Chapter 10, Clustering

Young-Rae Cho
Associate Professor
Department of Computer Science
Baylor University

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>q</th>
<th>r</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>q</td>
<td>r</td>
<td>q+r</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>s</td>
<td>t</td>
<td>s+t</td>
<td></td>
</tr>
<tr>
<td>sum</td>
<td>q+s</td>
<td>r+t</td>
<td>p</td>
<td></td>
</tr>
</tbody>
</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
  - 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**
  - Theoretical method: Enumerate exhaustively all possible partitions and select the best one
  - Heuristic method: \( k \)-means, \( k \)-medoids (PAM), CLARANS
**k-Means**

**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.

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

---

**Example of k-Means**

1. Random partition
2. Compute cluster means
3. Re-assign objects
4. Compute cluster means
5. Re-assign objects
Summary of *k*-Means

### 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
- Sometimes 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

- 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 swaped with \(O_{\text{random}}\)

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

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

- **Weakness**
  - Not scalable to a large data set
  - $O(k(n-k)^2)$ where $n$ objects and $k$ clusters
    - To improve the efficiency, need a heuristic algorithm
      e.g., sampling algorithm such as CLARA and CLARANS

**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)
Hierarchical Methods

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

Step 0  Step 1  Step 2  Step 3  Step 4

agglomerative (AGNES): bottom-up approach

`a b c d e`

Step 4  Step 3  Step 2  Step 1  Step 0
divisive (DIANA): top-down approach

divide and conquer

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
- Complete-Link Distance
- Average-Link Distance
- Centroid Distance
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.
Summary of Hierarchical Methods

- **Strength**
  - Not require the number of clusters, $k$, in advance

- **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
    - To improve the scalability, use a special data structure, e.g., BIRCH

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 (leaf nodes of the 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
  - \( \text{CF} = \langle n, LS, SS \rangle \) 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)
\( \text{CF} = \langle 5, (16,30), (54,190) \rangle \)

**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**
- \( \text{CF}_1 = \langle n_1, LS_1, SS_1 \rangle \)
- \( \text{CF}_2 = \langle n_2, LS_2, SS_2 \rangle \)
- \( \text{CF}_1 + \text{CF}_2 = \langle n_1+n_2, LS_1+LS_2, SS_1+SS_2 \rangle \)
**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

```
  CF1  CF2  ...  CFk
  |    |    |     |
  CF11 CF12 ... CF1k
  |    |    |     |
  CF21 CF22 ... CF2k
  ...  ...  ...  ...
```

- **Parameters**
  - Branching factor: the maximum number of children per non-leaf node
  - Threshold: the maximum distance of sub-clusters at the leaf nodes

**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\}
  - \( Link(\{a,b,c\}, \{c,d,e\}) = 2 \quad (sim(\{a,b,c\}, \{c,d,e\}) = 0.2) \)

- **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\}
  - \( Link(\{a,b,c\}, \{c,d,e\}) = 0 \quad (sim(\{a,b,c\}, \{c,d,e\}) = 0.2) \)
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**

---

CSI 4352, Introduction to Data Mining

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

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

\( \text{MinPts} = 7 \)
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**

Summary of DBSCAN

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

- **Weakness**
  - Cannot handle varying densities
  - Sensitive to parameters, \( \epsilon \) and MinPts
    - Need a density-based algorithm without pre-setting parameters, e.g., OPTICS

- **Reference**