  - User’s feedback based on domain knowledge makes the clustering results more interesting
    - User-guided clustering analysis
  - Less parameters, but more user-desired constraints

- **Advantages**
  - User flexibility
    - Users can provide any constraints on what to be clustered
  - System optimization
    - It reduces the search space for efficient clustering
Constraint Types

- **Constraints on Individual Objects**
  - Selection of the set of objects to be clustered

- **Constraints on Clustering Parameters**
  - Selection of a desired range for each clustering parameter, e.g., $k$, $\varepsilon$

- **Constraints on Distance or Similarity Function**
  - Selection of the distance or similarity measure for clustering

- **Constraints on Individual Clusters**
  - Selection of the clusters interesting based on their properties

- **Semi-Supervised Clustering**
  - Giving small training sets as constraints

User-Guided Clustering

Diagram showing relationships between different entities such as students, professors, courses, and publications, with a highlighted target of clustering and user hints.
**Semi-Supervised Clustering (1)**

- **Semi-Supervised Clustering**
  - Provides a small training set consisting of similar objects (must link) and dissimilar objects (cannot link)

**Semi-Supervised Clustering (2)**

- **Why Not Semi-Supervised Clustering?**
  - More information (in multiple relations) is needed to judge whether two data objects (tuples) are similar
  - A user may not be able to provide a good training set
  - It is much easier for a user to specify an attribute as a hint, such as a student’s research area
Chapter 10, Clustering

- Partition-based Methods
- Hierarchical Methods
- Density-based Methods
- Grid-based Methods
- Pattern-based Methods
- Constraint-based Methods

Cluster Validation

- Definition
  - Assessing the quality of clustering results

- Why Validating?
  - To avoid finding clusters formed by chance
  - To compare clustering algorithms
  - To choose clustering parameters

- Methods
  - External index: when "ground truth" is available
  - Internal index: when "ground truth" is unavailable
Internal Index

➢ Error Measures
  • Absolute error = |x_i - x_i'|
  • Squared error = (x_i - x_i')^2

➢ Sum of Squared Error (SSE)
  • Measure of cohesiveness by within-cluster sum of squared error
    \[ WSS = \sum_{i} \sum_{x \in C_i} (x - m_i)^2 \]
  • Measure of separability by between-cluster sum of squared error
    \[ BSS = \sum_{i} |C_i| \cdot (m - m_i)^2 \]
  • Relationship between WSS and BSS ?

External Index (1)

➢ Notations
  • N: the total number of data objects
  • C = \{C_1, C_2, ..., C_n\}: the set of clusters reported by a clustering algorithm
  • P = \{P_1, P_2, ..., P_m\}: the set of "ground truth" clusters

➢ Incident Matrix
  • (N x N) matrix
  • C_{ij} = 1 if two data objects O_i and O_j belong to the same cluster in C
  • C_{ij} = 0 otherwise
  • P_{ij} = 1 if O_i and O_j belong to the same "ground truth" cluster in P
  • P_{ij} = 0 otherwise
External Index (2)

- **Result Categories**
  - **SS**: $C_i = 1$ and $P_j = 1$ (agree)
  - **DD**: $C_i = 0$ and $P_j = 0$ (agree)
  - **SD**: $C_i = 1$ and $P_j = 0$ (disagree)
  - **DS**: $C_i = 0$ and $P_j = 1$ (disagree)

- **Rand Index**
  - $Rand = \frac{|SS|+|DD|}{|SS|+|DD|+|SD|+|DS|}$

- **Jaccard Index**
  - Jaccard Coefficient = $\frac{|SS|}{|SS|+|SD|+|DS|}$

f-Measure

- **Recall & Precision**
  - Comparison between an output cluster and a ground-truth cluster
  - Let an output cluster $X$, and a ground-truth cluster $Y$
  - Recall (Sensitivity or True positive rate) = $\frac{|X \cap Y|}{|Y|}$
  - Precision (Positive predictive value) = $\frac{|X \cap Y|}{|X|}$

- **f-Measure**
  - Harmonic mean of Recall and Precision
  - $f$-measure = $2 \times (\text{Recall} \times \text{Precision}) / (\text{Recall} + \text{Precision})$
Statistical P-Value

- **P-value of Hyper-geometric Distribution**
  - Let the set of all data objects, \( N \)
  - Let an output cluster \( X \), and a ground-truth cluster \( Y \)
  - Probability that at least \( k \) data objects in \( X \) are included in \( Y \)
  
  \[
  P = 1 - \sum_{i=0}^{k-1} \binom{|Y|}{i} \binom{|N| - |Y|}{|X| - i} \binom{|N|}{|X|}
  \]
  where \( k = |X \cap Y| \)

  - A low P-value indicates it is less probable that the cluster \( X \) is produced by chance
  - \(-log(P)\) is usually used for clustering evaluation

Questions?

- Lecture Slides on the Course Website,
  "www.ecs.baylor.edu/faculty/cho/4352"